МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Web-додаток підтримки дистанційного навчання»

Здобувачки групи ІТ-91 Подус Катерини Олександрівни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Катерина ПОДУС
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Володимир НАГОРНИЙ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Подус Катерині Олександрівні

1 Тема роботи *Web-додаток підтримки дистанційного навчання*

керівник роботи *Нагорний Володимир В'ячеславович, к.т.н., доцент* ,

затверджені наказом по університету від «*29*» травня 2023 р. №0588-VI

2 Строк подання студентом роботи «7» червня 2023 р.

3 Вхідні дані до роботи документації бібліотек і фреймворків, матеріали дослідження та статистики, продукти-аналоги

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз і дослідження теми; дослідження продуктів-аналогів; визначення функціональних вимог; моделювання та проєктування; розробка функціоналу; повне тестування.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) аналіз сучасного стану дистанційного навчання; постановка задачі; аналіз продуктів-аналогів; порівняння продуктів-аналогів; функціональні вимоги; контекстна діаграма в нотації IDEF0; діаграма декомпозиції першого рівня в нотації IDEF0; діаграма варіантів використання; діаграма послідовності; діаграма діяльності; інтерфейси; фізична модель даних; засоби реалізації; сторінка профілю та авторизації; створення освітнього закладу, перегляд власних і тих, до яких під'єднано; форма створення навчального класу та курсу й закріплення за ним викладачів і класів, перегляд учасників курсу; додавання та

перегляд навчальних матеріалів (лекції, практики, тести); надсилання робіт, перегляд журналу оцінок, перегляд і оцінювання робіт, проходження тестування; апробація; висновки.

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7. Дата видачі завдання 22.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Проведення аналізу й дослідження даної теми	до 20.02.2023	
2	Проведення дослідження продуктів-аналогів	до 25.03.2023	
3	Визначення функціональних вимог	до 01.04.2023	
4	Виконання моделювання та проєктування	до 17.04.2023	
5	Розробка функціоналу	до 20.05.2023	
6	Проведення тестування	до 31.05.2023	

Студент _____ Катерина ПОДУС
(підпис)

Керівник роботи _____ к.т.н., доц. Володимир НАГОРНИЙ
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підтримки дистанційного навчання».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 21 найменування, додатків. Загальний обсяг роботи – 212 сторінок, у тому числі 69 сторінок основного тексту, 3 сторінки списку використаних джерел, 139 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку підтримки дистанційного навчання.

У першому розділі роботи було проведено аналіз сучасного стану дистанційного навчання, а саме: огляд останніх досліджень і публікацій, аналіз програмних продуктів-аналогів і постановка задачі.

У другому розділі роботи було виконано моделювання та проєктування web-додатку підтримки дистанційного навчання, а саме: функціональне моделювання, моделювання варіантів використання та проєктування моделі бази даних.

Третій розділ містить архітектуру програмного додатку, програмну реалізацію та приклади використання web-додатку.

Результатом проведеної роботи став web-додаток підтримки дистанційного навчання.

Практичне значення роботи полягає у використанні web-додатку для надання та засвоєння освітніх матеріалів.

Ключові слова: web-додаток дистанційного навчання, освіта, освітній заклад, вчитель, учень, навчальний клас, курси, лекції, практичні завдання, тести, журнал, оцінка.

ЗМІСТ

ВСТУП

Впродовж останніх десятиліть сфера освіти стрімко розвивається завдяки впровадженню інноваційних технологій. Однією з найрозповсюдженіших інновацій є застосування web-додатків дистанційного навчання. Їх використання підвищує ефективність роботи як учнів, так і вчителів, бо дозволяє підлаштовуватись до будь-яких життєвих змін, зберігаючи всі освітні матеріали в одному місці.

Наскільки відомо, одна з перших форм дистанційного навчання була запропонована в 1840 році Ісааком Пітманом у вигляді обміну листами для навчання студентів Англії. Пізніше такий підхід до навчання стали пробувати також Чарльз Тюссе та Густав Лангеншейдт у Німеччині [1]. Після цього світ поступово дізнавався про нові технології дистанційного навчання й проводив їх апробацію для подальшого впровадження в освітніх закладах.

Оскільки отримання освіти є постійною та невід'ємною складовою життя людини, то не дивно, що всі прагнуть зробити цей процес максимально зручним. Тому в наш час кожен має можливість використовувати web-додатки дистанційного навчання, які містять велику кількість різноманітних освітніх матеріалів у форматі відео, лекцій, практичних завдань, тестів тощо. Такий спосіб навчання особливо підходить людям, які не можуть відвідувати свій навчальний заклад через певні обставини.

Процес дистанційного навчання для багатьох може здатися незвичним, бо для цього потрібно опановувати нові технічні навички, але, як показав попередній досвід такого навчання під час пандемії, учасники освітнього процесу швидко звикають до нового формату. Згідно з даними опитування, проведеного центром інноваційної освіти «Про.Світ» серед 5410 респондентів з 24 областей України, 70% опитуваних вважають, що реалізація дистанційного навчання не є складною [2]. Особливо зручним є те, що більше не потрібно залежати від місця проживання та витрачати зайві кошти, не потрібно мати

велику кількість паперів, бо вся необхідна інформація буде зберігатися лише в одному місці. Крім цього, за допомогою web-додатку досить зручно переглядати успішність виконання робіт та їх термінів, проводити тестування та оцінювання учнів.

Також одним із варіантів застосування web-додатків для дистанційної освіти є їх використання в змішаному навчанні. Тобто, освітні процеси можуть поділятися на два види робіт, одні з яких відбуваються очно, наприклад практики, а інші дистанційно, такі як: засвоєння лекційних матеріалів і проходження тестування. Такий розподіл робіт обумовлений тим, щоб не витрачати зайвого часу, знаходячись в освітньому закладі та надавати можливість учням опрацьовувати матеріали самостійно.

Отже, метою даного проєкту є створення web-додатка підтримки дистанційного навчання для надання та засвоєння освітніх матеріалів.

Для досягнення мети необхідно виконати наступні поставлені задачі:

–виконати аналіз і дослідження даної теми;

–провести дослідження продуктів-аналогів;

–визначити функціональні вимоги;

–виконати моделювання та проєктування;

–розробити функціонал;

–провести повне тестування.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДИСТАНЦІЙНОГО НАВЧАННЯ

1.1 Огляд останніх досліджень і публікацій

Останнім часом в світі зростає тенденція використання додатків дистанційного навчання через велику кількість різних факторів, які впливають на можливість навчатися очно. Потреба в таких додатках в Україні збільшилась з початку пандемії коронавірусу та триває досі через війну. Багато людей вимушені покидати свій дім у зв'язку з небезпекою, але навіть знаходячись на відстані, залишається необхідність продовжувати навчання [3]. Через це, багато шкіл, університетів або коледжів потребують зберігати всі свої матеріали в одному місці та поширювати доступ для своїх учнів.

Web-додаток підтримки дистанційного навчання дозволяє контролювати процес освіти та надавати доступ до проходження тестів для контролю та засвоєння теоретичного матеріалу [4]. Такі додатки можуть повністю замінити очне навчання, тому що, зазвичай, більшість з них дають змогу вчителям створювати курси та навчальні класи, надавати освітні матеріали, створювати тести та перевіряти роботи своїх учнів. В свою чергу, учні мають можливість переглядати всю інформацію стосовно курсу, матеріалів, оцінок, можуть проходити тести та завантажувати свої роботи на перевірку. Тому, онлайн-навчання має на меті полегшити процес здобування освіти шляхом використання web-додатків дистанційного навчання. В такому виді здобування знань можна виділити наступні переваги:

–можна займатися у вільний час, враховуючи терміни робіт;

–всі матеріали можна переглядати скільки завгодно разів для кращого засвоєння інформації;

–учень має змогу навчатися з будь-якого місця, маючи при собі лише гаджет.

Але варто також пам'ятати про деякі недоліки:

–ніколи достовірно не відомо, що учень сам виконував певні види робіт або тести;

–втрачена жива комунікація з іншими учнями та вчителями;

–відсутність повного практичного досвіду в тих учнів, що потребують очної практики, наприклад, студентів медичних спеціальностей [5].

Значимість використання онлайн-навчання тепер змушує все частіше обирати правильний додаток, спрямований на зручний підхід роботи як для вчителів, так і для учнів. Кожен освітній заклад сам для себе вирішує за якими параметрами можна обрати додаток для навчання, але Міністерство освіти і науки України рекомендує обирати за такими критеріями:

–додаток має бути спрямований на будь-які види робіт, такі як: створення та проходження тестування, доступ до всіх освітніх матеріалів курсу, можливість здачі робіт та їх перевірки шляхом виставлення оцінок;

–легкий та зручний інтерфейс для користувачів без попереднього досвіду;

–використання додатку з будь-якого пристрою;

–можливість використовувати додаток для будь-яких закладів освіти таких, як: школи, університети, коледжі;

–безпека стосовно збирання власних даних користувачів [6].

Отже, під час створення web-додатку дистанційного навчання потрібно враховувати найбільш поширені потреби та вимоги людей на основі раніше проведених досліджень та аналогів відповідної програми.

1.2 Аналіз програмних продуктів – аналогів

Для вибору задовільного навчального web-додатку існує велика кількість продуктів-аналогів зі схожим функціоналом. Кожен з них виділяється серед інших своїми певними перевагами. Саме тому, для виявлення основного важливого функціоналу, що повинен бути інтегрований у власний проєкт, було проведено аналіз серед таких застосунків:

–Google Classroom

–Moodle

–Mix

1.2.1 Google Classroom

Google Classroom – дуже популярний сервіс для навчання, яким можна користуватися як з комп'ютера, так і з мобільного пристрою [7]. Програма має легкий інтерфейс і достатню кількість функціоналу. На головній сторінці додатку, яка зображена на рисунку 1.1, відразу можна побачити весь список курсів, до яких під'єднаний користувач, а також є можливість перейти до перегляду всього списку наявних завдань або термінів здачі робіт в календарю.

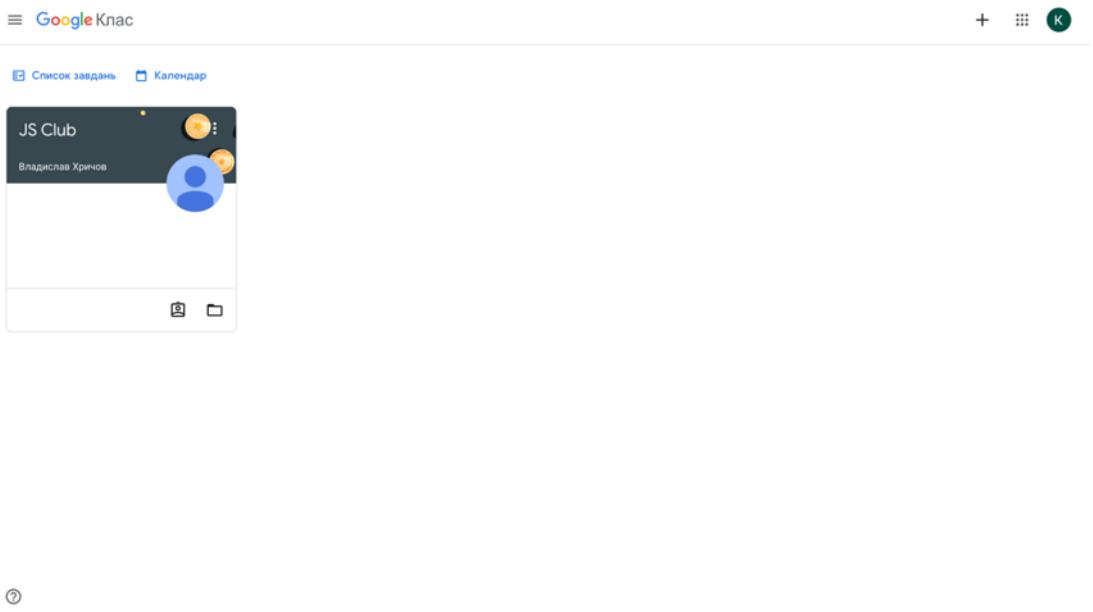


Рисунок 1.1 – Головна сторінка додатку «Google Classroom»

Основне наповнення курсу можна переглянути на рисунку 1.2. Всередині курсу можна гортати стрічку з усіма доступними матеріалами, у вкладці з завданнями можна переглядати умови робіт і надсилати їх на перевірку, а вкладка «Користувачі» відображає всіх учасників курсу. Також для зручної навігації по web-додатку зліва відкривається меню з переліком доступних сторінок. В Google Classroom досить незручно шукати потрібну лекцію серед всіх матеріалів у стрічці, було би краще зробити окремо лекції та завдання з можливістю фільтрації і сортування. Оцінку за роботу можна переглянути, якщо

перейти на відповідне завдання, тобто не можна подивитися успішність виконання робіт в цілому.

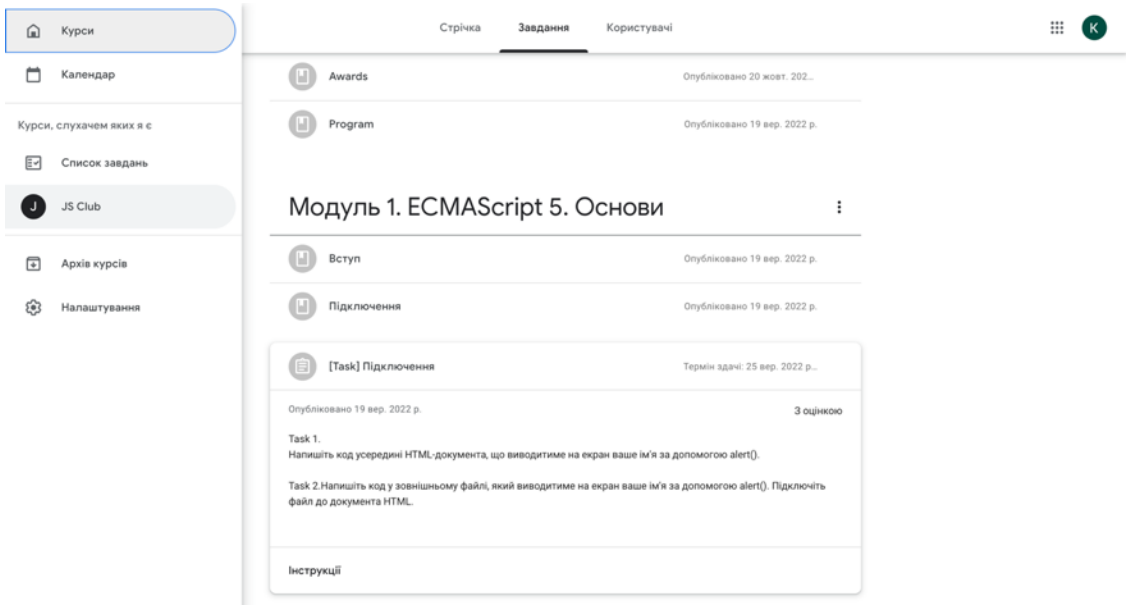


Рисунок 1.2 – Сторінка курсу «Google Classroom»

1.2.2 Moodle

Далі розглянемо ще одну відому систему управління навчанням – «Moodle» [8]. Дане навчальне середовище є одним з головних конкурентів серед інших аналогів, бо має потужний функціонал, який сприяє зручній праці з будь-яким видом навчальної діяльності. Moodle, також як і Classroom, має дві версії відображення: веб-версія та з мобільних пристроїв. Основними можливостями, що надає дана програма є: створення курсів; надання освітніх матеріалів різних форматів; створення різних видів тестів для перевірки знань; оцінювання; зворотній зв'язок між вчителем та учнем; створення форумів або чатів; планування різного виду занять, включаючи відео та аудіо конференції. Для використання даної програми потрібно заздалегідь мати впевнені навички роботи з різними web-застосунками для легшого розуміння роботи в подібних середовищах. Сторінка списку курсів зображена на рисунку 1.3.

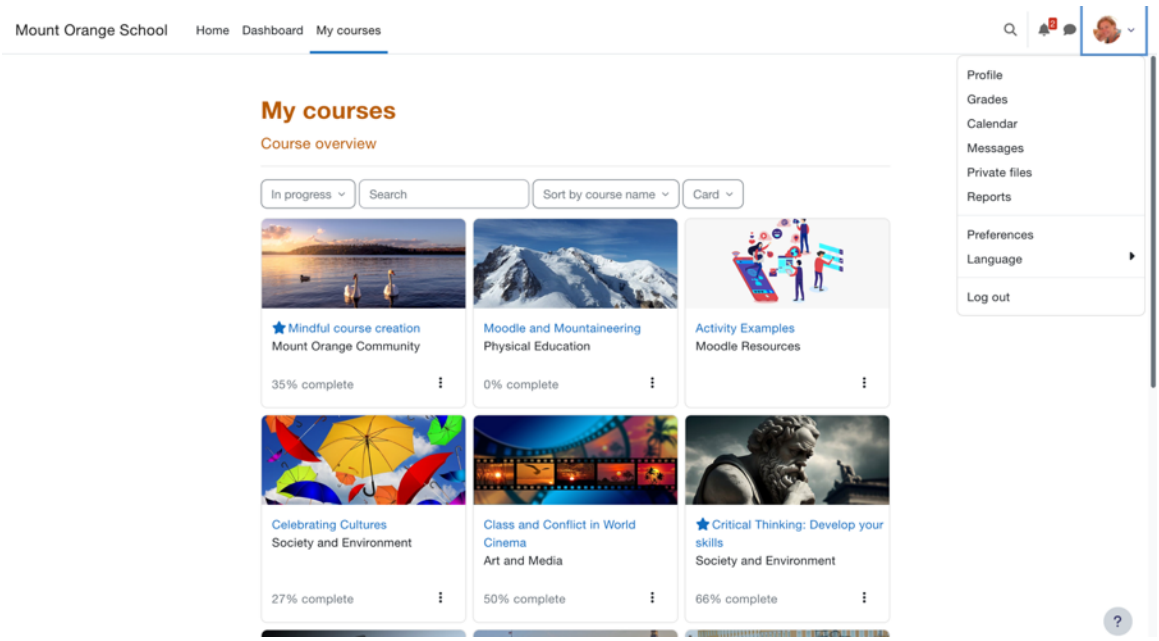


Рисунок 1.3 – Сторінка списку курсів «Moodle»

В самому курсі можна переглядати всю основну інформацію про курс, всіх учасників, виставлені оцінки та багато іншої додаткової інформації. Сторінка наповнення курсу наведена на рисунку 1.4.

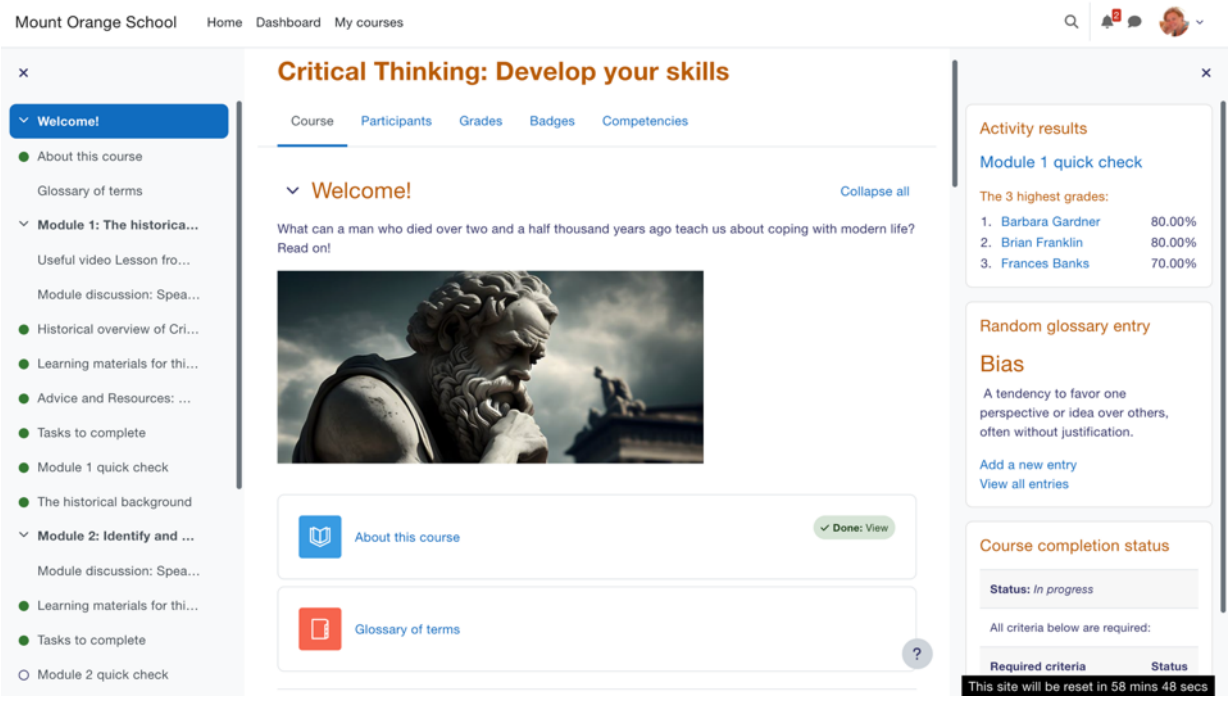


Рисунок 1.4 – Сторінка певного курсу «Moodle»

В журналі легко подивитися успішність перевірених робіт за категоріями по темі та виду завдання. Вигляд журналу в курсі наведений на рисунку 1.5.

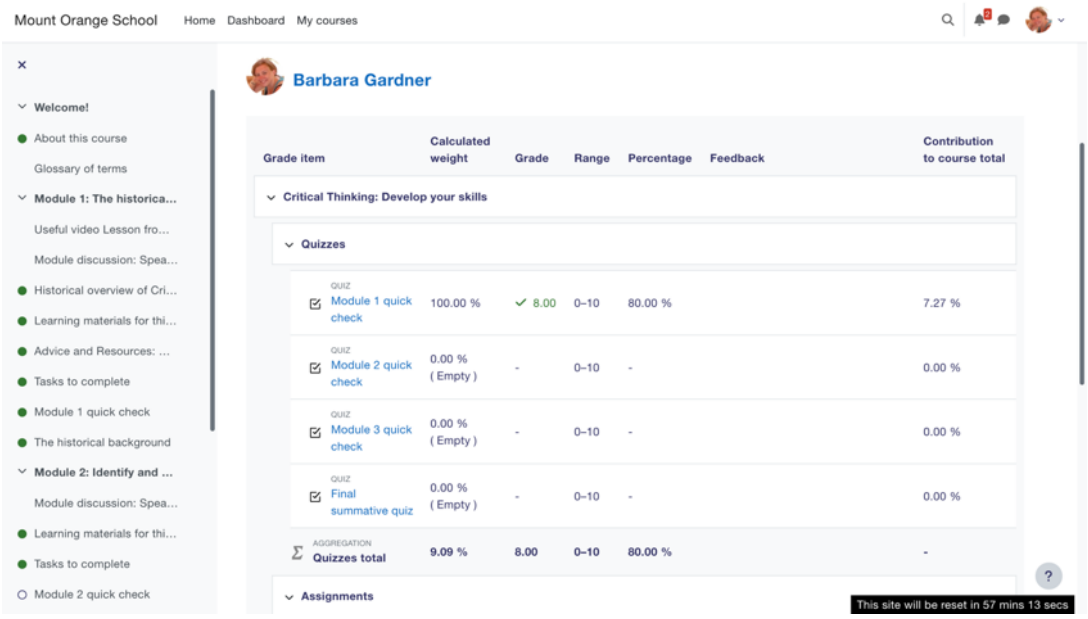


Рисунок 1.5 – Журнал курсу «Moodle»

Також в даній програмі присутній блок тестування, який може передбачати як один варіант відповіді, так і декілька. Вкладка з тестуванням знань зображена на рисунку 1.6.

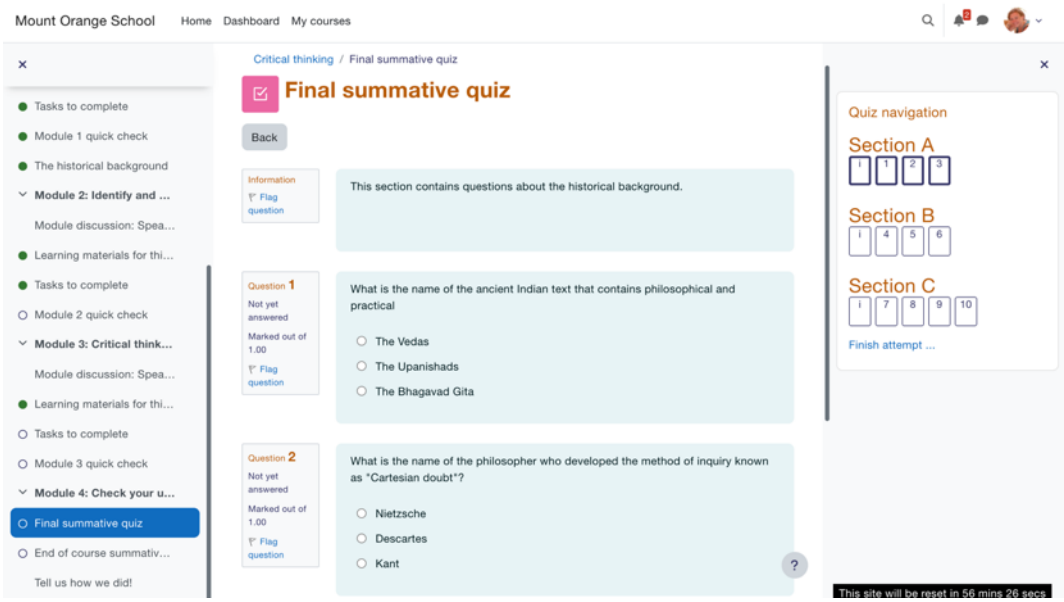


Рисунок 1.6 – Блок тестування знань «Moodle»

1.2.3 Mix

Наступним продуктом-аналогом беремо до розгляду систему «Mix» [9]. Цей додаток повністю спрямований на використання тільки для Сумського державного університету, але містить достатньо функціоналу і повністю задовольняє потреби користувача до дистанційного навчання. Даний продукт містить всі необхідні інструменти для такого функціоналу як: створення курсів; надання освітніх матеріалів у вигляді лекцій, практик і тестів; підтримки зв'язку з вчителями у вигляді особистого спілкування та оцінювання. «Mix» як і попередні додатки добре відображається з різних пристроїв. Нижче, на рисунку 1.7, зображена особиста сторінка з курсами, до яких користувач є під'єднаним.

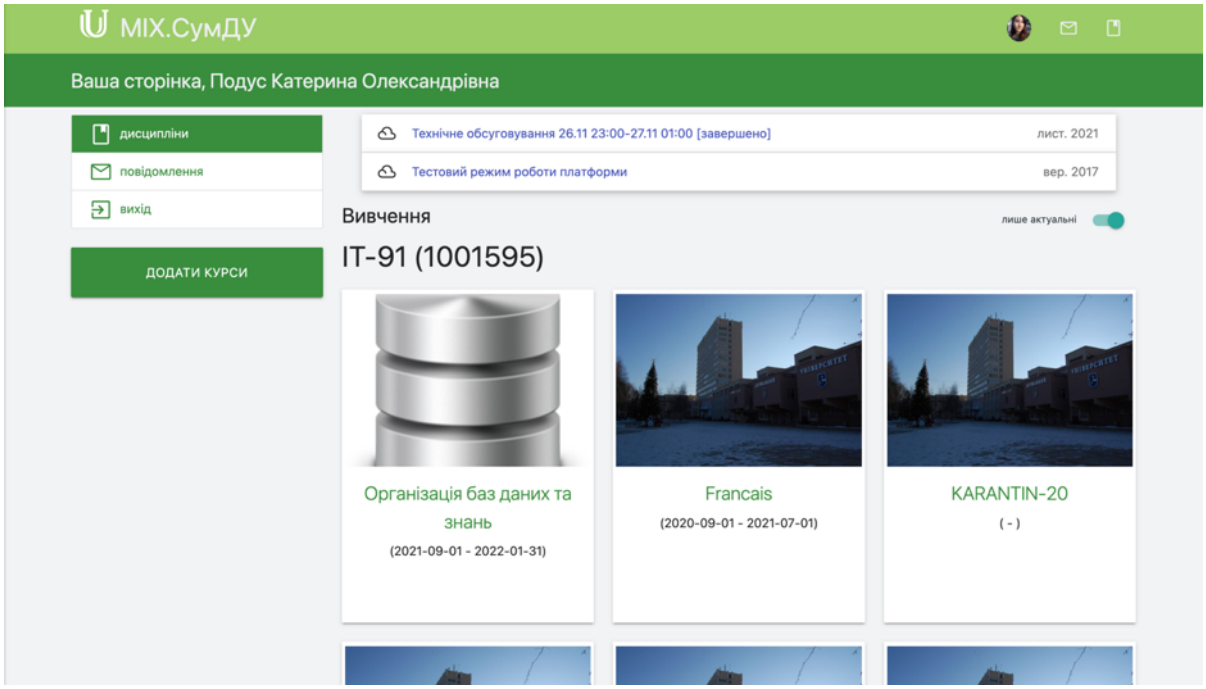


Рисунок 1.7 – Особиста сторінка з курсами в системі «Mix»

Основними частинами курсу в «Mix» є лекції та план з завданнями. В плані також можна бачити свою загальну успішність по предмету та у виконанні всіх завдань окремо. Концепція курсу наведена на рисунку 1.8.

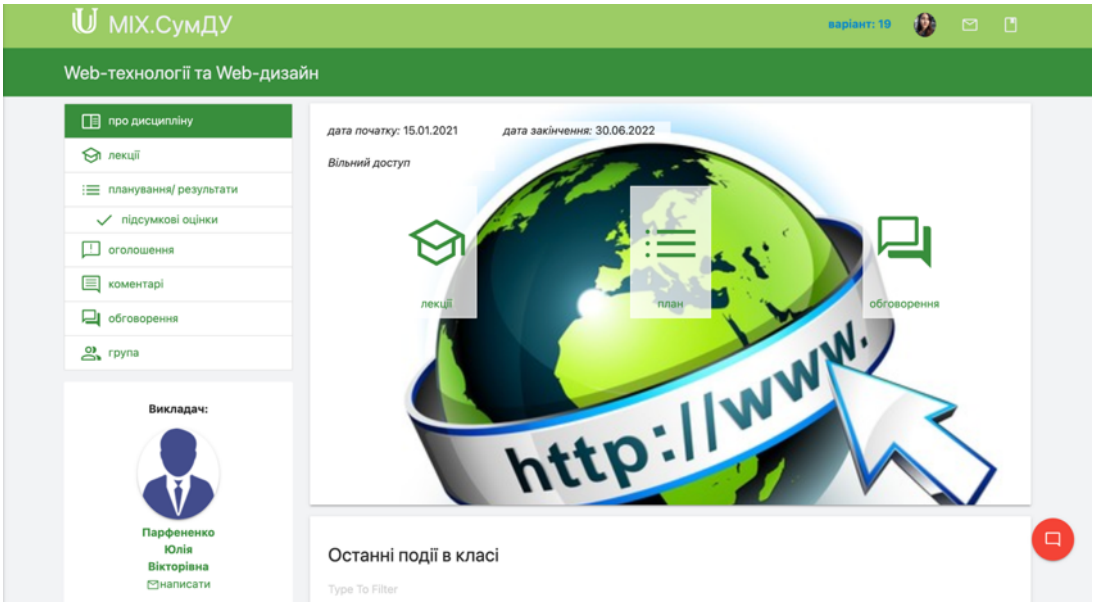


Рисунок 1.8 – Сторінка певного курсу в «Мix»

Блок з тестуванням наведений на рисунку 1.9. Дане тестування може передбачати різні варіанти відповідей.

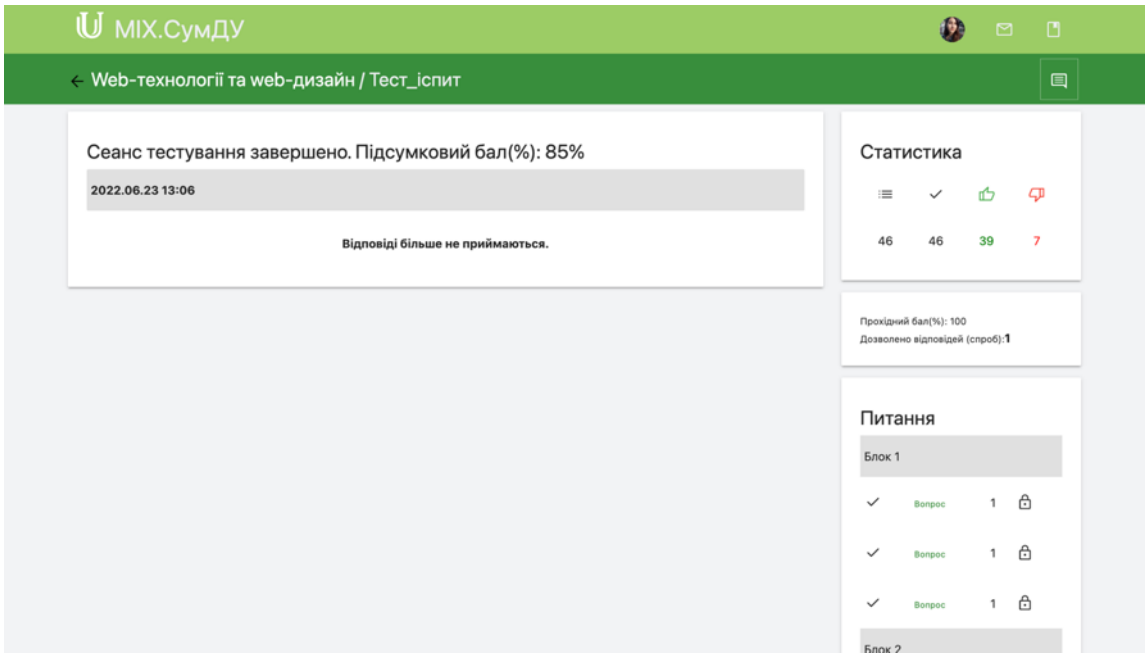


Рисунок 1.8 – Сторінка тесту в «Мix»

1.2.4 Порівняння характеристик продуктів-аналогів

Для порівняння характеристик продуктів-аналогів «Google Classroom», «Moodle», «Mix» була складена таблиця 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів платформ

Характеристика платформ дистанційного навчання	**Google Class-room**	**Moodle**	**Mix**	**Власна розробка**
Сучасний дизайн	-	+	+	+
Легкий інтерфейс і навігація	+	-	+	+
Відображення з різних пристроїв	+	+	+	+
Можливість використовувати додаток для будь-якого освітнього закладу або курсу	+	+	-	+
Фільтрація освітніх матеріалів курсу	-	+	-	+
Наявність перегляду успішності учня по курсу за кожне завдання	+	+	+	+
Наявність блоку тестування	-	+	+	+

Підсумовуючи всі загальні характеристики продуктів-аналогів, буде розроблено власний додаток дистанційного навчання з урахуванням переваг та недоліків альтернативних застосунків.

1.3 Постановка задачі

Метою даного проєкту є створення web-додатку підтримки дистанційного навчання для надання та засвоєння освітніх матеріалів.

Для досягнення мети, спочатку потрібно детально проаналізувати предметну область, знаходячи свіжі матеріали, публікації та статистики з даної теми. Виходячи з результатів дослідження, буде зрозуміло які саме питання зараз

постають щодо дистанційного навчання та його актуальність в цілому. Завдяки цьому, можна буде легко адаптувати застосунок під визначені цілі.

Наступним кроком важливо переглянути альтернативні продукти, щоб виділити в кожного з них кращі сторони для розуміння потреб користувача.

Після того як предметна область була досліджена та розглянуті продукти-аналоги, можна визначити функціональні вимоги з урахуванням всіх встановлених аспектів.

Маючи розуміння про кінцевий продукт, потрібно розбити його на декілька етапів, які можна чітко описати та візуально представити у вигляді діаграм, тобто провести моделювання web-додатку. Зазвичай, в таких діаграмах відображаються взаємозв'язки між компонентами програми та описуються ролі користувачів. Також, потрібно виконати проєктування web-додатку для побудови моделі бази даних.

Одним з основних етапів створення web-додатку є розробка функціоналу програми. Для цього необхідно визначити інструменти для написання коду та розробити сам функціонал, згідно до визначених функціональних вимог.

Після того як написання програми завершено, необхідно виконати мануальне тестування web-додатку.

Результатом даного проєкту повинен бути web-додаток підтримки дистанційного навчання, завдяки якому всі освітні процеси будуть доступні для кожного в будь-який час і з будь-якого місця.

Цільовою аудиторію розроблюваного продукту є вчителі та учні різних освітніх закладів або люди, які прагнуть створювати власні курси.

Враховуючи всі переваги та недоліки продуктів-аналогів, а також сформовані потреби з результатів дослідження, було визначено основні функціональні вимоги до web-додатку:

–можливість створення та редагування власного аккаунту;

–створення будь-якого дистанційного освітнього закладу, наприклад, школи або університету, в якому адміністратор може створювати курси і навчальні класи;

–можливість адміністратору закладу закріпляти за певним курсом вчителя та клас;

–можливість вчителю надавати освітні матеріали для курсу у вигляді лекцій або практик та зазначати терміни здачі робіт;

–можливість учням переглядати освітні матеріали курсу та завантажувати роботи;

–можливість вчителю заносити оцінки до журналу для спільного перегляду успішності учнів;

–можливість створення та проходження тестів для контролю та засвоєння теоретичного матеріалу [4].

Повний перелік поставлених задач для розробки проєкту наведений в технічному завданні в додатку А. Також календарний план всіх робіт наведений в додатку Б.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ WEB-ДОДАТКУ ПІДТРИМКИ ДИСТАНЦІЙНОГО НАВЧАННЯ

2.1 Структурно-функціональне моделювання

Одним з важливих етапів розробки проєкту є моделювання, оскільки, завдяки ньому з'являється можливість завчасно спланувати правильний хід робіт, визначити структуру проєкту, проаналізувати та оцінити всі функції системи і взаємозв'язки між ними.

Для опису всіх бізнес-процесів найкращим варіантом є їх подання у вигляді контекстної діаграми у нотації IDEF0. Дана методологія функціонального моделювання має ієрархічну структуру, яка логічно поєднує вхідні та вихідні потоки даних, функції та підфункції. Компонентами даної діаграми виступають функції у вигляді блоків, стрілки у вигляді даних або матеріальних об'єктів. Стрілки зліва, що поступають на вхід відображають змінювані дані, а стрілки виходу, що справа, відображають результат виконання. Зверху знаходяться стрілки управління, які визначать правила виконання робіт, а стрілки знизу відображають механізми, завдяки яким виконується робота [14].

Діаграма IDEF0 для web-додатку підтримки дистанційного навчання з точки зору вчителя зображена на рисунку 2.1.

Рисунок 2.1 – Контекстна діаграма бізнес-процесів у нотації IDEF0 з точки зору вчителя

Виходячи з отриманої діаграми бізнес-процесів на рисунку 2.1 можна побачити, що на вхід подаються запити користувачів на створення курсів, додавання матеріалів курсу, створення тестів, питань та варіантів відповідей, оцінювання робіт учнів. Результатами даних запитів на виході є: курс, матеріали курсу, тести для контролю засвоєння знань, оцінка за практичну роботу учня. Механізмами виступають вчитель, база даних та апаратне забезпечення. До методів управління процесами відноситься інструкція по використанню web-додатка.

Наступим кроком до кращого розуміння функціонування програми є декомпозиція IDEF0, яка дозволяє детально розглянути вже визначені бізнес-процеси, шляхом їх розбиття на підпроцеси. Таким чином, стане легше аналізувати всі елементи процесів, чітко визначити їх призначення та назначити кожному ролі. Декомпозиція IDEF0 з точки зору вчителя представлена на рисунку 2.2.

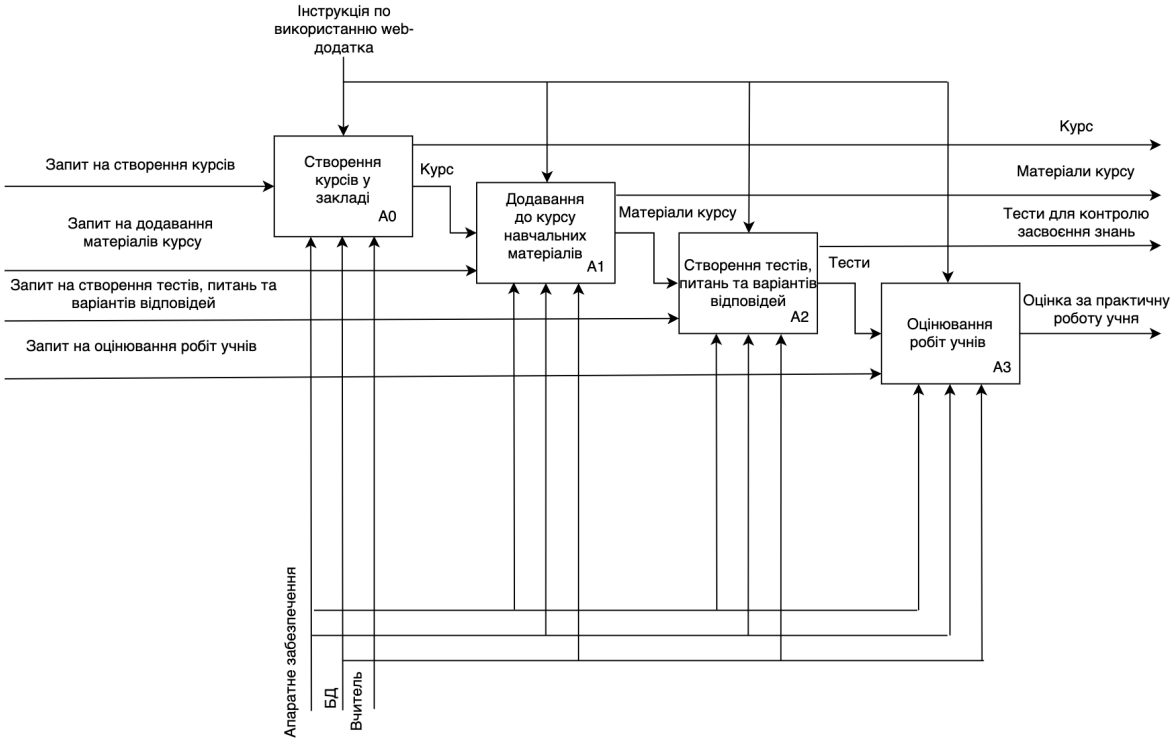


Рисунок 2.2 – Перший рівень декомпозиції IDEF0-діаграми з точки зору вчителя

При декомпозиції IDEF0 для функціонування кожного блоку були визначені вхідні дані, механізми та правила. Загальним для всіх блоків методом управління процесами є інструкція по використанню web-додатка. Загальними для всіх блоків механізмами є вчитель, база даних та апаратне забезпечення.

Першим блоком є створення курсу, він потребує відповідного запиту. На виході буде створено новий курс.

Наступний блок по додаванню навчальних матеріалів включає в себе запити по створенню лекцій, практик, тестів. На виході маємо матеріали курсу.

Для блоку створення тестів з питаннями та варіантами відповідей необхідно мати відповідний запит. На виході маємо тести для контролю засвоєння знань.

Блок для оцінювання робіт учнів повинен мати відповідний запит для виставлення оцінки. На виході будуть оцінки за практичні роботи учнів.

2.2Моделювання варіантів використання

Розуміння взаємодії користувачів з системою необхідно для побудови правильної логіки програми, саме тому створюють діаграми варіантів використання (Use Case), які наглядно показують всі зв’язки між акторами і варіантами використання системи.

Акторами називаються користувачі, які взаємодіють з системою, також це можуть бути інші системи, програмне забезпечення, організації тощо [15].

Варіант використання – це очікувана поведінка системи на певну дію актора, завдяки чому показується результат їх взаємодії [16].

На діаграмі 2.3 зображено зв’язок акторів з системою.

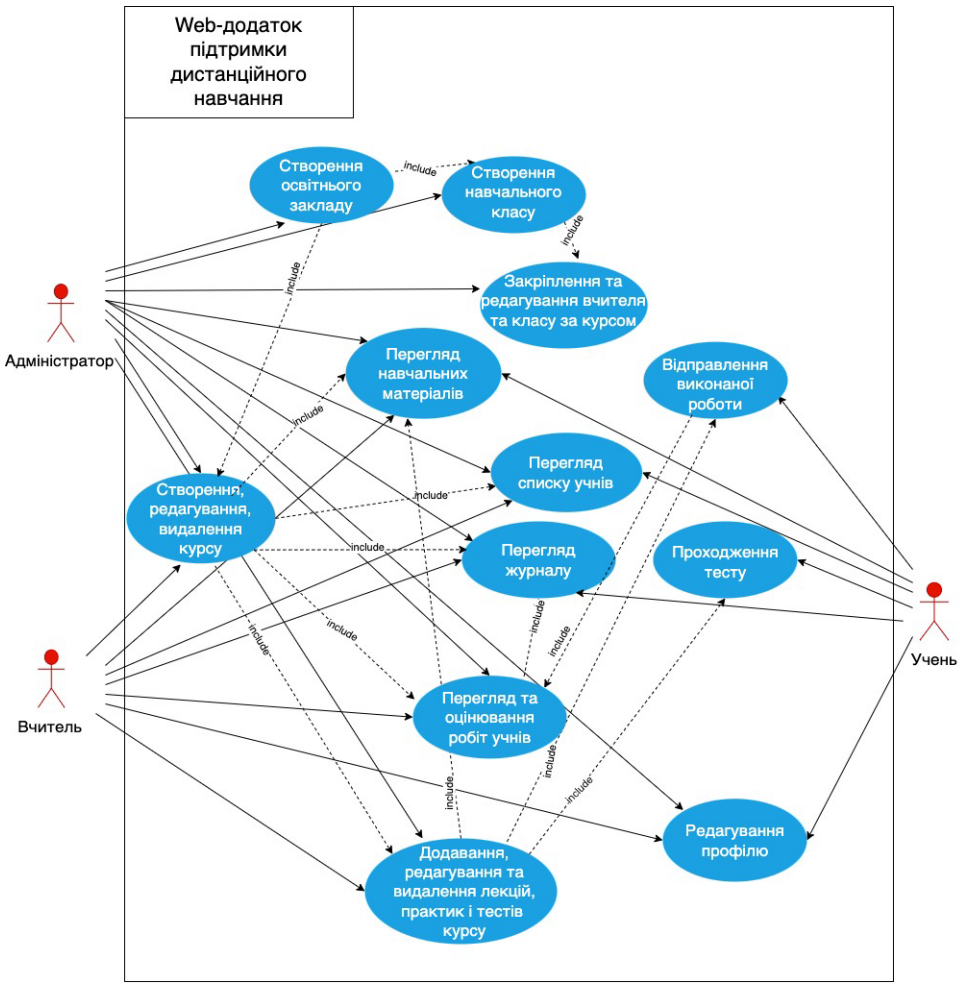


Рисунок 2.3 – Діаграма варіантів використання

Акторами виступають адміністратор закладу, викладач та учень. Спільні сценарії для всіх акторів є перегляд навчальних матеріалів, перегляд списку учнів, перегляд журналу, редагування профілю. У адміністратора та викладача є можливість взаємодії з такими ВВ: створення, редагування та видалення курсу; перегляд та оцінювання робіт учнів; додавання, редагування та видалення лекцій, практик і тестів курсу. Лише адміністратор може взаємодіяти з ВВ по створенню освітнього закладу, навчального класу і закріпленню вчителя та класу за курсом. Учень також має зв’язок з ВВ відправлення виконаних робіт і проходження тесту.

2.2.1 Діаграми послідовності

Діаграми послідовності дозволяють графічно представити проміжки часу у вигляді вертикальних ліній, в період яких одночасно задіяні певні процеси та обмін повідомленнями між ними, який зображений у вигляді горизонтальних ліній [17]. Для кожного варіанту використання було розроблено власну діаграму послідовності.

На рисунку 2.4 зображена діаграма послідовності для ВВ «Створення освітнього закладу». Під час взаємодії користувача з web-додатком, першим кроком потрібно перейти на сторінку всіх освітніх закладів, для цього до бази даних надсилається запит на отримання повної інформації про власні освітні заклади і ті, до яких користувач є під'єднаним. Після завантаження даних, користувач повинен відкрити форму для заповнення даних. Далі, після підтвердження введеної інформації, до бази надсилається запит на створення нового освітнього закладу. В результаті користувач бачить на сторінці новий щойно створений заклад.

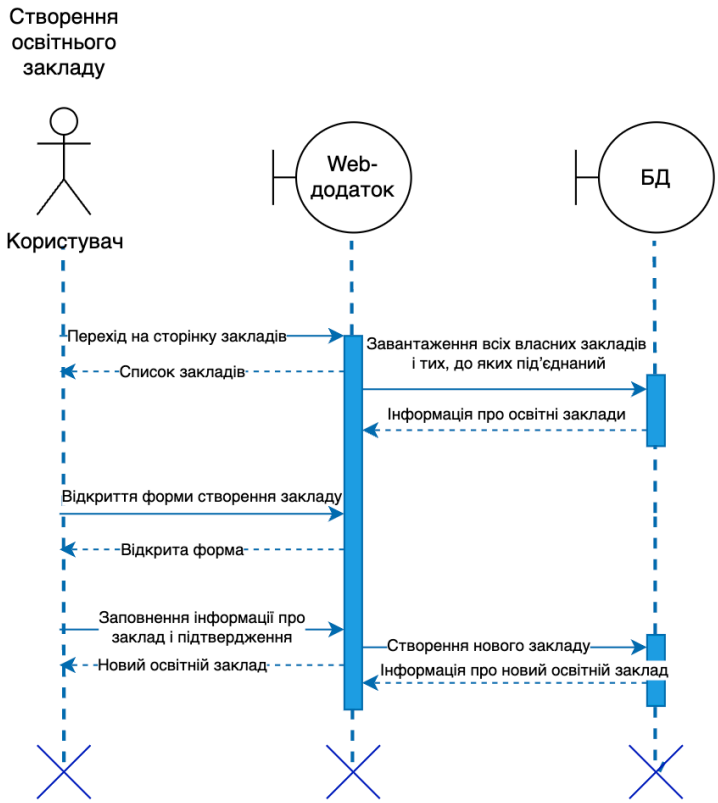


Рисунок 2.4 – Діаграма послідовності для ВВ «Створення освітнього закладу»

Для ВВ «Створення навчального класу» користувач спочатку повинен перейти на сторінку окремого закладу, для цього надсилається запит до БД на отримання інформації про всі курси закладу до яких під'єднано користувача. Після цього, адміністратор може перейти до налаштування курсу та відкрити форму створення навчального класу. Після підтвердження заповненої інформації, надсилається запит до БД і створюється новий клас. Діаграма зображена на рисунку 2.5.

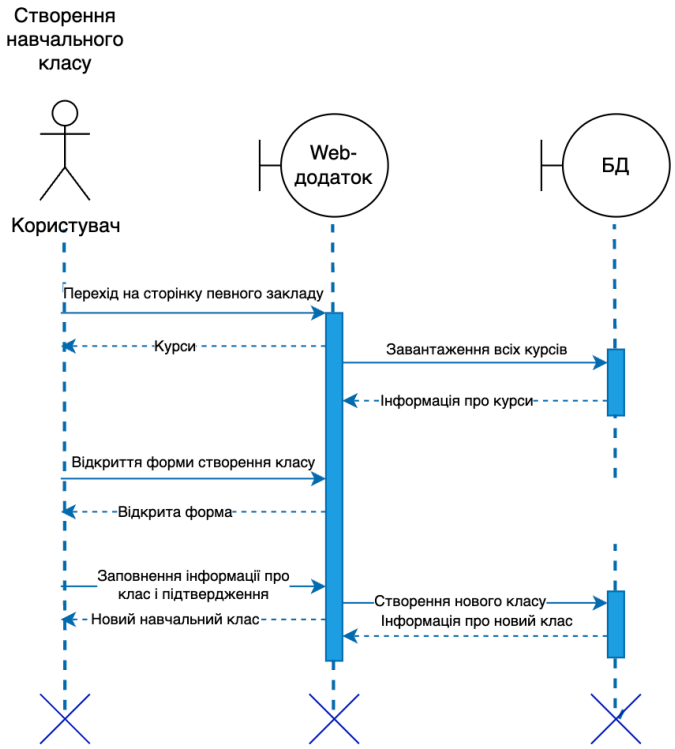


Рисунок 2.5 – Діаграма послідовності для ВВ «Створення навчального класу»

Наступна діаграма на рисунку 2.6 створена для ВВ «Створення курсу, закріплення вчителя та класу». Після переходу користувачем на певний заклад, завантажуються дані про курси. Далі адміністратор переходить до форми створення курсу та заповнює необхідні дані. Також обирається вчитель та навчальні класи. Після цього, натискаючи на кнопку підтвердження даних, надсилаються відповідні запити на створення курсу та закріплення вчителів і класів за ним. В результаті отримуємо новий курс.

Рисунок 2.6 – Діаграма послідовності для ВВ «Створення курсу, закріплення вчителя та класу»

Далі була розроблена діаграма послідовності для ВВ «Додавання матеріалів до курсу». Спочатку користувач переходить до певного курсу, після чого до БД надсилається запит на завантаження всіх матеріалів. Потім користувач може відкрити форму додавання нового матеріалу та заповнити дані. Після підтвердження введеної інформації, до БД надходить запит на створення матеріалу в залежності від його типу (лекції, практики, тести) і повертається новий створений матеріал. Діаграма представлена на рисунку 2.7.

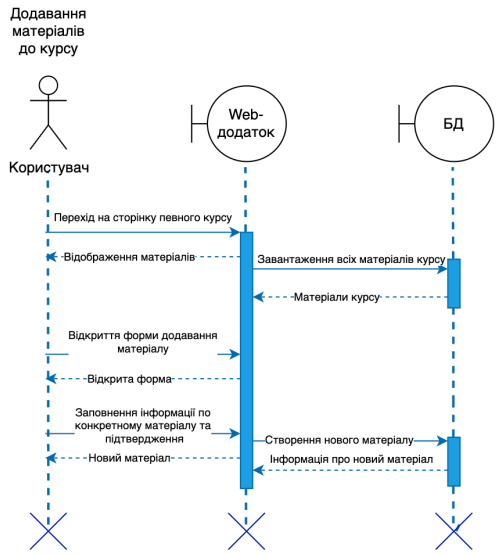


Рисунок 2.7 – Діаграма послідовності для ВВ «Додавання матеріалів до курсу»

Для ВВ «Перегляд навчальних матеріалів» діаграми на рисунку 2.8, першим кроком в діаграмі послідовності є перехід на сторінку курсу та завантаження всіх його матеріалів. Потім користувач повинен перейти до певного розділу, тобто до лекцій, практик або тестів, щоб завантажити конкретні дані. Зі списку лекцій, практик або тестів можна обрати будь-який елемент та клацнути на нього, щоб перейти на сторінку матеріалу та надіслати запит до БД на отримання контенту по даному матеріалу.

Рисунок 2.8 – Діаграма послідовності для ВВ «Перегляд навчальних матеріалів»

В діаграмі ВВ «Перегляд робіт та оцінювання учнів» описується, що першим кроком є перехід на сторінку курсу та завантаження всіх його матеріалів. Далі вчитель повинен перейти на сторінку зі списком робіт учнів, після чого БД завантажує та повертає дані про них, завдання та їх роботу. Вчитель може завантажити роботу на перевірку та виставити оцінку, надсилаючи запит до БД на додавання оцінки до роботи учня. В результаті всі перевірені роботи мають оцінку. Діаграма зображена на рисунку 2.9.

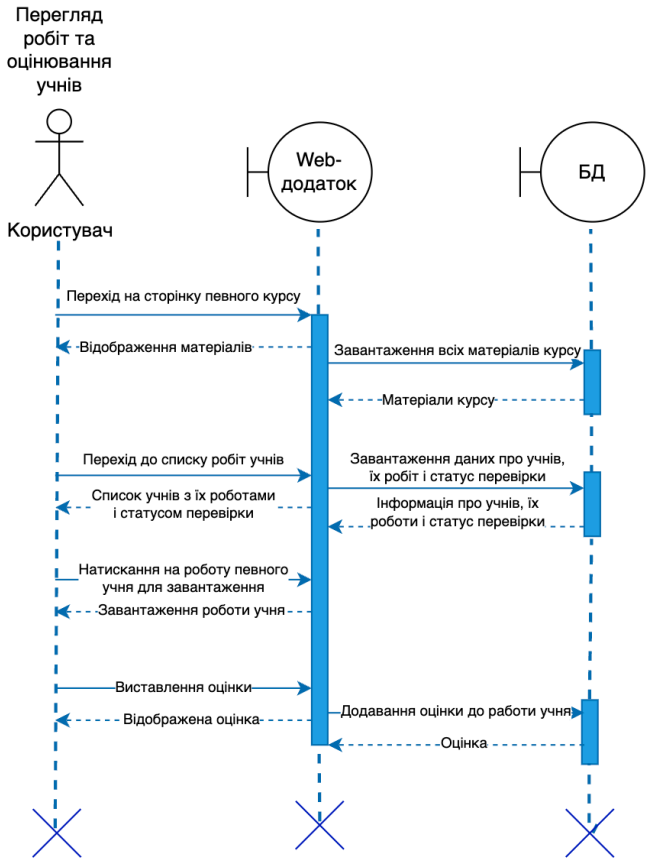


Рисунок 2.9 – Діаграма послідовності для ВВ «Перегляд робіт та оцінювання учнів»

На рисунку 2.10 представлена діаграма для ВВ «Перегляд списку учнів», де після переходу на сторінку курсу та завантаження матеріалів, користувач переходить до списку учнів і надсилається запит на отримання даних про кожного з них. У відповідь отримуємо список учнів з інформацією про кожного.

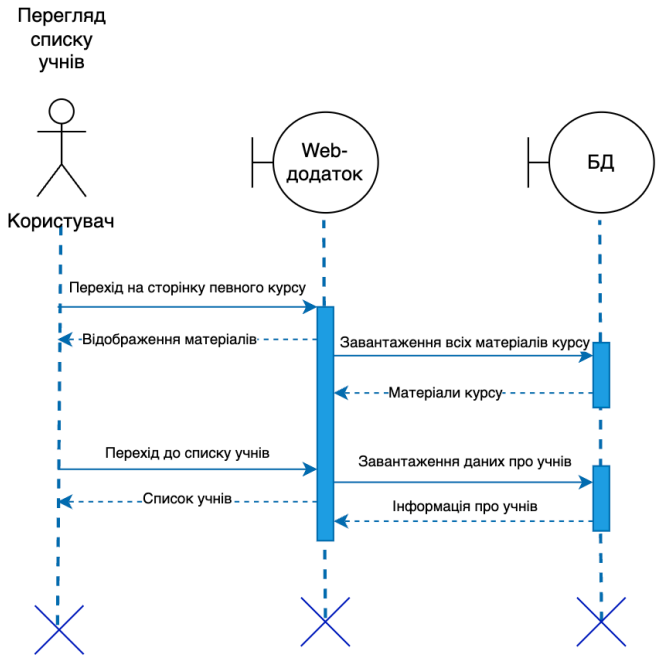


Рисунок 2.10 – Діаграма послідовності для ВВ «Перегляд списку учнів»

Для ВВ «Перегляд журналу» також спочатку необхідно перейти на потрібний курс та отримати від БД матеріали. Потім користувач може перейти до журналу з оцінками для його перегляду, де надсилається запит на завантаження завдання та оцінки і повертається відповідна інформація. Діаграма показана на рисунку 2.11.

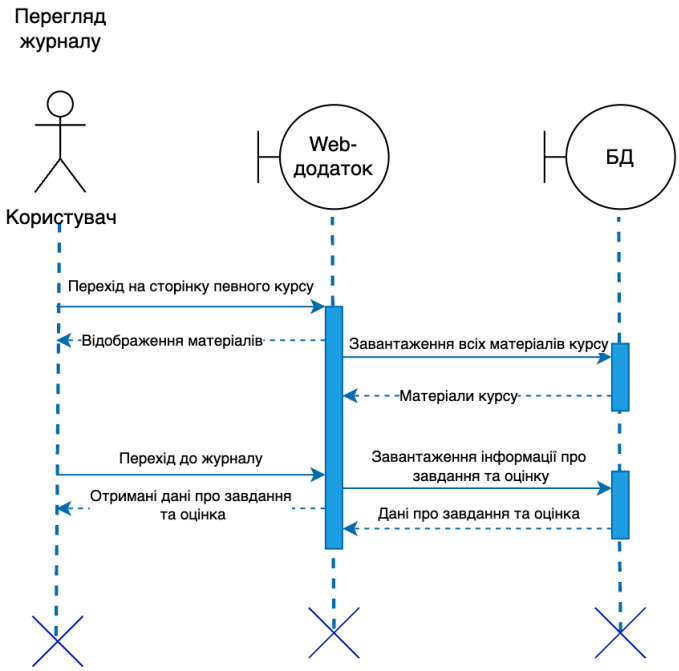


Рисунок 2.11 – Діаграма послідовності для ВВ «Перегляд журналу»

Далі, для діаграми на рисунку 2.12, яка описує ВВ «Відправлення робіт на перевірку» таким же чином, як і для попередньої діаграми в 2.11 завантажуються матеріали, після чого користувач переходить до практик і завантажується з БД їх список. Далі необхідно перейти до конкретної практики, щоб завантажити її вміст і мати змогу відправити власну роботу, потім надсилається запит на зберігання нової роботи і повертається оновлений список робіт.

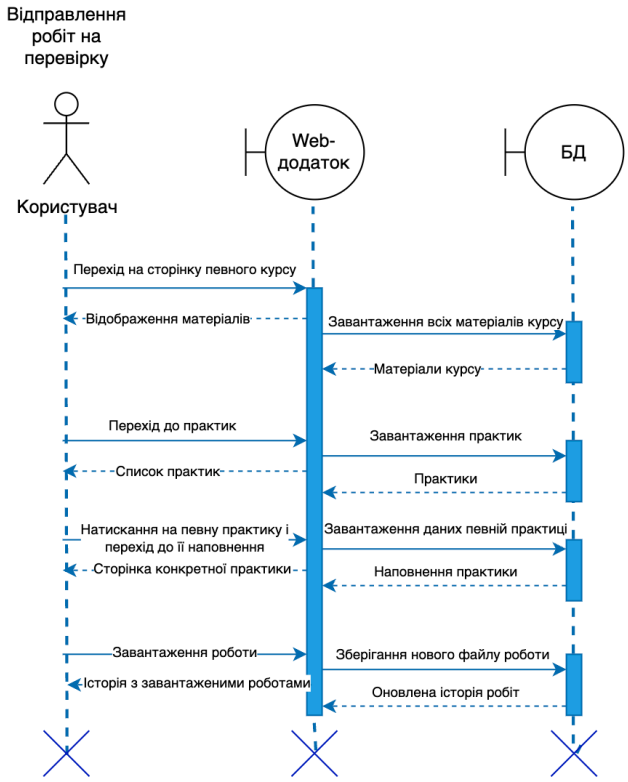


Рисунок 2.12 – Діаграма послідовності для ВВ «Відправлення робіт на перевірку»

Так само як і в попередніх діаграмах, для ВВ «Проходження тесту» після отримання матеріалів курсу переходимо до тестів і отримуємо їх список з бази даних. При переході на конкретний тест надсилається запит і завантажується його інформація та питання. Користувач на кожне питання дає відповідь, кожна з яких потім зберігається. Після підтвердження завершення тесту або при закінченні часу надсилається запит на отримання оцінки. В результаті повертається оцінка за тест. Діаграма зображена на рисунку 2.13.

Рисунок 2.13 – Діаграма послідовності для ВВ «Проходження тесту»

Для ВВ «Редагування профілю» була розроблена діаграма, яка представлена на рисунку 2.14. Після переходу на сторінку користувача, завантажується вся інформація про нього. При внесенні нових даних і підтвердження редагування відправляється запит до БД для зміни даних користувача. У відповідь повертається оновлена інформація.

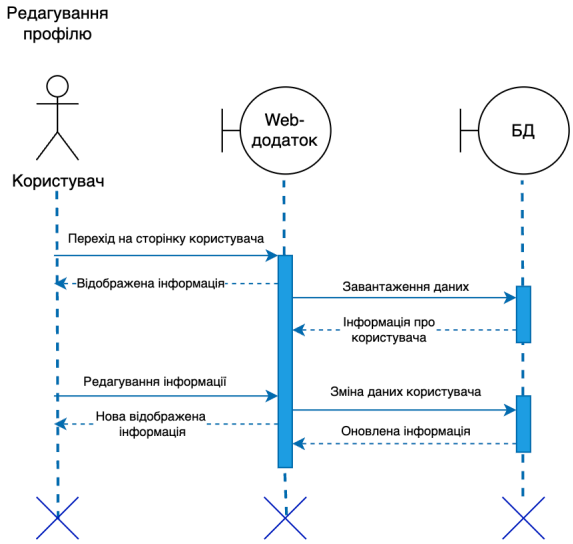


Рисунок 2.14 – Діаграма послідовності для ВВ «Редагування профілю»

2.2.2 Діаграми діяльності

Також не менш важливим етапом є створення діаграм діяльності, які надають візуальне представлення про те які дії виконуються в залежності від тієї чи іншої умови.

Для ВВ «Створення освітнього закладу» була створена власна діаграма діяльності, зображена на рисунку 2.15. Коли web-додаток відкритий, користувач переходить до сторінки закладів для їх перегляду. Далі система завантажує всі доступні користувачу заклади і в разі помилки виводить відповідне повідомлення. У разі успішного виконання, користувач створює новий заклад, після чого ці дані оброблюються і також виводиться або помилка, або успішне збереження нового закладу.

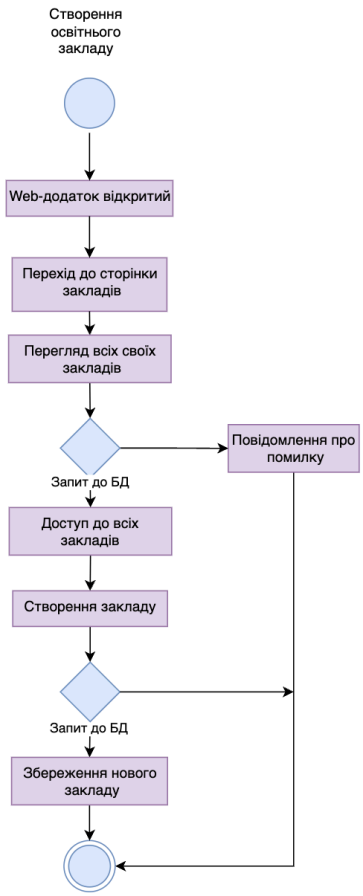


Рисунок 2.15 – Діаграма діяльності для ВВ «Створення освітнього закладу»

Наступна діаграма діяльності розроблена для ВВ «Створення навчального класу», яка представлена на рисунку 2.16. Після початку роботи web-додатку,

користувач переходить до певного закладу, там він може переглянути список всіх своїх курсів закладу, для цього надсилається запит до БД. Якщо виникає помилка, користувач отримує відповідне повідомлення, якщо все правильно, то користувач отримує доступ до всіх курсів і до можливості створення навчального класу. При успішному створенні класу отримуємо та зберігаємо новий, інакше виводиться повідомлення про помилку.

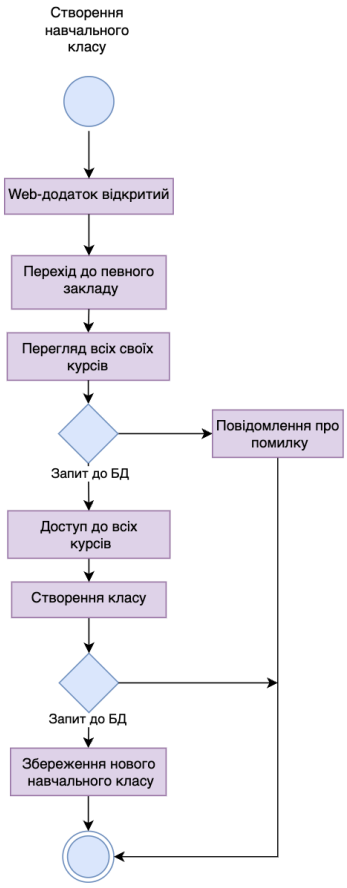


Рисунок 2.16 – Діаграма діяльності для ВВ «Створення навчального класу»

Далі, на рисунку 2.17 показана діаграма для ВВ «Створення курсу, закріплення вчителя та класу». Після відкриття додатку та переходу до певного закладу надсилаємо запит до БД на перегляд всіх курсів. У разі успіху отримуємо доступ до всіх курсів закладу, інакше повідомлення про помилку. Наступним кроком створюємо курс. Якщо всі дані були коректними і не виникла помилка, то отримуємо новий курс, інакше повідомлення з помилкою. Такі ж самі кроки

виконуються для обирання вчителів і навчальних класів. В результаті завершення всіх процесів маємо курс і закріплених за ним вчителів і класи.

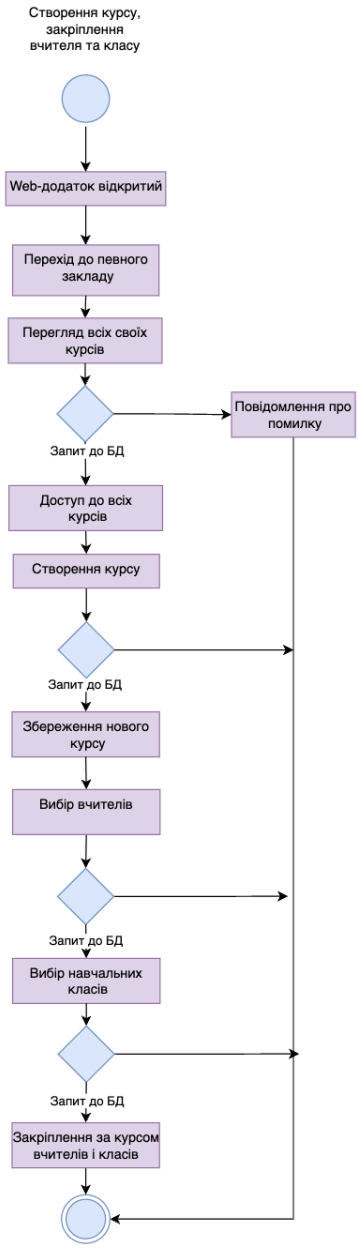


Рисунок 2.17 – Діаграма діяльності для ВВ «Створення курсу, закріплення вчителя та класу»

Діаграма діяльності для ВВ «Додавання матеріалів до курсу» зображена на рисунку 2.18. При відкритому додатку та переході на певний курс закладу, надсилаємо запит на перегляд його матеріалів. У випадку виникнення помилки з’являється відповідне повідомлення, в іншому випадку отримуємо доступ до

матеріалів курсу. Далі користувач створює новий певний матеріал, де у разі помилки також показується повідомлення, а у разі успіху отримуємо новий збережений матеріал.

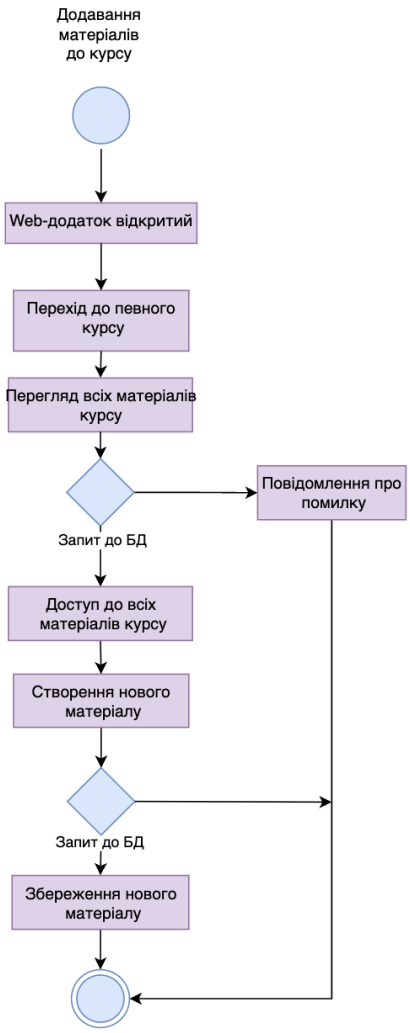


Рисунок 2.18 – Діаграма діяльності для ВВ «Додавання матеріалів до курсу»

Для діаграми діяльності ВВ «Перегляд навчальних матеріалів» всі процеси до отримання доступу матеріалів курсу є однаковими з тими, що зображені на рисунку 2.18. Після отримання доступу, користувач повинен перейти до конкретного розділу (лекції, практики, тести), де в успішному випадку з’явиться сторінка зі списком конкретних матеріалів, на кожну з яких можна перейти і переглянути зміст. У випадку помилки відображається відповідне повідомлення. Діаграма представлена на рисунку 2.19.

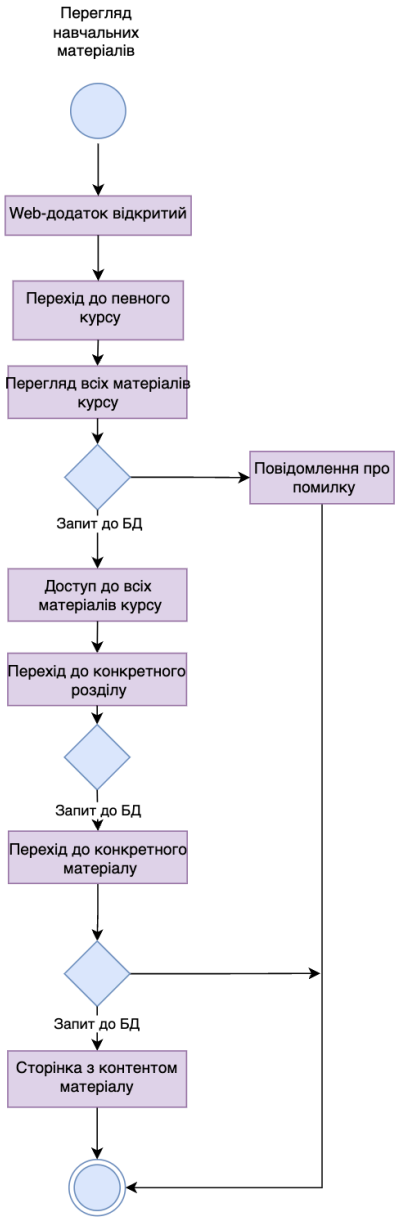


Рисунок 2.19 – Діаграма діяльності для ВВ «Перегляд навчальних матеріалів»

Наступна діаграма діяльності створена для ВВ «Перегляд робіт та оцінювання учнів». Початок процесу та деяка їх послідовність співпадає з діаграмою на рисунку 2.19 до моменту отримання матеріалів курсу. Потім користувач переходить до списку робіт учнів, де в результаті запиту отримується або доступ до робіт, або повідомлення з помилкою. Далі користувач виставляє оцінку, де також перевіряється успішність операції. Якщо все правильно, отримуємо виставлену оцінку, інакше повідомлення про помилку. Діаграму показано на рисунку 2.20.

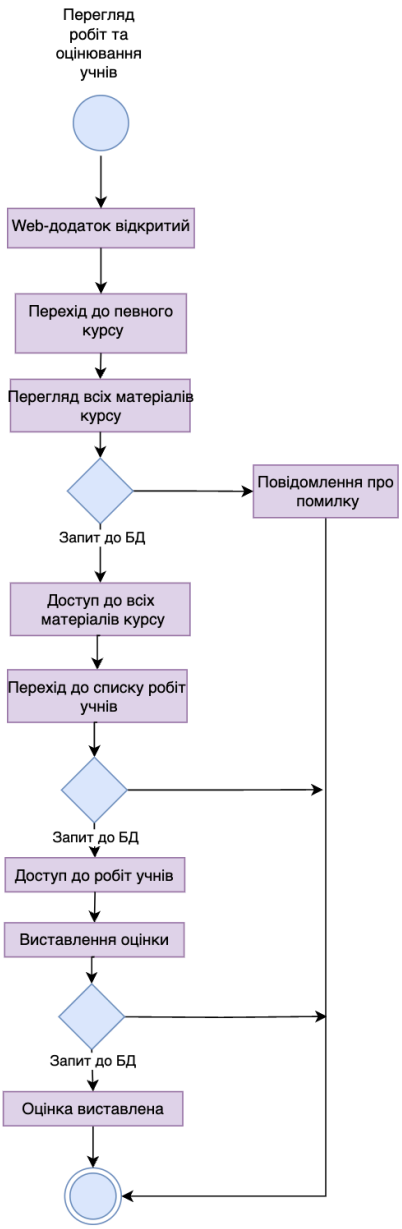


Рисунок 2.20 – Діаграма діяльності для ВВ «Перегляд робіт та оцінювання учнів»

Для ВВ «Перегляд списку учнів» була розроблена діаграма діяльності та представлена на рисунку 2.21. Процеси до отримання матеріалів повторюються з попередньою діаграмою. Після цього користувач переходить до списку учнів і у разі успішної обробки даних повертається інформація про учасників курсу, а інакше показується повідомлення з помилкою.

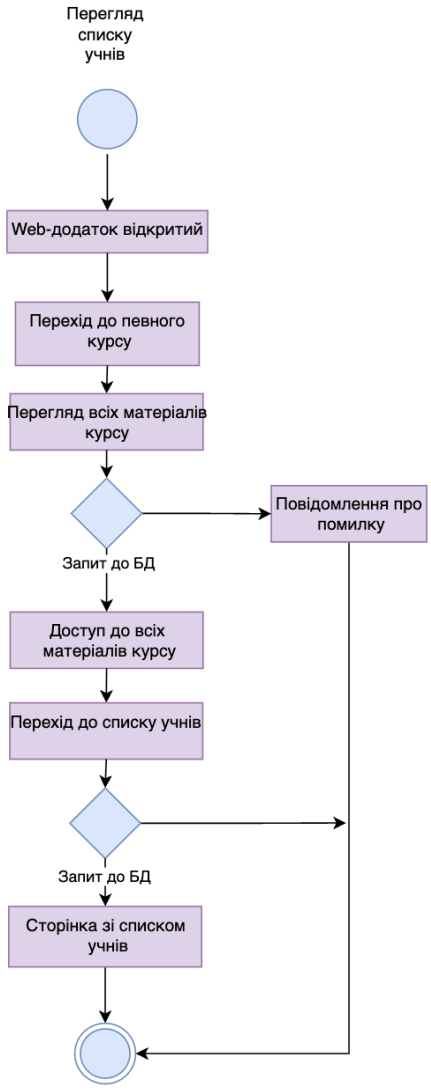


Рисунок 2.21 – Діаграма діяльності для ВВ «Перегляд списку учнів»

Далі, для ВВ «Перегляд журналу» була також розроблена діаграма діяльності та представлена на рисунку 2.22. Процеси до моменту отримання матеріалів також є в описі попередніх діаграм. Наступним кроком користувач переходить до журналу і в успішному випадку повертається інформація про завдання та оцінки учня, інакше показується повідомлення про помилку.

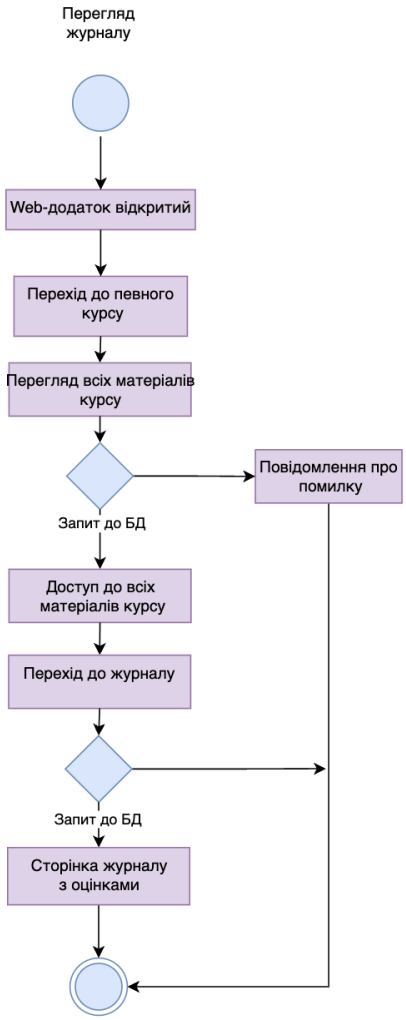


Рисунок 2.22 – Діаграма діяльності для ВВ «Перегляд журналу»

Також була створена діаграма діяльності для ВВ «Відправлення робіт на перевірку», яка представлена на рисунку 2.23. Як і в попередніх діаграмах, після доступу до матеріалів курсу користувач переходить саме до практик. Після виконання запиту, в успішному випадку отримуємо список всіх практик, інакше повідомлення з помилкою. Далі потрібно перейти до однієї з практик, щоб отримати її внутрішню інформацію, також проводиться перевірка на успішність операції. Потім користувач відправляє на перевірку свою роботу, яка зберігається в базі даних при успішному запиті, інакше знову з’являється повідомлення з помилкою. В результаті на сторінці бачимо оновлену історію робіт.

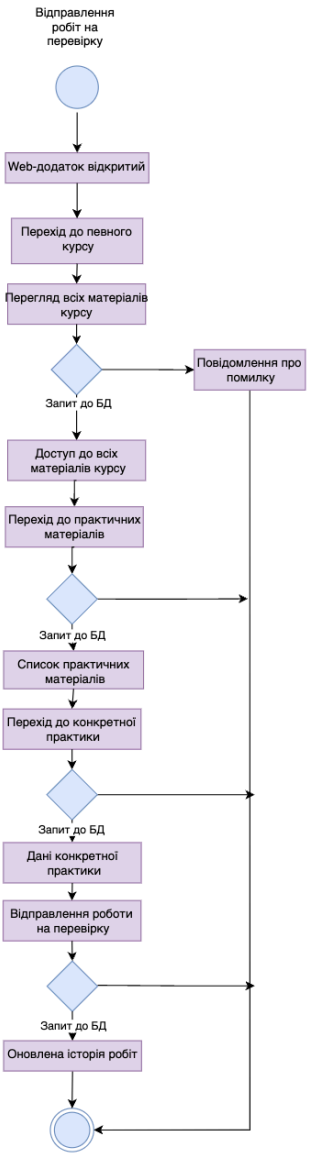


Рисунок 2.23 – Діаграма діяльності для ВВ «Відправлення робіт на перевірку»

Для ВВ «Проходження тесту» діаграма діяльності зображена на рисунку 2.24. Так само як і з попередніми діаграмами, після отримання доступу до матеріалів курсу необхідно перейти до тестів, де після успішного запиту показується їх список. Далі необхідно перейти до конкретного тесту, щоб почати обирати відповіді на питання. При помилках завантаження інформації про тест або обиранні відповіді відображається віконце з цією помилкою. Потім відповіді зберігаються і після завершення тесту або вичерпаного часу надсилається запит на отримання оцінки. У разу успіху отримуємо оцінку, інакше повідомлення з помилкою.

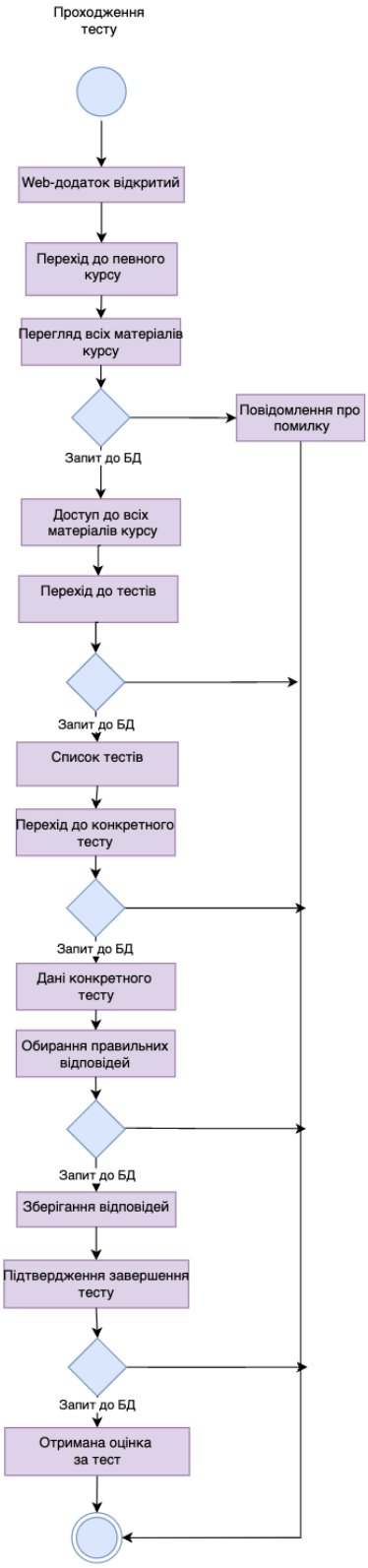


Рисунок 2.24 – Діаграма діяльності для ВВ «Проходження тесту»

Остання діаграма діяльності присвячена ВВ «Редагування профілю», саму діаграму представлено на рисунку 2.25. Після відкриття web-додатку, користувач переходить на особисту сторінку, де завантажується вся інформація

про нього. Якщо під час завантаження даних виникає помилка, то з’являється відповідне вікно. Далі користувач заповнює форму для зміни власних даних та надсилає новий запит після підтвердження інформації. У кращому випадку, дані оновлюються, у гіршому виникає повідомлення з помилкою.

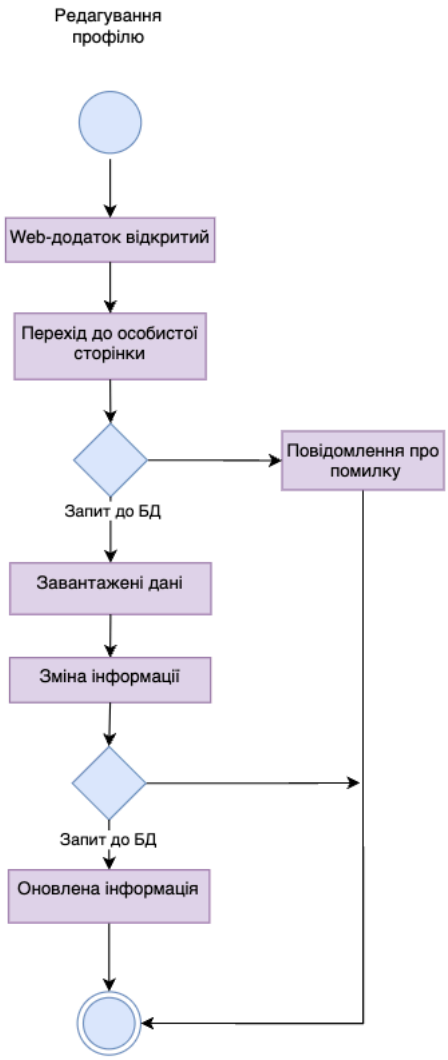


Рисунок 2.25 – Діаграма діяльності для ВВ «Редагування профілю»

2.2.3 Інтерфейси

З метою візуального представлення роботи веб-додатку, були розроблені інтерфейси для кожної функціональної вимоги.

Для створення освітнього закладу, при натисненні на відповідну кнопку, відкривається модальне вікно, в якому потрібно обов’язково ввести дані про його назву та опис, за бажанням можна додати зображення. При натисканні кнопки «Submit» повинна відбуватися валідація введеної інформації. Після натискання модальне вікно закривається. Якщо трапляється помилка, то з’являється відповідне повідомлення. Інтерфейс для ВВ «Створення освітнього закладу» зображено на рисунку 2.26.

Рисунок 2.26 – Інтерфейс для ВВ «Створення освітнього закладу»

При створенні навчального класу для початку потрібно відкрити його форму і ввести назву нового класу. Далі необхідно зі списку користувачів знайти кожного учня. Пошук повинен здійснюватися гортанням списку, введенням ім’я та прізвища або пошти. Після підтвердження даних по кнопці «Submit» відбувається валідація даних і в успішному випадку поля форми очищуються, інакше показується помилка. Форму представлено на рисунку 2.27.

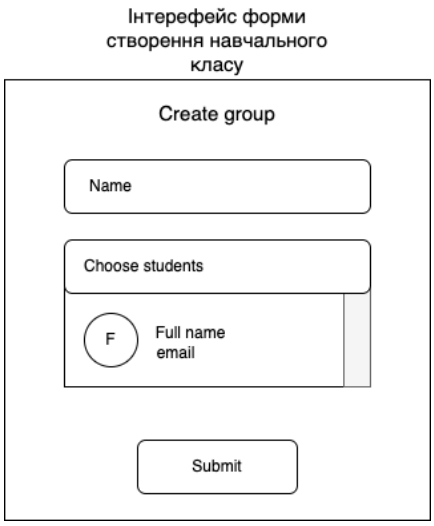


Рисунок 2.27 – Інтерфейс для ВВ «Створення навчального класу»

Для створення курсу відбуваються схожі дії, як і для створення класу. Спочатку заповнюємо дані про назву, опис і можемо додати зображення, валідація здійснюється після підтвердження даних. Потім обираємо зі списків вчителів і групи. Після натискання кнопки «Submit», якщо все правильно, вона очищується, якщо виникне помилка, то з’явиться повідомлення. Форма зображена на рисунку 2.28.

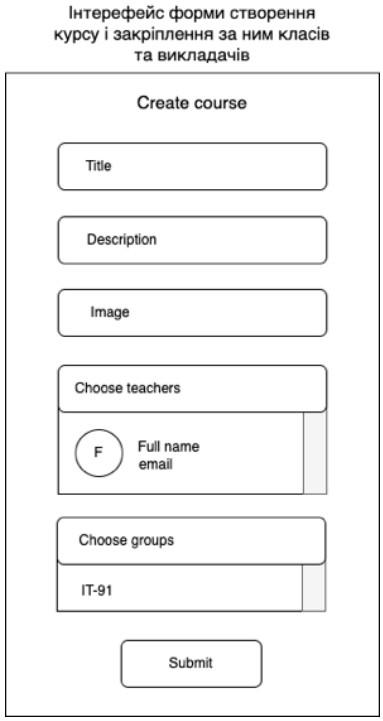


Рисунок 2.28 – Інтерфейс для ВВ «Створення курсу і закріплення за ним класів та викладачів»

Для ВВ «Додавання матеріалів до курсу – лекції» створений інтерфейс у вигляді модального вікна, де заповнюються дані про назву та номер лекції, основний контент. При натисканні на кнопку «Save» інформація зберігається та закривається модальне вікно. У разі помилки з’являється повідомлення. Інтерфейс зображений на рисунку 2.29.

Рисунок 2.29 – Інтерфейс для ВВ «Додавання матеріалів до курсу – лекції»

Таким же чином як і для додавання лекцій, створений інтерфейс для додавання практик. Єдиною відмінністю є вказування терміну здачі роботи. Інтерфейс представлено на рисунку 2.30.

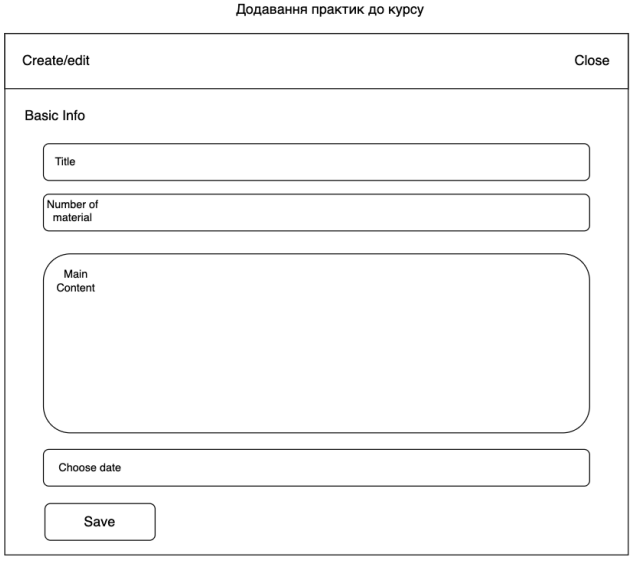
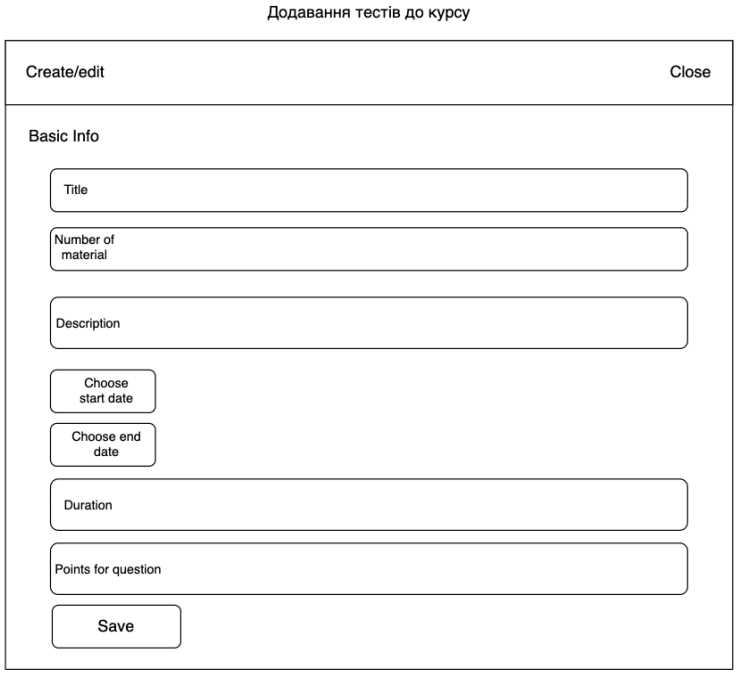


Рисунок 2.30 – Інтерфейс для ВВ «Додавання матеріалів до курсу – практики»

Подібним до попередніх інтерфейсів є також ескіз ВВ «Додавання матеріалів до курсу – тести». Крім назви, номеру та опису додаються поля для обирання дати початку та завершення тесту, його тривалість та кількість балів за одне питання. Інтерфейс модульного вікна показаний на рисунку 2.31. Також окремою є форма створення питань, де можна вписати сам текст питання та додати будь-яку кількість відповідей за допомогою кнопки «Add option». Серед всіх варіантів відповідей можна правильною позначити лише одну. Після створеного блоку питання та натиснення «Save», дані зберігаються та показується повідомлення з успішно виконаною операцією або помилкою. Форма створення питань зображена на рисунку 2.32.



Рисунки 2.31-2.32 – Інтерфейси для ВВ «Додавання матеріалів до курсу – тести»

Інтерфейс по перегляду робіт та оцінювання учнів представлений на рисунку 2.33 у вигляді сторінки, на якій відображається список робіт учнів, де в кожному елементі вказується інформація про учня, завдання, файл роботи та виставлена оцінка. Якщо робота ще не була оцінена, то замість оцінки є поле для її введення та кнопка для підтвердження операції. При натисненні на файл, він автоматично завантажується. Оцінені роботи також підсвічуються блакитним кольором.

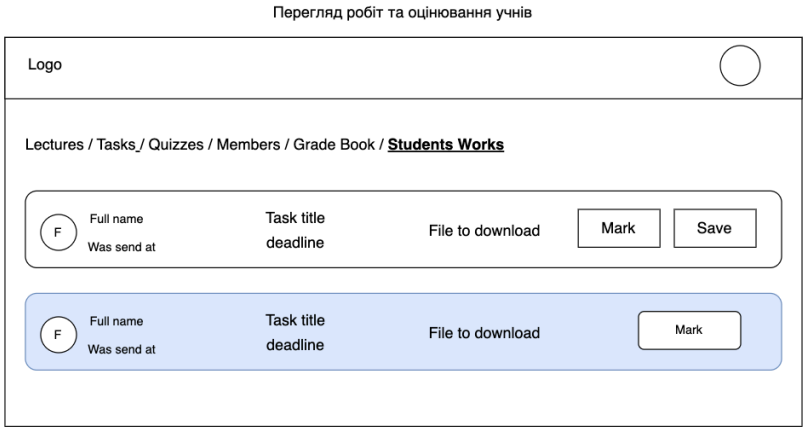


Рисунок 2.33 – Інтерфейс для ВВ «Перегляд робіт та оцінювання учнів»

На рисунку 2.34 зображено інтерфейс для ВВ «Перегляд навчальних матеріалів». Переходячи на певну вкладку матеріалів, відкривається список лекцій, практик або тестів. Список навчальних матеріалів можна сортувати за датою створення і нумерацією в порядку зростання та спадання.

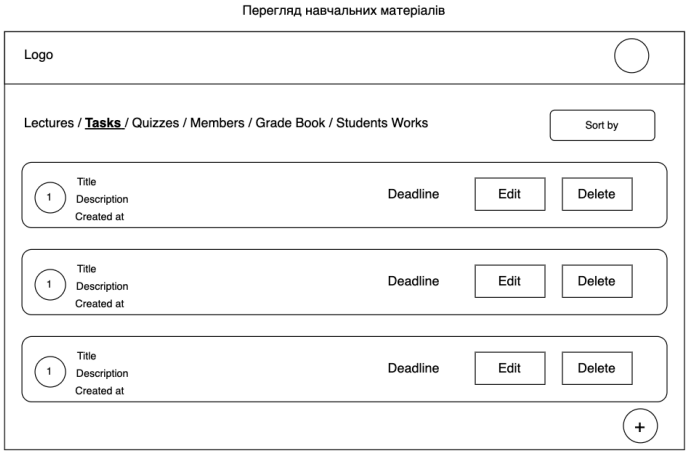


Рисунок 2.34 – Інтерфейс для ВВ «Перегляд навчальних матеріалів» на прикладі практик

Для перегляду списку учнів курсу необхідно перейти до відповідної вкладки у курсі, де відобразиться список карточок з інформацією про кожного з них. Інтерфейс для ВВ «Перегляд списку учнів» зображено на рисунку 2.35

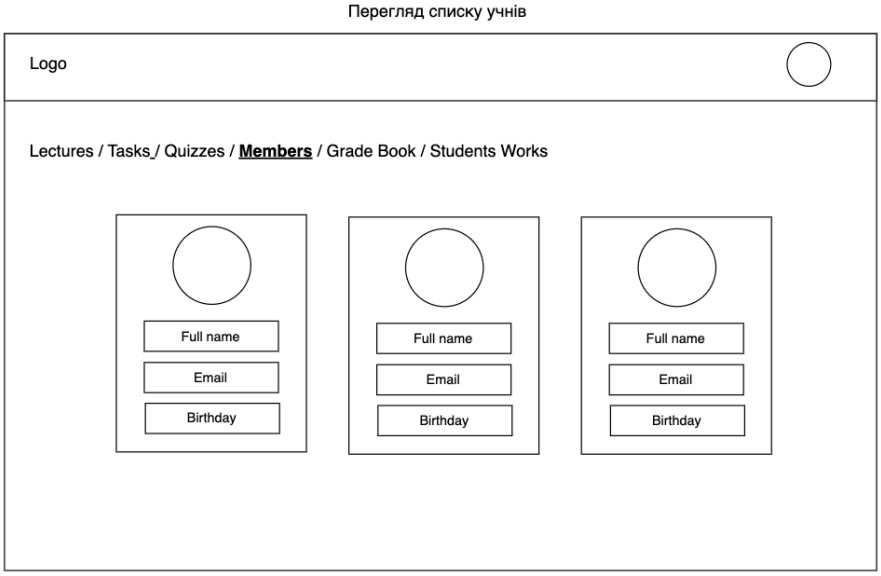


Рисунок 2.35 – Інтерфейс для ВВ «Перегляд списку учнів»

На рисунку 2.36 показаний інтерфейс для ВВ «Редагування профілю». На сторінці профіля є карточка з повною інформацією про користувача, а також карточка для зміни цих даних. Після внесення змін, вони перевіряються на правильність. В результаті отримуємо оновлену інформацію або повідомлення про помилку.

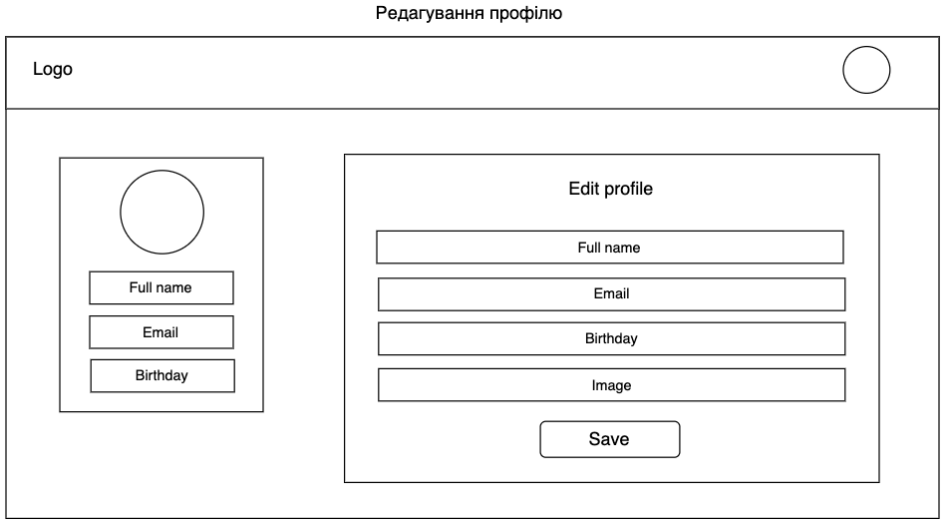


Рисунок 2.36 – Інтерфейс для ВВ «Редагування профілю»

Для ВВ «Проходження тесту» було створено інтерфейс, представлений на рисунку 2.37. В даному інтерфейсі присутній список блоків з питанням, в яких є сам текст питання та варіанти відповідей, серед яких можна обрати лише одну. Також справа розташована панель, в якій є перелік всіх питань, відлік часу та кнопка завершення тесту.

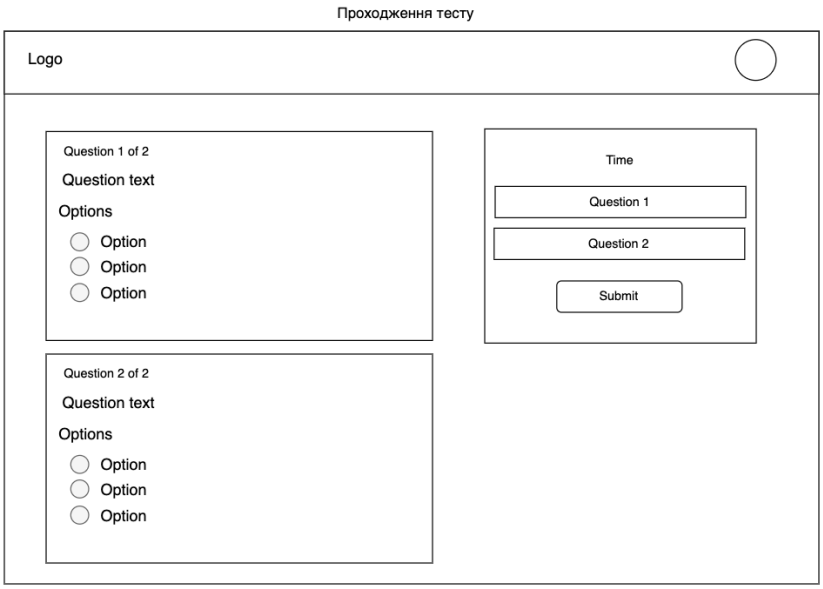


Рисунок 2.37 – Інтерфейс для ВВ «Проходження тесту»

Для ВВ «Відправлення робіт на перевірку» створено інтерфейс, що показаний на рисунку 2.38. Він містить сторінку зі змістом практики, поле для завантаженні роботи будь-якого типу, кнопку підтвердження та історію завантажень. Після підтвердження завантаження робіт, історія оновлюється та відображається від нових до старих версій.

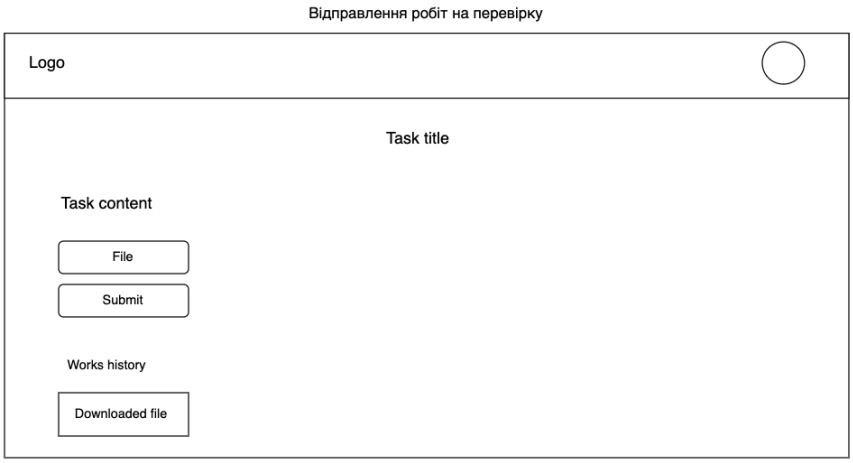


Рисунок 2.38 – Інтерфейс для ВВ «Відправлення робіт на перевірку»

Інтерфейс перегляду журналу наведено на рисунку 2.39. Користувач може переглянути оцінки за всі свої роботи, обираючи певний вид завдання. Якщо обрати тести, то відкривається список робіт з відповідними оцінками за пройдені тести, а роботи практичних завдань закриваються.

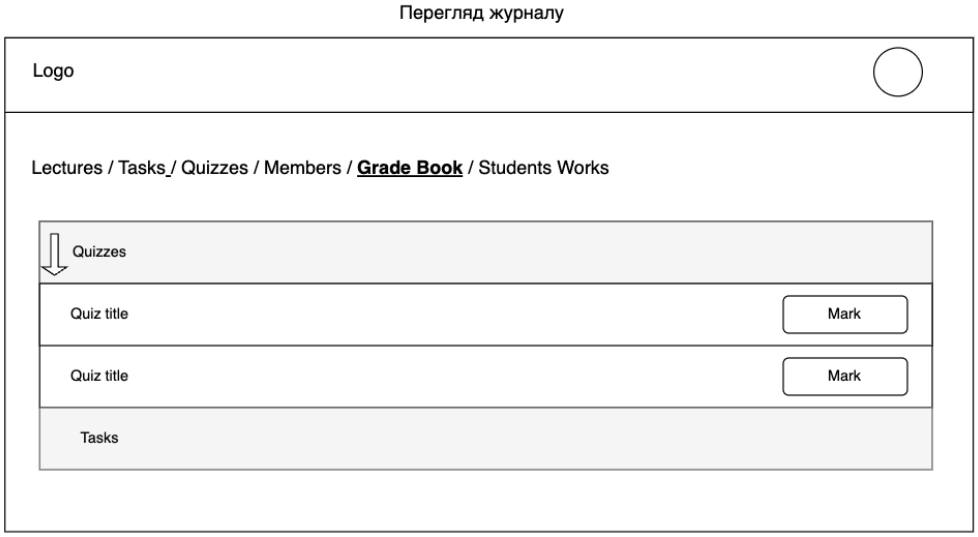


Рисунок 2.39 – Інтерфейс для ВВ «Перегляд журналу»

2.3Проєктування моделі бази даних

Визначення правильної структури для зберігання даних в системі потребує важливого етапу розробки проєкту – проведення проєктування бази даних. В результаті проєктування отримується модель з логічно пов’язаними між собою сутностями, в яких зазначаються всі необхідні атрибути для формування таблиці.

Крім загальної інформації в моделі про наведені дані в таблицях, також можна вказувати тип цих даних та їх обмеження, наприклад як: унікальність, можливість бути не визначеними (перевірка на можливість бути Null), визначення зберігання кількості символів для типу даних «Varchar» та інші.

Головним фактором для розуміння взаємодії сутностей моделі є їх зв’язки, які можуть набувати таких типів:

One-to-One – кожен запис таблиці відповідає одному запису з іншої;

One-to-Many – кожен запис таблиці відповідає багатьом записам з іншої;

Many-to-Many – кожен запис таблиці може відповідати багатьом записам з іншої [18].

Отже, під час проєктування була створена фізична модель бази даних, представлена на рисунку 2.40, яка відображає взаємозв'язки між сутностями.

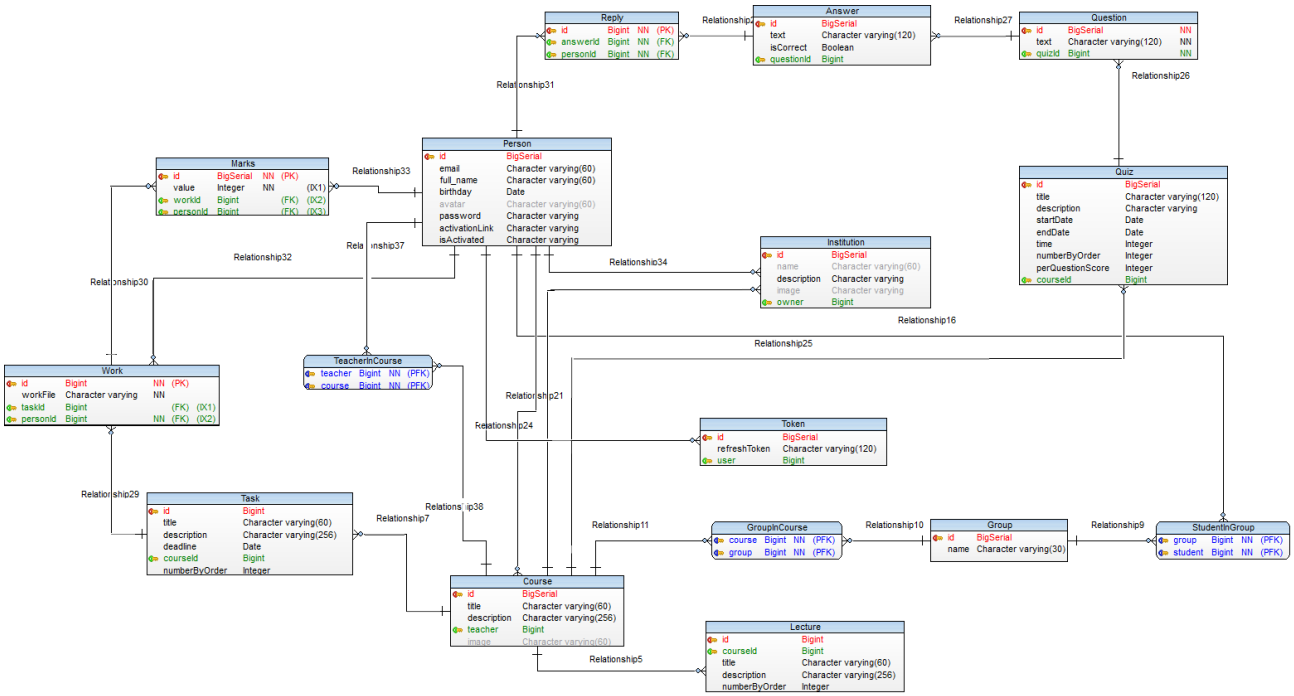


Рисунок 2.40 – Фізична модель бази даних

До переліку сутностей даної моделі входять наступні: користувач системи (Person), освітній заклад (Institution), курс (Course), навчальний клас (Group), студенти в класі (StudentInGroup), класи в курсі (GroupInCourse), вчителі в курсі (TeacherInCourse), практики (Task), практичне завдання (Work), оцінка (Mark), лекції (Lecture), тести (Quiz), питання (Question), варіант відповіді на тест (Answer), можлива відповідь учня (Reply), токени (Token).

Зв'язки між сутностями:

– сутності «Person» та «Token»: Користувач може мати декілька токенів, а токен повинен відноситись до одного користувача;

– сутності «Person» та «Institution»: Користувач може мати декілька освітніх закладів, а заклад повинен відноситись до одного користувача;

– сутності «Person» та «Group»: Користувач може належати до декількох навчальних класів, а навчальний клас може відноситись до декількох користувачів;

– сутності «Institution» та «Course»: Освітній заклад може мати декілька курсів, а курс повинен відноситись до одного закладу;

– сутності «Course» та «Lecture»: Курс може мати декілька лекцій, а лекція повинна відноситись до одного курсу;

– сутності «Course» та «Task»: Курс може мати декілька практик, а практика повинна відноситись до одного курсу;

– сутності «Course» та «Quiz»: Курс може мати декілька тестів, а тест повинен відноситись до одного курсу;

– сутності «Task» та «Work»: Практика може мати декілька практичних завдань, а практичне завдання повинно відноситись до однієї практики;

– сутності «Person» та «Work»: Користувач може мати декілька практичних завдань, а практичне завдання повинно відноситись до одного користувача;

– сутності «Person» та «Mark»: Користувач може мати декілька оцінок, а оцінка повинна відноситись до одного користувача;

– сутності «Answer» та «Reply»: Варіант відповіді на тест може мати декілька можливих відповідей учня на тест, а можлива відповідь повинна відноситись до одного варіанту відповіді на тест;

– сутності «Person» та «Reply»: Користувач може мати декілька можливих відповідей учня на тест, а можлива відповідь повинна відноситись до одного користувача;

– сутності «Quiz» та «Question»: Тест може мати декілька питань, а питання повинно відноситись до одного тесту;

– сутності «Question» та «Answer»: Питання може мати декілька варіантів відповідей, а варіант відповіді повинен відноситись до одного питання;

– сутності «Group» та «Course»: Навчальний клас може належати до декількох курсів, а курс може відноситись до декількох навчальних класів;

– сутності «Person» та «Course»: Користувач може належати до декількох курсів, а курс може відноситись до декількох користувачів.

3 РОЗРОБКА WEB-ДОДАТКУ ПІДТРИМКИ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1 Архітектура програмного додатку

Для розроблюваного додатку було обрано стандартну архітектуру у вигляді «Клієнт-сервер». Дана модель розподіляє завдання між клієнтом та сервером, які можуть бути як в одній системі, так і спілкуватись через мережу. Розподіл обов'язків допомагає уникнути перенавантаження системи та робить її більш ефективною [19].

Взаємодія користувача з додатком виконується через наданий клієнтом інтерфейс програмного забезпечення. Під час цього процесу, клієнт може звертатися до сервера для надання, обробки або збереження необхідних користувачеві даних. Для підтримки комунікації між клієнтом і сервером було створено набір правил та форматів для визначення способу обміну даних. Серед найпоширеніших протоколів можна виділити HTTP, TCP/IP, WebSocket.

Схема архітектури web-додатку підтримки дистанційного навчання представлена на рисунку 3.1.

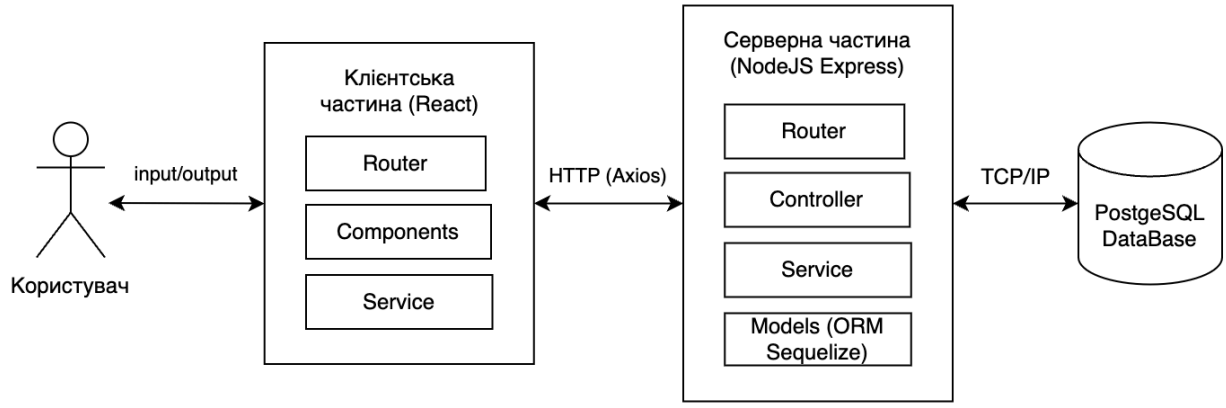


Рисунок 3.1 – Архітектура web-додатку підтримки дистанційного навчання

Клієнтська частина була розроблена за допомогою бібліотеки React для підтримки структури односторінкового web-додатку, тобто завантаження його

вмісту відбувається лише один раз при відкритті застосунку, а подальші зміни не викликають перезавантаження всього додатку. Клієнтська частина складається з таких елементів:

– роутер, який здійснює навігацію по додатку;

– компоненти для відображення контенту та можливості взаємодії з ним;

– сервіси для зв’язку з сервером.

Для надсилання HTTP-запитів до серверної частини використовується JavaScript бібліотека – Axios. Дана технологія підтримує всі основні функції управління даними CRUD (create, read, update, delete) та дозволяє перехоплювати запити і відповіді [20].

Серверна частина додатку реалізована за допомогою NodeJS з використанням Express фреймворку, який надає можливість визначати маршрутизацію застосунку та надавати відповідні запити на кожен з маршрутів (URL), а також дозволяє оброблювати запити та надавати відповіді. Складовими серверу є такі елементи:

– роути для маршрутизації запитів;

– контроллери для обробки запитів і відповідей;

– сервіси для створення запитів до бази даних;

– моделі, які створені за допомогою ORM-бібліотеки (Object-Relational Mapping) «Sequelize», що відображають таблиці бази даних.

Створення та надсилання запитів до БД відбувається за допомогою Sequelize з протоколом TCP/IP. Обраною базою даних є PostgreSQL.

3.2 Програмна реалізація

Першим кроком для розробки web-додатку є створення та налаштування серверної частини за допомогою NodeJS. Після встановлення всіх залежностей та виконання загальних налаштувань, потрібно визначити REST API, тобто які саме дані повинні надаватися клієнту. Для простішого процесу створення серверу було обрано фреймворк Express, завдяки якому можна легко визначати маршрути для обробки HTTP-методів, створювати Middleware, тобто функції які

виконуються до чи після обробки запиту для оброблення сесій, помилок, автентифікації та іншого [21]. Додатковими інструментами під час розробки стали наступні: bcrypt – бібліотека для хешування паролів; cors – бібліотека для налаштування CORS; pg – пакет для можливості встановлення зв'язку з базою даних PostgreSQL; jsonwebtoken – бібліотека для створення чи перевірки JWT токенів.

Відштовхуючись від визначених функціональних вимог до системи і проведеного моделювання, були створені відповідні ендпоінти. У роутах визначаються маршрути, які будуть оброблюватись за допомогою контроллерів. Контроллери в свою чергу отримують дані, що до них поступають при маршрутизації та оброблюють їх, використовуючи сервіси. В результаті обробки повертається відповідь. Сервіси потрібні для виконання більш низькорівневих задач, таких як надсилання запитів до бази даних.

Візьмемо до розгляду приклад створення ендпоінтів для авторизації. Для підтримання сесії аутентифікованого користувача, йому необхідно мати два токени: «access» та «refresh». Перший має досить швидкий термін життя, приблизно до 15 хвилин для збереження безпеки користувача. Після закінчення дії «access» токену, для його оновлення використовується «refresh» токен, який має довгий термін життя та в нашому випадку зберігається в базі даних для підвищення рівня безпеки. За допомогою «refresh» токену перевіряється валідність даних користувача і в успішному випадку надається новий «access» токен без необхідності повторно авторизуватись в системі.

Отже, першим етапом було визначення основних маршрутів для реєстрації, логіну, виходу, оновлення «refresh» токену та активації аккаунту. Для кожного з них, за допомогою об'єкту Router з фреймворку Express було визначено маршрут з відповідним типом запиту (get, post, update, delete) та кожному надано функцію-обробник з класу контроллера для авторизації.

Наступним кроком розглянемо сам контроллер авторизації. Кожен метод цього класу на вхід приймає дані під час маршрутизації та певним чином оброблює їх. Логіка пов'язана з взаємодією з БД виноситься до класу сервісу та

відповідні методи викликаються в контроллері. В результаті повертається відповідь на запит. Для прикладу розглянемо як влаштований метод реєстрації в контроллері. На вхід у нас подаються такі параметри:

– request – об’єкт, що приходить від клієнта, який може зберігатися в тілі запиту, якщо це звичайні поля (request.body), в параметрах (request.params), наприклад як ID користувача та у випадку файлів використовується request.files;

– response – об’єкт, який використовується для надання відповіді клієнту. У нашому випадку майже кожен метод контроллера повертає JSON об’єкт, тому частіше всього у response викликаємо метод response.json(). Також використовуються такі методи як перенаправлення на сторінку – response.redirect() і очищення кукі – response. clearCookie();

– next – функція для передачі управління наступному Middleware.

Спочатку потрібно перевірити наявність помилок у наданих даних від клієнта. Якщо вони виникли, то за допомогою next передаємо їх до потрібного Middleware на обробку. Якщо помилки відсутні, то викликаємо метод реєстрації сервісу, куди передаємо тіло запиту – request.body та зображення користувача – request.files.avatar. Результат виконання зберігаємо та викликаємо функцію створення «refresh» токену. Дана функція до нашої відповіді додає токен. Кінцеву версію повертаємо у вигляді JSON об’єкту.

Для створення моделей і забезпечення зв’язку з базою даних було використано технологію ORM (Object-Relational Mapping) – Sequelize. Завдяки ORM можна описувати таблиці у вигляді об’єктів і досить просто створювати зв’язки між ними, наприклад, за допомогою методів hasMany, belongsTo. Крім створення моделей, за допомогою Sequelize можна легко налаштувати автоматичні міграції бази даних і створювати складні запити до БД.

Далі перейдемо до розгляду сервісу авторизації на прикладі методу реєстрації. На вхід маємо об’єкт з даними про користувача та його зображення. Спочатку перевіримо наявність аккаунту користувача, для цього у моделі Person викликаємо метод findOne, завдяки якому шукаємо людину в базі даних за поштою. Якщо аккаунт вже було створено, то генеруємо виключення, в іншому

випадку потрібно захешувати пароль за допомогою бібліотеки bcrypt та методу «hash». Далі для створення посилання для активації аккаунту необхідно скористатися пакетом uuid.v4 для створення унікального ідентифікатора. Після цього використовуємо функцію для збереження зображення на сервері та отримуємо ім’я файлу. Потім викликаємо у моделі користувача метод «create», куди передаємо дані про користувача з оновленим захешованим паролем, ім’я файлу зображення та посилання для активації аккаунту. Наступним кроком викликаємо функцію для надсилання листа на пошту. В кінці повертаємо результат функції створення токенів, що надає загальні дані користувача з «access» та «refresh» токенами.

Після визначення REST API та створення основних ендпоінтів можна розгортати клієнтську частину додатку за допомогою бібліотеки React. Спочатку необхідно встановити залежності, зокрема бібліотеку Axios для надсилання HTTP-запитів до серверу. Для цього на клієнті також створюються власні сервіси, що будуть взаємодіяти з визначеним API.

Для прикладу також розглянемо авторизацію. В класі сервісу опишемо основні методи: логін, реєстрація та вихід з аккаунту. За допомогою Axios визначаємо для кожного тип запиту та передаємо маршрут з параметрами або тілом запиту, дані яких подаються на вхід методу. Для обробки запитів перед відправкою або після отримання відповіді Axios надає змогу використовувати так звані Response Interceptors, тобто перехоплювачі відповідей. З їх допомогою, наприклад, можна прикріпити токен до запиту, який потребує захисту від неавторизованих користувачів. Також вони можуть бути використані для перевірки валідності «access» токену та його оновлення в разі потреби.

Маючи основні інтерфейси програми та можливість звертатися до серверу, можна легко створити відображення функціоналу, шляхом написання компонентів. Для використання готових компонентів та їх стилізації було обрано бібліотеку MUI. Готові рішення даної бібліотеки можна легко налаштовувати під власний дизайн, а також це значно заощаджує час на створення інтерфейсу.

Для навігації по додатку використовується бібліотека React Router DOM. Визначення маршрутів у даному web-додатку відбувається завдяки використанню хука useRoutes, який представляє маршрути у вигляді об’єктів. Також використовувались такі хуки як useLocation для отримання параметрів сортування та useParams для отримання параметрів з URL, наприклад, id курсу чи освітнього закладу.

Додатковими інструментами для написання клієнтської частини були наступні: yup – бібліотека для валідації даних; formik – бібліотека для легкої взаємодії з формами в React; file-saver – бібліотека для завантаження файлів.

3.3 Використання програмного додатку

Основний функціонал програми доступний лише для авторизованих користувачів, тому перед початком роботи з web-додатком необхідно пройти реєстрацію або виконати вхід у систему, у разі вже наявного аккаунту. На рисунках 3.2-3.3 представлено модальні вікна логіну та реєстрації. Після підтвердження ведених даних відбувається валідація полів форми. Якщо дані було ведено правильно, операція виконується успішно. У випадку вже наявного аккаунту під час реєстрації або неправильно ведених даних під час логіну з’являється відповідне повідомлення про це.

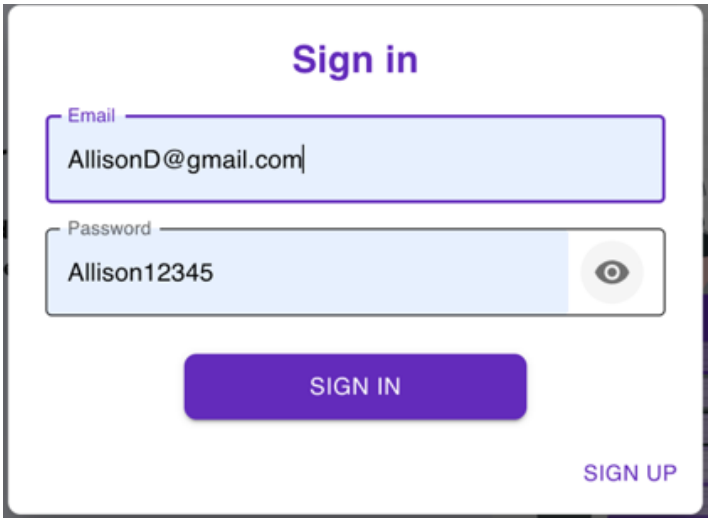


Рисунок 3.2 – Модальне вікно логіну

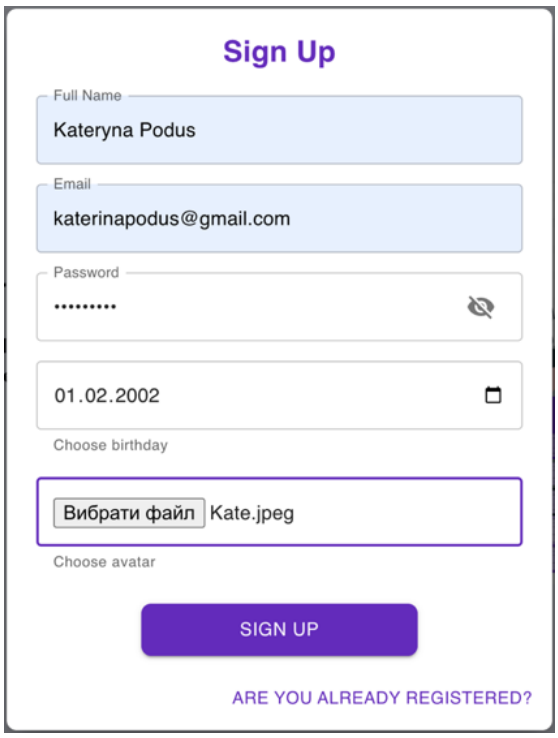


Рисунок 3.3 – Модальне вікно реєстрації

Якщо користувач у ролі гостя намагається отримати доступ до захищених матеріалів, що потребують авторизації, то виводиться вікно з можливістю доєднатися до системи (рис. 3.4).

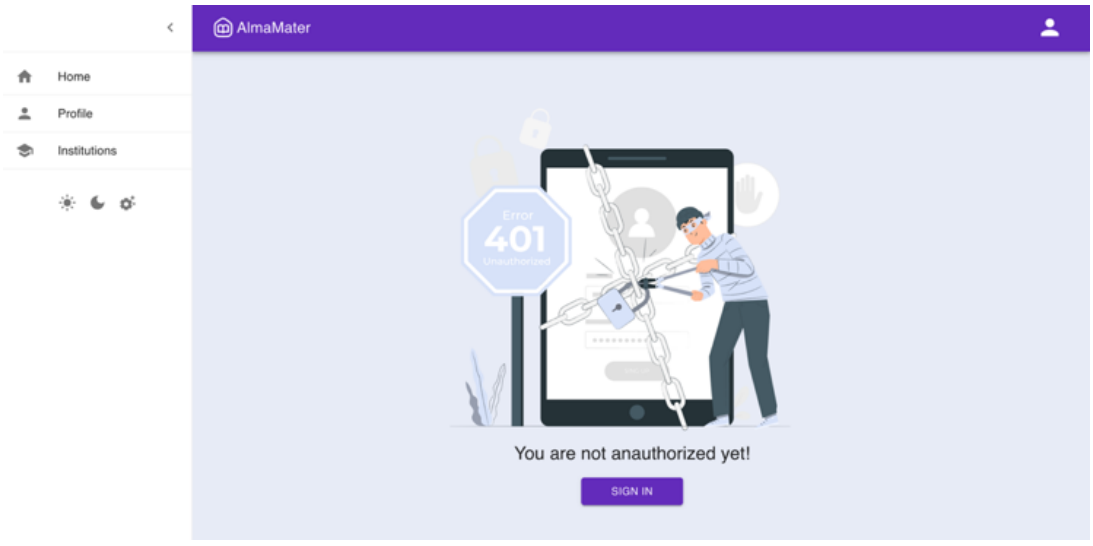


Рисунок 3.4 – Вікно для не авторизованого користувача

Після створення аккаунта користувач потрапляє на свій профіль, зображений на рисунку 3.5, де він може побачити всю інформацію про себе у

карточці. Також є можливість зміни даних за допомогою редагування форми. При натисканні аватарки у шапці додатку відкривається меню з опціями переходу на власний профіль або можливістю вийти з аккаунту. Для переходу по сторінкам додатку в шапці можна відкрити бокове навігаційне меню.

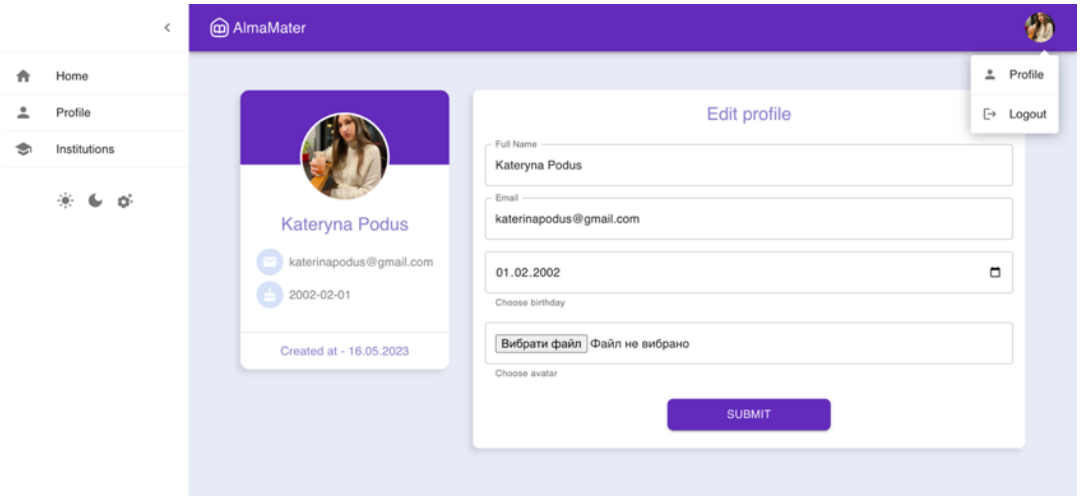


Рисунок 3.5 – Сторінка профілю користувача

Перейшовши на вкладку «Institutions» можна побачити всі освітні заклади, до яких під’єднано користувача та власні створені. Для створення нового закладу освіти потрібно натиснути на кнопку «Create Institution» та заповнити форму даними (рис. 3.6).

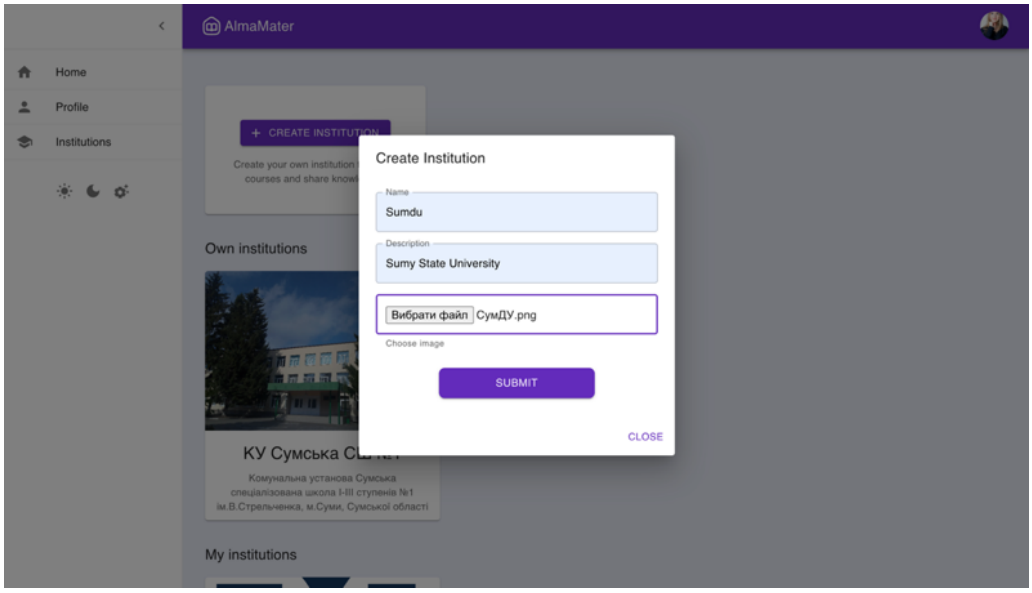


Рисунок 3.6 – Сторінка освітніх закладів і форма створення нового

Відкривши потрібний заклад користувач відразу побачить його курси. Для адміністратора, тобто власника закладу, буде можливість видаляти курси та редагувати їх. Список курсів закладу представлено на рисунку 3.7.

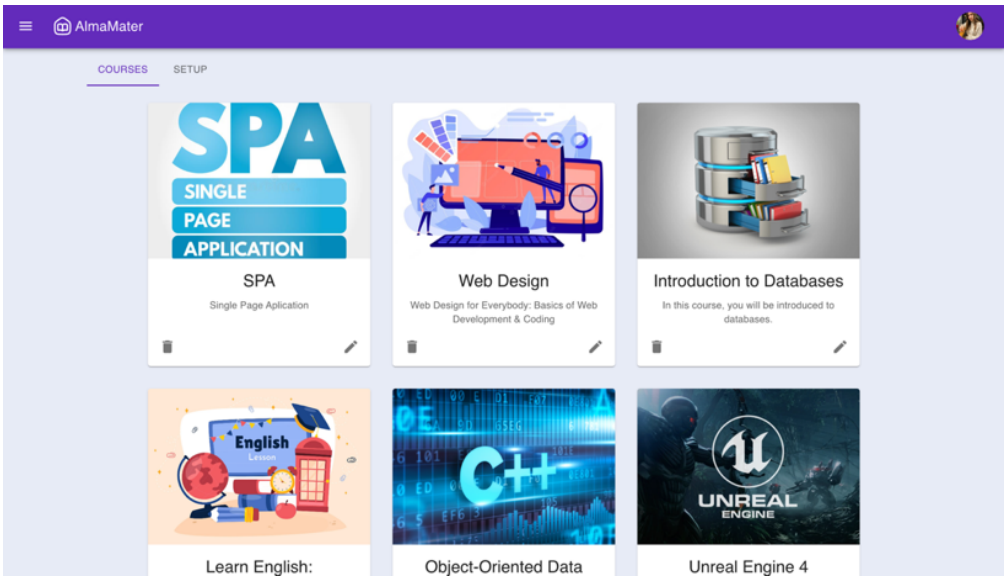


Рисунок 3.7 – Сторінка курсів освітнього закладу

На сторінці закладу у вкладці «Setup» адміністратор може створювати нові курси та групи. Для створення навчального класу необхідно ввести назву та обрати зі списку учнів. Пошук може здійснюватися шляхом обирання потрібних учнів або веденням їх імені. Приклад надведено на рисунку 3.8.

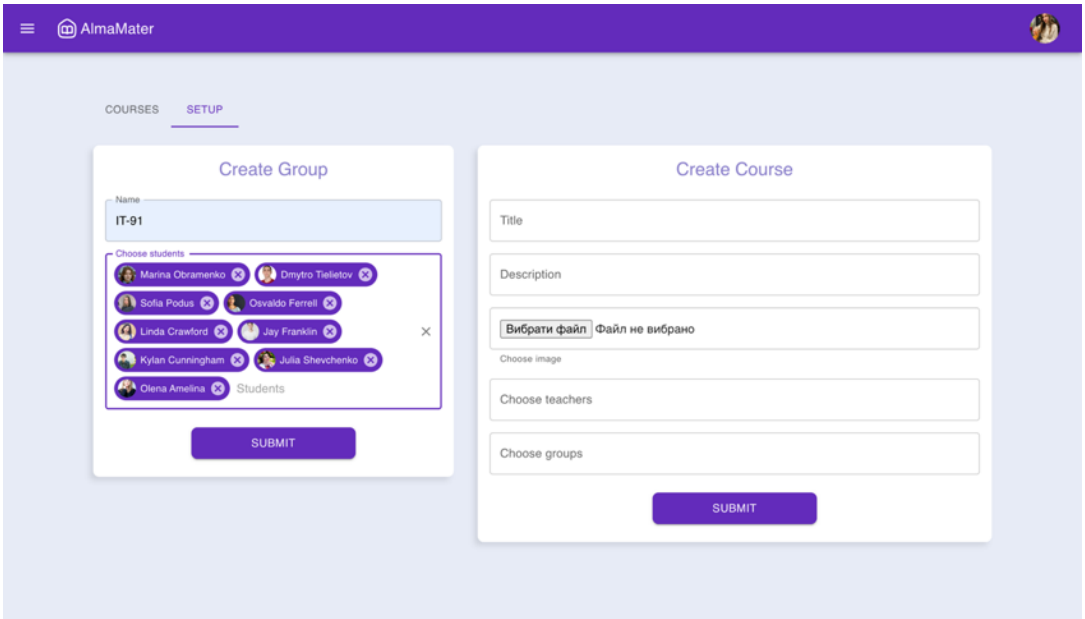


Рисунок 3.8 – Створення нового навчального класу

При створенні курсу також вказується загальна інформація про нього та обираються викладачі і класи, щоб закріпити їх за курсом (рис. 3.9).

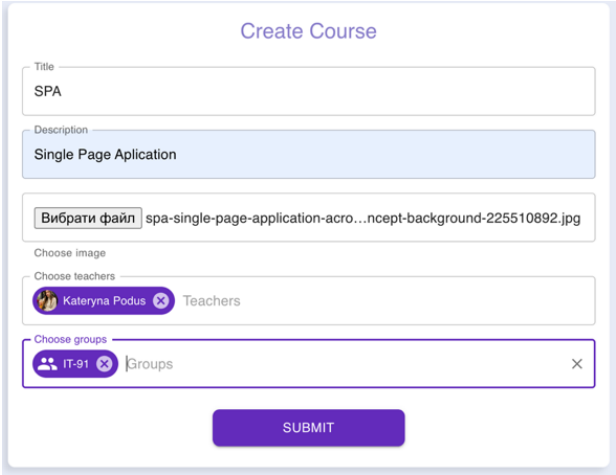


Рисунок 3.9 – Створення нового курсу

Після переходу на сторінку окремого курсу користувач може переглянути список лекцій, практичних завдань, тестів контролю засвоєння знань, учасників курсу та журнал з оцінками. Для вчителя також надається доступ до перегляду списку робіт студентів. Також, якщо користувач у ролі вчителя, він може створювати, редагувати та видаляти певні матеріали курсу. На рисунку 3.10 зображено список практичних матеріалів для користувача у ролі вчителя.

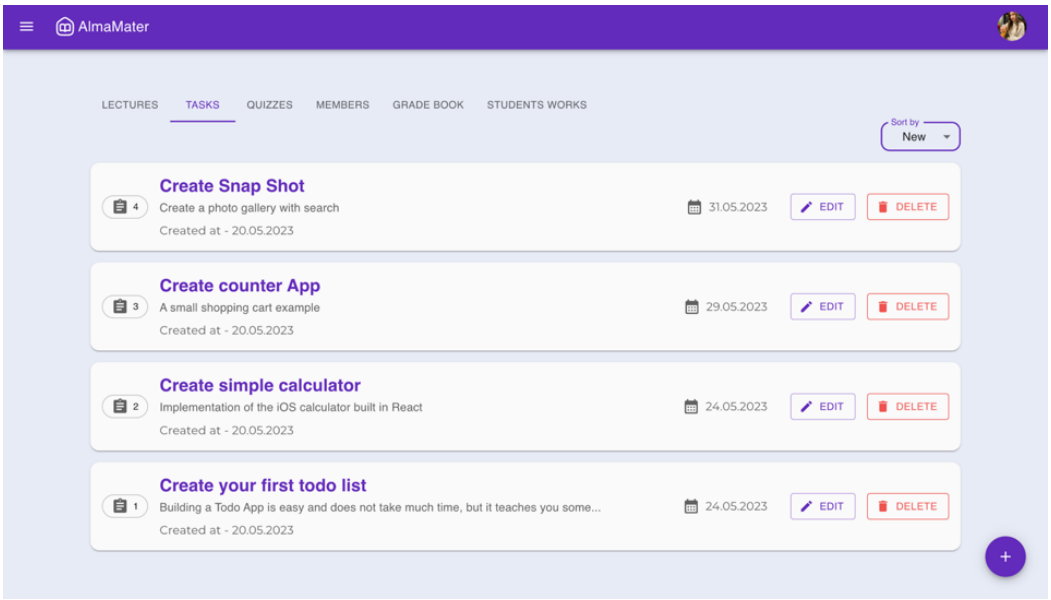


Рисунок 3.10 – Список практичних завдань

На рисунку 3.11 показано як адміністратор може створювати новий практичний матеріал, вказуючи назву, опис, номер завдання та термін здачі роботи. Модальне вікно схожим чином працює для редагування даних, тільки відразу містить заповнену інформацію в полях форми.

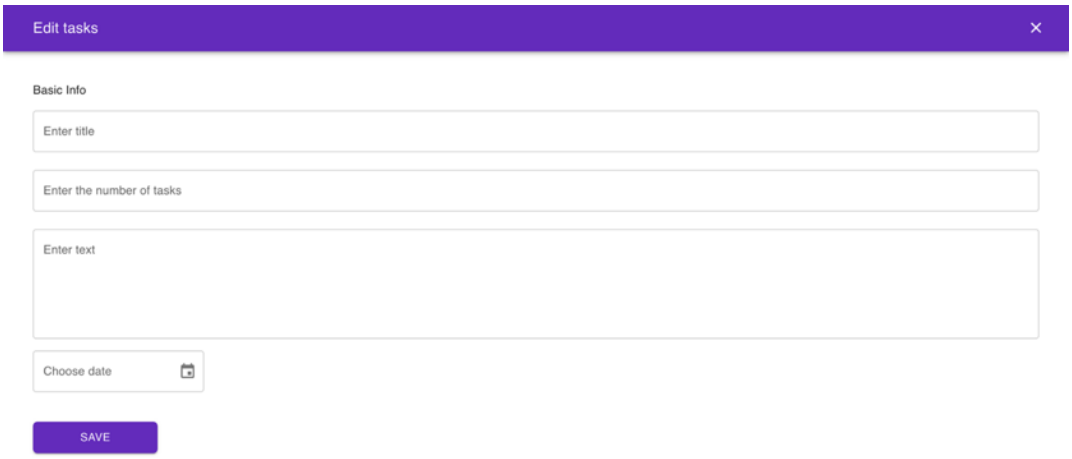


Рисунок 3.11 – Створення та редагування практичного матеріалу

Якщо учень переходить до конкретного завдання, він може надіслати свої роботи. Після надсилання оновлюється історія збережених файлів, які можна скачати та переглянути (рис. 3.12).

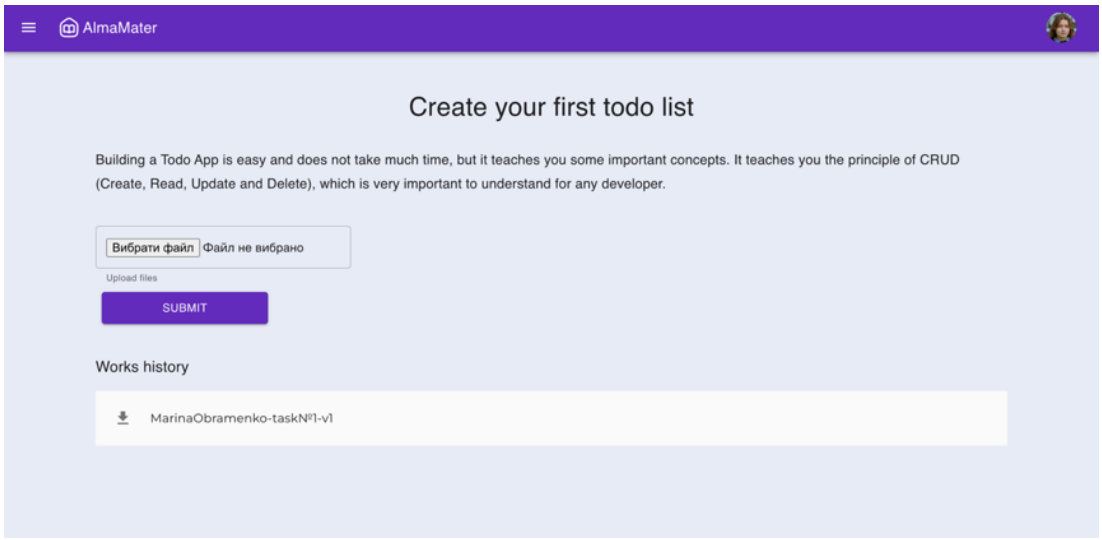


Рисунок 3.12 – Перегляд конкретного матеріалу та завантаження виконаних робіт

Далі, на рисунку 3.13 показано список лекцій для користувача у ролі вчителя та на рисунку 3.14 зображено модальне вікно для створення нової лекції.

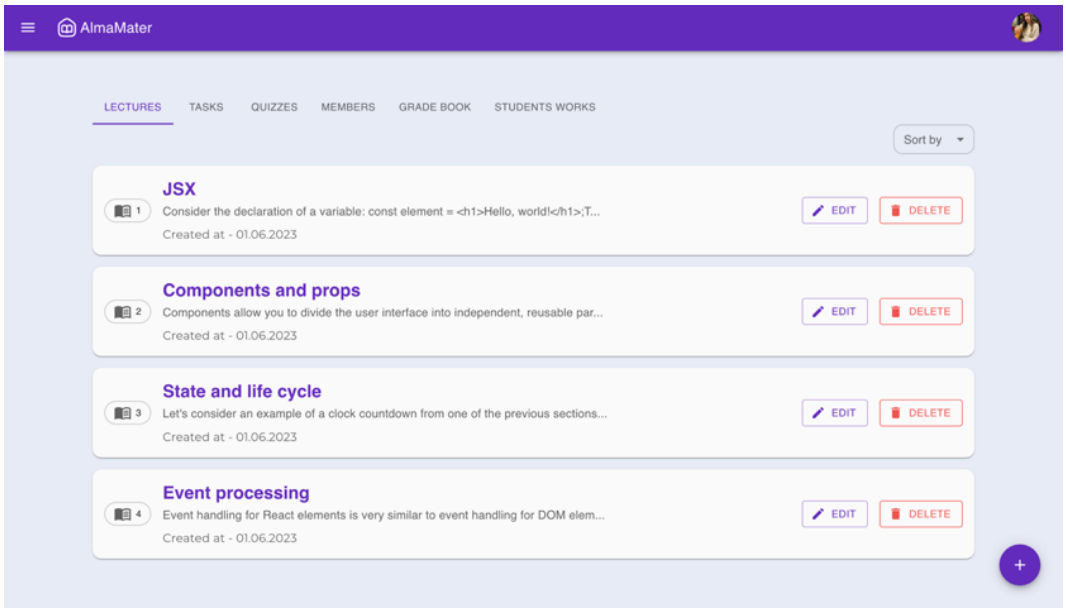


Рисунок 3.13– Список лекцій

Для створення лекції необхідно ввести назву, номер і саме наповнення.

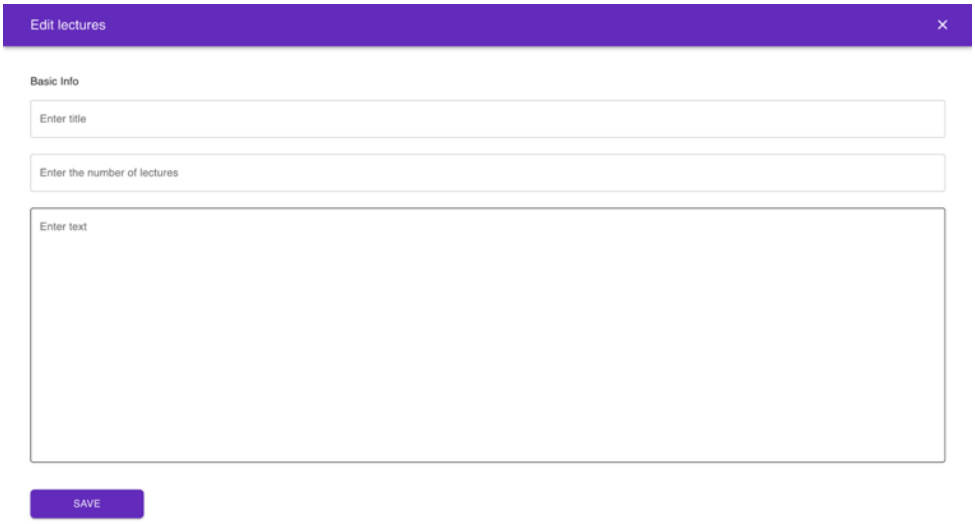


Рисунок 3.14 – Створення та редагування лекції

Переходячи до списку тестів, користувач у ролі учня може бачити активні тести, які мають білий колір, червоні тести це ті, які учень пропустив. Якщо

учень вже пройшов тест і отримав оцінку, тест становиться неактивним і показує результат. Приклад сторінки з тестами зображено на рисунку 3.15.

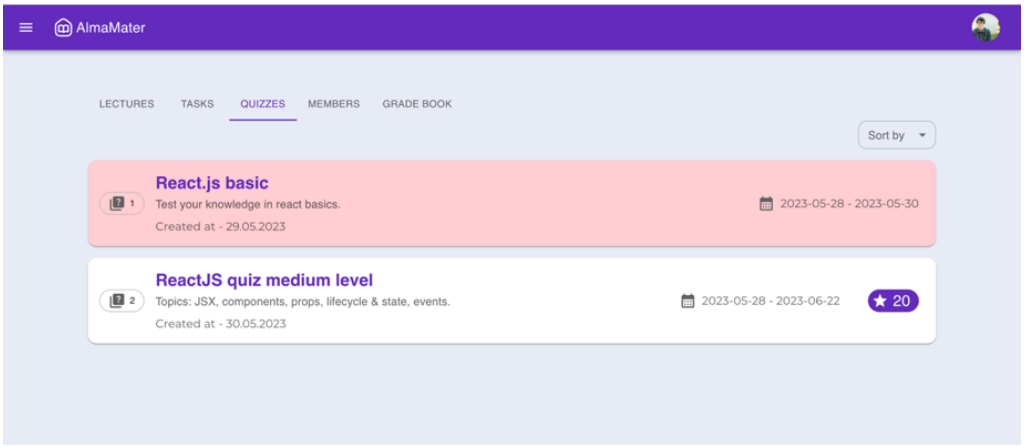


Рисунок 3.15 – Список тестів

Приклад модального вікна для створення та редагування тесту представлено на рисунку 3.16. Необхідно заповнити такі дані: назва, опис, номер тесту, дата початку та кінцю, час на виконання та кількість балів за одне питання.

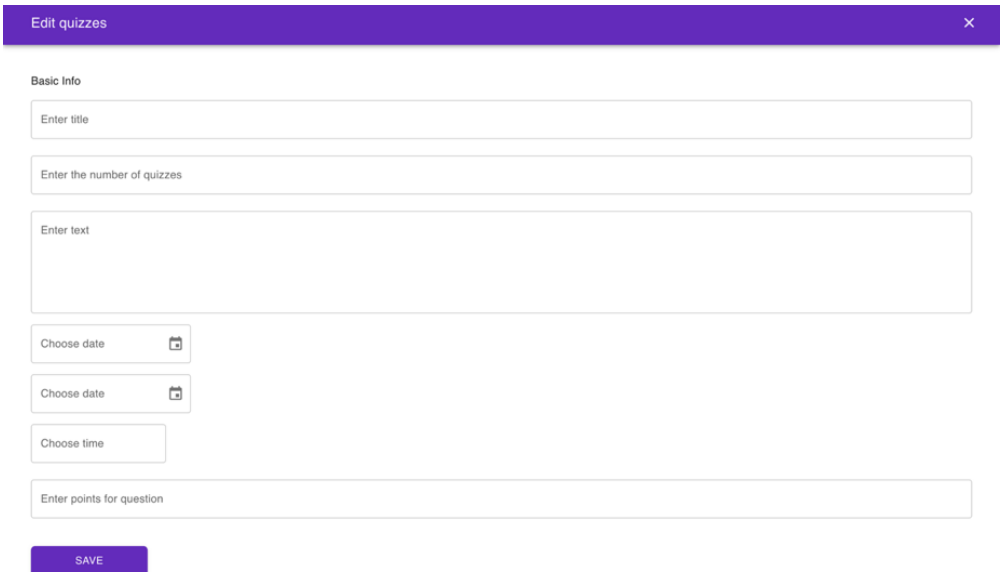


Рисунок 3.16 – Створення та редагування нового тесту

Для створення питань викладачу потрібно перейти до конкретного тесту, де він побачить форму з можливістю додавати нове питання. В даному блоці можна вписувати текст питання та додавати необхідну кількість варіантів

відповідей, натискаючи кнопку «Add option». Серед відповідей можна вказати правильною лише одну (рис. 3.17). Після заповнення всіх даних потрібно натиснути кнопку «Save». У разі необхідності видалити питання, можна скористатися кнопкою «Delete» на ньому.

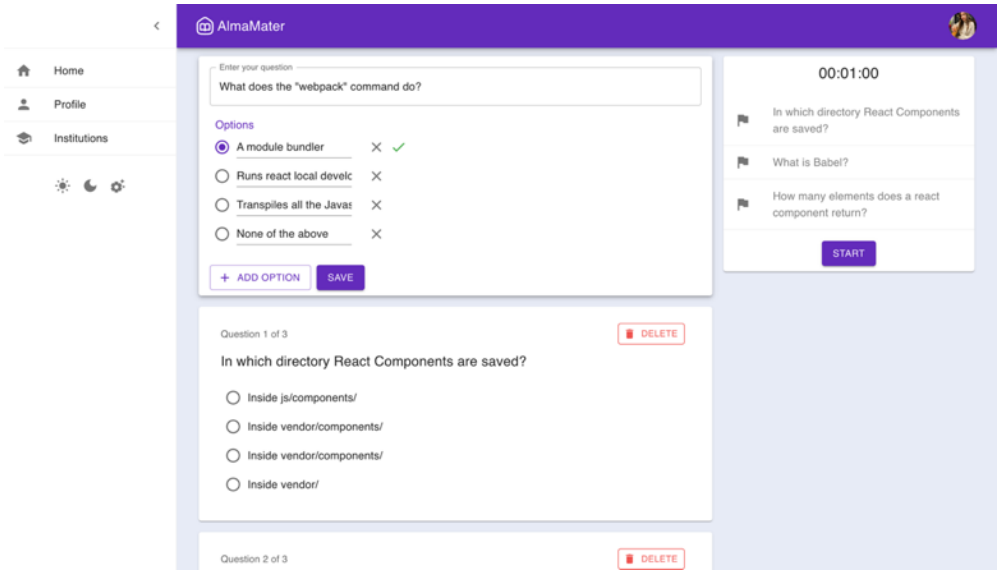


Рисунок 3.17 – Блок для створення питань і варіантів відповідей в тесті

Для учня при переході на тест відображаються лише самі питання та панель з таймером зворотного відліку (рис. 3.18).

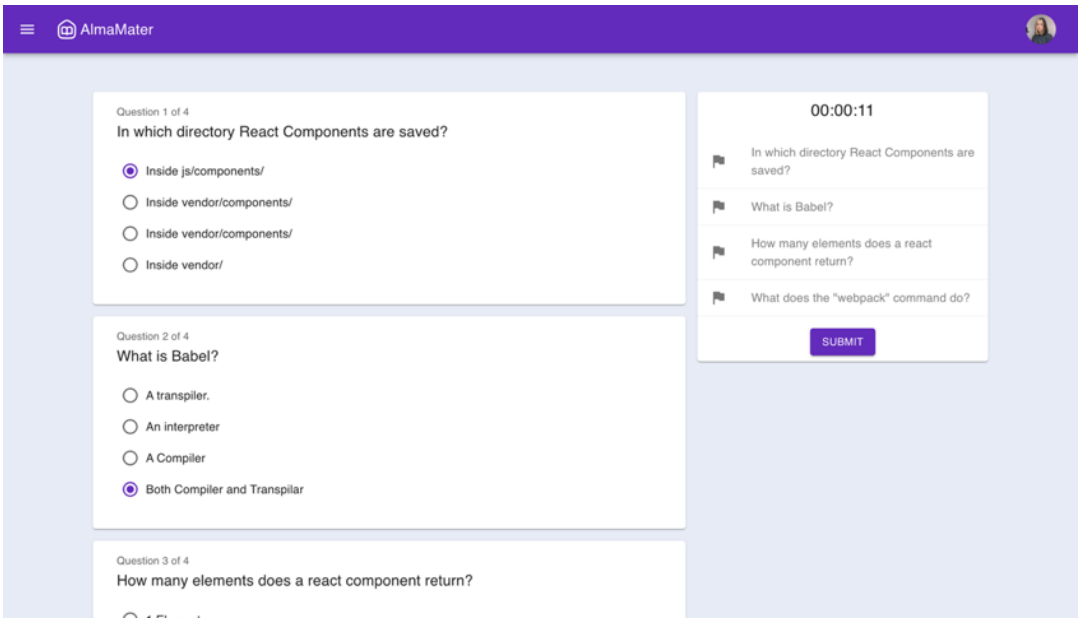


Рисунок 3.18 – Проходження тесту учнем

Після натиснення кнопки «Submit» на панелі з питаннями або при вичерпанні часу з’являється оцінка за тест (рис. 3.19).

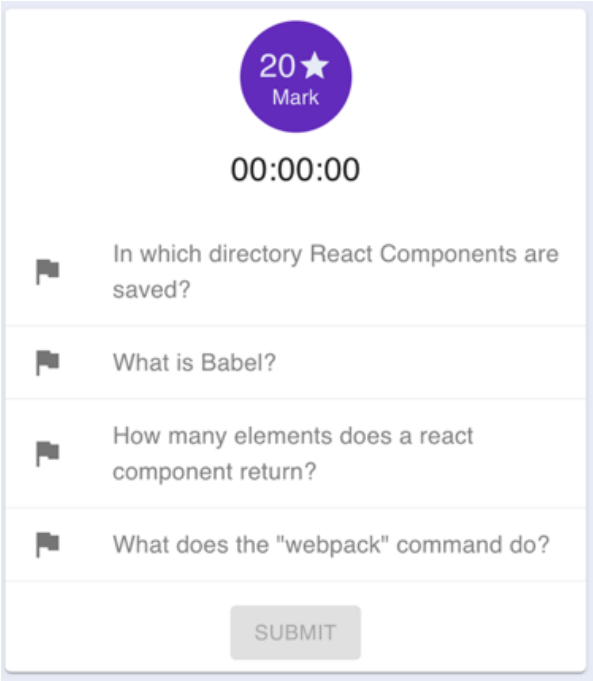


Рисунок 3.19 – Оцінка за пройдений тест

Якщо учень намагається оновити сторінку для повторного проходження тесту з’являється повідомлення, зображене на рисунку 3.20.

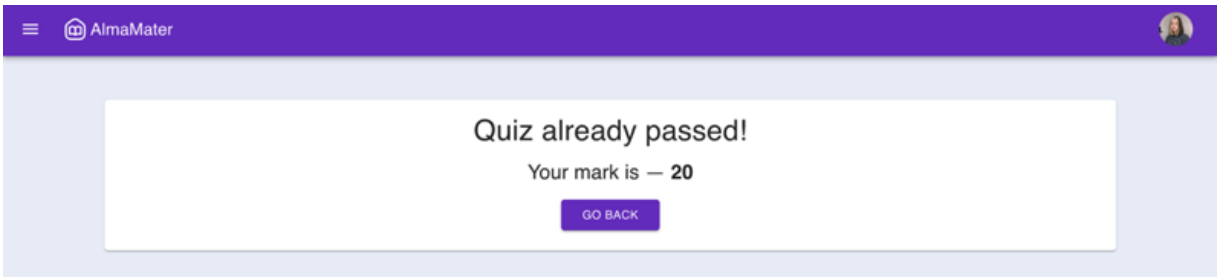


Рисунок 3.20 – Повідомлення про пройдений тест під час оновлення сторінки відразу після проходження

У вкладці «Members» курсу можна переглянути всіх його учасників з деякою інформацією про них (рис. 3.21).

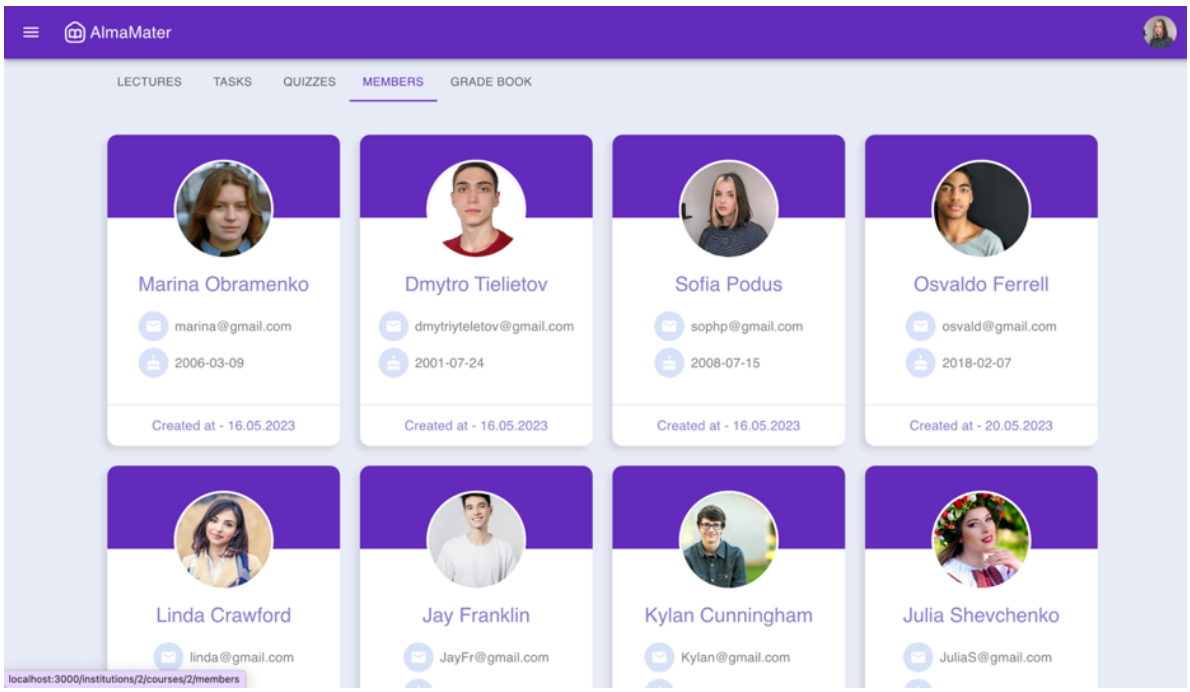


Рисунок 3.21 – Учасники курсу

Кожен учень може також переглянути журнал зі своїми оцінками. Журнал поділяється на такі категорії як тести та практичні завдання. Після проходження тестування оцінка автоматично заноситься в журнал, а оцінки за практичні роботи виставляє вчитель. Журнал представлено на рисунку 3.22.

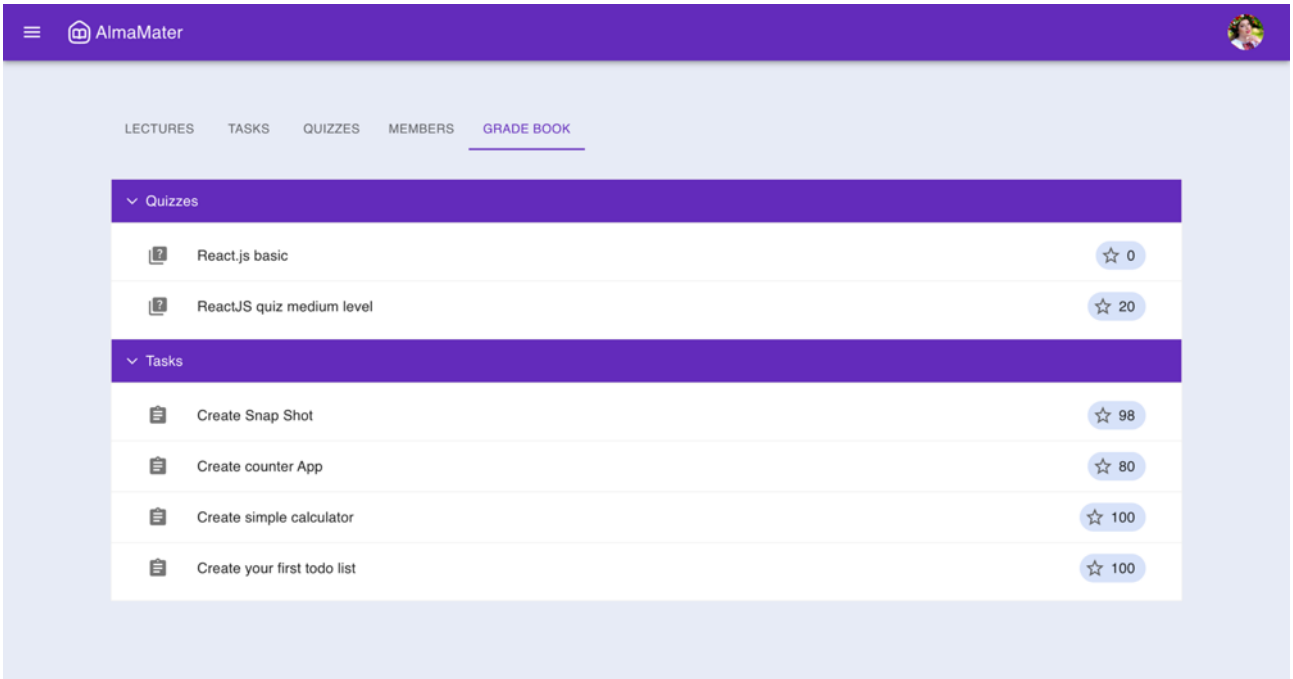


Рисунок 3.22 – Журнал з оцінками

Для вчителів також доступна вкладка «Students works», де відображається список з інформацією про учня, дату надсилання роботи, назву завдання та термін виконання, сам файл завдання з можливістю його завантажити та поле для виставлення оцінки. Роботи які були перевірені мають оцінку та підсвічуються блакитним кольором. Сторінка списку робіт представлена на рисунку 3.23.

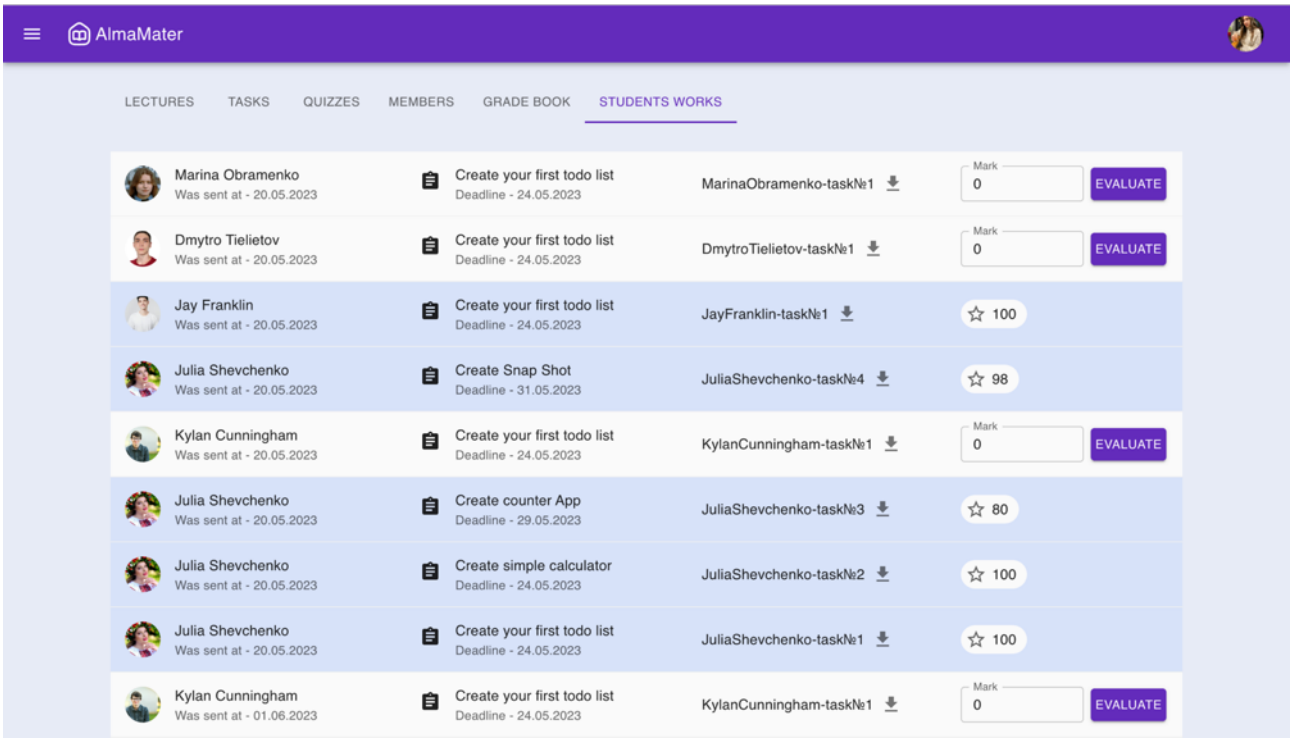


Рисунок 3.23 – Перегляд та оцінювання робіт учнів

ВИСНОВКИ

У ході розробки дипломного проєкту було створено web-додаток підтримки дистанційного навчання для надання та засвоєння освітніх матеріалів. Під час виконання було розглянуто предметну область даної теми, визначено актуальність і проблематику, базуючись на останніх дослідженнях і публікаціях. Зважаючи на великий попит в дистанційному навчанні, особливо в Україні, у зв'язку з пандемією та війною, додатки онлайн-навчання стають все більше в нагоді вчителям та учням. Порівнюючи між собою продукти-аналоги, виявлено, що деякі з них орієнтовані лише на один конкретний освітній заклад, інші мають недостатньо можливостей, наприклад відсутнє виставлення оцінок у журнал, а деякі мають занадто великий і складний для розуміння функціонал. Тому при розробці проєкту були прийняті до уваги всі важливі для навчання процеси й переваги інших розглянутих продуктів, щоб реалізувати їх у власному web-додатку.

На основі аналізу даної теми були визначені функціональні вимоги: можливість створення та редагування власного аккаунту; створення будь-якого дистанційного освітнього закладу, наприклад, школи або університету, в якому адміністратор може створювати курси й навчальні класи; можливість адміністратору закладу закріпляти за певним курсом вчителя та клас; можливість вчителю надавати освітні матеріали для курсу у вигляді лекцій або практик та зазначати терміни здачі робіт; можливість учням переглядати освітні матеріали курсу та завантажувати роботи; можливість вчителю заносити оцінки до журналу для спільного перегляду успішності учнів; можливість створення та проходження тестів для контролю та засвоєння теоретичного матеріалу.

Також було проведено моделювання web-додатку та проєктування бази даних. У результаті структурно-функціонального моделювання отримали контекстну діаграму бізнес-процесів у нотації IDEF0 та діаграму першого рівня декомпозиції IDEF0. У моделюванні варіантів використання було створено

діаграми послідовності, діяльності та інтерфейси для кожного ВВ. Під час проєктування створили фізичну модель даних з описом сутностей та їх зв’язку.

Під час опису програмної архітектури було встановлено архітектуру у вигляді «клієнт-сервер». Взаємозв’язок між компонентами даної архітектури було представлено на схемі.

У пункті програмної реалізації було описано, як влаштована серверна та клієнтська частина й за допомогою яких інструментів була створена кожна з них. Основними інструментами клієнтської частини стали React, Axios, MUI. Для серверної основними було виділено NodeJS, Express. Для прикладу структури серверу та зв’язку клієнта з ним через сервіси було розглянуто реалізацію авторизації користувача.

Отже, після роботи над дипломним проєктом отримали web-додаток підтримки дистанційного навчання.

Результати даної роботи були представлені у вигляді тез на конференції ІМА-2023, які можна переглянути в додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розвиток дистанційної освіти (1998–2021 рр.) | Національна бібліотека України імені В. І. Вернадського. Національна бібліотека України імені В. І. Вернадського. URL: http://nbuv.gov.ua/node/5652 (дата звернення: 03.05.2023).

2. Дослідження стану реалізації дистанційного навчання в Україні. Центр інноваційної освіти «Про.Світ». URL: https://nus.org.ua/wp-content/uploads/2020/05/Research2020_ProSvit_MF1.pdf (дата звернення: 03.05.2023).

3. Якісне дистанційне навчання в умовах війни: поради директору школи. Державна служба якості освіти України. URL: https://sqe.gov.ua/yakisne-distanciyne-navchannya-v-umovakh/ (дата звернення: 03.05.2023).

4. Подус К. О., Нагорний В. В. Web-додаток підтримки дистанційного навчання. Міжнародна наукова конференція IMA-2023, м. Суми, 28 квіт. 2023 р. Суми, 2023.

5. Advantages And Disadvantages Of eLearning. eLearning Industry. URL: https://elearningindustry.com/advantages-and-disadvantages-of-elearning (дата звернення: 03.05.2023).

6. Міністерство освіти і науки України - Дистанційне навчання. Міністерство освіти і науки України. URL: https://mon.gov.ua/ua/osvita/pozashkilna-osvita/distancijne-navchannya (дата звернення: 03.05.2023).

7. Classroom Management Tools & Resources - Google for Education. Google for Education. URL: https://classroom.google.com/ (дата звернення: 04.05.2023).

8. Moodle - Open-source learning platform | Moodle.org. Moodle - Open-source learning platform | Moodle.org. URL: https://moodle.org/ (дата звернення: 05.05.2023).

9. Mix. eLearning@SumyStateUniversity. URL: https://mix.sumdu.edu.ua/ (дата звернення: 05.05.2023).

10. Постановка цілей по SMART – приклади, критерії. ЗВО «Подільський державний університет». URL: https://pdatu.edu.ua/images/vihovna-robota/psiholog/ps10.pdf (дата звернення: 06.05.2023).

11. What is Work Breakdown Structure (WBS) Diagram? - Edraw. [OFFICIAL] Edraw Software: Unlock Diagram Possibilities. URL: https://www.edrawsoft.com/what-is-work-breakdown-structure-diagram.html (дата звернення: 06.05.2023).

12. What Is an Organizational Breakdown Structure (OBS)?. Small Business - Chron.com. URL: https://smallbusiness.chron.com/organizational-breakdown-structure-obs-72523.html (дата звернення: 06.05.2023).

13. Gantt Chart: The Ultimate Guide (Definitions & Examples). ProjectManager. URL: https://www.projectmanager.com/guides/gantt-chart (дата звернення: 06.05.2023).

14. Лекція 6. Нотація IDEF0. Головна | Elib LNTU. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Кондіус%202%20готовва/page9.html (дата звернення: 02.06.2023).

15. USE CASE-діаграма. Приклади використання. Новини високих технологій. URL: https://hi-news.pp.ua/kompyuteri/8924-use-case-dagrama-prikladi-vikoristannya.html (дата звернення: 02.06.2023).

16. Діаграми UML для моделювання процесів і архітектури проекту. Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 02.06.2023).

17. Учасники проектів Вікімедіа. Діаграма послідовності – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_послідовності (дата звернення: 02.06.2023).

18. Tables Relations: One-to-One, One-to-Many, Many-to-Many. TutorialsTeacher - Learn Technologies. URL: https://www.tutorialsteacher.com/sqlserver/tables-relations (date of access: 02.06.2023).

19. Client Server Architecture - Detailed Explanation. InterviewBit. URL: https://www.interviewbit.com/blog/client-server-architecture/ (date of access: 02.06.2023).

20. Getting Started | What is Axios?. Axios. URL: https://www.axios-http.cn/en/docs/intro (date of access: 02.06.2023).

21. What Is the Express Node.js Framework? Solve Hard Problems Faster | heynode.com. URL: https://heynode.com/tutorial/what-express-nodejs-framework/ (date of access: 02.06.2023).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

web-додатку підтримки дистанційного навчання

1 Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Web-додаток підтримки дистанційного навчання має надавати повний доступ до всіх процесів навчання в будь-який зручний час та з будь-якого місця.

1.2 Мета створення web-додатку

Головна мета проєкту – підтримка дистанційного навчання для надання та засвоєння освітніх матеріалів.

1.3 Цільова аудиторія

Цільовою аудиторією даного проєкту можуть бути працівники освітніх закладів, таких як: школи, університети, коледжі. Також це люди, які хочуть створювати власні навчальні курси та здобувачі освіти.

2 Вимоги до web-додатку

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток підтримки дистанційного навчання повинен забезпечувати визначений набір функціональних можливостей, таких як:

– Навігація по додатку повинна бути зрозумілою кожному користувачу.

– Дизайн платформи має бути адаптивним для його перегляду з більшості девайсів.

– Дизайн веб-сайту має правильно відображатись в більшості поширених браузерів.

Кінцевий продукт даного проєкту має бути представлений web-додатком, який містить якісне інформаційне наповнення та графічні матеріали.

2.1.2 Вимоги до персоналу

Для успішної підтримки та експлуатації web-додатку дистанційного навчання необхідні загальні навички роботи з персональним комп'ютером та стандартним веб-браузером. Додаткові технічні навички не є обов'язковими, оскільки програма розроблена з метою забезпечення легкої та зрозумілої роботи з нею. Однак, бажано мати загальне розуміння технологій web-розробки та знання принципів роботи web-додатків, що допоможе персоналу краще зрозуміти проблеми користувачів та швидше їх вирішувати.

2.1.3 Вимоги до збереження інформації

Уся інформація, що міститься у web-додатку повинна зберігатися у базі даних реалізованій засобами системи управління базами даних PostgreSQL.

2.1.4 Вимоги до розмежування доступу

Web-додаток підтримки дистанційного навчання повинен бути загальнодоступним у мережі Інтернет.

Права доступу до інформації розподілені за групами користувачів: адміністратор закладу, вчитель та учень. Роль адміністратора автоматично надається людині, що створює дистанційний освітній заклад. Адміністратор в свою чергу підключає людей до закладу, надає деяким учасникам закладу роль вчителя та створює курси. Далі він створює класи, надаючи учасникам закладу роль учня та курс. За курсом потім закріпляються клас і вчитель.

У кожної людини є можливість створювати, редагувати та видаляти власний аккаунт, переглядати список з деякою інформацією про інших учнів, вчителів.

Вчитель має можливість надавати освітні матеріали для курсу у вигляді лекцій або практик та зазначати терміни здачі робіт. Також він має право на перевірку робіт учнів і занесення оцінок до журналу. Крім цього, вчитель може створювати тести для контролю та засвоєння теоретичного матеріалу.

В свою чергу, учень може переглядати всі освітні матеріали, бачити терміни здачі робіт та завантажувати виконані роботи вчителю на перевірку.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

Структура web-додатку складається з набору основних вкладок у боковій панелі та шапці, в якій міститься логотип та можливість авторизуватися або вийти з аккаунту.

Основні блоки програми:

Головна сторінка – загальна інформація про додаток.

Профіль – особиста сторінка користувача з інструментами редагування та видалення профілю, можливістю створення закладу.

Заклади – сторінка з переліком власних закладів користувача або тих закладів, до яких користувач під’єднаний.

Заклад – при переході зі сторінки закладів на окремий заклад для адміністратора є можливість створювати навчальні класи та курси, назначаючи вчителя та клас для курсу, переглядати та переходити на окремі курси. Учні відразу бачать список курсів, до яких вони під’єднані

Курси – сторінка з переліком власних курсів або тих, до яких користувач під’єднаний.

Курс – в курсі вчителі мають можливість надавати освітні матеріали та створювати тести для контролю та засвоєння теоретичного матеріалу Також для вчителів є окрема вкладка для перевірки робіт учнів. Учні мають можливість переглядати та завантажувати матеріали.

Лекція – сторінка в курсі з лекційними матеріалами.

Практика – сторінка в курсі з практичними матеріалами.

Тестування – сторінка в курсі з тестами для контролю знань.

Журнал – сторінка в курсі для перегляду оцінок учня.

Учасники курсу – сторінка в курсі для перегляду всіх учасників курсу.

2.2.2 Навігація

Для навігації по додатку повинна бути створена зручна бокова панель, що забезпечить швидке переміщення користувача по всім доступним блокам. Панель можна відкрити по натисканню на спеціальну навігаційну кнопку, що знаходиться в шапці додатку, разом з логотипом та ще деякими елементами навігації. Шапка завжди залишається доступною для взаємодії з нею.

2.2.3 Наповнення web-додатку (контент)

Управління контентом здійснюється в залежності від завантажених матеріалів користувачами в профілі, освітніх закладах або курсах, це можуть бути файли різних форматів.

2.2.4 Дизайн та структура додатку

Дизайн web-додатку має бути виконаний у мінімалістичному та сучасному стилі. Серед основних кольорів обрано фіолетовий, білий та сірий відтінки. Види і розміри шрифтів повинні бути комфортними для перегляду.

Інформаційні блоки, графічні матеріали та інші елементи блоків повинні мати зручне і логічне розташування. Навігація повинна бути зрозумілою для користувачів без попереднього досвіду роботи з додатком. Розташування основних блоків додатку схематично зображено на рисунках А.1-А.11.

Рисунок А.1 – Схема головної сторінки

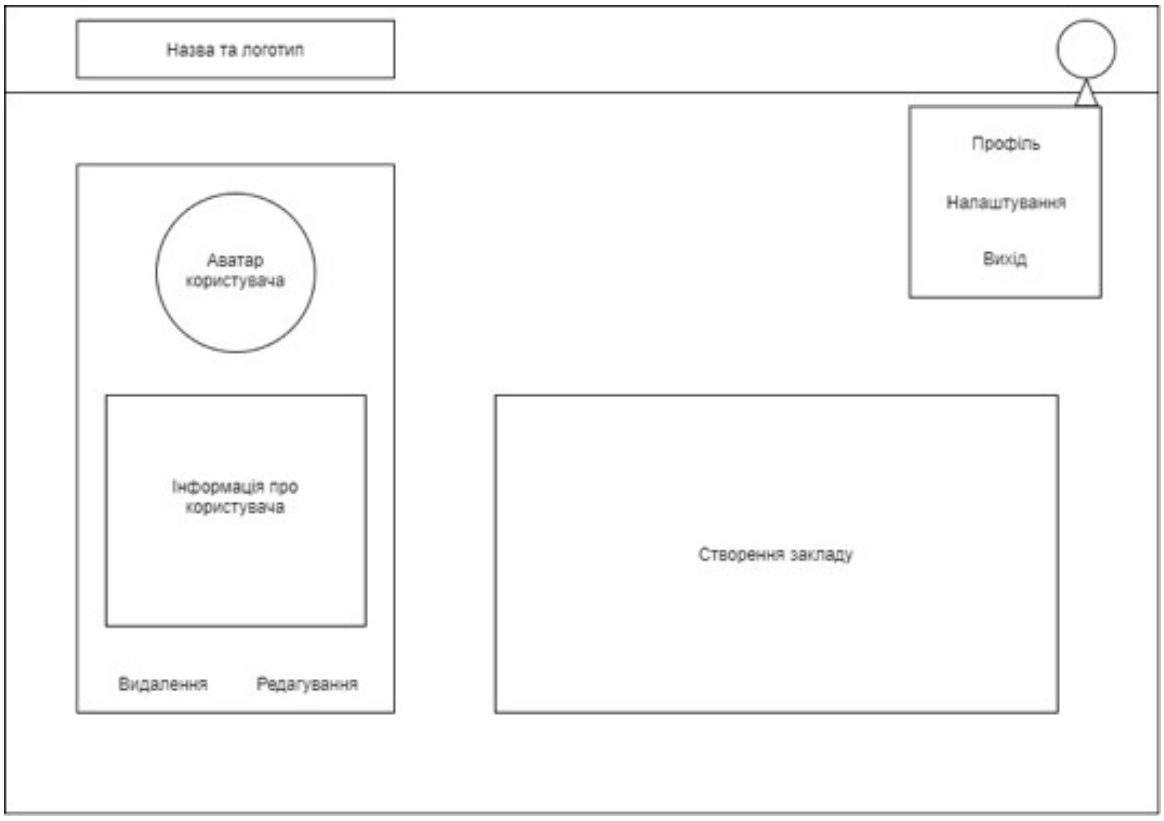


Рисунок А.2 – Схема сторінки профілю

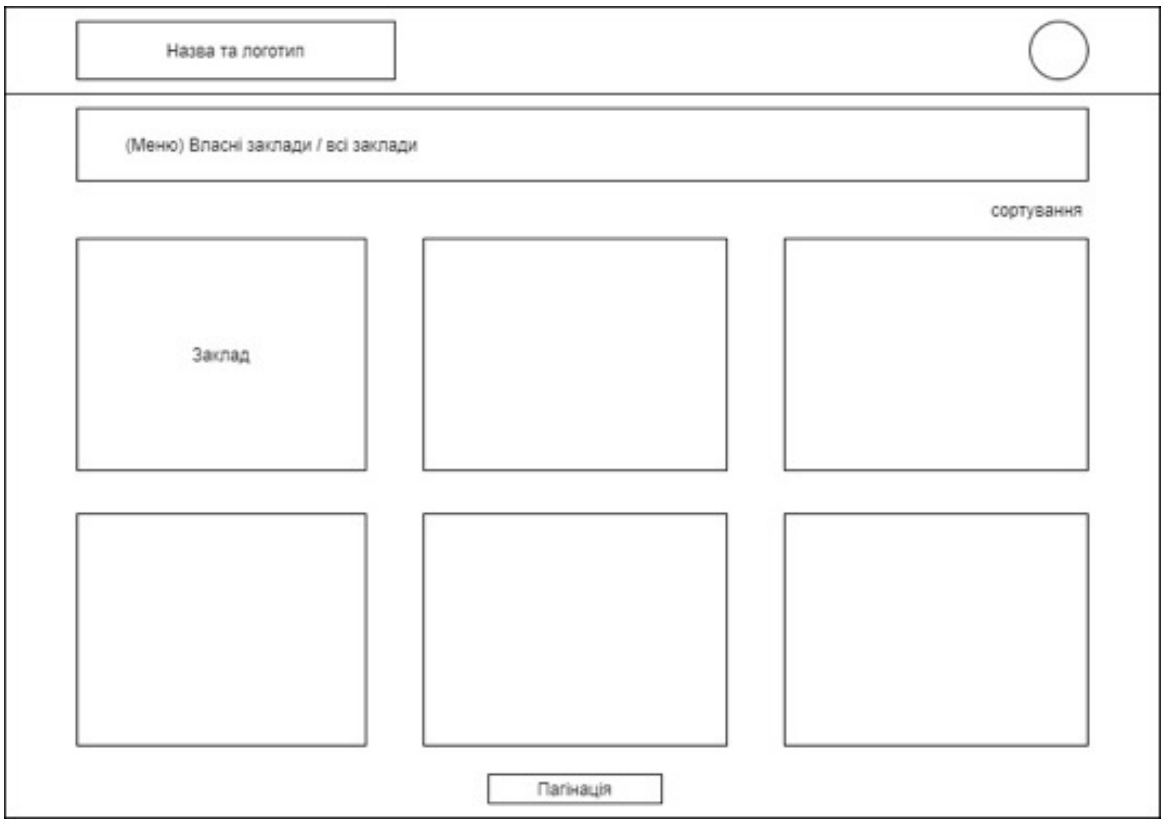


Рисунок А.3 – Схема сторінки закладів

Рисунок А.4 – Схема сторінки окремого закладу

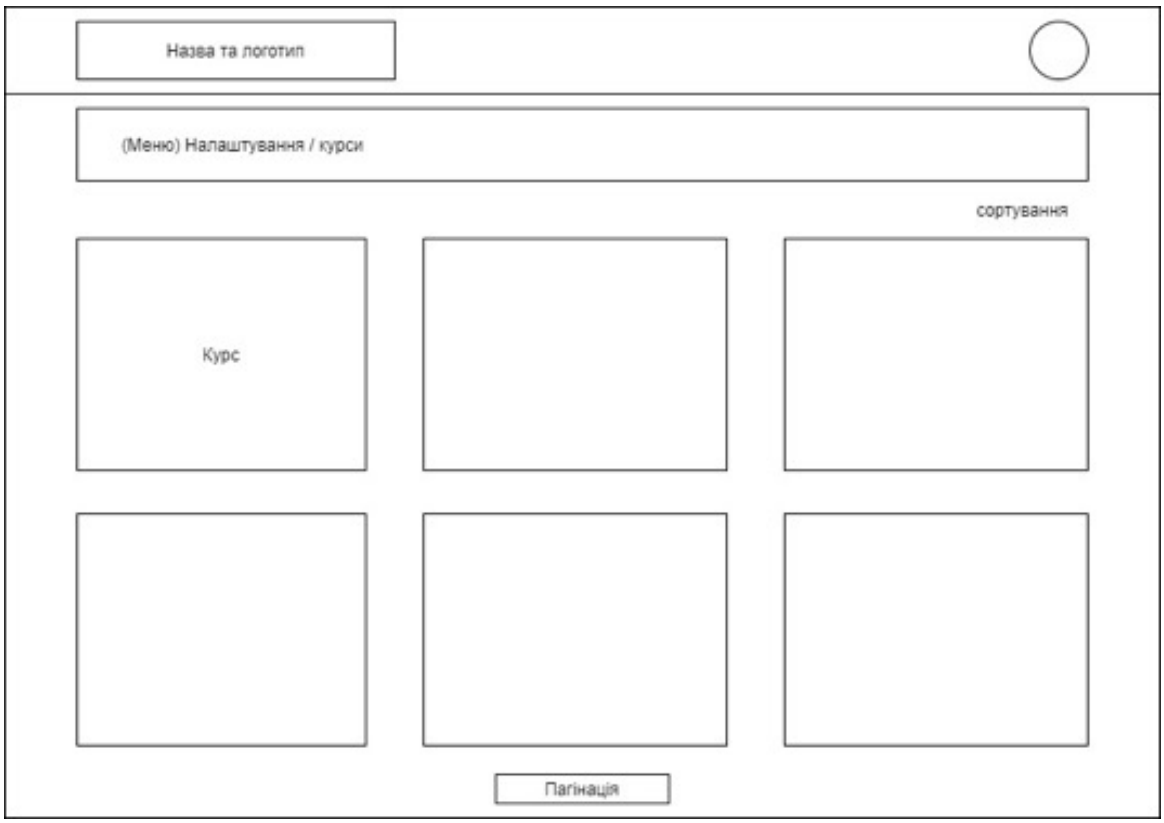


Рисунок А.5 – Схема сторінки курсів

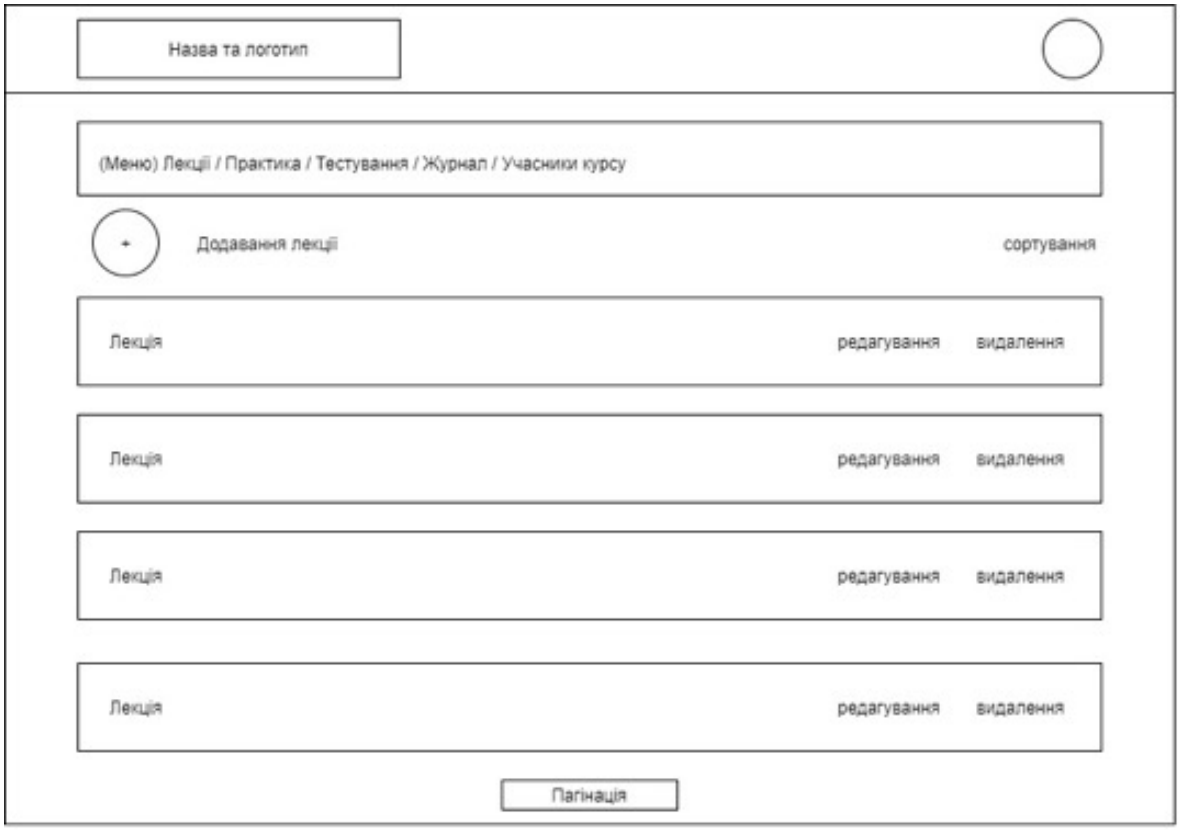


Рисунок А.6 – Схема сторінки окремого курсу

Рисунок А.7 – Схема сторінки окремої лекції

Рисунок А.8 – Схема сторінки окремого тестування

Рисунок А.9 – Схема сторінки окремої практики

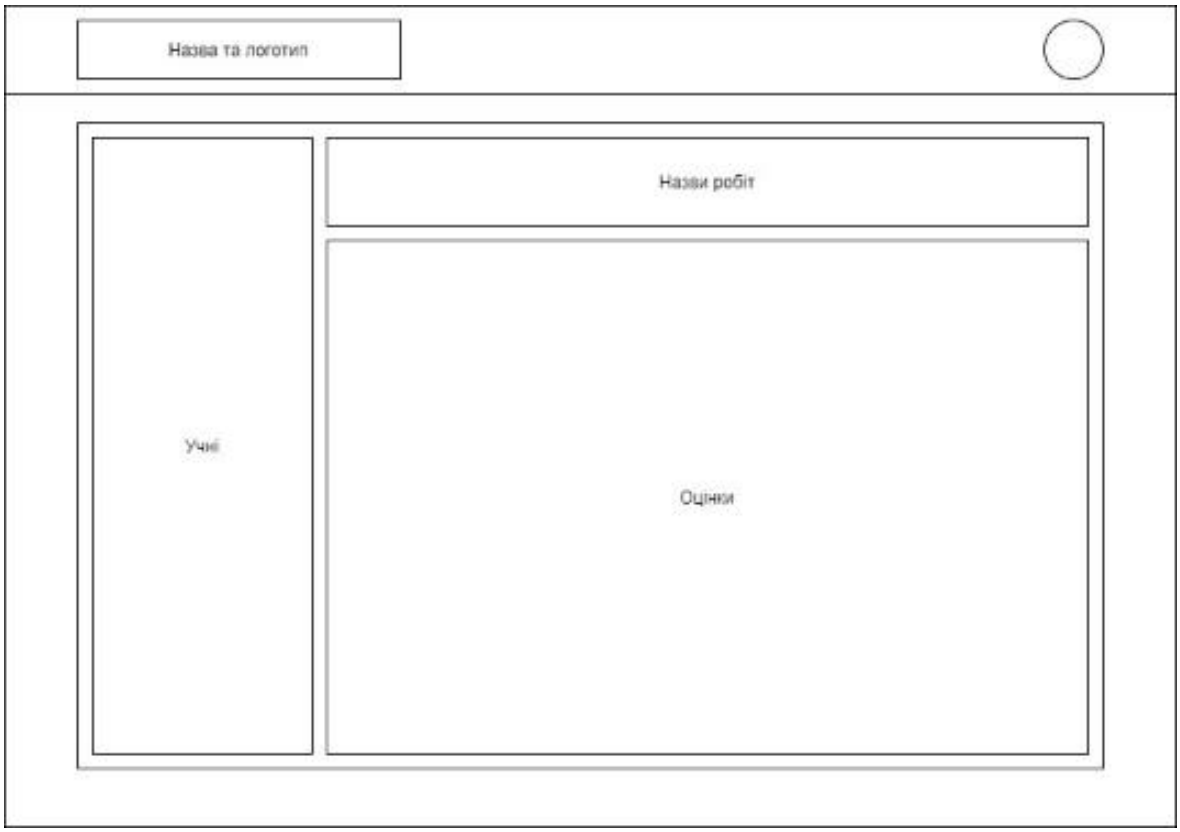


Рисунок А.10 – Схема сторінки журналу

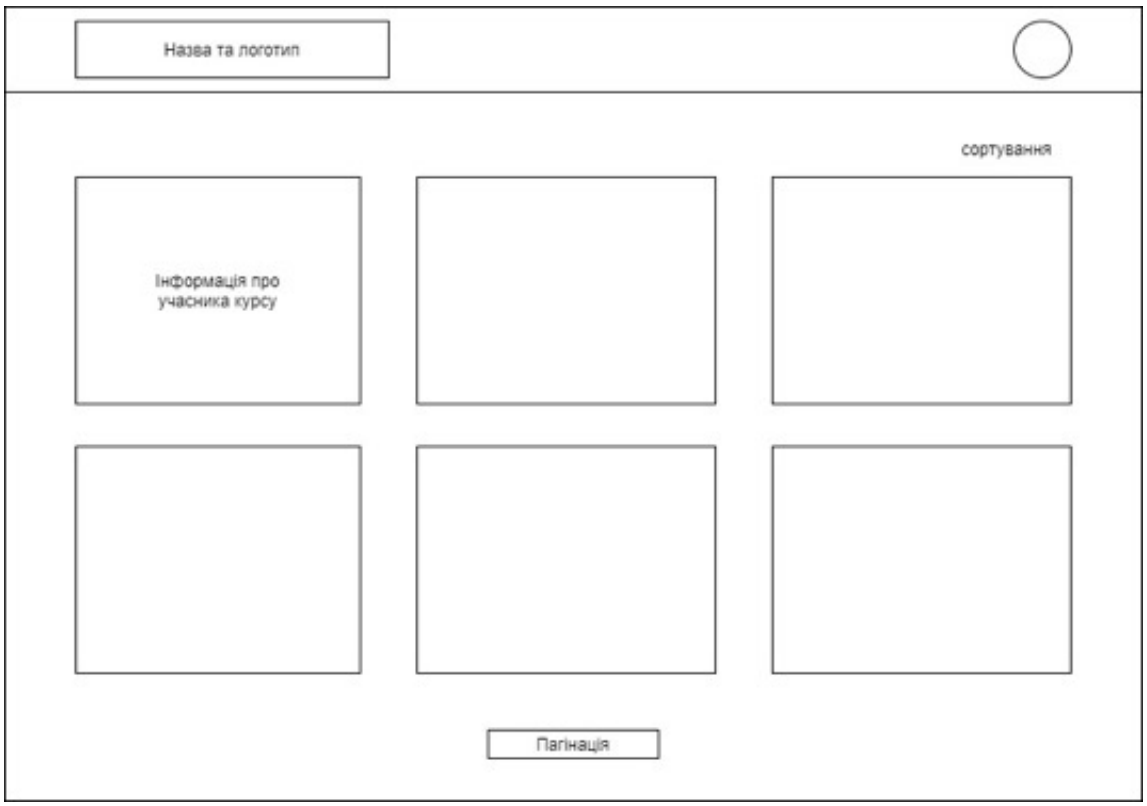


Рисунок А.11 – Схема головної учасників курсу

2.2.5 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.12.

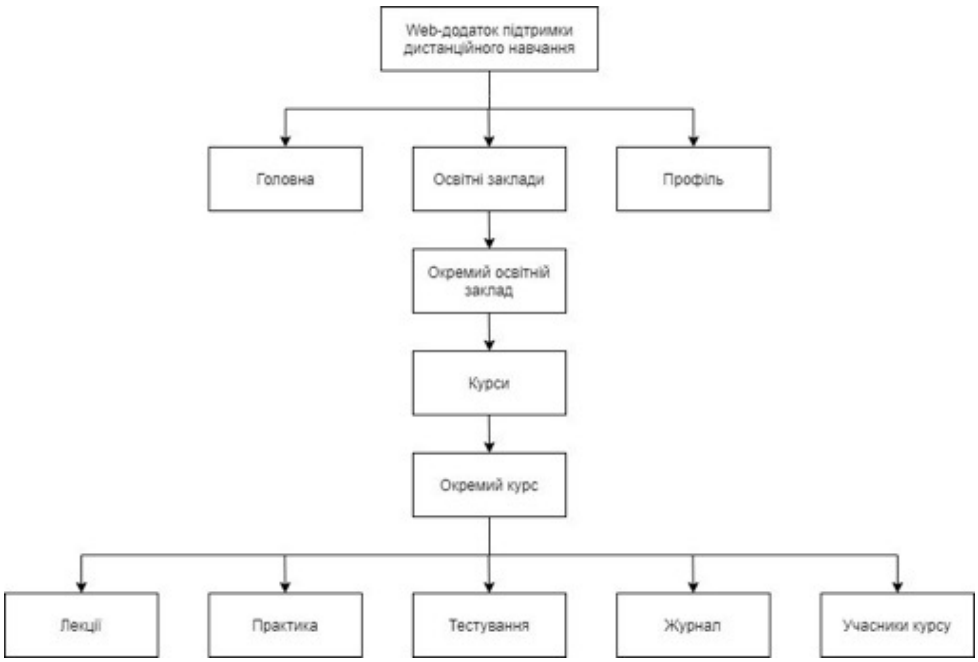


Рисунок А.12 – Карта web-додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Створення дистанційних освітніх закладів	Адміністратор
UN-02	Створення курсів	Адміністратор
UN-03	Наповнення курсів освітніми матеріалами лекцій, практик, тестів	Вчитель
UN-04	Створення навчальних класів	Адміністратор
UN-05	Закріплення за курсом вчителя та навчального класу	Адміністратор
UN-06	Перегляд всіх власних закладів і тих, до яких користувач під’єднаний	Адміністратор, вчитель, учень
UN-07	Перегляд всіх власних курсів і тих, до яких користувач під’єднаний	Адміністратор, вчитель, учень
UN-08	Завантаження виконаних робіт до практичних завдань курсу	Учень
UN-09	Оцінювання учнів та виставлення оцінки в журнал	Вчитель
UN-10	Перегляд журналу	Адміністратор, вчитель, учень
UN-11	Створення, редагування та видалення профілю	Адміністратор, вчитель, учень

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

– можливість створення та редагування власного аккаунту;

– створення будь-якого дистанційного освітнього закладу, наприклад, школи або університету, в якому адміністратор може створювати курси і навчальні класи;

– можливість адміністратору закладу закріпляти за певним курсом вчителя та клас;

– можливість вчителю надавати освітні матеріали для курсу у вигляді лекцій або практик та зазначати терміни здачі робіт;

– можливість учням переглядати освітні матеріали курсу та завантажувати роботи;

– можливість вчителю заносити оцінки до журналу для спільного перегляду успішності учнів;

– можливість створення та проходження тестів для контролю та засвоєння теоретичного матеріалу.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	**Системні вимоги**	**Пріоритет**	**Опис**
SR-01	Наявність форм для авторизації користувача	M	Надає можливість клієнту зареєструватися або просто увійти до свого аккаунту

Продовження таблиці А.2

ID	**Системні вимоги**	**Пріоритет**	**Опис**
SR-02	Перегляд, редагування та видалення власного профілю	M	Надає можливість переглядати, змінювати дані в профілі або видаляти його.
SR-03	Освітні заклади	M	Можливість створювати або переглядати дистанційні освітні заклади
SR-04	Курси	M	Можливість створювати або переглядати курси в закладі
SR-05	Освітні матеріали курсів	M	Можливість завантаження освітніх матеріалів до курсу
SR-06	Блок з тестами	M	Надає можливість створювати або проходити тестування для контролю знань
SR-07	Журнал з оцінками	M	Можливість занесення та перегляду оцінок в журналі

Продовження таблиці А.2

ID	Системні вимоги	Пріоритет	Опис
SR-08	Панель адміністратора	M	Відповідає за створення та редагування освітніх закладів, курсів, навчальних класів, закріплення вчителя та класу за певним курсом
SR-09	Перегляд успішності кожного учня	C	Можливість вчителю переглядати успішність кожного учня

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проєкту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням:

–написання клієнтської частини – React, JavaScript, HTML, CSS;

–написання серверної частини – Node.JS, Express;

–база даних – PostgreSQL.

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути виконаний українською або англійською мовою.

2.4.3 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи додатку рекомендовано наступні мінімальні системні вимоги:

–web-додаток може працювати на різних операційних системах, таких як Windows, MacOS, Linux та інші;

–web-додаток повинен підтримувати роботу з різними веб-браузерами, такими як Google Chrome, Mozilla Firefox, Safari, Opera та інші;

–web-додаток вимагає постійного інтернет-з'єднання з достатньою швидкістю передачі даних, залежно від типу контенту, який надається користувачам.

3 Склад і зміст робіт зі створення web-додатку

Докладний опис етапів роботи зі створення web-додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення web-додатку

№	**Склад і зміст робіт**	**Строк розробки**
1	Розробка шаблону додатку	10 днів
2	Розробка бази даних	5 днів
3	Написання клієнтської частини	30 днів
4	Написання серверної частини	30 днів
5	Наповнення контентом додатку	3 дня
6	Розміщення на хостингу	1 день
7	Перевірка працездатності	3 дня
8	Написання супровідної документації	1 день
9	Реліз додатку	1 день

Продовження таблиці А.3

№	**Склад і зміст робіт**	**Строк розробки**
	Загальна тривалість робіт	84 дні

4 Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Для використання web-додатку користувачами, його потрібно розмістити у мережі Інтернет. Для цієї процедури необхідно придбати доменне ім’я та місце на хостингу, куди буде переміщено сам web-додаток і база даних.

ДОДАТОК Б
Планування робіт

Деталізація мети проєкту методом SMART [10]. Щоб досягти успішності проєкту, спочатку потрібно чітко сформулювати його мету, для цього використовують SMART-метод. Результати даного методу представлені в таблиці Б.1.

Таблиця Б.1 – Деталізація мети проєкту SMART-методом

Критерій	**Мета**
Specific (конкретна)	Створити web-додаток для підтримки дистанційного навчання
Measurable (вимірювана)	Близько 10 під’єднаних до web-додатку освітніх закладів
Achievable (досяжна)	Досвід роботи з подібними проєктами; визначені функціональні вимоги та інструменти для створення додатку; знання необхідних мов, фреймворків і баз даних: JavaScript, React, NodeJS, PostgreSQL; мається необхідна техніка та ПЗ.
Relevant (реалістична)	Web-додаток підтримки дистанційного навчання надасть змогу будь-яким освітнім закладам і людям, що прагнуть викладати свої курси виконувати всі навчальні процеси в зручний час і з будь-якого місця.
Time-framed (обмежена в часі)	Термін виконання проєкту – до кінця 25 травня 2023 року

Планування змісту структури робіт. WBS діаграма – це згруповані та пов’язані між собою частини робіт проєкту, які представлені у графічному виді [11]. Сама діаграма поділяється на різну кількість рівнів, на першому якої знаходиться сам розроблюваний продукт, а на інших упорядковані види робіт. На рисунку Б.1 представлена WBS діаграма з розробки web-додатку підтримки дистанційного навчання.

Планування структури організації, для впровадження готового проєкту (OBS). На основі раніше описаної WBS діаграми створюється організаційна структура виконавців проєкту – OBS [12]. Замість кожного виду робіт, в діаграмі вказуються виконавці, відповідальні за певне завдання. OBS діаграму зображено на рисунку Б.2. В таблиці Б.2 вказані всі учасники проєкту.

Таблиця Б.2 – Виконавці проєкту

Роль	**Ім’я**	**Проєктна роль**
Розробник	Подус К.О.	Виконує розробку клієнтської та серверної частини
Проєктувальник	Подус К.О.	Виконує проєктування бази даних та розробляє структуру web-додатку
Тестувальник	Подус К.О.	Відповідає за тестування функціоналу web-додатку
Керівник проєкту	Нагорний В.В.	Формує завдання на розробку проєкту
Менеджер проєкту	Подус К.О.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних

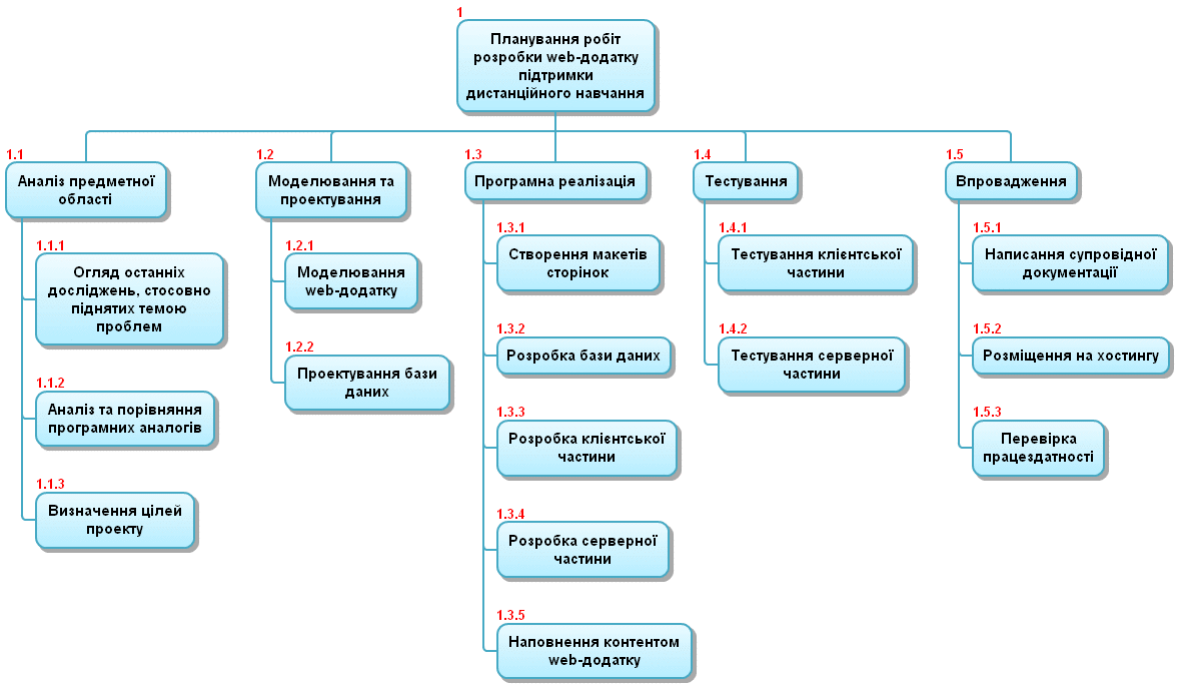


Рисунок Б.1 – WBS-структура робіт проєкту

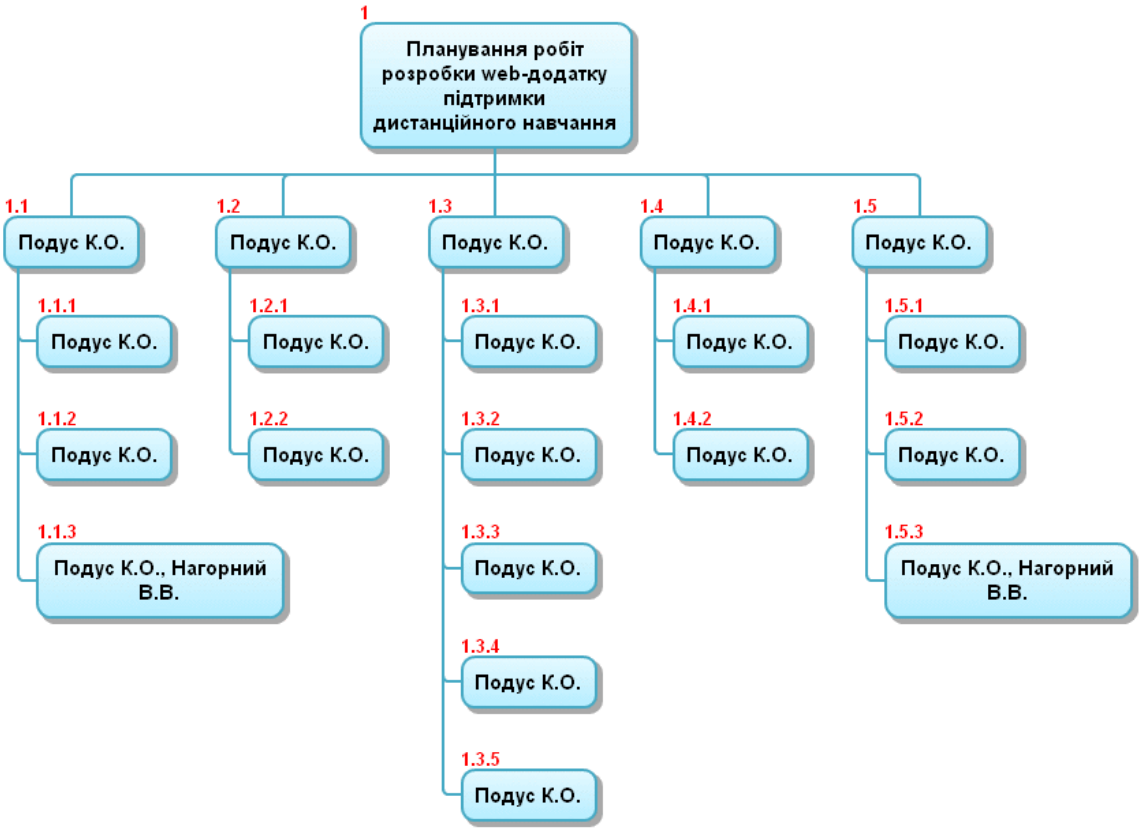


Рисунок Б.2 – OBS-структура робіт проєкту

Діаграма Ганта. Один з поширених способів візуального представлення календарного плану робіт – це діаграма Ганта [13]. Вона дозволяє описати всі

робочі процеси, включаючи їх дати початку та кінця, без урахування святкових і вихідних днів. В результаті чітко видно скільки часу займає розроблення проєкту в цілому та окремо по кожному пункту. Завдяки діаграмі Ганта просто контролювати та планувати свій час на вирішення питань розробки продукту. Календарний графік представлено на рисунку Б.3

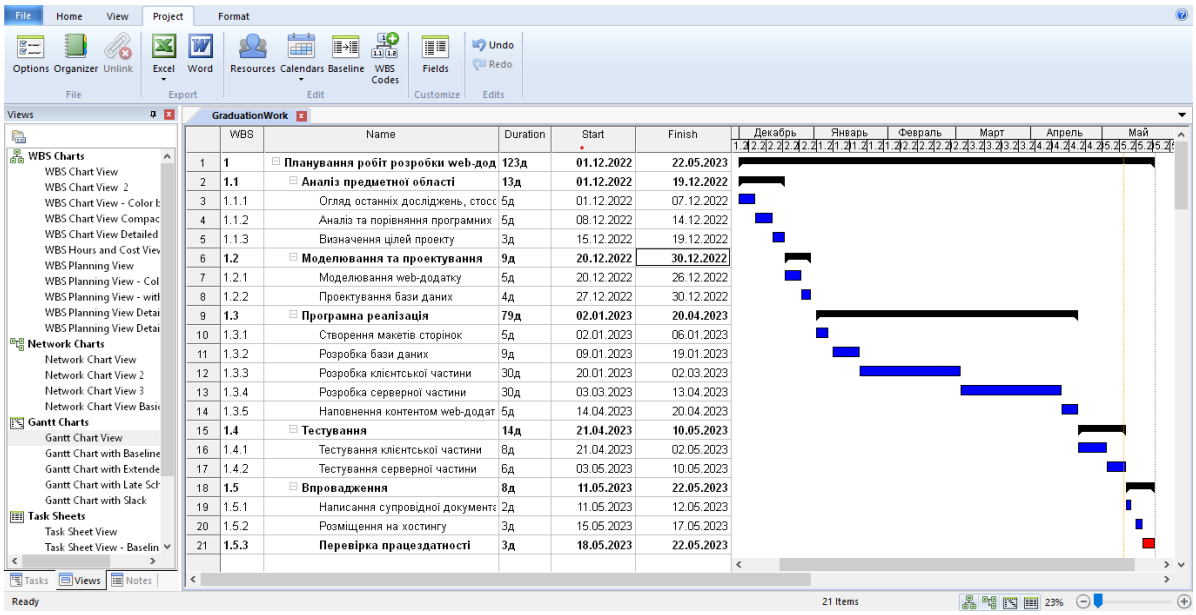


Рисунок Б.3 – Календарний план проєкту

Аналіз ризиків. Майже кожен проєкт рано чи пізно зіштовхується з своїми ризиками, які можуть по-різному в подальшому вплинути на отримані результати. Для ефективної роботи над проєктом, потрібно завчасно передбачити та оцінити можливі ризики, щоб відразу винайти їх методи вирішення. Ризики можна поділити за шкалою їх впливу, типом та ймовірністю виникнення, що можна побачити у таблиці Б.3.

Таблиця Б.3 – Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проєкту

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Ідентифікація ризиків проєкту. Для web-додатку підтримки дистанційного навчання було передбачено можливі ризики, що можуть надати негативного впливу. Ймовірні ризики описані в таблиці Б.4.

Таблиця Б.4 – Ризики проєкту

№ ризику	**Назва ризику**
1	Обрані не підходящі інструменти розробки
2	Постійні зміни ТЗ
3	Технічні помилки
4	Неправильний розподіл часу
5	Недостатня кількість знань та навичок для вирішення проблеми
6	Проблеми з ПЗ

Для повного розуміння оцінки ризиків, визначимо окремо кожний з них за впливом, ймовірністю виникнення та рангом. Всі дані описані в таблиці Б.5.

Таблиця Б.5 – Визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	**Назва ризику**	**Ймовірність (0.1-0.9)**	**Вплив (0.05-0.8)**	**Ранг**
1	Обрані не підходящі інструменти розробки	0.3	0.4	0.12
2	Постійні зміни ТЗ	0.5	0.2	0.1
3	Технічні помилки	0.9	0.05	0.045
4	Неправильний розподіл часу	0.7	0.4	0.28
5	Недостатня кількість знань та навичок для вирішення проблеми	0.1	0.1	0.01
6	Проблеми з ПЗ	0.1	0.8	0.08

На основі отриманих значень в таблиці Б.5 можна скласти матрицю ймовірності та впливу ризиків на проєкт, яка зображена в таблиці Б.6. Прийнятними ризиками можна вважати ті, які знаходяться в зелених комірках, виправдані – в жовтих і недопустимі в червоних.

Таблиця Б.6 – Матриця ймовірності та впливу ризиків

Ймовірність ризику	**Вплив загрози**				
	Дуже малий	**Малий**	**Середній**	**Великий**	**Дуже великий**
	0,05	0,1	0,2	0,4	0,8
0,9	R3(0,045)				
0,7				R4(0,28)	
0,5			R2(0,1)		
0,3				R1(0,12)	
0,1		R5(0,01)			R6(0,08)

Зважаючи на отримані значення індексів, ризики можна поділити за рівнями, які описуються в таблиці Б.7. Додатково було створено засоби по запобіганню створення ризиків і вирішення їх наслідків в разі потреби. Детально про стратегії реагування описано в таблиці Б.8.

Таблиця Б.7 – Шкала оцінювання за рівнем ризику

№	**Назва**	**Межі**	**Ризики, які входять (номера)**
1	Прийнятні	$0,005 \leq R \leq 0,05$	3, 5
2	Виправдані	$0,05 < R \leq 0,14$	6, 2, 1
3	Недопустимі	$0,14 < R \leq 0,72$	4

Таблиця Б.8 – Ризики та стратегії реагування

ID	**Статус**	**Опис**	**Ймовірність виникнення**	**Вплив**	**Ранг**	**План А (заходи запобігання виникненню ризику)**	**Тип стратегії реагування**	**План Б (заходи усунення наслідків ризику)**
RS_1	Відкритий	Обрані не підходящі інструменти розробки	0.3	0.4	0.12	Завчасно добре дослідити предметну область та інструменти написання продуктів-аналогів	Попередження	Зміна інструментів розробки
RS_2	Відкритий	Постійні зміни ТЗ	0.5	0.2	0.1	Провести повний аналіз предметної області та визначити чітке ТЗ	Попередження	Перепланування ТЗ

Продовження таблиці Б.8

ID	**Статус**	**Опис**	**Ймовірність виникнення**	**Вплив**	**Ранг**	**План А (заходи запобігання виникненню ризику)**	**Тип стратегії реагування**	**План Б (заходи усунення наслідків ризику)**
RS_3	Відкритий	Технічні помилки	0.9	0.05	0.045	Зберігати копії версій коду	Зменшення	Пошук і виправлення помилок
RS_4	Відкритий	Неправильний розподіл часу	0.7	0.4	0.28	Розрахувати час, включаючи всі фактори впливу	Попередження	Розрахувати новий час для роботи, включаючи нові фактори впливу

Продовження таблиці Б.8

ID	**Статус**	**Опис**	**Ймовірність виникнення**	**Вплив**	**Ранг**	**План А (заходи запобігання виникненню ризику)**	**Тип стратегії реагування**	**План Б (заходи усунення наслідків ризику)**
RS_5	Відкритий	Недостатня кількість знань та навичок для вирішення проблеми	0.1	0.1	0.01	Завчасно провести аналіз предметної області для визначення необхідних знань та навичок для роботи	Зменшення	Пошук необхідної інформації для вирішення проблеми
RS_6	Відкритий	Проблеми з ПЗ	0,1	0.8	0.08	Робити копії всіх даних і мати додаткове ПЗ	Попередження	Вирішити та усунути виявлені проблеми або знайти нове ПЗ

ДОДАТОК В

Лістинг програмного коду

В.1 Серверна частина

server.js:

```
import config from './config';
import express from 'express';
import fileUpload from 'express-fileupload';
import cors from 'cors';
import * as path from 'path';
import sequelize from './db';
import mainRoutes from './routes/mainRoutes';
import cookieParser from 'cookie-parser';
import ErrorMiddleware from './middlewares/ErrorMiddleware';

const app = express();

app.use(express.json());
app.use(cookieParser());
app.use(
  cors({
    origin: config.CLIENT_URL,
    credentials: true,
  }),
);
app.use(express.static(path.resolve('static')));
app.use(fileUpload({}));
app.use('/api', mainRoutes);
app.use(ErrorMiddleware);

const start = async () => {
  try {
    await sequelize.authenticate();
    await sequelize.sync();
    app.listen(config.PORT, () => {
      console.log(`App listening on port ${config.PORT}`);
    });
  } catch (error) {
    console.log(error.message);
  }
};

start();
```

db.js:

```
import { Sequelize } from 'sequelize';
import config from './config';

const sequelize = new Sequelize(config.DB_NAME, config.DB_USER, con-
fig.DB_PASSWORD, {
  dialect: 'postgres',
  host: config.DB_HOST,
  port: config.DB_PORT,
```

```
});

export default sequelize;
```

config.js:

```
const PORT = 3001;

const config = {
  PORT,
  DB_NAME: 'distance-learning-platform',
  DB_USER: 'postgres',
  DB_PASSWORD: 'admin',
  DB_HOST: 'localhost',
  DB_PORT: 5432,
  //To send an email to verify your account
  SMPT_HOST: 'smtp.gmail.com',
  SMPT_PORT: 587,
  SMPT_USER: 'katerinapodus@gmail.com',
  SMPT_PASSWORD: 'cokicreakmvudbpo',
  API_URL: 'http://localhost:' + PORT,
  CLIENT_URL: 'http://localhost:3000',
};

export default config;
```

Models.js:

```
import sequelize from '../db';
import { DataTypes } from 'sequelize';

export const Person = sequelize.define('person', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  email: { type: DataTypes.STRING, unique: true, allowNull: false },
  password: { type: DataTypes.STRING, allowNull: false },
  isActivated: {
    type: DataTypes.BOOLEAN,
    defaultValue: false,
  },
  activationLink: { type: DataTypes.STRING },
  fullName: { type: DataTypes.STRING, allowNull: false },
  birthday: { type: DataTypes.DATEONLY, allowNull: true },
  avatar: { type: DataTypes.STRING, allowNull: true },
});

export const Token = sequelize.define('token', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  refreshToken: { type: DataTypes.STRING, allowNull: false },
});

export const Institution = sequelize.define('institution', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, allowNull: true },
  description: { type: DataTypes.STRING, allowNull: false },
  image: { type: DataTypes.STRING, allowNull: true },
});

export const Group = sequelize.define('group', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, allowNull: false },
});
```

```
export const Lecture = sequelize.define('lecture', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
  description: { type: DataTypes.STRING, allowNull: false },
  numberByOrder: { type: DataTypes.INTEGER, allowNull: false },
  file: { type: DataTypes.STRING, allowNull: true },
});

export const Task = sequelize.define('task', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
  description: { type: DataTypes.STRING, allowNull: true },
  deadline: { type: DataTypes.DATEONLY, allowNull: false },
  numberByOrder: { type: DataTypes.INTEGER, allowNull: false },
});

export const Course = sequelize.define('course', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
  description: { type: DataTypes.STRING, allowNull: true },
  image: { type: DataTypes.STRING, allowNull: true },
});

export const File = sequelize.define('file', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  path: { type: DataTypes.STRING, unique: true },
});

export const Quiz = sequelize.define('quiz', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
  description: { type: DataTypes.STRING, allowNull: true },
  startDate: { type: DataTypes.DATEONLY, allowNull: true },
  endDate: { type: DataTypes.DATEONLY, allowNull: true },
  time: { type: DataTypes.DATE, allowNull: true },
  perQuestionScore: { type: DataTypes.INTEGER, allowNull: false },
  numberByOrder: { type: DataTypes.INTEGER, allowNull: false },
});

export const Question = sequelize.define('question', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  text: { type: DataTypes.STRING, allowNull: false },
});

export const Answer = sequelize.define('answer', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  text: { type: DataTypes.STRING, allowNull: false },
  isCorrect: { type: DataTypes.BOOLEAN, allowNull: false },
});

export const Work = sequelize.define('work', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  workFile: { type: DataTypes.STRING, allowNull: false },
});

export const Mark = sequelize.define('mark', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  value: { type: DataTypes.INTEGER, allowNull: false },
});

export const Reply = sequelize.define('reply', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
```

```
});

export const StudentInGroup = sequelize.define('student_in_group', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

export const GroupInCourse = sequelize.define('group_in_course', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

export const TeachersInCourse = sequelize.define('teachers_in_course', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

export const FileInLecture = sequelize.define('file_in_lecture', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

export const FileInTask = sequelize.define('file_in_task', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

Person.hasMany(Token);
Token.belongsTo(Person, {
  foreignKey: {
    allowNull: false,
  },
});

Person.hasMany(Institution);
Institution.belongsTo(Person);

Person.belongsToMany(Group, { through: StudentInGroup });
Group.belongsToMany(Person, { through: StudentInGroup });

Institution.hasMany(Course);
Course.belongsTo(Institution, {
  foreignKey: {
    allowNull: false,
  },
});

Course.hasMany(Lecture);
Lecture.belongsTo(Course, {
  foreignKey: {
    allowNull: false,
  },
});

Course.hasMany(Task);
Task.belongsTo(Course, {
  foreignKey: {
    allowNull: false,
  },
});

Course.hasMany(Quiz);
Quiz.belongsTo(Course, {
  foreignKey: {
    allowNull: false,
  },
});
```

```
Task.hasMany(Work);
Work.belongsTo(Task, {
  foreignKey: {
    allowNull: false,
  },
});

Person.hasMany(Work);
Work.belongsTo(Person, {
  foreignKey: {
    allowNull: false,
  },
});

Work.hasMany(Mark);
Mark.belongsTo(Work, {
  foreignKey: {
    allowNull: false,
  },
});

Person.hasMany(Mark);
Mark.belongsTo(Person, {
  foreignKey: {
    allowNull: false,
  },
});

Answer.hasMany(Reply);
Reply.belongsTo(Answer, {
  foreignKey: {
    allowNull: false,
  },
});

Person.hasMany(Reply);
Reply.belongsTo(Person, {
  foreignKey: {
    allowNull: false,
  },
});

Quiz.hasMany(Question);
Question.belongsTo(Quiz, {
  foreignKey: {
    allowNull: false,
  },
});

Question.hasMany(Answer, { onDelete: 'CASCADE' });
Answer.belongsTo(Question, {
  foreignKey: {
    allowNull: false,
  },
});

Group.belongsToMany(Course, { through: GroupInCourse });
Course.belongsToMany(Group, { through: GroupInCourse });

Person.belongsToMany(Course, { through: TeachersInCourse });
Course.belongsToMany(Person, { through: TeachersInCourse });

Lecture.belongsToMany(File, { through: FileInLecture });
```

```
File.belongsToMany(Lecture, { through: FileInLecture });

Task.belongsToMany(File, { through: FileInTask });
File.belongsToMany(Task, { through: FileInTask });
```

mainRoutes.js:

```
import { Router } from 'express';
import personRoutes from './personRoutes';
import institutionRoutes from './institutionRoutes';
import courseRoutes from './courseRoutes';
import groupRoutes from './groupRoutes';
import lectureRoutes from './lectureRoutes';
import taskRoutes from './taskRoutes';
import authRoutes from './authRoutes';
import AuthMiddleware from '../middlewares/AuthMiddleware';
import quizRoutes from './quizRoutes';
import questionRoutes from './questionRoutes';
import answerRoutes from './answerRoutes';
import markRoutes from './markRoutes';
import replyRoutes from './replyRoutes';

const mainRoutes = new Router();

mainRoutes.use('/auth', authRoutes);
mainRoutes.use('/person', AuthMiddleware, personRoutes);
mainRoutes.use('/institution', AuthMiddleware, institutionRoutes);
mainRoutes.use('/course', AuthMiddleware, courseRoutes);
mainRoutes.use('/group', AuthMiddleware, groupRoutes);
mainRoutes.use('/lecture', AuthMiddleware, lectureRoutes);
mainRoutes.use('/task', AuthMiddleware, taskRoutes);
mainRoutes.use('/quiz', AuthMiddleware, quizRoutes);
mainRoutes.use('/question', AuthMiddleware, questionRoutes);
mainRoutes.use('/answer', AuthMiddleware, answerRoutes);
mainRoutes.use('/mark', AuthMiddleware, markRoutes);
mainRoutes.use('/reply', AuthMiddleware, replyRoutes);

export default mainRoutes;
```

answerRoutes.js:

```
import { Router } from 'express';
import AnswerController from '../controllers/AnswerController';

const answerRoutes = new Router();
const answerController = new AnswerController();

answerRoutes.get('/', answerController.getAllAnswers);
answerRoutes.get('/:id', answerController.getAnswer);
answerRoutes.post('/', answerController.createAnswer);
answerRoutes.post('/reply', answerController.createReply);
answerRoutes.put('/:id', answerController.updateAnswer);
answerRoutes.delete('/:id', answerController.deleteAnswer);

export default answerRoutes;
```

authRoutes.js:

```
import { Router } from 'express';
import { body } from 'express-validator';
import AuthController from '../controllers/AuthController';
```

```
const authRoutes = new Router();
const authController = new AuthController();

authRoutes.post(
  '/registration',
  body('email').isEmail(),
  body('password').isLength({ min: 5, max: 15 }),
  authController.registration,
);
authRoutes.post('/login', authController.login);
authRoutes.post('/logout', authController.logout);
authRoutes.get('/refresh', authController.refresh);
authRoutes.get('/activate/:link', authController.activate);

export default authRoutes;
```

courseRoutes.js:

```
import { Router } from 'express';
import CourseController from '../controllers/CourseController';

const courseRoutes = new Router();
const courseController = new CourseController();

courseRoutes.get('/', courseController.getAllCourses);
courseRoutes.get('/:institutionId/all', courseController.getAllCoursesByInstitu-
tion);
courseRoutes.get('/:id/members', courseController.getAllMembersByCourse);
courseRoutes.get('/:id/teachers', courseController.getTeachersFromCourse);
courseRoutes.get('/:id', courseController.getCourse);
courseRoutes.get('/:id/groups', courseController.getGroupsInCourse);
courseRoutes.post('/', courseController.createCourse);
courseRoutes.post('/teachers', courseController.addTeachersToCourse);
courseRoutes.put('/teachers', courseController.updateTeachersInCourse);
courseRoutes.post('/groups', courseController.addGroupsToCourse);
courseRoutes.put('/groups', courseController.updateGroupsInCourse);
courseRoutes.put('/:id', courseController.updateCourse);
courseRoutes.delete('/:id', courseController.deleteCourse);

export default courseRoutes;
```

groupRoutes.js:

```
import { Router } from 'express';
import GroupController from '../controllers/GroupController';

const groupRoutes = new Router();
const groupController = new GroupController();

groupRoutes.get('/', groupController.getAllGroups);
groupRoutes.get('/:id/students', groupController.getStudentsInGroup);
groupRoutes.get('/:id', groupController.getGroup);
groupRoutes.post('/', groupController.createGroup);
groupRoutes.post('/students', groupController.createStudentsInGroup);
groupRoutes.put('/:id', groupController.updateGroup);
groupRoutes.delete('/:id', groupController.deleteGroup);

export default groupRoutes;
```

institutionRoutes.js:

```
import { Router } from 'express';
```

```
import InstitutionController from '../controllers/InstitutionController';

const institutionRoutes = new Router();
const institutionController = new InstitutionController();

institutionRoutes.get('/', institutionController.getAllInstitutions);
institutionRoutes.get('/own/:personId', institutionController.getAllInstitu-
tionsByPerson);
institutionRoutes.get('/:userId', institutionController.getAllInstitu-
tionsByUser);
institutionRoutes.get('/:id', institutionController.getInstitution);
institutionRoutes.post('/', institutionController.createInstitution);
institutionRoutes.put('/:id', institutionController.updateInstitution);
institutionRoutes.delete('/:id', institutionController.deleteInstitution);

export default institutionRoutes;
```

lectureRoutes.js:

```
import { Router } from 'express';
import LectureController from '../controllers/LectureController';

const lectureRoutes = new Router();
const lectureController = new LectureController();

lectureRoutes.get('/', lectureController.getAllLectures);
lectureRoutes.get('/course/:courseId', lectureController.getAllLecturesBy-
Course);
lectureRoutes.get('/:id', lectureController.getLecture);
lectureRoutes.post('/', lectureController.createLecture);
lectureRoutes.put('/:id', lectureController.updateLecture);
lectureRoutes.delete('/:id', lectureController.deleteLecture);

export default lectureRoutes;
```

markRoutes.js:

```
import { Router } from 'express';
import MarkController from '../controllers/MarkController';

const markRoutes = new Router();
const markController = new MarkController();

markRoutes.get('/', markController.getAllMarks);
markRoutes.get('/marks', markController.getMark);
markRoutes.post('/', markController.createMark);

export default markRoutes;
```

personRoutes.js:

```
import { Router } from 'express';
import PersonController from '../controllers/PersonController';

const personRoutes = new Router();
const personController = new PersonController();

personRoutes.get('/', personController.getAllPersons);
personRoutes.get('/:id', personController.getPerson);
personRoutes.put('/:id', personController.updatePerson);
personRoutes.delete('/:id', personController.deletePerson);
```

```
export default personRoutes;
```

questionRoutes.js:

```
import { Router } from 'express';
import QuestionController from '../controllers/QuestionController';

const questionRoutes = new Router();
const questionController = new QuestionController();

questionRoutes.get('/:quizId', questionController.getAllQuestions);
questionRoutes.get('/:id', questionController.getQuestion);
questionRoutes.post('/', questionController.createQuestion);
questionRoutes.put('/:id', questionController.updateQuestion);
questionRoutes.delete('/:id', questionController.deleteQuestion);

export default questionRoutes;
```

quizRoutes.js:

```
import { Router } from 'express';
import QuizController from '../controllers/QuizController';

const quizRoutes = new Router();
const quizController = new QuizController();

quizRoutes.get('/course/:courseId', quizController.getAllQuizzesByCourse);
quizRoutes.get('/course/:courseId/:personId/marks', quizController.getAllQuiz-
zesMarks);
quizRoutes.get('/:id', quizController.getQuiz);
quizRoutes.get('/:id/:personId/mark', quizController.getOneWithMark);
quizRoutes.post('/', quizController.createQuiz);
quizRoutes.post('/questions', quizController.createQuestionsInQuiz);
quizRoutes.put('/:id', quizController.updateQuiz);
quizRoutes.delete('/:id', quizController.deleteQuiz);

export default quizRoutes;
```

replyRoutes.js:

```
import { Router } from 'express';
import ReplyController from '../controllers/ReplyController';

const replyRoutes = new Router();
const replyController = new ReplyController();

replyRoutes.get('/', replyController.getAllReplies);
replyRoutes.get('/replies', replyController.getReply);
replyRoutes.post('/', replyController.createReply);

export default replyRoutes;
```

taskRoutes.js:

```
import { Router } from 'express';
import TaskController from '../controllers/TaskController';

const taskRoutes = new Router();
const taskController = new TaskController();

taskRoutes.get('/all/:personId', taskController.getAllTasks);
taskRoutes.get('/work', taskController.getAllWork);
```

```
taskRoutes.get('/course/:courseId', taskController.getAllTasksByCourse);
taskRoutes.get('/:id', taskController.getTask);
taskRoutes.get('/:id/:personId/works', taskController.getTaskWorks);
taskRoutes.get('/:id/files', taskController.getFilesInTask);
taskRoutes.post('/', taskController.createTask);
taskRoutes.post('/work', taskController.createWork);
taskRoutes.put('/:id', taskController.updateTask);
taskRoutes.delete('/:id', taskController.deleteTask);

export default taskRoutes;
```

AnswerController.js:

```
import AnswerService from '../services/AnswerService';

export default class AnswerController {
  constructor() {
    this.getAllAnswers = this.getAllAnswers.bind(this);
    this.getAnswer = this.getAnswer.bind(this);
    this.createAnswer = this.createAnswer.bind(this);
    this.updateAnswer = this.updateAnswer.bind(this);
    this.deleteAnswer = this.deleteAnswer.bind(this);
  }

  async getAllAnswers(req, res) {
    try {
      const Answers = await AnswerService.getAllAnswers();
      return res.json(Answers);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAnswer(req, res) {
    try {
      const answer = await AnswerService.getAnswer(req.params.id);
      return res.json(answer);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createAnswer(req, res) {
    try {
      const answer = await AnswerService.createAnswer(req.body);
      res.json(answer);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createReply(req, res) {
    try {
      const reply = await AnswerService.createReply(req.body);
      res.json(reply);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateAnswer(req, res) {
    try {
```

```
      const updatedAnswers = await AnswerService.updateAnswer(req.params.id,
req.body);
      return res.json(updatedAnswers);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteAnswer(req, res) {
    try {
      const answer = await AnswerService.deleteAnswer(req.params.id);
      return res.json(answer);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

AuthController.js:

```
import config from '../config';
import { validationResult } from 'express-validator';
import ApiError from '../exceptions/ApiError';
import AuthService from '../services/AuthService';

export default class AuthController {
  constructor() {
    this.registration = this.registration.bind(this);
    this.activate = this.activate.bind(this);
    this.login = this.login.bind(this);
    this.logout = this.logout.bind(this);
    this.refresh = this.refresh.bind(this);
  }

  async registration(req, res, next) {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        return next(ApiError.BadRequest('Validation error', errors.array()));
      }
      const userData = await AuthService.registration(req.body, req.files.ava-
tar);
      this.__addRefreshCookies(res, userData.refreshToken);
      return res.json(userData);
    } catch (error) {
      next(error);
    }
  }

  async activate(req, res, next) {
    try {
      const activationLink = req.params.link;
      await AuthService.activate(activationLink);
      return res.redirect(config.CLIENT_URL);
    } catch (error) {
      console.log(error);
      next(error);
    }
  }

  async login(req, res, next) {
    try {
```

```
      const { email, password } = req.body;
      const userData = await AuthService.login(email, password);
      this.__addRefreshCookies(res, userData.refreshToken);
      return res.json(userData);
    } catch (error) {
      next(error);
    }
  }

  async logout(req, res, next) {
    try {
      const { refreshToken } = req.cookies;
      const token = await AuthService.logout(refreshToken);
      res.clearCookie('refreshToken');
      return res.json(token);
    } catch (error) {
      next(error);
    }
  }

  async refresh(req, res, next) {
    try {
      const { refreshToken } = req.cookies;
      const userData = await AuthService.refresh(refreshToken);
      this.__addRefreshCookies(res, userData.refreshToken);
      return res.json(userData);
    } catch (error) {
      next(error);
    }
  }

  __addRefreshCookies(res, refreshToken) {
    res.cookie('refreshToken', refreshToken, {
      maxAge: 30 * 24 * 60 * 60 * 1000,
      httpOnly: true,
    });
  }
}
```

CourseController.js:

```
import CourseService from '../services/CourseService';

export default class CourseController {
  constructor() {
    this.getAllCourses = this.getAllCourses.bind(this);
    this.getCourse = this.getCourse.bind(this);
    this.createCourse = this.createCourse.bind(this);
    this.updateCourse = this.updateCourse.bind(this);
    this.deleteCourse = this.deleteCourse.bind(this);
  }

  async getAllCourses(req, res) {
    try {
      const courses = await CourseService.getAllCourses();
      return res.json(courses);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllCoursesByInstitution(req, res) {
```

```
    try {
      const courses = await CourseService.getAllCoursesByInstitu-
tion(req.params.institutionId);
      return res.json(courses);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllMembersByCourse(req, res) {
    try {
      const members = await CourseService.getAllMembersByCourse(req.params.id);
      return res.json(members);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getTeachersFromCourse(req, res) {
    try {
      const teachers = await CourseService.getTeachersFromCourse(req.params.id);
      return res.json(teachers);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getCourse(req, res) {
    try {
      const course = await CourseService.getCourse(req.params.id);
      return res.json(course);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getGroupsInCourse(req, res) {
    try {
      const groups = await CourseService.getGroupsInCourse(req.params.id);
      return res.json(groups);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createCourse(req, res, next) {
    try {
      const course = await CourseService.createCourse(req.body, req.files.im-
age);
      res.json(course);
    } catch (error) {
      next(error);
    }
  }
  async updateCourse(req, res) {
    try {
      const updatedCourse = await CourseService.updateCourse(req.params.id,
req.body);
      return res.json(updatedCourse);
    } catch (error) {
      res.status(500).json(error);
    }
  }
```

```
  async deleteCourse(req, res) {
    try {
      const course = await CourseService.deleteCourse(req.params.id);
      return res.json(course);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async addTeachersToCourse(req, res) {
    try {
      const { personId, courseId } = req.body;
      const teachers = await CourseService.addTeachersToCourse(personId,
courseId);
      return res.json(teachers);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateTeachersInCourse(req, res) {
    try {
      const { personId, courseId } = req.body;
      const teachers = await CourseService.updateTeachersInCourse(personId,
courseId);
      return res.json(teachers);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async addGroupsToCourse(req, res) {
    try {
      const { groupId, courseId } = req.body;
      const groups = await CourseService.addGroupsToCourse(groupId, courseId);
      return res.json(groups);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateGroupsInCourse(req, res) {
    try {
      const { groupId, courseId } = req.body;
      const groups = await CourseService.updateGroupsInCourse(groupId,
courseId);
      return res.json(groups);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

GroupController.js:

```
import GroupService from '../services/GroupService';

export default class GroupController {
  constructor() {
    this.getAllGroups = this.getAllGroups.bind(this);
    this.getGroup = this.getGroup.bind(this);
```

```
    this.createGroup = this.createGroup.bind(this);
    this.updateGroup = this.updateGroup.bind(this);
    this.deleteGroup = this.deleteGroup.bind(this);
  }

  async getAllGroups(req, res) {
    try {
      const groups = await GroupService.getAllGroups();
      return res.json(groups);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getGroup(req, res) {
    try {
      const group = await GroupService.getGroup(req.params.id);
      return res.json(group);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createGroup(req, res) {
    try {
      const group = await GroupService.createGroup(req.body);
      res.json(group);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateGroup(req, res) {
    try {
      const updatedGroup = await GroupService.updateGroup(req.body);
      return res.json(updatedGroup);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteGroup(req, res) {
    try {
      const group = await GroupService.deleteGroup(req.params.id);
      return res.json(group);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createStudentsInGroup(req, res) {
    try {
      const { personId, groupId } = req.body;
      const students = await GroupService.createStudentsInGroup(personId,
groupId);
      return res.json(students);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getStudentsInGroup(req, res) {
    try {
```

```
      const students  = await GroupService.getStudentsInGroup(req.params.id);
      return res.json(students);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

InstitutionController.js:

```
import InstitutionService from '../services/InstitutionService';

export default class InstitutionController {
  constructor() {
    this.getAllInstitutions = this.getAllInstitutions.bind(this);
    this.getInstitution = this.getInstitution.bind(this);
    this.createInstitution = this.createInstitution.bind(this);
    this.updateInstitution = this.updateInstitution.bind(this);
    this.deleteInstitution = this.deleteInstitution.bind(this);
  }

  async getAllInstitutions(req, res) {
    try {
      const institutions = await InstitutionService.getAllInstitutions();
      return res.json(institutions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllInstitutionsByPerson(req, res) {
    try {
      const institutions = await InstitutionService.getAllInstitutionsByPer-
son(req.params.personId);
      return res.json(institutions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllInstitutionsByUser(req, res) {
    try {
      const institutions = await InstitutionService.getAllInstitu-
tionsByUser(req.params.userId);
      return res.json(institutions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getInstitution(req, res) {
    try {
      const institution = await InstitutionService.getInstitu-
tion(req.params.id);
      return res.json(institution);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createInstitution(req, res, next) {
    try {
```

```
      const institution = await InstitutionService.createInstitution(req.body,
req.files.image, req.person.id);
      res.json(institution);
    } catch (error) {
      next(error);
    }
  }

  async updateInstitution(req, res) {
    try {
      const updatedInstitution = await InstitutionService.updateInstitu-
tion(req.body);
      return res.json(updatedInstitution);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteInstitution(req, res) {
    try {
      const institution = await InstitutionService.deleteInstitu-
tion(req.params.id);
      return res.json(institution);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

LectureController.js:

```
import LectureService from '../services/LectureService';

export default class LectureController {
  constructor() {
    this.getAllLectures = this.getAllLectures.bind(this);
    this.getLecture = this.getLecture.bind(this);
    this.createLecture = this.createLecture.bind(this);
    this.updateLecture = this.updateLecture.bind(this);
    this.deleteLecture = this.deleteLecture.bind(this);
  }

  async getAllLectures(req, res) {
    try {
      const Lectures = await LectureService.getAllLectures();
      return res.json(Lectures);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllLecturesByCourse(req, res) {
    try {
      const lectures = await LectureService.getAllLecturesBy-
Course(req.params.courseId, req.query.sort);
      return res.json(lectures);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getLecture(req, res) {
```

```
    try {
      const Lecture = await LectureService.getLecture(req.params.id);
      return res.json(Lecture);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createLecture(req, res) {
    try {
      const Lecture = await LectureService.createLecture(req.body);
      res.json(Lecture);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateLecture(req, res) {
    try {
      const updatedLecture = await LectureService.updateLecture(req.params.id,
req.body);
      return res.json(updatedLecture);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteLecture(req, res) {
    try {
      const Lecture = await LectureService.deleteLecture(req.params.id);
      return res.json(Lecture);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

MarkController.js:

```
import MarkService from '../services/MarkService';

export default class MarkController {
  constructor() {
    this.getAllMarks = this.getAllMarks.bind(this);
    this.getMark = this.getMark.bind(this);
    this.createMark = this.createMark.bind(this);
  }

  async getAllMarks(req, res) {
    try {
      const marks = await MarkService.getAllMarks();
      return res.json(marks);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getMark(req, res) {
    try {
      const mark = await MarkService.getMark(req.query.work, req.query.person);
      return res.json(mark);
    } catch (error) {
```

```
      res.status(500).json(error);
    }
  }

  async createMark(req, res) {
    try {
      const mark = await MarkService.createMark(req.body);
      res.json(mark);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

PersonController.js:

```
import PersonService from '../services/PersonService';

export default class PersonController {
  constructor() {
    this.getAllPersons = this.getAllPersons.bind(this);
    this.getPerson = this.getPerson.bind(this);
    this.updatePerson = this.updatePerson.bind(this);
    this.deletePerson = this.deletePerson.bind(this);
  }

  async getAllPersons(req, res, next) {
    try {
      const persons = await PersonService.getAllPersons();
      return res.json(persons);
    } catch (error) {
      next(error);
    }
  }

  async getPerson(req, res, next) {
    try {
      const person = await PersonService.getPerson(req.params.id);
      return res.json(person);
    } catch (error) {
      next(error);
    }
  }

  async updatePerson(req, res, next) {
    try {
      const updatedPerson = await PersonService.updatePerson(req.params.id,
req.body, req.files?.avatar);
      return res.json(updatedPerson);
    } catch (error) {
      next(error);
    }
  }

  async deletePerson(req, res, next) {
    try {
      const person = await PersonService.deletePerson(req.params.id);
      return res.json(person);
    } catch (error) {
      next(error);
    }
  }
```

```
}
```

QuestionController.js:

```
import QuestionService from '../services/QuestionService';

export default class QuestionController {
  constructor() {
    this.getAllQuestions = this.getAllQuestions.bind(this);
    this.getQuestion = this.getQuestion.bind(this);
    this.createQuestion = this.createQuestion.bind(this);
    this.updateQuestion = this.updateQuestion.bind(this);
    this.deleteQuestion = this.deleteQuestion.bind(this);
  }

  async getAllQuestions(req, res) {
    try {
      const questions = await QuestionService.getAllQues-
tions(req.params.quizId);
      return res.json(questions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getQuestion(req, res) {
    try {
      const question = await QuestionService.getQuestion(req.params.id);
      return res.json(question);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createQuestion(req, res) {
    try {
      const question = await QuestionService.createQuestion(req.body);
      res.json(question);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateQuestion(req, res) {
    try {
      const updatedQuestions = await QuestionService.updateQues-
tion(req.params.id, req.body);
      return res.json(updatedQuestions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteQuestion(req, res) {
    try {
      const question = await QuestionService.deleteQuestion(req.params.id);
      return res.json(question);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

QuizController.js:

```
import QuizService from '../services/QuizService';

export default class QuizController {
  constructor() {
    this.getQuiz = this.getQuiz.bind(this);
    this.createQuiz = this.createQuiz.bind(this);
    this.updateQuiz = this.updateQuiz.bind(this);
    this.deleteQuiz = this.deleteQuiz.bind(this);
  }

  async getAllQuizzesByCourse(req, res) {
    try {
      const quizzes = await QuizService.getAllQuizzesBy-
Course(req.params.courseId, req.query.sort);
      return res.json(quizzes);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllQuizzesMarks(req, res) {
    try {
      const quizzes = await QuizService.getAllQuizzesMarks(req.params.courseId,
req.params.personId);
      return res.json(quizzes);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getQuestionsInQuiz(req, res) {
    try {
      const questions = await QuizService.getQuestionsInQuiz(req.params.id);
      return res.json(questions);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getQuiz(req, res) {
    try {
      const quiz = await QuizService.getQuiz(req.params.id);
      return res.json(quiz);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getOneWithMark(req, res) {
    try {
      const quiz = await QuizService.getOneWithMark(req.params.id,
req.params.personId);
      return res.json(quiz);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createQuiz(req, res) {
```

```
    try {
      const quiz = await QuizService.createQuiz(req.body);
      res.json(quiz);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateQuiz(req, res) {
    try {
      const updatedQuiz = await QuizService.updateQuiz(req.params.id, req.body);
      return res.json(updatedQuiz);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteQuiz(req, res) {
    try {
      const quiz = await QuizService.deleteQuiz(req.params.id);
      return res.json(quiz);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createQuestionsInQuiz(req, res) {
    try {
      const { questionId, quizId } = req.body;
      const questions = await QuizService.createQuestionsInQuiz(questionId,
quizId);
      return res.json(questions);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

ReplyController.js:

```
import ReplyService from '../services/ReplyService';

export default class ReplyController {
  constructor() {
    this.getAllReplies = this.getAllReplies.bind(this);
    this.getReply = this.getReply.bind(this);
    this.createReply = this.createReply.bind(this);
  }

  async getAllReplies(req, res) {
    try {
      const replies = await ReplyService.getAllReplys();
      return res.json(replies);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getReply(req, res) {
    try {
      const reply = await ReplyService.getReply(req.query.answer, req.query.per-
son);
```

```
      return res.json(reply);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createReply(req, res) {
    try {
      const reply = await ReplyService.createReply(req.body);
      res.json(reply);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

TaskController.js:

```
import TaskService from '../services/TaskService';

export default class TaskController {
  constructor() {
    this.getAllTasks = this.getAllTasks.bind(this);
    this.getTask = this.getTask.bind(this);
    this.createTask = this.createTask.bind(this);
    this.updateTask = this.updateTask.bind(this);
    this.deleteTask = this.deleteTask.bind(this);
  }

  async getAllTasks(req, res) {
    try {
      const tasks = await TaskService.getAllTasks(req.params.personId);
      return res.json(tasks);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllTasksByCourse(req, res) {
    try {
      const tasks = await TaskService.getAllTasksByCourse(req.params.courseId,
req.query.sort);
      return res.json(tasks);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getTask(req, res) {
    try {
      const task = await TaskService.getTask(req.params.id);
      return res.json(task);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getTaskWorks(req, res) {
    try {
      const task = await TaskService.getTaskWorks(req.params.id, req.params.per-
sonId);
      return res.json(task);
```

```
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createTask(req, res) {
    try {
      const task = await TaskService.createTask(req.body);
      res.json(task);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async createWork(req, res) {
    try {
      const { taskId, personId } = req.body;
      const works = await TaskService.createWork(taskId, personId,
req.files.workFile);
      res.json(works);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async getAllWork(req, res) {
    try {
      const { taskId, personId } = req.body;
      const works = await TaskService.getAllWork(taskId, personId);
      return res.json(works);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async updateTask(req, res) {
    try {
      const updatedTask = await TaskService.updateTask(req.params.id, req.body);
      return res.json(updatedTask);
    } catch (error) {
      res.status(500).json(error);
    }
  }

  async deleteTask(req, res) {
    try {
      const Task = await TaskService.deleteTask(req.params.id);
      return res.json(Task);
    } catch (error) {
      res.status(500).json(error);
    }
  }
}
```

AnswerService.js:

```
import { Answer, Reply } from '../models/Models';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class AnswerService {
```

```
  async getAllAnswers() {
    return await Answer.findAll();
  }

  async getAnswer(id) {
    checkID(id);
    return await Answer.findOne({
      where: { id: id },
    });
  }

  async createAnswer(answer) {
    return await Answer.create(answer);
  }

  async createReply(reply) {
    return await Reply.create(reply);
  }

  async updateAnswer(id, answer) {
    checkID(id);
    delete answer.id;
    await Answer.update(answer, {
      where: {
        id: id,
      },
    });

    return await Answer.findOne({
      where: { id: id },
    });
  }

  async deleteAnswer(id) {
    checkID(id);
    return await Answer.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new AnswerService();
```

AuthService.js:

```
import bcrypt from 'bcrypt';
import * as uuid from 'uuid';
import { Person } from '../models/Models';
import FileService from './FileService';
import MailService from './MailService';
import TokenService from './TokenService';
import config from '../config';
import ApiError from '../exceptions/ApiError';

class AuthService {
  async registration(person, avatar) {
    const personAlreadyExists = await Person.findOne({
      where: { email: person.email },
    });
    if (personAlreadyExists) {
```

```
      throw ApiError.BadRequest(`The user with email "${person.email}" already
exists in the database`);
    }

    const hashPassword = await bcrypt.hash(person.password, 3);
    const activationLink = uuid.v4();
    const fileName = FileService.saveFile(avatar);
    person.password = hashPassword;

    const user = await Person.create({
      ...person,
      avatar: fileName,
      activationLink,
    });

    await MailService.sendActivationMail(user.email, config.API_URL +
'/api/auth/activate/' + activationLink);

    return this.createTokens(user);
  }

  async activate(activationLink) {
    const user = await Person.findOne({
      where: { activationLink: activationLink },
    });
    if (!user) {
      throw ApiError.BadRequest('Incorrect link for account activation');
    }
    user.isActivated = true;
    await user.save();
  }

  async login(email, password) {
    const user = await Person.findOne({
      where: { email: email },
    });
    if (!user) {
      throw ApiError.BadRequest(`User with this ${email} not found`);
    }
    const isPassEquals = await bcrypt.compare(password, user.password);
    if (!isPassEquals) {
      throw ApiError.BadRequest('Incorrect password');
    }

    return this.createTokens(user);
  }

  async logout(refreshToken) {
    const token = await TokenService.removeToken(refreshToken);
    return token;
  }

  async refresh(refreshToken) {
    if (!refreshToken) {
      throw ApiError.UnauthorizedError();
    }
    const userData = TokenService.validateRefreshToken(refreshToken);
    const tokenFromDB = await TokenService.findToken(refreshToken);
    if (!userData || !tokenFromDB) {
      throw ApiError.UnauthorizedError();
    }
    const user = await Person.findOne({
      where: { id: userData.id },
```

```
    });

    return this.createTokens(user);
  }

  makePersonDto({ id, fullName, email, avatar, birthday, isActivated, createdAt
}) {
    return { id, fullName, email, avatar, birthday, isActivated, createdAt };
  }

  async createTokens(user) {
    const { id, email, isActivated } = user;
    const tokens = TokenService.generateTokens({ id, email, isActivated });
    await TokenService.saveToken(id, tokens.refreshToken);
    return {
      ...tokens,
      user: this.makePersonDto(user),
    };
  }
}

export default new AuthService();
```

CourseService.js:

```
import { Course, GroupInCourse } from '../models/Models';
import FileService from './FileService';
import sequelize from '../db';
import { QueryTypes } from 'sequelize';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class CourseService {
  async getAllCourses() {
    return await Course.findAll();
  }

  async getAllCoursesByInstitution(institutionId) {
    return await Course.findAll({
      where: { institutionId: institutionId },
    });
  }

  async getAllMembersByCourse(id) {
    return await sequelize.query(
      `SELECT people.* FROM people JOIN student_in_groups ON people.id = stu-
dent_in_groups."personId" JOIN group_in_courses ON group_in_courses."groupId" =
student_in_groups."groupId" WHERE group_in_courses."courseId" = ${id}`,
      { type: QueryTypes.SELECT },
    );
  }

  async getCourse(id) {
    checkID(id);
    return await Course.findOne({
      where: { id: id },
    });
  }

  async getGroupsInCourse(id) {
```

```
    checkID(id);
    return await GroupInCourse.findAll({
      where: { courseId: id },
    });
  }

  async createCourse(course, image) {
    const fileName = FileService.saveFile(image);
    return await Course.create({
      ...course,
      image: fileName,
    });
  }

  async updateCourse(id, course) {
    checkID(id);
    delete course.id;
    await Course.update(course, {
      where: {
        id: id,
      },
    });

    return await Course.findOne({
      where: { id: id },
    });
  }

  async deleteCourse(id) {
    checkID(id);
    return await Course.destroy({
      where: {
        id: id,
      },
    });
  }

  async addTeachersToCourse(personId, courseId) {
    return await sequelize.query(
      `INSERT INTO teachers_in_courses("createdAt", "updatedAt", "personId",
"courseId") VALUES(NOW(), NOW(), ${personId}, ${courseId})`,
      { type: QueryTypes.INSERT },
    );
  }

  async updateTeachersInCourse(personId, courseId) {
    return await sequelize.query(
      `UPDATE teachers_in_courses SET "personId"=${personId} WHERE teach-
ers_in_courses."courseId" = ${courseId}`,
      { type: QueryTypes.UPDATE },
    );
  }

  async getTeachersFromCourse(courseId) {
    return await sequelize.query(
      `SELECT teachers_in_courses.* FROM teachers_in_courses JOIN courses ON
teachers_in_courses."courseId" = courses."id" WHERE courses."id" = ${courseId}`,
      { type: QueryTypes.SELECT },
    );
  }

  async addGroupsToCourse(groupId, courseId) {
    return await sequelize.query(
```

```
      `INSERT INTO group_in_courses("createdAt", "updatedAt", "groupId",
"courseId") VALUES(NOW(), NOW(), ${groupId}, ${courseId})`,
      { type: QueryTypes.INSERT },
    );
  }

  async updateGroupsInCourse(groupId, courseId) {
    return await sequelize.query(
      `UPDATE group_in_courses SET "groupId"=${groupId} WHERE
group_in_courses."courseId" = ${courseId}`,
      { type: QueryTypes.UPDATE },
    );
  }
}

export default new CourseService();
```

GroupService.js:

```
import sequelize from '../db';
import { QueryTypes } from 'sequelize';
import { Group } from '../models/Models';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class GroupService {
  async getAllGroups() {
    return await Group.findAll();
  }

  async getGroup(id) {
    checkID(id);
    return await Group.findOne({
      where: { id: id },
    });
  }

  async createGroup(group) {
    return await Group.create(group);
  }

  async updateGroup(group) {
    checkID(group.id);
    return await Group.update(group, {
      where: {
        id: group.id,
      },
    });
  }

  async deleteGroup(id) {
    checkID(id);
    return await Group.destroy({
      where: {
        id: id,
      },
    });
  }

  async createStudentsInGroup(personId, groupId) {
```

```
    return await sequelize.query(
      `INSERT INTO student_in_groups("createdAt", "updatedAt", "personId",
"groupId") VALUES(NOW(), NOW(), ${personId}, ${groupId});`,
      { type: QueryTypes.POST },
    );
  }

  async getStudentsInGroup(id) {
    checkID(id);
    return await sequelize.query(
      `SELECT people.id, people.email, people."fullName", people.birthday, peo-
ple.avatar FROM people JOIN student_in_groups ON people.id = stu-
dent_in_groups."personId" WHERE "groupId" = ${id}`,
      { type: QueryTypes.SELECT },
    );
  }
}

export default new GroupService();
```

InstitutionService.js:

```
import { QueryTypes } from 'sequelize';
import sequelize from '../db';
import { Institution } from '../models/Models';
import FileService from './FileService';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class InstitutionService {
  async getAllInstitutions() {
    return await Institution.findAll();
  }

  async getAllInstitutionsByPerson(personId) {
    checkID(personId);
    return await Institution.findAll({
      where: { personId: personId },
    });
  }

  async getAllInstitutionsByUser(userId) {
    checkID(userId);
    return await sequelize.query(
      `SELECT distinct institutions.* FROM institutions JOIN courses ON
courses."institutionId" = institutions."id" JOIN group_in_courses ON courses.id
= group_in_courses."courseId" JOIN student_in_groups ON stu-
dent_in_groups."groupId" = group_in_courses."groupId" WHERE stu-
dent_in_groups."personId" = ${userId}`,
      { type: QueryTypes.SELECT },
    );
  }

  async getInstitution(id) {
    checkID(id);
    return await Institution.findOne({
      where: { id: id },
    });
  }
```

```
  async createInstitution(institution, image, id) {
    const fileName = FileService.saveFile(image);
    return await Institution.create({ ...institution, image: fileName, personId:
id });
  }

  async updateInstitution(institution) {
    checkID(institution.id);
    return await Institution.update(institution, {
      where: {
        id: institution.id,
      },
    });
  }

  async deleteInstitution(id) {
    checkID(id);
    return await Institution.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new InstitutionService();
```

LectureService.js:

```
import { Lecture } from '../models/Models';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class LectureService {
  async getAllLectures() {
    return await Lecture.findAll();
  }

  async getAllLecturesByCourse(courseId, sort) {
    let order;

    switch (sort) {
    case 'old':
      order = [['createdAt', 'ASC']];
      break;
    case 'new':
      order = [['createdAt', 'DESC']];
      break;
    case 'numberFromFirst':
      order = [['numberByOrder', 'ASC']];
      break;
    case 'numberFromLast':
      order = [['numberByOrder', 'DESC']];
      break;
    default:
      break;
    }

    const options = { where: { courseId: courseId } };
```

```
    if (sort) {
      options.order = order;
    }

    return await Lecture.findAll(options);
  }

  async getLecture(id) {
    checkID(id);
    return await Lecture.findOne({
      where: { id: id },
    });
  }

  async createLecture(lecture) {
    return await Lecture.create(lecture);
  }

  async updateLecture(id, lecture) {
    checkID(id);
    delete lecture.id;
    await Lecture.update(lecture, {
      where: {
        id: id,
      },
    });

    return await Lecture.findOne({
      where: { id: id },
    });
  }

  async deleteLecture(id) {
    checkID(id);
    return await Lecture.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new LectureService();
```

MailService.js:

```
import nodemailer from 'nodemailer';
import config from '../config';

class MailService {
  constructor() {
    this.transporter = nodemailer.createTransport({
      host: config.SMPT_HOST,
      port: config.SMPT_PORT,
      secure: false,
      auth: {
        user: config.SMPT_USER,
        pass: config.SMPT_PASSWORD,
      },
    });
  }
```

```
  async sendActivationMail(to, link) {
    await this.transporter.sendMail({
      from: config.SMPT_USER,
      to,
      subject: 'Account activation + ' + config.API_URL,
      text: '',
      html: `<div>
                <h1>To activate your account, follow the link:</h1>
                <a href=${link}>${link}</a>
            </div>`,
    });
  }
}

export default new MailService();
```

MarkService.js:

```
import { Mark } from '../models/Models';

class MarkService {
  async getAllMarks() {
    return await Mark.findAll();
  }

  async getMark(work, person) {
    return await Mark.findOne({
      where: { workId: work, personId: person },
    });
  }

  async createMark(mark) {
    return await Mark.create(mark);
  }
}

export default new MarkService();
```

PersonService.js:

```
import { Person } from '../models/Models';
import FileService from './FileService';
import ApiError from '../exceptions/ApiError';

function checkID(id) {
  if (!id) throw ApiError.BadRequest('No id specified');
}

class PersonService {
  async getAllPersons() {
    return await Person.findAll();
  }

  async getPerson(id) {
    checkID(id);
    return await Person.findOne({
      where: { id: id },
    });
  }

  async updatePerson(id, person, avatar) {
    checkID(id);
```

```
    delete person.id;
    delete person.avatar;
    if (avatar) {
      person.avatar = FileService.saveFile(avatar);
    }
    await Person.update(person, {
      where: {
        id: id,
      },
    });

    return await Person.findOne({
      where: { id: id },
    });
  }

  async deletePerson(id) {
    checkID(id);
    return await Person.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new PersonService();
```

QuestionService.js:

```
import { Answer, Question } from '../models/Models';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class QuestionService {
  async getAllQuestions(quizId) {
    return await Question.findAll({
      where: { quizId: quizId },
      include: Answer,
    });
  }

  async getQuestion(id) {
    checkID(id);
    return await Question.findOne({
      where: { id: id },
    });
  }

  async createQuestion(question) {
    return await Question.create(question, {
      include: [Answer],
    });
  }

  async updateQuestion(id, question) {
    checkID(id);
    delete question.id;
    await Question.update(question, {
      where: {
```

```
        id: id,
      },
    });

    return await Question.findOne({
      where: { id: id },
    });
  }

  async deleteQuestion(id) {
    checkID(id);
    return await Question.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new QuestionService();
```

QuizService.js:

```
import { Answer, Question, Quiz, Reply } from '../models/Models';
import sequelize from '../db';
import { QueryTypes } from 'sequelize';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class QuizService {
  async getAllQuizzesByCourse(courseId, sort) {
    let order;

    switch (sort) {
    case 'deadline':
      order = [['deadline', 'ASC']];
      break;
    case 'old':
      order = [['createdAt', 'ASC']];
      break;
    case 'new':
      order = [['createdAt', 'DESC']];
      break;
    case 'numberFromFirst':
      order = [['numberByOrder', 'ASC']];
      break;
    case 'numberFromLast':
      order = [['numberByOrder', 'DESC']];
      break;
    default:
      break;
    }

    const options = {
      where: { courseId: courseId },
    };

    if (sort) {
      options.order = order;
    }
```

```
    return await Quiz.findAll(options);
  }

  async getAllQuizzesMarks(courseId, personId) {
    const options = {
      where: { courseId: courseId },
      group: ['quiz.id', 'personId'],
      attributes: [
        'title',
        'startDate',
        'endDate',
        [sequelize.literal('quiz."perQuestionScore" * count("questions->an-
swers"."isCorrect")'), 'mark'], // Custom field: age * 2
      ],
      include: [
        {
          model: Question,
          attributes: [],
          include: [
            {
              model: Answer,
              required: true,
              attributes: [],
              include: { model: Reply, attributes: [], required: true, where:
{personId: personId} },
            },
          ],
        },
      ],
    };
    return await Quiz.findAll(options);
  }

  async getQuiz(id) {
    checkID(id);
    return await Quiz.findOne({where: { id: id }});
  }

  async getOneWithMark(id, personId) {
    checkID(id);
    const options = {
      where: { id: id },
      group: ['quiz.id', 'personId'],
      attributes: [
        'title',
        'description',
        'startDate',
        'endDate',
        'time',
        'perQuestionScore',
        'numberByOrder',
        'createdAt',
        'updatedAt',
        'courseId',
        [sequelize.literal('quiz."perQuestionScore" * count("questions->an-
swers"."isCorrect")'), 'mark'], // Custom field: age * 2
      ],
      include: [
        {
          model: Question,
          attributes: [],
          include: [
```

```
            {
              model: Answer,
              required: true,
              attributes: [],
              include: { model: Reply, attributes: [], required: true, where:
{personId: personId} },
            },
          ],
        },
      ],
    };
    return await Quiz.findAll(options);
  }

  async createQuiz(quiz) {
    return await Quiz.create(quiz);
  }

  async updateQuiz(id, quiz) {
    checkID(id);
    delete quiz.id;
    await Quiz.update(quiz, {
      where: {
        id: id,
      },
    });

    return await Quiz.findOne({
      where: { id: id },
    });
  }

  async deleteQuiz(id) {
    checkID(id);
    return await Quiz.destroy({
      where: {
        id: id,
      },
    });
  }

  async createQuestionsInQuiz(questionId, quizId) {
    console.log(questionId, quizId);
    return await sequelize.query(
      `INSERT INTO question_in_quizzes("createdAt", "updatedAt", "questionId",
"quizId") VALUES(NOW(), NOW(), ${questionId}, ${quizId})`,
      { type: QueryTypes.POST },
    );
  }

  async getQuestionsInQuiz(id) {
    checkID(id);
    return await sequelize.query(
      `SELECT questions.* FROM questions JOIN question_in_quizzes ON ques-
tions.id = question_in_quizzes."questionId" WHERE "quizId" = ${id}`,
      { type: QueryTypes.SELECT },
    );
  }
}

export default new QuizService();
```

ReplyService.js:

```
import { Reply } from '../models/Models';

class ReplyService {
  async getAllReplies() {
    return await Reply.findAll();
  }

  async getReply(answer, person) {
    return await Reply.findOne({
      where: { answerId: answer, personId: person },
    });
  }

  async createReply(reply) {
    return await Reply.create(reply);
  }
}

export default new ReplyService();
```

TaskService.js:

```
import { Task, Work } from '../models/Models';
import FileService from './FileService';

function checkID(id) {
  if (!id) throw new Error('No id specified');
}

class TaskService {
  async getAllTasks(personId) {
    return await Task.findAll({
      include: {
        model: Work,
        where: {
          personId: personId,
        },
      },
      order: [[Work, 'createdAt', 'DESC']],
    });
  }

  async getAllTasksByCourse(courseId, sort) {
    let order;

    switch (sort) {
    case 'deadline':
      order = [['deadline', 'ASC']];
      break;
    case 'old':
      order = [['createdAt', 'ASC']];
      break;
    case 'new':
      order = [['createdAt', 'DESC']];
      break;
    case 'numberFromFirst':
      order = [['numberByOrder', 'ASC']];
      break;
    case 'numberFromLast':
      order = [['numberByOrder', 'DESC']];
```

```
      break;
    default:
      break;
    }

    const options = { where: { courseId: courseId } };

    if (sort) {
      options.order = order;
    }

    return await Task.findAll(options);
  }

  async getTask(id) {
    checkID(id);
    return await Task.findOne({
      where: { id: id },
    });
  }

  async getTaskWorks(id, personId) {
    checkID(id);
    return await Task.findOne({
      where: { id: id },
      include: {
        model: Work,
        where: {
          personId: personId,
        },
      },
      order: [[Work, 'createdAt', 'DESC']],
    });
  }

  async createTask(task) {
    return await Task.create(task);
  }

  async createWork(taskId, personId, workFile) {
    const fileName = FileService.saveFile(workFile);
    return await Work.create({ personId: personId, taskId: taskId, workFile:
fileName });
  }

  async getAllWork(taskId, personId) {
    return await Work.findAll({ where: { personId: personId, taskId: taskId }
});
  }

  async updateTask(id, task) {
    checkID(id);
    delete task.id;
    await Task.update(task, {
      where: {
        id: id,
      },
    });

    return await Task.findOne({
      where: { id: id },
    });
  }
```

```
  async deleteTask(id) {
    checkID(id);
    return await Task.destroy({
      where: {
        id: id,
      },
    });
  }
}

export default new TaskService();
```

TokenService.js:

```
import jwt from 'jsonwebtoken';
const { sign, verify } = jwt;
import { Token } from '../models/Models';
import { JWT_ACCESS_SECRET_KEY, JWT_REFRESH_SECRET_KEY } from '../consts/jwtCon-
sts';

class TokenService {
  generateTokens(payload) {
    const accessToken = sign(payload, JWT_ACCESS_SECRET_KEY, {
      expiresIn: '10m',
    });
    const refreshToken = sign(payload, JWT_REFRESH_SECRET_KEY, {
      expiresIn: '30d',
    });
    return {
      accessToken,
      refreshToken,
    };
  }

  validateAccessToken(token) {
    try {
      const userData = verify(token, JWT_ACCESS_SECRET_KEY);
      return userData;
    } catch (error) {
      return null;
    }
  }

  validateRefreshToken(token) {
    try {
      const userData = verify(token, JWT_REFRESH_SECRET_KEY);
      return userData;
    } catch (error) {
      return null;
    }
  }

  async saveToken(userId, refreshToken) {
    const tokenData = await Token.findOne({
      where: { personId: userId },
    });
    if (tokenData) {
      tokenData.refreshToken = refreshToken;
      return tokenData.save();
    }
    return await Token.create({ personId: userId, refreshToken });
```

```
  }

  async removeToken(refreshToken) {
    return await Token.destroy({
      where: {
        refreshToken: refreshToken,
      },
    });
  }

  async findToken(refreshToken) {
    return await Token.findOne({
      where: {
        refreshToken: refreshToken,
      },
    });
  }
}

export default new TokenService();
```

jwtConsts.js:

```
export const JWT_ACCESS_SECRET_KEY = 'jwt-access-secret-key';
export const JWT_REFRESH_SECRET_KEY = 'jwt-refresh-secret-key';
```

ApiError.js:

```
export default class ApiError extends Error {
  status;
  errors;

  constructor(status, message, errors = []) {
    super(message);
    this.status = status;
    this.errors = errors;
  }

  static UnauthorizedError() {
    return new ApiError(401, 'User not authorized');
  }

  static BadRequest(message, errors = []) {
    return new ApiError(400, message, errors);
  }
}
```

AuthMiddleware.js:

```
import ApiError from '../exceptions/ApiError';
import TokenService from '../services/TokenService';

export default function (req, res, next) {
  try {
    const authoHeader = req.headers.authorization;
    if (!authoHeader) {
      return next(ApiError.UnauthorizedError());
    }
    const accessToken = authoHeader.split(' ')[1];
    if (!accessToken) {
      return next(ApiError.UnauthorizedError());
    }
```

```
    const userData = TokenService.validateAccessToken(accessToken);

    if (!userData) {
      return next(ApiError.UnauthorizedError());
    }
    req.person = userData;
    next();
  } catch (error) {
    return next(ApiError.UnauthorizedError());
  }
}
```

ErrorMiddleware.js:

```
import ApiError from '../exceptions/ApiError';

export default function (err, req, res, next) {
  if (err instanceof ApiError) {
    return res.status(err.status).json({ message: err.message, errors: err.er-
rors });
  }
  return res.status(500).json({ message: err.message || 'Unexpected error' });
}
```

B.2 Клієнтська частина

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
);

reportWebVitals();
```

App.js:

```
import './App.css';
import Navbar from './components/Navbar/Navbar';
import AuthWrapper from './components/Authorization/AuthWrapper';
import ThemeWrapper from './components/Theme/ThemeWrapper';
import { Router } from './components/Shared/Router/Router';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
import { LocalizationProvider } from '@mui/x-date-pickers/LocalizationProvider';

function App() {
  return (
    <div className="App">
      <LocalizationProvider dateAdapter={AdapterDayjs}>
        <ThemeWrapper>
          <AuthWrapper>
```

```
            <Navbar>
              <Router />
            </Navbar>
          </AuthWrapper>
        </ThemeWrapper>
      </LocalizationProvider>
    </div>
  );
}

export default App;
```

Api.js:

```
import axios from "axios";
import configClient from "../configClient";

const $api = axios.create({
  withCredentials: true,
  baseURL: configClient.API_URL + "/api/",
});

$api.interceptors.request.use((config) => {
  config.headers.Authorization = `Bearer ${localStorage.getItem(
    "access-token"
  )}`;
  return config;
});

$api.interceptors.response.use(
  (config) => {
    return config;
  },
  async (error) => {
    const originalRequest = error.config;
    if (
      error.response.status === 401 &&
      error.config &&
      !error.config._isRetry
    ) {
      originalRequest._isRetry = true;
      try {
        const response = await axios.get(
          configClient.API_URL + "/api/auth/refresh",
          { withCredentials: true }
        );
        localStorage.setItem("access-token", response.data.accessToken);
        return $api.request(originalRequest);
      } catch (error) {
        console.log("Not authorized");
      }
    }
    throw error;
  }
);

export default $api;
```

AnswerService.js:

```
import $api from '../http/Api';
```

```
export default class AnswerService {
  static async create(answer) {
    const response = await $api.post('answer/', answer);
    return response.data;
  }

  static async createReply(answerId, personId) {
    const response = await $api.post('question/reply', { answerId, personId });
    return response.data;
  }
}
```

AuthService.js:

```
import $api from '../http/Api';

export default class AuthService {
  static async login(email, password) {
    return $api.post('auth/login', { email, password });
  }

  static async registration(person) {
    return $api.post('auth/registration', person);
  }

  static async logout() {
    return $api.post('auth/logout');
  }
}
```

CourseService.js:

```
import $api from '../http/Api';

export default class CourseService {
  static async getAll() {
    const response = await $api.get('course/');
    return response.data;
  }

  static async getTeachersFromCourse(courseId) {
    const response = await $api.get(`course/${courseId}/teachers`);
    return response.data;
  }

  static async getAllCoursesByInstitution(institutionId) {
    const response = await $api.get(`course/${institutionId}/all`);
    return response.data;
  }

  static async getAllMembersByCourse(courseId) {
    const response = await $api.get(`course/${courseId}/members`);
    return response.data;
  }

  static async getOne(courseId) {
    const response = await $api.get(`course/${courseId}`);
    return response.data;
  }

  static async getGroupsInCourse(courseId) {
    const response = await $api.get(`course/${courseId}/groups`);
```

```
    return response.data;
  }

  static async create(course) {
    return $api.post('course/', course);
  }

  static async update(courseId, newCourse) {
    return $api.put(`course/${courseId}`, newCourse);
  }

  static async addTeachersToCourse(personId, courseId) {
    return $api.post('course/teachers', { personId, courseId });
  }

  static async updateTeachersInCourse(personId, courseId) {
    return $api.put('course/teachers', { personId, courseId });
  }

  static async addGroupsToCourse(groupId, courseId) {
    return $api.post('course/groups', { groupId, courseId });
  }

  static async updateGroupsInCourse(groupId, courseId) {
    return $api.put('course/groups', { groupId, courseId });
  }

  static async delete(courseId) {
    return $api.post(`course/${courseId}`);
  }

  static async getTeacherAccess(personId, courseId) {
    return (await this.getTeachersFromCourse(courseId)).some((teacher) =>
teacher.personId === personId);
  }
}
```

FileService.js:

```
import $api from '../http/Api';

export default class FileService {
  static async downloadFile(file) {
    return $api.get(`file/${file}`, { responseType: 'blob' });
  }
}
```

GroupService.js:

```
import $api from '../http/Api';

export default class GroupService {
  static async getAll() {
    const response = await $api.get('group/');
    return response.data;
  }

  static async getOne(groupId) {
    const response = await $api.get(`group/${groupId}`);
    return response.data;
  }
```

```
  static async getStudentsInGroup(groupId) {
    const response = await $api.get(`group/${groupId}/students`);
    return response.data;
  }

  static async create(group) {
    return $api.post('group/', group);
  }

  static async createStudentsInGroup(personId, groupId) {
    return $api.post('group/students', { personId, groupId });
  }
}
```

InstitutionService.js:

```
import $api from '../http/Api';

export default class InstitutionService {
  static async getAll() {
    const response = await $api.get('institution/');
    return response.data;
  }

  static async getAllByPerson(personId) {
    const response = await $api.get(`institution/own/${personId}`);
    return response.data;
  }

  static async getAllByUser(userId) {
    const response = await $api.get(`institution/${userId}`);
    return response.data;
  }

  static async create(institution) {
    return $api.post('institution/', institution);
  }
}
```

LectureService.js:

```
import $api from '../http/Api';

export default class LectureService {
  static async getAllByCourse(courseId, sort) {
    const response = await $api.get(`lecture/course/${courseId}${sort ?
`?sort=${sort}` : ''}`);
    return response.data;
  }

  static async getOne(lectureId) {
    const response = await $api.get(`lecture/${lectureId}`);
    return response.data;
  }

  static async create(lecture) {
    const response = await $api.post('lecture/', lecture);
    return response.data;
  }

  static async update(id, lecture) {
    const response = await $api.put(`lecture/${id}`, lecture);
```

```
    return response.data;
  }

  static async delete(id) {
    const response = await $api.delete(`lecture/${id}`);
    return response.data;
  }
}
```

MarkService.js:

```
import $api from '../http/Api';

export default class MarkService {
  static async getAll() {
    const response = await $api.get(`mark/`);
    return response.data;
  }

  static async getOne(workId, personId) {
    const response = await $api.get(`mark/marks?work=${workId}&person=${per-
sonId}`);
    return response.data;
  }

  static async create(mark) {
    const response = await $api.post('mark/', mark);
    return response.data;
  }
}
```

QuestionService.js:

```
import $api from '../http/Api';

export default class QuestionService {
  static async getAll(quizId) {
    const response = await $api.get(`question/${quizId}`);
    return response.data;
  }

  static async create(question) {
    const response = await $api.post('question/', question);
    return response.data;
  }

  static async delete(id) {
    const response = await $api.delete(`question/${id}`);
    return response.data;
  }
}
```

QuizService.js:

```
import $api from '../http/Api';

export default class QuizService {
  static async getAllByCourse(quizId, sort) {
    const response = await $api.get(`quiz/course/${quizId}${sort ?
`?sort=${sort}` : ''}`);
    return response.data;
  }
```

```
  static async getAllQuizzesMarks(quizId, personId) {
    const response = await $api.get(`quiz/course/${quizId}/${personId}/marks`);
    return response.data;
  }

  static async getOne(id) {
    const response = await $api.get(`quiz/${id}`);
    return response.data;
  }

  static async getOneWithMark(id, personId) {
    const response = await $api.get(`quiz/${id}/${personId}/mark`);
    return response.data;
  }

  static async getQuestionsInQuiz(id) {
    const response = await $api.get(`quiz/${id}/questions`);
    return response.data;
  }

  static async createQuestionsInQuiz(questionId, quizId) {
    const response = await $api.post(`quiz/questions`, { questionId, quizId });
    return response.data;
  }

  static async create(quiz) {
    const response = await $api.post('quiz/', quiz);
    return response.data;
  }

  static async update(id, quiz) {
    const response = await $api.put(`quiz/${id}`, quiz);
    return response.data;
  }

  static async delete(id) {
    const response = await $api.delete(`quiz/${id}`);
    return response.data;
  }
}
```

ReplyService.js:

```
import $api from '../http/Api';

export default class ReplyService {
  static async getAll() {
    const response = await $api.get(`reply/`);
    return response.data;
  }

  static async getOne(answerId, personId) {
    const response = await $api.get(`reply/replies?answer=${answerId}&per-
son=${personId}`);
    return response.data;
  }

  static async create(reply) {
    const response = await $api.post('reply/', reply);
    return response.data;
  }
```

```
}
```

TaskService.js:

```
import $api from '../http/Api';

export default class TaskService {
  static async getAllByCourse(taskId, sort) {
    const response = await $api.get(`task/course/${taskId}${sort ?
`?sort=${sort}` : ''}`);
    return response.data;
  }

  static async getAll(personId) {
    const response = await $api.get(`task/all/${personId}`);
    return response.data;
  }

  static async getOne(id) {
    const response = await $api.get(`task/${id}`);
    return response.data;
  }

  static async getTaskWorks(id, personId) {
    const response = await $api.get(`task/${id}/${personId}/works`);
    return response.data;
  }

  static async create(task) {
    const response = await $api.post('task/', task);
    return response.data;
  }

  static async createWork(work) {
    const response = await $api.post('task/work', work);
    return response.data;
  }

  static async getAllWork(work) {
    const response = await $api.get('task/work', work);
    return response.data;
  }

  static async update(id, task) {
    const response = await $api.put(`task/${id}`, task);
    return response.data;
  }

  static async delete(id) {
    const response = await $api.delete(`task/${id}`);
    return response.data;
  }
}
```

UserService.js:

```
import $api from '../http/Api';

export default class UserService {
  static async getAllPersons() {
    return $api.get('person/');
  }
```

```
  static async getPerson(id) {
    const response = await $api.get(`person/${id}`);
    return response.data;
  }

  static async updatePerson(id, person) {
    return $api.put(`person/${id}`, person);
  }
}
```

AuthModal.js:

```
import { useState } from 'react';
import RegistrationForm from './RegistrationForm';
import LoginForm from './LoginForm';
import { Dialog, DialogTitle, DialogContent, DialogActions, Button, Box } from
'@mui/material';

export default function AuthDialog(props) {
  const [userRegistered, setUserRegistered] = useState(true);

  const onClose = () => {
    props.onClose();
    setUserRegistered(true);
  };

  return (
    <Dialog
      open={props.open}
      onClose={onClose}
      aria-labelledby="contained-Dialog-title-vcenter"
      scroll={'paper'}
      maxWidth="xs"
      fullWidth
      PaperProps={{
        sx: { borderRadius: 2 },
      }}
    >
      <Box sx={{ display: 'flex', justifyContent: 'center' }}>
        <DialogTitle id="contained-Dialog-title-vcenter" sx={{ pt: 2, pb: 0,
fontSize: '24px' }} color="primary">
          <strong>{userRegistered ? 'Sign in' : 'Sign Up'}</strong>
        </DialogTitle>
      </Box>
      <DialogContent sx={{ py: 0 }}>
        {userRegistered ? <LoginForm onClose={onClose} /> : <RegistrationForm
onClose={onClose} />}
      </DialogContent>
      <DialogActions>
        <Button onClick={() => setUserRegistered((prev) => !prev)}>
          {userRegistered ? 'Sign Up' : 'Are you already registered?'}
        </Button>
      </DialogActions>
    </Dialog>
  );
}
```

AuthWrapper.js:

```
import { useState, useEffect } from 'react';
import axios from 'axios';
```

```
import configClient from '../../configClient';
import UserContext from './UserContext';
import AuthModal from './AuthModal';

function AuthWrapper(props) {
  const [user, setUser] = useState(null);
  const [showAuthModal, setShowAuthModal] = useState(false);
  const [isLoaded, setIsLoaded] = useState(true);

  const isUserAuthorized = () => !!user;
  const openAuthModal = () => {
    setShowAuthModal(true);
  };

  const checkAuth = async () => {
    try {
      const response = await axios.get(configClient.API_URL + '/api/auth/re-
fresh', { withCredentials: true });
      localStorage.setItem('access-token', response.data.accessToken);
      setUser(response.data.user);
    } catch (error) {
      console.log(error.message);
    } finally {
      setIsLoaded(false);
    }
  };

  useEffect(() => {
    if (localStorage.getItem('access-token')) {
      checkAuth();
    }
  }, []);

  return (
    <UserContext.Provider value={{ user, setUser, isUserAuthorized, openAu-
thModal }}>
      <AuthModal
        open={showAuthModal}
        onClose={() => {
          setShowAuthModal(false);
        }}
      />
      {props.children}
    </UserContext.Provider>
  );
}

export default AuthWrapper;
```

LoginForm.js:

```
import { useContext, useState } from 'react';
import AuthService from '../../services/AuthService';
import UserContext from './UserContext';
import * as yup from 'yup';
import StackForm from '../Shared/StackForm';
import { Alert, LinearProgress } from '@mui/material';
import { useNavigate } from 'react-router-dom';

const initialValues = {
  email: '',
  password: '',
```

```
};

const labels = {
  email: 'Email',
  password: 'Password',
};

const types = {
  email: 'email',
  password: 'password',
};

const validationSchema = yup.object({
  email: yup.string('Enter your email').email('Enter a valid email').re-
quired('Email is required'),
  password: yup
    .string('Enter your password')
    .min(8, 'Password should be of minimum 8 characters length')
    .required('Password is required'),
});

function LoginForm(props) {
  const [error, setError] = useState('');
  const [isLoaded, setIsLoaded] = useState(false);
  const { setUser } = useContext(UserContext);
  const navigate = useNavigate();

  const onLoginHandler = async (values) => {
    try {
      setIsLoaded(() => true);
      const response = await AuthService.login(values.email, values.password);
      localStorage.setItem('access-token', response.data.accessToken);
      setUser(response.data.user);
      props.onClose();
      navigate('./profile');
    } catch (error) {
      if (error.response.status === 401) {
        setError(() => 'Wrong email or password');
      } else {
        setError(() => 'Oops, something is wrong');
      }
    } finally {
      setIsLoaded(() => false);
    }
  };

  return (
    <StackForm
      initialValues={initialValues}
      validationSchema={validationSchema}
      onSubmit={onLoginHandler}
      types={types}
      labels={labels}
      submitButtonLabel={'Sign In'}
    >
      {error && <Alert severity="error">{error}</Alert>}
      {isLoaded && <LinearProgress color="inherit" />}
    </StackForm>
  );
}

export default LoginForm;
```

RegistrationForm.js:

```
import { useContext, useState } from 'react';
import AuthService from '../../services/AuthService';
import UserContext from './UserContext';
import * as yup from 'yup';
import StackForm from '../Shared/StackForm';
import { Alert, LinearProgress } from '@mui/material';
import { useNavigate } from 'react-router-dom';

const initialValues = {
  fullName: '',
  email: '',
  password: '',
  birthday: '',
  avatar: null,
};

const labels = {
  fullName: 'Full Name',
  email: 'Email',
  password: 'Password',
  birthday: 'Choose birthday',
  avatar: 'Choose avatar',
};

const types = {
  fullName: 'text',
  email: 'email',
  password: 'password',
  birthday: 'date',
  avatar: 'file',
};

const validationSchema = yup.object({
  fullName: yup
    .string('Enter your full name')
    .min(3, 'Full name must be more than 3 characters')
    .max(64, 'Full name must be less than 64 characters')
    .matches(/^[A-Za-z\s'-]+$/, 'Full name must not contain symbols or numbers')
    .required('Full name is required'),
  email: yup
    .string('Enter your email')
    .email('Enter a valid email')
    .matches(/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/, 'Enter a valid
email')
    .required('Email is required'),
  password: yup
    .string('Enter your password')
    .min(8, 'Password must be more than 8 characters')
    .max(128, 'Password must not be more than 128 characters')
    .matches(
      /^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)\S+$/,
      'Password must contain at least one uppercase letter, one lowercase let-
ter, one number, and no spaces',
    )
    .required('Password is required'),
  birthday: yup.string('Enter your birthday'),
  avatar: yup.string('Upload your avatar').required('Avatar is required'),
});

function RegistrationForm(props) {
```

```
  const [error, setError] = useState('');
  const [isLoaded, setIsLoaded] = useState(false);
  const { setUser } = useContext(UserContext);
  const navigate = useNavigate();

  const onRegistrationHandler = async (values) => {
    try {
      const formData = new FormData();
      for (let value in values) {
        formData.append(value, values[value]);
      }
      const response = await AuthService.registration(formData);
      localStorage.setItem('access-token', response.data.accessToken);
      setUser(response.data.user);
      props.onClose();
      navigate('./profile');
    } catch (error) {
      if (error.response.status === 409) {
        setError(() => 'An account is already registered with your email');
      } else {
        setError(() => 'Oops, something is wrong');
      }
    } finally {
      setIsLoaded(() => false);
    }
  };

  return (
    <StackForm
      initialValues={initialValues}
      validationSchema={validationSchema}
      onSubmit={onRegistrationHandler}
      types={types}
      labels={labels}
      submitButtonLabel={'Sign Up'}
    >
      {error && <Alert severity="error">{error}</Alert>}
      {isLoaded && <LinearProgress color="inherit" />}
    </StackForm>
  );
}

export default RegistrationForm;
```

UserContext.js:

```
import { createContext } from "react";
const UserContext = createContext({ user: null, isUserAuthorized: false });
export default UserContext;
```

CoursePage.js:

```
import { Tabs, Tab, Box, Grid } from '@mui/material';
import { Link, useParams } from 'react-router-dom';
import { useContext, useEffect, useMemo, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import materials from '../Materials/materials';
import CourseService from '../../../services/CourseService';
import UserContext from '../../Authorization/UserContext';
import CourseContext from '../CourseContext';
import { ErrorPage } from '../../Shared/ErrorPages/ErrorPage';
import GroupService from '../../../services/GroupService';
```

```
function CoursePage() {
  const params = useParams();
  const type = params.material;
  const navigate = useNavigate();
  const { user } = useContext(UserContext);
  const [teachers, setTeachers] = useState([]);
  const [students, setStudents] = useState([]);
  const [groups, setGroups] = useState([]);
  const teacherAccess = useMemo(() => teachers.some((teacher) => teacher.per-
sonId === user?.id), [teachers, user?.id]);

  useEffect(() => {
    if (!type) {
      navigate('./lectures', { replace: true });
    }
  }, [type]);

  useEffect(() => {
    CourseService.getTeachersFromCourse(params.course).then((teachers) => {
      setTeachers(() => teachers);
    });
    CourseService.getGroupsInCourse(params.course).then((groups) => {
      setGroups(() => groups);
      Promise.all(groups.map((group) => GroupService.getStuden-
tsInGroup(group.id))).then((students) =>
        setStudents(() => students.flat()),
      );
    });
  }, [params.course]);

  return (
    <CourseContext.Provider value={{ teacherAccess, groups, students }}>
      <Grid item>
        <Box sx={{ width: '100%' }}>
          <Box>
            {type && (
              <Tabs value={type} aria-label="basic tabs example">
                {Object.keys(materials).map((material) => {
                  if (material === 'studentsWorks' && !teacherAccess) {
                    return null;
                  } else {
                    return (
                      <Tab
                        key={material}
                        value={material}
                        label={materials[material].label}
                        component={Link}
                        to={`/institutions/${params.institu-
tion}/courses/${params.course}/${material}`}
                      />
                    );
                  }
                })}
              </Tabs>
            )}
          </Box>
          {materials[type]?.render() || <ErrorPage />}
        </Box>
      </Grid>
    </CourseContext.Provider>
  );
}
```

```
export default CoursePage;
```

GradeBook.js:

```
import { styled } from '@mui/material/styles';
import ArrowForwardIosSharpIcon from '@mui/icons-material/ArrowForwardIosSharp';
import MuiAccordion from '@mui/material/Accordion';
import MuiAccordionSummary from '@mui/material/AccordionSummary';
import MuiAccordionDetails from '@mui/material/AccordionDetails';
import Typography from '@mui/material/Typography';
import React, { useContext, useEffect, useState } from 'react';
import { Box, Chip, Divider, List, ListItem, ListItemIcon, ListItemText } from
'@mui/material';
import { Assignment, Quiz, StarBorder } from '@mui/icons-material';
import UserContext from '../../Authorization/UserContext';
import TaskService from '../../../services/TaskService';
import MarkService from '../../../services/MarkService';
import QuizService from '../../../services/QuizService';
import { useParams } from 'react-router-dom';

const Accordion = styled((props) => <MuiAccordion disableGutters elevation={0}
square {...props} />)(({ theme }) => ({
  border: `1px solid ${theme.palette.divider}`,
  '&:not(:last-child)': {
    borderBottom: 0,
  },
  '&:before': {
    display: 'none',
  },
}));

const AccordionSummary = styled((props) => (
  <MuiAccordionSummary
    expandIcon={<ArrowForwardIosSharpIcon sx={{ fontSize: '0.9rem', color:
'#ffff' }} />}
    {...props}
  />
))(({ theme }) => ({
  backgroundColor: theme.palette.mode === 'dark' ? 'rgba(255, 255, 255, .05)' :
'#632BBB',
  color: '#ffff',
  flexDirection: 'row-reverse',
  '& .MuiAccordionSummary-expandIconWrapper.Mui-expanded': {
    transform: 'rotate(90deg)',
  },
  '& .MuiAccordionSummary-content': {
    marginLeft: theme.spacing(1),
  },
}));

const AccordionDetails = styled(MuiAccordionDetails)(({ theme }) => ({
  padding: theme.spacing(2),
  borderTop: '1px solid rgba(0, 0, 0, .125)',
}));

const GradeBook = () => {
  const [expanded, setExpanded] = useState('panel1');
  const [quizzes, getQuizzes] = useState([]);
  const [tasks, setTasks] = useState([]);
  const { course: courseId } = useParams();
  const { user } = useContext(UserContext);
```

```
  useEffect(() => {
    QuizService.getAllQuizzesMarks(courseId, user?.id).then((quizzes) =>
getQuizzes(() => quizzes));
    if (user) {
      TaskService.getAll(user?.id).then((tasks) => {
        tasks.forEach((task) =>
          task.works.forEach((work) => {
            MarkService.getOne(work?.id, user?.id).then((mark) => {
              setTasks((prev) => [...prev, { title: task?.title, mark:
mark?.value }]);
            });
          }),
        );
      });
    }
  }, [courseId, user]);

  console.log("quizzes", quizzes);

  const handleChange = (panel) => (event, newExpanded) => {
    setExpanded(newExpanded ? panel : false);
  };

  const values = [
    {
      expanded: 'panel1',
      title: 'Quizzes',
      icon: <Quiz />,
      tasks: quizzes,
    },
    {
      expanded: 'panel2',
      title: 'Tasks',
      icon: <Assignment />,
      tasks: tasks,
    },
  ];

  return (
    <Box sx={{ my: 4 }}>
      {values.map((item, index) => (
        <Accordion key={index} expanded={expanded === item.expanded} on-
Change={handleChange(item.expanded)}>
          <AccordionSummary>
            <Typography>{item.title}</Typography>
          </AccordionSummary>
          <AccordionDetails sx={{ p: 0 }}>
            <List>
              {item.tasks.map((task, index) => (
                <React.Fragment key={index}>
                  <ListItem sx={{ px: 5 }}>
                    <ListItemIcon>{item.icon}</ListItemIcon>
                    <ListItemText primary={task.title} />
                    {task?.mark && (
                      <ListItemText
                        primary={
                          <Chip
                            label={task?.mark}
                            sx={{ bgcolor: '#d7e2f9', fontSize: '16px' }}
                            icon={<StarBorder />}
                          />
                        }
```

```
                        sx={{ textAlign: 'right' }}
                      />
                    )}
                  </ListItem>
                  {item.tasks.length - 1 !== index && <Divider />}
                </React.Fragment>
              ))}
            </List>
          </AccordionDetails>
        </Accordion>
      ))}
    </Box>
  );
};

export default GradeBook;
```

Lectures.js:

```
import MaterialsList from '../Materials/MaterialsList';

const Lectures = () => {
  return (
    <>
      <MaterialsList type="lectures"/>
    </>
  );
};

export default Lectures;
```

EditModal.js:

```
import * as yup from 'yup';
import { Dialog, AppBar, Toolbar, IconButton, Typography, Slide, Box, TextField,
Button } from '@mui/material';
import CloseIcon from '@mui/icons-material/Close';
import { forwardRef } from 'react';
import { Form } from '../../Shared/components/Form';
import DateField from '../../Shared/Fields/DateField/DateField';
import dayjs from 'dayjs';
import TimeCustomField from '../../Shared/Fields/TimeField/TimeCustomField';

const Transition = forwardRef(function Transition(props, ref) {
  return <Slide direction="up" ref={ref} {...props} />;
});

const EditModal = ({ type, open, onClose, item, courseId, onUpdate, service,
children }) => {
  const title = `Edit ${type}`;

  const schemaObj = {
    title: yup.string(`Enter the title of ${type}`),
    numberByOrder: yup.string(`Enter the number of order of your ${type}`),
    description: yup.string(`Enter the text of ${type}`),
  };

  const labels = {
    title: 'Title',
    numberByOrder: 'Number by order',
    description: 'Description',
  };
```

```
  const types = {
    title: 'text',
    numberByOrder: 'number',
    description: type === 'quiz' ? 'text' : 'textarea',
  };

  const initialValues = {
    title: item?.title || '',
    numberByOrder: item?.numberByOrder || '',
    description: item?.description || '',
  };

  const fieldsRenderers = {
    title: (props) => <TextField label="Enter title" {...props} />,
    numberByOrder: (props) => <TextField type="number" label={`Enter the number
of ${type}`} {...props} />,
    description: (props) => <TextField label="Enter text" multiline rows={type
=== 'lectures' ? 15 : 5} {...props} />,
  };

  if (type === 'tasks') {
    schemaObj.deadline = yup.string(`Enter the start date of ${type}`);
    labels.deadline = 'Deadline';
    types.deadline = 'date';
    initialValues.deadline = dayjs(item?.deadline || Date.now());
    fieldsRenderers.deadline = (props) => <DateField {...props} />;
  }

  if (type === 'quizzes') {
    schemaObj.startDate = yup.string(`Enter the start date of ${type}`);
    schemaObj.endDate = yup.string(`Enter the end date of ${type}`);
    schemaObj.time = yup.string(`Enter the duration of ${type}`);
    schemaObj.perQuestionScore = yup.string(`Enter the number of points for
question`);

    labels.startDate = 'Start date';
    labels.endDate = 'End date';
    labels.time = 'Time';
    labels.perQuestionScore = 'Points for question';

    types.startDate = 'date';
    types.endDate = 'date';
    types.time = 'time';
    types.perQuestionScore = 'number';

    initialValues.startDate = dayjs(item?.startDate || Date.now());
    initialValues.endDate = dayjs(item?.endDate || Date.now());
    initialValues.time = dayjs(item?.time || Date.now());
    initialValues.perQuestionScore = item?.perQuestionScore || '';

    fieldsRenderers.startDate = (props) => <DateField {...props} />;
    fieldsRenderers.endDate = (props) => <DateField {...props} />;
    fieldsRenderers.time = (props) => <TimeCustomField {...props} />;
    fieldsRenderers.perQuestionScore = (props) => (
      <TextField type="number" label={`Enter points for question`} {...props} />
    );
  }

  const validationSchema = yup.object(schemaObj);

  const onEditHandler = async (values) => {
    if (values.courseId === undefined) {
```

```
      values.courseId = courseId;
    }

    let newItem;

    if (item) {
      newItem = await service.update(item.id, values);
    } else {
      newItem = await service.create(values);
    }

    onUpdate(newItem);
    onClose();
  };

  return (
    <Dialog fullScreen open={open} onClose={onClose} TransitionComponent={Tran-
sition}>
      <AppBar sx={{ position: 'relative' }}>
        <Toolbar>
          <Typography sx={{ ml: 2, flex: 1 }} variant="h6" component="div">
            {title}
          </Typography>
          <IconButton edge="start" color="inherit" onClick={onClose} aria-la-
bel="close">
            <CloseIcon />
          </IconButton>
        </Toolbar>
      </AppBar>
      <Box sx={{ p: 5 }}>
        <Typography>Basic Info</Typography>
        <Form
          id="edit"
          initialValues={initialValues}
          types={types}
          validationSchema={validationSchema}
          fieldsRenderers={fieldsRenderers}
          onSubmit={onEditHandler}
        >
          <Button type="submit" variant="contained" size="large" sx={{ mt: 4,
px: 8, borderRadius: '6px' }}>
            Save
          </Button>
        </Form>
      </Box>
    </Dialog>
  );
};

export default EditModal;
```

Material.js:

```
import { Box, Card, CardActionArea, CardContent, Typography, Button, Chip, Car-
dActions, Avatar } from '@mui/material';
import { Delete, Edit, CalendarMonth, Star } from '@mui/icons-material';
import { useContext, useEffect, useState } from 'react';
import DeleteModal from '../../Shared/components/DeleteModal';
import { Link } from 'react-router-dom';
import EditModal from './EditModal';
import materials from './materials';
import QuizService from '../../../services/QuizService';
```

```
import UserContext from '../../Authorization/UserContext';

const TEXT_MAX_LENGTH = 80;

const Material = ({ type, item, onDelete, onUpdate, teacherAccess, children })
=> {
  const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);
  const [isEditModalOpen, setIsEditModalOpen] = useState(false);
  const { user } = useContext(UserContext);
  const [mark, setMark] = useState(null);

  useEffect(() => {
    if (type === 'quizzes') {
      QuizService.getOneWithMark(item.id, user?.id).then((quiz) => {
        setMark(() => quiz[0].mark);
      });
    }
  }, [type, item.id]);

  const onDeleteModalOpen = () => {
    setIsDeleteModalOpen(() => true);
  };

  const onDeleteModalClose = () => {
    setIsDeleteModalOpen(() => false);
  };

  const onEditModalOpen = () => {
    setIsEditModalOpen(() => true);
  };

  const onEditModalClose = () => {
    setIsEditModalOpen(() => false);
  };

  const onDeleteHandler = async () => {
    await materials[type]?.service.delete(item.id);
    onDelete(item.id);
  };

  const currentDate = new Date(Date.now()).toLocaleDateString('en-US');
  const checkDate =
    currentDate >= new Date(item.startDate).toLocaleDateString('en-US') &&
    currentDate <= new Date(item.endDate).toLocaleDateString('en-US');

  return (
    <>
      <Card
        sx={{
          bgcolor: type !== 'quizzes' ? '#fafafa' : !checkDate && '#ffcdd2',
          width: '100%',
          my: 1,
          borderRadius: 3,
        }}
      >
        <Box sx={{ display: 'flex', flexDirection: 'row', justifyContent:
'space-between' }}>
          <CardActionArea
            sx={{ p: 2 }}
            component={Link}
            to={`${item.id}`}
            disabled={type === 'quizzes' && (!checkDate || (+mark !== 0 &&
!teacherAccess))}
```

```
          >
            <CardContent sx={{ p: 0, textAlign: 'left', mr: 1, display: 'flex',
justifyContent: 'space-between' }}>
              <Box sx={{ display: 'flex' }}>
                <Box sx={{ display: 'flex', alignItems: 'center', mr: 2 }}>
                  <Chip variant="outlined" label={item.numberByOrder} icon={ma-
terials[type]?.icon()} sx={{ pl: 1 }} />
                </Box>
                <Box>
                  <Typography variant="h5" color="primary">
                    <strong>
                      {item.title.length > TEXT_MAX_LENGTH
                        ? item.title.substring(0, TEXT_MAX_LENGTH) + '...'
                        : item.title}
                    </strong>
                  </Typography>
                  <Typography variant="subtitle1" color="text.secondary" gutter-
Bottom>
                    {item.description.length > TEXT_MAX_LENGTH
                      ? item.description.substring(0, TEXT_MAX_LENGTH) + '...'
                      : item.description}{' '}
                  </Typography>
                  <Typography variant="body" color="text.secondary">
                    Created at - {new Date(item.createdAt).toLocaleDateString()}
                  </Typography>
                </Box>
              </Box>
              {type === 'tasks' && (
                <Box sx={{ display: 'flex', alignItems: 'center' }}
color="#616161">
                  <CalendarMonth sx={{ mr: 1 }} />
                  <Typography variant="body" color="text.secondary">
                    {new Date(item.deadline).toLocaleDateString()}
                  </Typography>
                </Box>
              )}
              {type === 'quizzes' && (
                <Box sx={{ display: 'flex', alignItems: 'center' }}
color="#616161">
                  <CalendarMonth sx={{ mr: 1 }} />
                  <Typography variant="body" color="text.secondary">
                    {`${item.startDate} - ${item.endDate}`}
                  </Typography>
                </Box>
              )}
            </CardContent>
          </CardActionArea>
          {+mark !== 0 && (
            <Box sx={{ display: 'flex', alignItems: 'center', mr: 1 }}>
              <Chip icon={<Star />} color="primary" label={mark} variant="con-
tained" sx={{ fontSize: '22px', mx: 2 }} />
            </Box>
          )}
          {teacherAccess && (
            <CardActions sx={{ display: 'flex', alignItems: 'center', mr: 1 }}>
              <Button variant="outlined" startIcon={<Edit />} sx={{ mr: 1 }} on-
Click={onEditModalOpen}>
                Edit
              </Button>
              <Button
                variant="outlined"
                startIcon={<Delete />}
                sx={{ color: '#ef5350', border: '1px solid #ef5350' }}
```

```
                onClick={onDeleteModalOpen}
              >
                Delete
              </Button>
            </CardActions>
          )}
        </Box>
      </Card>
      <DeleteModal
        open={isDeleteModalOpen}
        onClose={onDeleteModalClose}
        onDelete={onDeleteHandler}
        title={`Delete ${type}?`}
        description={`Do you want to delete this ${type}?`}
      />
      <EditModal
        type={type}
        open={isEditModalOpen}
        onClose={onEditModalClose}
        item={item}
        courseId={item.courseId}
        onUpdate={onUpdate}
        service={materials[type]?.service}
      >
        {children}
      </EditModal>
    </>
  );
};

export default Material;
```

MaterialPage.js:

```
import { Box, Button, Divider, IconButton, List, ListItem, ListItemIcon, Text-
Field, Typography } from '@mui/material';
import React, { useContext, useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import materials from './materials';
import TaskService from '../../../services/TaskService';
import UserContext from '../../Authorization/UserContext';
import { Download } from '@mui/icons-material';
import FileService from '../../../services/FileService';
import { saveAs } from 'file-saver';

const MaterialPage = () => {
  const params = useParams();
  const type = params.material;
  const id = params.materialId;
  const [item, setItem] = useState([]);
  const [file, setFile] = useState(null);
  const [works, setWorks] = useState([]);
  const { user } = useContext(UserContext);

  useEffect(() => {
    materials[type].service.getOne(id).then(
      (item) => {
        setItem(() => item);
        if (user && type === 'tasks') {
          TaskService.getTaskWorks(item.id, user?.id).then((task) => {
            if (task?.works) {
              setWorks(() => task.works);
```

```
            }
          });
        }
      },
      (error) => {
        console.log(error);
      },
    );
  }, [id, type, user]);

  const onSendAnswersClick = () => {
    const formData = new FormData();
    formData.append('taskId', item.id);
    formData.append('personId', user.id);
    formData.append('workFile', file);

    TaskService.createWork(formData).then(() => console.log('success'));
  };

  const onDownloadHandler = (workFile) => () => {
    FileService.downloadFile(workFile).then((response) =>
      saveAs(response.data, user.fullName.replace(/ /g,'') + '-task№' +
item.numberByOrder),
    );
  };

  return (
    <Box>
      <Typography variant="h4" sx={{ mb: 4 }}>
        {item.title}
      </Typography>
      <Typography variant="subtitle1" sx={{ fontSize: '18px', textAlign: 'left'
}}>
        {item.description}
      </Typography>
      {type === 'tasks' && (
        <>
          <Box sx={{ display: 'flex', flexDirection: 'column', alignItems:
'flex-start', mt: 5 }}>
            <TextField
              type="file"
              variant="outlined"
              helperText="Upload files"
              onChange={(e) => setFile(() => e.currentTarget.files[0])}
            />
            <Button color="primary" variant="contained" size="large" sx={{ m: 1,
px: 10 }} onClick={onSendAnswersClick}>
              Submit
            </Button>
          </Box>
          {works && works.length ? (
            <>
              <Typography sx={{ mt: 4, mb: 2, textAlign: 'left' }} variant="h6"
component="div">
                Works history
              </Typography>
              <List sx={{ bgcolor: '#fafafa' }}>
                {works.map((work, index) => (
                  <React.Fragment key={index}>
                    <ListItem>
                      <ListItemIcon>
                        <IconButton onClick={onDownloadHandler(work.workFile)}>
                          <Download />
```

```
                      </IconButton>
                    </ListItemIcon>
                    {`${user.fullName.replace(/ /g,'')}-task№${item.number-
ByOrder}-v${index + 1}`}
                  </ListItem>
                  {works.length - 1 !== index && <Divider />}
                </React.Fragment>
              ))}
            </List>
          </>
        ) : (
          <></>
        )}
      </>
    )}
  </Box>
 );
};

export default MaterialPage;
```

materials.js:

```
import { Assignment, MenuBook, Quiz } from '@mui/icons-material';
import LectureService from '../../../services/LectureService';
import QuizService from '../../../services/QuizService';
import TaskService from '../../../services/TaskService';
import UserService from '../../../services/UserService';
import Lectures from '../Lectures/Lectures';
import Members from '../Members/Members';
import Quizzes from '../Quizzes/Quizzes';
import Tasks from '../Tasks/Tasks';
import GradeBook from '../GradeBook/GradeBook';
import StudentsWorks from '../StudentsWorks/StudentsWorks';

const materials = {
  lectures: {
    label: 'Lectures',
    icon: () => <MenuBook />,
    render: () => <Lectures />,
    service: LectureService,
  },
  tasks: {
    label: 'Tasks',
    icon: () => <Assignment />,
    render: () => <Tasks />,
    service: TaskService,
  },
  quizzes: {
    label: 'Quizzes',
    icon: () => <Quiz />,
    render: () => <Quizzes />,
    service: QuizService,
  },
  members: {
    label: 'Members',
    render: () => <Members />,
    service: UserService,
  },
  gradeBook: {
    label: 'Grade book',
    render: () => <GradeBook />,
```

```
  },
  studentsWorks: {
    label: 'Students works',
    render: () => <StudentsWorks />,
  },
};

export default materials;
```

MaterialsList.js:

```
import { useContext, useEffect, useState } from 'react';
import { Backdrop, Box, CircularProgress, Fab, FormControl, InputLabel, Menu-
Item, Select } from '@mui/material';
import { Add } from '@mui/icons-material';
import { useParams, useSearchParams } from 'react-router-dom';
import Material from './Material';
import EditModal from './EditModal';
import materials from './materials';
import { EmptyPage } from '../../Shared/components/EmptyPage';
import CourseContext from '../CourseContext';

const MaterialsList = ({ type, children }) => {
  const [searchParams, setSearchParams] = useSearchParams();
  const [items, setItems] = useState([]);
  const [isLoaded, setIsLoaded] = useState(true);
  const [isCreateModalOpen, setIsCreateModalOpen] = useState(false);
  const sortValue = searchParams.get('sort');
  const { course: courseId } = useParams();
  const { teacherAccess } = useContext(CourseContext);

  useEffect(() => {
    materials[type]?.service.getAllByCourse(courseId, sortValue).then(
      (items) => {
        setItems(() => items);
        setIsLoaded(() => false);
      },
      (error) => {
        console.log(error);
      },
    );
  }, [courseId, sortValue]);

  const onCreateModalOpen = () => {
    setIsCreateModalOpen(() => true);
  };

  const onCreateModalClose = () => {
    setIsCreateModalOpen(() => false);
  };

  const deleteItem = (itemId) => {
    setItems((prev) => prev.filter((item) => item.id !== itemId));
  };

  const updateItem = (newItem) => {
    setItems((prev) => prev.map((item) => (item.id !== newItem.id ? item :
newItem)));
  };

  const createLecture = (newItem) => {
    setItems((prev) => prev.concat(newItem));
```

```
  };

  const onChangeHandler = (e) => {
    if (e.target.value) {
      setSearchParams({ sort: e.target.value });
    } else {
      setSearchParams({});
    }
  };

  return (
    <>
      <Box sx={{ width: '100%', display: 'flex', flexDirection: 'column' }}>
        {items.length || isLoaded ? (
          <>
            <Box sx={{ display: 'flex', justifyContent: 'flex-end' }}>
              <FormControl sx={{ mb: 1, minWidth: 110 }} size="small">
                <InputLabel id="demo-simple-select-autowidth-label">Sort by</In-
putLabel>
                <Select
                  labelId="demo-simple-select-autowidth-label"
                  id="demo-simple-select-autowidth"
                  value={sortValue || ''}
                  onChange={onChangeHandler}
                  autoWidth
                  label="Sorting"
                  sx={{ borderRadius: 3 }}
                >
                  <MenuItem value="">Default</MenuItem>
                  {type === 'task' && <MenuItem value="deadline">Deadline</Menu-
Item>}
                  <MenuItem value="old">Old</MenuItem>
                  <MenuItem value="new">New</MenuItem>
                  <MenuItem value="numberFromFirst">Number from first</MenuItem>
                  <MenuItem value="numberFromLast">Number from last</MenuItem>
                </Select>
              </FormControl>
            </Box>
            {items.map((item, index) => {
              return (
                <Material
                  key={index}
                  type={type}
                  item={item}
                  onDelete={deleteItem}
                  onUpdate={updateItem}
                  teacherAccess={teacherAccess}
                >
                  {children}
                </Material>
              );
            })}
          </>
        ) : (
          <EmptyPage />
        )}
      </Box>
      {teacherAccess && (
        <>
          <Fab
            color="primary"
            sx={{
              position: 'fixed',
```

```
              bottom: 30,
              right: 30,
              '&:hover': {
                bgcolor: '#311b92',
              },
            }}
            onClick={onCreateModalOpen}
          >
            <Add />
          </Fab>
          <EditModal
            type={type}
            open={isCreateModalOpen}
            onClose={onCreateModalClose}
            courseId={courseId}
            onUpdate={createLecture}
            service={materials[type]?.service}
          >
            {children}
          </EditModal>
        </>
      )}
    </>
  );
};

export default MaterialsList;
```

Members.js:

```
import { useEffect, useState } from 'react';
import { Grid } from '@mui/material';
import CourseService from '../../../services/CourseService';
import { useParams } from 'react-router-dom';
import UserData from '../../ProfilePage/UserData/UserData';

const Members = () => {
  const [studentsInGroup, setStudentsInGroup] = useState([]);
  const courseId = useParams().course;

  useEffect(() => {
    CourseService.getAllMembersByCourse(courseId).then((students) => {
      setStudentsInGroup(() => students);
    });
  }, [courseId]);

  return (
    <Grid container spacing={3} sx={{ mt: 2, justifyContent: 'center' }}>
      {studentsInGroup.map((student) => {
        return (
          <Grid key={student.id} item>
            <UserData user={student} />
          </Grid>
        );
      })}
    </Grid>
  );
};

export default Members;
```

AddQuestionCard.js:

```
import { Add, Check, Clear } from '@mui/icons-material';
import {
  Alert,
  Box,
  Button,
  Card,
  CardActions,
  CardContent,
  FormControl,
  FormControlLabel,
  FormLabel,
  IconButton,
  Radio,
  RadioGroup,
  TextField,
} from '@mui/material';
import { useState } from 'react';
import QuestionService from '../../../../services/QuestionService';
import { useParams } from 'react-router-dom';

export const Option = ({ index, option, onTextChange, onDelete }) => {
  const onChangeOptionHandler = (e) => {
    onTextChange(() => e.target.value);
  };

  return (
    <Box sx={{ display: 'flex', alignItems: 'center' }}>
      <FormControlLabel
        value={index}
        control={<Radio />}
        label={
          <TextField value={option.text} variant="standard" placeholder="Option"
onChange={onChangeOptionHandler} />
        }
      />
      <IconButton onClick={onDelete}>
        <Clear />
      </IconButton>
      {option.isCorrect && <Check sx={{ color: '#4caf50' }} />}
    </Box>
  );
};

const AddQuestionCard = ({ setQuestions }) => {
  const { quiz: quizId } = useParams();
  const [questionText, setQuestionText] = useState(null);
  const [options, setOptions] = useState([{ text: '', isCorrect: false }]);
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);

  const onChangeSelectedHandler = (event) => {
    setOptions((prev) =>
      prev.map((item, index) => {
        item.isCorrect = event.target.value == index;
        return item;
      }),
    );
  };

  const getTextChangeHandler = (id) => (callback) => {
    setOptions((prev) => prev.map((item, index) => (index === id ? { ...item,
text: callback() } : item)));
  };
```

```
  const getOptionDeleteHandler = (id) => () => {
    setOptions((prev) => prev.filter((_, index) => index !== id));
  };

  const onChangeQuestionHandler = (e) => {
    setQuestionText(e.target.value);
  };

  const onAddOptionHandler = () => {
    setOptions((prev) => [...prev, { text: '', isCorrect: false }]);
  };

  const onCreateQuestion = async () => {
    let newQuestion;
    setLoading(() => true);
    try {
      if (!questionText || !options.length) {
        setError(() => true);
      } else {
        newQuestion = await QuestionService.create({ text: questionText, an-
swers: options, quizId });
        setError(() => false);
      }
    } catch (error) {
      setError(() => true);
    } finally {
      setTimeout(() => setLoading(() => false), 3000);
    }
    setQuestionText(() => '');
    setOptions(() => [{ text: '', isCorrect: false }]);
    setQuestions((prev) => prev.concat(newQuestion));
  };

  return (
    <Card elevation={4}>
      <CardContent sx={{ pb: 0, display: 'flex', flexDirection: 'column' }}>
        <TextField
          error={!!error}
          fullWidth
          label="Enter your question"
          type="text"
          variant="outlined"
          value={questionText}
          helperText={error && 'Enter correct question'}
          onChange={onChangeQuestionHandler}
        />
        <FormControl sx={{ my: 2, ml: 1, alignItems: 'flex-start' }}>
          {options.length ? <FormLabel id="demo-controlled-radio-buttons-
group">Options</FormLabel> : <></>}
          <RadioGroup
            aria-labelledby="demo-controlled-radio-buttons-group"
            name="controlled-radio-buttons-group"
            value={options.findIndex((item) => item.isCorrect)}
            onChange={onChangeSelectedHandler}
          >
            {options.map((item, index) => (
              <Option
                key={index}
                index={index}
                option={item}
                onTextChange={getTextChangeHandler(index)}
                onDelete={getOptionDeleteHandler(index)}
```

```
              />
            ))}
          </RadioGroup>
        </FormControl>
        {loading && <Alert severity={error ? 'error' : 'success'}>This is an er-
ror alert — check it out!</Alert>}
      </CardContent>
      <CardActions sx={{ px: 2 }}>
        <Button variant="outlined" startIcon={<Add />} onClick={onAddOptionHan-
dler}>
          Add option
        </Button>
        <Button variant="contained" onClick={onCreateQuestion}>
          Save
        </Button>
      </CardActions>
    </Card>
  );
};

export default AddQuestionCard;
```

Question.js:

```
import { ArrowForward, Delete } from '@mui/icons-material';
import {
  Box,
  Button,
  Card,
  CardActions,
  FormControl,
  FormControlLabel,
  Radio,
  RadioGroup,
  Typography,
} from '@mui/material';
import { useEffect, useState } from 'react';
import DeleteModal from '../../../Shared/components/DeleteModal';
import QuestionService from '../../../../services/QuestionService';

const Question = ({ question, index, amount, setAnswers, setRepliesAnswersId,
teacherAccess, setQuestions }) => {
  const [selected, setSelected] = useState(null);
  const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);

  useEffect(() => {
    let id;
    question.answers.forEach((answer) => {
      if (answer.text === selected) {
        id = answer.id;
      }
      return id;
    });
    if (id) {
      setRepliesAnswersId((prev) => [...prev, id]);
    }
  }, [selected, setAnswers, question.id]);

  const onChangeSelectedHandler = (e) => {
    setSelected(e.target.value);
    question.answers.forEach((answer) => setRepliesAnswersId((prev) => prev.fil-
ter((id) => answer.id !== id)));
```

```
  };

  const onDeleteModalOpen = () => {
    setIsDeleteModalOpen(() => true);
  };

  const onDeleteModalClose = () => {
    setIsDeleteModalOpen(() => false);
  };

  const onDeleteHandler = async () => {
    await QuestionService.delete(question.id);
    setQuestions((prev) => prev.filter((item) => item.id !== question.id));
  };

  return (
    <>
      <Card sx={{ py: 2, px: 4, mb: 2 }}>
        <Box sx={{ display: 'flex', alignItems: 'center', justifyContent:
'space-between' }}>
          <Typography sx={{ fontSize: 14 }} color="text.secondary">
            {`Question ${index + 1} of ${amount}`}
          </Typography>
          {teacherAccess && (
            <CardActions sx={{ display: 'flex', alignItems: 'center' }}>
              <Button
                variant="outlined"
                size="small"
                startIcon={<Delete />}
                sx={{ color: '#ef5350', border: '1px solid #ef5350' }}
                onClick={onDeleteModalOpen}
              >
                Delete
              </Button>
            </CardActions>
          )}
        </Box>
        <Box sx={{ display: 'flex', flexDirection: 'column', alignItems: 'flex-
start' }}>
          <Typography variant="h6">{question.text}</Typography>
          <FormControl sx={{ my: 2, ml: 1, alignItems: 'flex-start' }}>
            <RadioGroup
              aria-labelledby="demo-controlled-radio-buttons-group"
              name="controlled-radio-buttons-group"
              value={selected}
              onChange={onChangeSelectedHandler}
            >
              {question.answers.map((answer, index) => (
                <FormControlLabel key={index} value={answer.text} control={<Ra-
dio />} label={answer.text} />
              ))}
            </RadioGroup>
          </FormControl>
        </Box>
      </Card>
      <DeleteModal
        open={isDeleteModalOpen}
        onClose={onDeleteModalClose}
        onDelete={onDeleteHandler}
        title={`Delete question?`}
        description={`Do you want to delete this question?`}
      />
    </>
```

```
  );
};

export default Question;
```

QuestionList.js:

```
import { Grid } from '@mui/material';
import Question from './Question';
import AddQuestionCard from './AddQuestionCard';
import { useContext, useEffect, useState } from 'react';
import UserContext from '../../../Authorization/UserContext';
import CourseService from '../../../../services/CourseService';
import { useParams } from 'react-router-dom';

const QuestionList = ({ questions, setAnswers, setRepliesAnswersId, setQues-
tions, isQuizStarted }) => {
  const { user } = useContext(UserContext);
  const { course: courseId } = useParams();
  const [teacherAccess, setTeacherAccess] = useState(false);

  useEffect(() => {
    CourseService.getTeacherAccess(user?.id, courseId).then((teacherAccess) => {
      setTeacherAccess(() => teacherAccess);
    });
  }, [user?.id, courseId]);

  return (
    <Grid container direction="column" justifyContent="center"
alignItems="stretch" spacing={2}>
      {teacherAccess && (
        <Grid item>
          <AddQuestionCard setQuestions={setQuestions} />
        </Grid>
      )}

      <Grid item>
        {questions.map((question, index) => (
          <Question
            key={index}
            question={question}
            index={index}
            amount={questions.length}
            setAnswers={setAnswers}
            setRepliesAnswersId={setRepliesAnswersId}
            teacherAccess={teacherAccess}
            setQuestions={setQuestions}
          />
        ))}
      </Grid>
    </Grid>
  );
};

export default QuestionList;
```

QuestionModal.js:

```
import * as yup from 'yup';
import Modal from '../../Shared/Modal';
import StackForm from '../../Shared/StackForm';
```

```
const initialValues = {
  text: '',
};

const labels = {
  text: 'Question',
};

const types = {
  text: 'text',
};

const validationSchema = yup.object({
  text: yup.string('Enter the question').required('Question is required'),
});

const QuestionModal = ({ open, onClose }) => {
  const onAddQuestionHandler = async (values) => {
    const formData = new FormData();
    for (let value in values) {
      formData.append(value, values[value]);
    }
    await onAddQuestionHandler.create(formData);
    onClose();
  };

  const title = 'Create Institution';

  return (
    <Modal title={title} open={open} onClose={onClose}>
      <StackForm
        initialValues={initialValues}
        validationSchema={validationSchema}
        onSubmit={onAddQuestionHandler}
        types={types}
        labels={labels}
      />
    </Modal>
  );
};

export default QuestionModal;
```

QuestionsPanel.js:

```
import { Flag, Star, StarOutline } from '@mui/icons-material';
import {
  Avatar,
  Box,
  Button,
  Card,
  CardActions,
  Chip,
  Divider,
  List,
  ListItem,
  ListItemButton,
  ListItemIcon,
  ListItemText,
  Typography,
} from '@mui/material';
import Timer from './Timer';
```

```
import React, { useContext, useState } from 'react';
import UserContext from '../../../Authorization/UserContext';
import ReplyService from '../../../../services/ReplyService';
import QuizService from '../../../../services/QuizService';

const TEXT_MAX_LENGTH = 90;

const QuestionsPanel = ({
  time,
  questions,
  repliesAnswersId,
  quizId,
  isQuizStarted,
  setIsQuizStarted,
  isQuizFinished,
  setIsQuizFinished,
}) => {
  const [mark, setMark] = useState(null);
  const { user } = useContext(UserContext);

  const onFinishQuizHandler = () => {
    setIsQuizFinished(() => true);
    Promise.all(
      repliesAnswersId.map((replyAnsweId) => ReplyService.create({ answerId: re-
plyAnsweId, personId: user.id })),
    ).then(() =>
      QuizService.getOneWithMark(quizId, user?.id).then((quiz) => {
        setMark(() => quiz[0].mark);
      }),
    );
  };

  return (
    <Card>
      <Box>
        {mark && (
          <Box sx={{ display: 'flex', justifyContent: 'center', alignItems:
'center', mt: 1 }}>
            <Avatar sx={{ bgcolor: 'primary.main', width: 75, height: 75 }}
alt=" ">
              <Box sx={{ display: 'flex', flexDirection: 'column' }}>
                <Box sx={{ display: 'flex', justifyContent: 'center',
alignItems: 'center' }}>
                  <Typography sx={{ fontSize: '22px' }}>{mark}</Typography>
                  <Star />
                </Box>
                <Typography sx={{ fontSize: '14px', mt: -0.8 }}>Mark</Typogra-
phy>
              </Box>
            </Avatar>
          </Box>
        )}
        <Timer
          time={time}
          isQuizStarted={isQuizStarted}
          isQuizFinished={isQuizFinished}
          setIsQuizStarted={setIsQuizStarted}
        />
      </Box>
      <List>
        {questions.map((question, index) => (
          <React.Fragment key={index}>
            <ListItem disablePadding>
```

```
              <ListItemButton>
                <ListItemIcon>
                  <Flag />
                </ListItemIcon>
                <ListItemText
                  primary={
                    question?.text.length > TEXT_MAX_LENGTH
                      ? question?.text.substring(0, TEXT_MAX_LENGTH) + '...'
                      : question?.text
                  }
                  sx={{ color: '#808080' }}
                />
              </ListItemButton>
            </ListItem>
            <Divider />
          </React.Fragment>
        ))}
      </List>
      <CardActions sx={{ justifyContent: 'center' }}>
        {!isQuizStarted && (
          <Button variant="contained" onClick={() => setIsQuizStarted(() =>
true)}>
            Start
          </Button>
        )}
        {isQuizStarted && (
          <Button variant="contained" onClick={onFinishQuizHandler} disa-
bled={isQuizFinished}>
            Submit
          </Button>
        )}
      </CardActions>
    </Card>
  );
};

export default QuestionsPanel;
```

Timer.js:

```
import { Box, Typography } from '@mui/material';
import React, { useState, useEffect } from 'react';

const Timer = ({ time, isQuizStarted, isQuizFinished, setIsQuizStarted }) => {
  const [remainingTime, setRemainingTime] = useState(time);

  useEffect(() => {
    let countdown;
    if (isQuizStarted) {
      countdown = setInterval(() => {
        setRemainingTime((prevTime) => {
          if (prevTime === '00:00:00') {
            clearInterval(countdown);
            return prevTime;
          }

          const timeArray = prevTime.split(':');
          const hours = parseInt(timeArray[0]);
          const minutes = parseInt(timeArray[1]);
          const seconds = parseInt(timeArray[2]);

          let newHours = hours;
```

```
          let newMinutes = minutes;
          let newSeconds = seconds - 1;

          if (newSeconds < 0) {
            newSeconds = 59;
            newMinutes -= 1;
          }
          if (newMinutes < 0) {
            newMinutes = 59;
            newHours -= 1;
          }

          return `${formatTimeUnit(newHours)}:${formatTimeUnit(new-
Minutes)}:${formatTimeUnit(newSeconds)}`;
        });
      }, 1000);
    } else {
      return () => clearInterval(countdown);
    }

  }, [isQuizStarted, isQuizFinished]);

  const formatTimeUnit = (unit) => {
    return unit < 10 ? `0${unit}` : unit.toString();
  };

  return (
    <Box sx={{ my: 1 }}>
      <Typography sx={{ fontSize: '22px' }}>
        {isQuizStarted ? (isQuizFinished ? '00:00:00' : remainingTime) : time}
      </Typography>
    </Box>
  );
};

export default Timer;
```

QuizPage.js:

```
import { Alert, AlertTitle, Box, Button, Card, CardContent, Grid, Typography }
from '@mui/material';
import QuestionList from './Question/QuestionList';
import QuestionsPanel from './Question/QuestionsPanel';
import { useNavigate, useParams } from 'react-router-dom';
import { useContext, useEffect, useState } from 'react';
import QuizService from '../../../services/QuizService';
import QuestionService from '../../../services/QuestionService';
import UserContext from '../../Authorization/UserContext';
import CourseService from '../../../services/CourseService';

const QuizPage = () => {
  const { quiz: quizId, course: courseId } = useParams();
  const [quiz, setQuiz] = useState({});
  const [mark, setMark] = useState(null);
  const [questions, setQuestions] = useState([]);
  const [answers, setAnswers] = useState([]);
  const [repliesAnswersId, setRepliesAnswersId] = useState([]);
  const [isQuizStarted, setIsQuizStarted] = useState(false);
  const [isQuizFinished, setIsQuizFinished] = useState(false);
  const { user } = useContext(UserContext);
  const [teacherAccess, setTeacherAccess] = useState(false);
  const navigate = useNavigate();
```

```
  useEffect(() => {
    CourseService.getTeacherAccess(user?.id, courseId).then((teacherAccess) => {
      setTeacherAccess(() => teacherAccess);
    });
    QuizService.getOne(quizId).then((quiz) => {
      setQuiz(() => quiz);
    });
    QuizService.getOneWithMark(quizId, user?.id).then((quiz) => {
      if (+quiz[0].mark !== 0) {
        setMark(() => quiz[0].mark);
      }
    });
    QuestionService.getAll(quizId).then((questions) => {
      setQuestions(() => questions);
    });
  }, [quizId, user?.id, courseId]);

  console.log('mark', mark);

  return (
    <>
      {!mark || teacherAccess ? (
        <Grid container spacing={2}>
          <Grid item xs={8}>
            <QuestionList
              questions={questions}
              setAnswers={setAnswers}
              setRepliesAnswersId={setRepliesAnswersId}
              setQuestions={setQuestions}
            />
          </Grid>
          {questions.length ? (
            <Grid item xs>
              <QuestionsPanel
                time={new Date(quiz?.time).toLocaleTimeString()}
                questions={questions}
                repliesAnswersId={repliesAnswersId}
                quizId={quizId}
                isQuizStarted={isQuizStarted}
                setIsQuizStarted={setIsQuizStarted}
                isQuizFinished={isQuizFinished}
                setIsQuizFinished={setIsQuizFinished}
              />
            </Grid>
          ) : null}
        </Grid>
      ) : (
        <Card>
          <CardContent sx={{ display: 'flex', flexDirection: 'column',
alignItems: 'center' }}>
            <Typography variant="h4" gutterBottom>
              Quiz already passed!
            </Typography>
            <Typography variant="h5" gutterBottom>
              Your mark is — <strong>{mark}</strong>
            </Typography>
            <Button variant="contained" sx={{ mt: 1, width: '10%' }} onClick={()
=> navigate(-1)}>
              Go back
            </Button>
          </CardContent>
        </Card>
```

```
      )}
    </>
  );
};

export default QuizPage;
```

Quizzes.js:

```
import MaterialsList from '../Materials/MaterialsList';

const Quizzes = () => {
  return <MaterialsList type="quizzes" />;
};

export default Quizzes;
```

StudentsWork.js:

```
import { useState } from 'react';
import {
  Avatar,
  Box,
  Button,
  Chip,
  IconButton,
  ListItem,
  ListItemAvatar,
  ListItemIcon,
  ListItemText,
  TextField,
  Tooltip,
  Typography,
} from '@mui/material';
import { saveAs } from 'file-saver';
import { Assignment, Download, StarBorder } from '@mui/icons-material';
import FileService from '../../../services/FileService';
import configClient from '../../../configClient';
import MarkService from '../../../services/MarkService';

const StudentsWork = ({ workInfo }) => {
  const [mark, setMark] = useState(0);

  console.log("workInfo", workInfo);

  const onDownloadHandler = (workFile) => () => {
    console.log(workFile);
    FileService.downloadFile(workFile).then((response) => saveAs(response.data,
workFile));
  };

  const onEvaluateWork = (markInfo) => async () => {
    await MarkService.create(markInfo);
  };

  return (
    <ListItem
      sx={{ bgcolor: !workInfo.mark?.value ? '#fafafa' : '#d7e2f9' }}
    >
      <Box sx={{ display: 'flex', alignItems: 'center', width: '250px', mr:
10}}>
        <ListItemAvatar>
```

```
          <Avatar src={configClient.API_URL + '/' + workInfo.student?.avatar}
alt={workInfo.student?.fullName} />
        </ListItemAvatar>
        <ListItemText
          primary={workInfo.student?.fullName}
          secondary={`Was sent at - ${new Date(workInfo.work?.createdAt).toLo-
caleDateString()}`}
        />
      </Box>
      <Box sx={{ display: 'flex', alignItems: 'center', width: '220px', mr: 12
}}>
        <ListItemAvatar>
          <Assignment />
        </ListItemAvatar>
        <ListItemText
          sx={{ ml: -2 }}
          primary={workInfo.task?.title}
          secondary={`Deadline - ${new Date(workInfo.task?.deadline).toLocale-
DateString()}`}
        />
      </Box>
      <Box sx={{ display: 'flex', alignItems: 'center', width: '250px', mr: 5
}}>
        <Typography>{`${workInfo.student.fullName.replace(/ /g,'')}-
task№${workInfo.task.numberByOrder}`}</Typography>
        <ListItemIcon>
          <Tooltip title="Download file" placement="top">
            <IconButton onClick={onDownloadHandler(workInfo.work?.workFile)}>
              <Download />
            </IconButton>
          </Tooltip>
        </ListItemIcon>
      </Box>
      <Box sx={{ display: 'flex', width: '230px',  alignItems: 'center' }}>
        {!workInfo.mark?.value ? (
          <>
            <TextField
              type="number"
              label="Mark"
              variant="outlined"
              size="small"
              value={mark}
              onChange={(e) => setMark(e.target.value)}
            />
            <Button
              variant="contained"
              sx={{ ml: 1 }}
              onClick={onEvaluateWork({ workId: workInfo.work?.id, personId:
workInfo.student?.id, value: mark })}
            >
              Evaluate
            </Button>
          </>
        ) : (
          <Chip label={workInfo.mark?.value} sx={{ bgcolor: '#fafafa', fontSize:
'16px' }} icon={<StarBorder />} />
        )}
      </Box>
    </ListItem>
  );
};

export default StudentsWork;
```

StudentsWorks.js:

```
import React, { useContext, useEffect, useState } from 'react';
import { Divider, List } from '@mui/material';
import TaskService from '../../../services/TaskService';
import CourseContext from '../CourseContext';
import StudentsWork from './StudentsWork';
import MarkService from '../../../services/MarkService';

const StudentsWorks = () => {
  const [works, setWorks] = useState([]);
  const { students } = useContext(CourseContext);

  useEffect(() => {
    students.forEach((student) => {
      TaskService.getAll(student.id).then((tasks) => {
        tasks.forEach((task) =>
          task.works.forEach((work) => {
            MarkService.getOne(work.id, student.id).then((mark) => {
              setWorks((prev) => [...prev, { student: student, work: work, task:
task, mark: mark }]);
            });
          }),
        );
      });
    });
  }, [students]);

  return (
    <List sx={{ mt: 3 }}>
      {students &&
        works.map((workInfo, index) => (
          <React.Fragment key={index}>
            <StudentsWork workInfo={workInfo} />
            {works.length - 1 !== index && <Divider />}
          </React.Fragment>
        ))}
    </List>
  );
};

export default StudentsWorks;
```

Tasks.js:

```
import MaterialsList from '../Materials/MaterialsList';

const Tasks = () => {
  return (
    <>
      <MaterialsList type="tasks" />
    </>
  );
};

export default Tasks;
```

AllCourses.js:

```
import { useEffect, useState } from 'react';
import { Grid } from '@mui/material';
```

```
import Course from './Course';
import CourseService from '../../services/CourseService';
import { useParams } from 'react-router-dom';

const AllCourses = ({ userId }) => {
  const [coursesList, setCoursesList] = useState([]);
  const { institution: institutionId } = useParams();

  useEffect(() => {
    CourseService.getAllCoursesByInstitution(institutionId).then(
      (courses) => {
        setCoursesList(() => courses);
      },
      (error) => {
        console.log(error);
      },
    );
  }, [institutionId]);

  return (
    <Grid container spacing={4} sx={{ justifyContent: 'center' }}>
      {coursesList.map((course) => {
        return <Course key={course.id} course={course} />;
      })}
    </Grid>
  );
};

export default AllCourses;
```

Course.js:

```
import {
  Card,
  CardContent,
  Typography,
  CardMedia,
  Box,
  CardActionArea,
  Grid,
  CardActions,
  IconButton,
  Tooltip,
} from '@mui/material';
import configClient from '../../configClient';
import { Link } from 'react-router-dom';
import { Delete, Edit } from '@mui/icons-material';
import { useContext, useEffect, useState } from 'react';
import CourseCreationForm from './CourseCreationForm';
import Modal from '../Shared/Modal';
import DeleteModal from '../Shared/components/DeleteModal';
import CourseService from '../../services/CourseService';
import UserContext from '../Authorization/UserContext';

function Course({ course }) {
  const { user } = useContext(UserContext);
  const [isEditOpen, setIsEditOpen] = useState(false);
  const [isDeleteOpen, setIsDeleteOpen] = useState(false);
  const [teacherAccess, setTeacherAccess] = useState(false);
  const IMG_URL = configClient.API_URL + '/' + course.image;
  const cardSize = 320;
```

```
  useEffect(() => {
    CourseService.getTeacherAccess(user?.id, course.id).then((teacherAccess) =>
{
      setTeacherAccess(() => teacherAccess);
    });
  }, [user?.id, course.id]);

  const onEditOpenHandler = () => {
    setIsEditOpen(() => true);
  };

  const onEditCloseHandler = () => {
    setIsEditOpen(() => false);
  };

  const onDeleteOpenHandler = () => {
    setIsDeleteOpen(() => true);
  };

  const onDeleteCloseHandler = () => {
    setIsDeleteOpen(() => false);
  };

  const onDeleteHandler = async () => {
    await CourseService.delete(course.id);
  };

  return (
    <Grid item>
      <Card sx={{ width: cardSize }}>
        <CardActionArea component={Link} to={`courses/${course.id}`}>
          <Box sx={{ width: cardSize, height: '14rem' }}>
            <CardMedia
              component="img"
              image={IMG_URL}
              alt="Course"
              sx={{ objectFit: 'cover', width: '100%', height: '100%' }}
            />
          </Box>
        </CardActionArea>
        <CardContent sx={{ pb: 0 }}>
          <Typography gutterBottom variant="h5" component="div">
            {course.title}
          </Typography>
          <Typography variant="body2" color="text.secondary" sx={{ height:
'50px', overflow: 'scroll' }}>
            {course.description}
          </Typography>
        </CardContent>
        {teacherAccess && (
          <CardActions sx={{ justifyContent: 'space-between', pt: 0 }}>
            <Tooltip title="Delete" placement="top">
              <IconButton onClick={onDeleteOpenHandler}>
                <Delete />
              </IconButton>
            </Tooltip>
            <Tooltip title="Edit" placement="top">
              <IconButton onClick={onEditOpenHandler}>
                <Edit />
              </IconButton>
            </Tooltip>
          </CardActions>
        )}
```

```
      </Card>
      <Modal title="Edit course" open={isEditOpen} onClose={onEditCloseHandler}>
        <CourseCreationForm item={course} />
      </Modal>
      <DeleteModal
        open={isDeleteOpen}
        onClose={onDeleteCloseHandler}
        onDelete={onDeleteHandler}
        title={`Delete course?`}
        description={`Do you want to delete this course?`}
      />
    </Grid>
  );
}

export default Course;
```

CourseContext.js:

```
import { createContext } from "react";

const CourseContext = createContext({ teacherAccess: false, group: null });

export default CourseContext;
```

CourseCreationForm.js:

```
import * as yup from 'yup';
import CourseService from '../../services/CourseService';
import GroupsAutocomplete from '../Group/GroupsAutocomplete';
import StackForm from '../Shared/StackForm';
import UsersAutocomplete from '../User/UsersAutocomplete';
import { Box, Typography } from '@mui/material';
import { useContext, useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import UserContext from '../Authorization/UserContext';
import UserService from '../../services/UserService';

const labels = {
  title: 'Title',
  description: 'Description',
  image: 'Choose image',
};

const types = {
  title: 'text',
  description: 'text',
  image: 'file',
};

const validationSchema = yup.object({
  title: yup.string('Enter the title of course').required('Name is required'),
  description: yup.string('Enter the description of course').required('Descrip-
tion is required'),
  image: yup.string('Enter the image of course').required('Image is required'),
});

const CourseCreationForm = ({ item }) => {
  const { institution: institutionId } = useParams();
  const [teachers, setTeachers] = useState([]);
  const [groups, setGroups] = useState([]);
```

```
const initialValues = {
  title: item?.title || '',
  description: item?.description || '',
  image: item?.image || '',
};

const onCreateCourseHandler = async (values, { resetForm }) => {
  const formData = new FormData();
  for (let value in values) {
    formData.append(value, values[value]);
  }
  formData.append('institutionId', institutionId);
  let res;
  if (item) {
    res = await CourseService.update(item.id, formData);
  } else {
    res = await CourseService.create(formData);
  }

  for (let teacher of teachers) {
    if (item) {
      await CourseService.updateTeachersInCourse(teacher.id, res.data.id);
    } else {
      await CourseService.addTeachersToCourse(teacher.id, res.data.id);
    }
  }
  for (let group of groups) {
    if (item) {
      await CourseService.updateGroupsInCourse(group.id, res.data.id);
    } else {
      await CourseService.addGroupsToCourse(group.id, res.data.id);
    }
  }
  resetForm(initialValues);
  setTeachers(() => []);
  setGroups(() => []);
};

return (
  <Box
    sx={
      !item && {
        p: 2,
        bgcolor: '#fff',
        borderRadius: 2,
        boxShadow: '-5px 5px 7px rgb(208, 212, 222)',
      }
    }
  >
    {!item && (
      <Typography variant="h5" gutterBottom color="#887dca">
        Create Course
      </Typography>
    )}

    <StackForm
      initialValues={initialValues}
      validationSchema={validationSchema}
      onSubmit={onCreateCourseHandler}
      types={types}
      labels={labels}
    >
```

```
        <UsersAutocomplete people={teachers} setPeople={setTeachers} la-
bel="teachers" />
        <GroupsAutocomplete groups={groups} setGroups={setGroups} label="groups"
/>
      </StackForm>
    </Box>
  );
};

export default CourseCreationForm;
```

GroupChip.js:

```
import { Chip } from '@mui/material';
import PeopleIcon from '@mui/icons-material/People';

function GroupChip({ group, ...props }) {
  return (
    <Chip
      key={group.id}
      sx={{ margin: 0.5 }}
      color="primary"
      icon={<PeopleIcon />}
      label={group.name}
      {...props}
    />
  );
}

export default GroupChip;
```

GroupCreationForm.js:

```
import { Autocomplete, Box, TextField, Typography } from '@mui/material';
import React, { useState, useEffect } from 'react';
import * as yup from 'yup';
import GroupService from '../../services/GroupService';
import UserService from '../../services/UserService';
import StackForm from '../Shared/StackForm';
import UserChip from '../User/UserChip';
import UserLineCard from '../User/UserLineCard';

const initialValues = {
  name: '',
};

const labels = {
  name: 'Name',
};

const types = {
  name: 'text',
};

const validationSchema = yup.object({
  name: yup.string('Enter the name of group').required('Name is required'),
});

const preventErase = (event) => {
  if (event.key === 'Backspace' || event.key === 'Delete') {
    event.stopPropagation();
  }
```

```
};

function GroupFormModal() {
  useEffect(() => {
    UserService.getAllPersons().then((people) => {
      setUsers(people.data);
    });
  }, []);

  const [users, setUsers] = useState([]);
  const [students, setStudents] = useState([]);

  const onCreateGroupHandler = async (values, { resetForm }) => {
    const res = await GroupService.create(values);
    for (let student of students) {
      await GroupService.createStudentsInGroup(student.id, res.data.id);
    }
    resetForm(initialValues);
    setStudents(() => []);
  };

  return (
    <Box
      sx={{
        p: 2,
        bgcolor: '#fff',
        borderRadius: 2,
        boxShadow: '-5px 5px 7px rgb(208, 212, 222)',
      }}
    >
      <Typography variant="h5" gutterBottom color="#887dca">
        Create Group
      </Typography>
      <StackForm
        initialValues={initialValues}
        validationSchema={validationSchema}
        onSubmit={onCreateGroupHandler}
        types={types}
        labels={labels}
      >
        <Autocomplete
          multiple
          freeSolo
          noOptionsText={'No students'}
          limitTags={1}
          id="multiple-limit-tags"
          options={users}
          isOptionEqualToValue={(option, value) => option.id === value.id}
          filterSelectedOptions
          value={students}
          onChange={(event, value) => setStudents((prev) => value)}
          getOptionLabel={(user) => user.fullName}
          renderInput={(params) => (
            <TextField label="Choose students" placeholder="Students"
onKeyDown={preventErase} {...params} />
          )}
          renderTags={(value, getTagProps) =>
            value.map((user, index) => <UserChip key={user.id} user={user}
{...getTagProps({ index })} />)
          }
          renderOption={(props, user, state) => <UserLineCard key={user.id}
user={user} {...props} />}
        />
```

```
      </StackForm>
    </Box>
  );
}

export default GroupFormModal;
```

GroupLineCard.js:

```
import { Divider, ListItem, ListItemButton, ListItemText } from '@mui/material';

function GroupLineCard({ group, ...props }) {
  return (
    <>
      <ListItem disablePadding>
        <ListItemButton {...props}>
          <ListItemText primary={group.name} />
        </ListItemButton>
      </ListItem>
      <Divider variant="inset" component="li" />
    </>
  );
}

export default GroupLineCard;
```

GroupsAutocomplete.js:

```
import { Autocomplete, TextField } from '@mui/material';
import { useState, useEffect } from 'react';
import GroupService from '../../services/GroupService';
import GroupChip from './GroupChip';
import GroupLineCard from './GroupLineCard';

const preventErase = (event) => {
  console.log(event.key);
  if (event.key === 'Backspace' || event.key === 'Delete') {
    event.stopPropagation();
  }
  console.log(event.isPropagationStopped());
};

function GroupsAutocomplete({ groups, setGroups, label }) {
  const [allGroups, setAllGroups] = useState([]);

  useEffect(() => {
    GroupService.getAll().then((groups) => {
      setAllGroups(groups);
    });
  }, []);

  return (
    <Autocomplete
      multiple
      freeSolo
      noOptionsText={`No ${label}`}
      limitTags={1}
      id="multiple-limit-tags"
      options={allGroups}
      isOptionEqualToValue={(option, value) => option.id === value.id}
      filterSelectedOptions
      value={groups}
```

```
      onChange={(event, value) => setGroups((prev) => value)}
      getOptionLabel={(group) => group.name}
      renderInput={(params) => (
        <TextField
          label={`Choose ${label}`}
          placeholder={label[0].toUpperCase() + label.slice(1)}
          onKeyDown={preventErase}
          {...params}
        />
      )}
      renderTags={(value, getTagProps) =>
        value.map((group, index) => <GroupChip key={group.id} group={group}
{...getTagProps({ index })} />)
      }
      renderOption={(props, group, state) => <GroupLineCard key={group.id}
group={group} {...props} />}
    />
  );
}

export default GroupsAutocomplete;
```

Institution.js:

```
import { Card, CardContent, Typography, CardMedia, Box, CardActionArea, Grid }
from '@mui/material';
import configClient from '../../configClient';
import { Link } from 'react-router-dom';

function Institution({ institution }) {
  const IMG_URL = configClient.API_URL + '/' + institution.image;
  const cardSize = 330;

  return (
    <Grid item>
      <Card sx={{ width: cardSize }}>
        <CardActionArea component={Link} to={`${institution.id}`} sx={{ height:
350 }}>
          <Box sx={{ width: cardSize, height: '14rem' }}>
            <CardMedia
              component="img"
              image={IMG_URL}
              alt="Institution"
              sx={{ objectFit: 'cover', width: '100%', height: '100%' }}
            />
          </Box>
          <CardContent >
            <Typography gutterBottom variant="h5" component="div">
              {institution.name}
            </Typography>
            <Typography variant="body2" color="text.secondary">
              {institution.description}
            </Typography>
          </CardContent>
        </CardActionArea>
      </Card>
    </Grid>
  );
}

export default Institution;
```

InstitutionCreationCard.js:

```
import CreationCard from '../Shared/CreationCard';

function InstitutionCreationCard(props) {
  const actionTitle = 'Create Institution';
  const actionDescription = 'Create your own institution to create courses and
share knowledges';
  const onClickHandler = () => {
    props.setIsInstitutionModalOpen(!props.isInstitutionModalOpen);
  };
  return <CreationCard actionTitle={actionTitle} actionDescription={actionDe-
scription} onClick={onClickHandler} />;
}

export default InstitutionCreationCard;
```

InstitutionFormModal.js:

```
import * as yup from 'yup';
import InstitutionService from '../../services/InstitutionService';
import Modal from '../Shared/Modal';
import StackForm from '../Shared/StackForm';

const initialValues = {
  name: '',
  description: '',
  image: '',
};

const labels = {
  name: 'Name',
  description: 'Description',
  image: 'Choose image',
};

const types = {
  name: 'text',
  description: 'text',
  image: 'file',
};

const validationSchema = yup.object({
  name: yup.string('Enter the name of institution').required('Name is re-
quired'),
  description: yup.string('Enter the description of institution').required('De-
scription is required'),
  image: yup.string('Enter the image of institution').required('Image is re-
quired'),
});

function InstitutionFormModal(props) {
  const onCreateInstitutionHandler = async (values) => {
    const formData = new FormData();
    for (let value in values) {
      formData.append(value, values[value]);
    }
    await InstitutionService.create(formData);
    onClose();
  };

  const onClose = () => {
```

```
    props.onClose();
  };

  const title = 'Create Institution';

  return (
    <Modal title={title} open={props.open} onClose={onClose}>
      <StackForm
        initialValues={initialValues}
        validationSchema={validationSchema}
        onSubmit={onCreateInstitutionHandler}
        types={types}
        labels={labels}
      />
    </Modal>
  );
}

export default InstitutionFormModal;
```

InstitutionMainPage.js:

```
import Institution from './Institution';
import InstitutionService from '../../services/InstitutionService';
import { useContext, useEffect, useState } from 'react';
import { Grid, Typography } from '@mui/material';
import UserContext from '../Authorization/UserContext';
import { UnauthorizedPage } from '../Shared/ErrorPages/UnauthorizedPage';
import InstitutionCreationCard from './InstitutionCreationCard';
import InstitutionFormModal from './InstitutionFormModal';

const InstitutionMainPage = () => {
  const [isInstitutionModalOpen, setIsInstitutionModalOpen] = useState(false);
  const [ownInstitutionsList, setOwnInstitutionsList] = useState([]);
  const [institutionsList, setInstitutionsList] = useState([]);
  const { user } = useContext(UserContext);

  useEffect(() => {
    if (user) {
      InstitutionService.getAllByPerson(user.id).then((institutions) => {
        setOwnInstitutionsList(
          institutions.map((institution) => {
            return <Institution key={institution.id} institution={institution}
/>;
          }),
        );
      });
      InstitutionService.getAllByUser(user.id).then((institutions) => {
        setInstitutionsList(
          institutions.map((institution) => {
            return <Institution key={institution.id} institution={institution}
/>;
          }),
        );
      });
    }
  }, [user]);

  if (!user) {
    return <UnauthorizedPage />;
  }
```

```
  return (
    <Grid container spacing={4} flexDirection="column">
      <Grid item>
        <InstitutionCreationCard
          isInstitutionModalOpen={isInstitutionModalOpen}
          setIsInstitutionModalOpen={setIsInstitutionModalOpen}
        />
        {isInstitutionModalOpen ? (
          <InstitutionFormModal open={isInstitutionModalOpen} onClose={setIsIn-
stitutionModalOpen} />
        ) : (
          <></>
        )}
      </Grid>
      {ownInstitutionsList.length || institutionsList.length ? (
        <>
          {ownInstitutionsList.length ? (
            <Grid item sx={{ display: 'flex', alignItems: 'flex-start',
flexDirection: 'column' }}>
              <Typography variant="h6" sx={{ mb: 2 }}>
                Own institutions
              </Typography>
              <Grid container spacing={2}>
                <Grid item>{ownInstitutionsList}</Grid>
              </Grid>
            </Grid>
          ) : (
            <></>
          )}
          {institutionsList.length ? (
            <Grid item sx={{ display: 'flex', alignItems: 'flex-start',
flexDirection: 'column' }}>
              <Typography variant="h6" sx={{ mb: 2 }}>
                My institutions
              </Typography>{' '}
              {institutionsList}
            </Grid>
          ) : (
            <></>
          )}
        </>
      ) : (
        <></>
      )}
    </Grid>
  );
};

export default InstitutionMainPage;
```

InstitutionOwnPage.js:

```
import { Box, Grid, Tab, Tabs } from '@mui/material';
import { useContext, useState } from 'react';
import AllCourses from '../Course/AllCourses';
import CourseCreationForm from '../Course/CourseCreationForm';
import GroupCreationForm from '../Group/GroupCreationForm';
import { TabPanel, a11yProps } from '../Shared/components/TabPanel';
import UserContext from '../Authorization/UserContext';

function InstitutionOwnPage() {
  const [value, setValue] = useState(0);
```

```
  const { user } = useContext(UserContext);

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Box>
        <Tabs value={value} onChange={handleChange} aria-label="basic tabs exam-
ple">
          <Tab label="Courses" {...a11yProps(0)} />
          <Tab label="Setup" {...a11yProps(1)} />
        </Tabs>
      </Box>
      <TabPanel value={value} index={0}>
        {user && <AllCourses userId={user.id} />}
      </TabPanel>
      <TabPanel value={value} index={1}>
        <Grid container spacing={4}>
          <Grid item xs={5}>
            <GroupCreationForm />
          </Grid>
          <Grid item xs={7}>
            <CourseCreationForm />
          </Grid>
        </Grid>
      </TabPanel>
    </Box>
  );
}

export default InstitutionOwnPage;
```

MainPage.js:

```
import { Box, Button, Grid, SvgIcon, Typography } from '@mui/material';
import { ReactComponent as Education } from '../../icons/Education.svg';
import { useContext } from 'react';
import UserContext from '../Authorization/UserContext';

const MainPage = () => {
  const { openAuthModal } = useContext(UserContext);

  return (
    <Grid container spacing={2} justifyContent="center">
      <Grid item sx={{ display: 'flex' }}>
        <Box
          sx={{
            display: 'flex',
            flexDirection: 'column',
            alignItems: 'flex-start',
            justifyContent: 'center',
            textAlign: 'left',
          }}
        >
          <Typography variant="h2" gutterBottom>
            Online education
          </Typography>
          <Typography variant="subtitle" sx={{ width: '60%' }}>
            <strong>
```

```
              Become one of the many who have joined us to learn easily and con-
veniently at any time and from anywhere.
            </strong>
          </Typography>
          <Button onClick={openAuthModal} sx={{ mt: 4, px: 10, borderRadius: 10
}} variant="contained">
            Sign In
          </Button>
        </Box>
        <SvgIcon component={Education} inheritViewBox sx={{ fontSize: 550, ml: -
40 }} />
      </Grid>
    </Grid>
  );
};

export default MainPage;
```

Navbar.js:

```
import React, { useContext, useState } from 'react';
import { SvgIcon } from '@mui/material';
import { ReactComponent as AlmaMater } from '../../icons/AlmaMater.svg';
import { Link, useNavigate } from 'react-router-dom';
import { styled, useTheme } from '@mui/material/styles';
import {
  Box,
  Drawer,
  ListItemText,
  CssBaseline,
  AppBar as MuiAppBar,
  Toolbar,
  List,
  Typography,
  Divider,
  IconButton,
  ListItem,
  ListItemButton,
  ListItemIcon,
  Menu,
  MenuItem,
  Avatar,
  Container,
} from '@mui/material';
import {
  Menu as MenuIcon,
  ChevronLeft as ChevronLeftIcon,
  ChevronRight as ChevronRightIcon,
  Person as PersonIcon,
  Logout as LogoutIcon,
  School as SchoolIcon,
  Person,
  Home,
  LightMode,
  DarkMode,
  SettingsSuggest,
  Notifications,
  Message,
} from '@mui/icons-material';
import UserContext from '../Authorization/UserContext';
import configClient from '../../configClient';
import AuthService from '../../services/AuthService';
```

```
const drawerWidth = 250;

const Main = styled('main', { shouldForwardProp: (prop) => prop !== 'open' })(({
theme, open }) => ({
  flexGrow: 1,
  padding: theme.spacing(4),
  transition: theme.transitions.create('margin', {
    easing: theme.transitions.easing.sharp,
    duration: theme.transitions.duration.leavingScreen,
  }),
  marginLeft: `-${drawerWidth}px`,
  ...(open && {
    transition: theme.transitions.create('margin', {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
    marginLeft: 0,
  }),
}));

const AppBar = styled(MuiAppBar, {
  shouldForwardProp: (prop) => prop !== 'open',
})(({ theme, open }) => ({
  transition: theme.transitions.create(['margin', 'width'], {
    easing: theme.transitions.easing.sharp,
    duration: theme.transitions.duration.leavingScreen,
  }),
  ...(open && {
    width: `calc(100% - ${drawerWidth}px)`,
    marginLeft: `${drawerWidth}px`,
    transition: theme.transitions.create(['margin', 'width'], {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
  }),
}));

const DrawerHeader = styled('div')(({ theme }) => ({
  display: 'flex',
  alignItems: 'center',
  padding: theme.spacing(0, 1),
  ...theme.mixins.toolbar,
  justifyContent: 'flex-end',
}));

const paperProps = {
  elevation: 0,
  sx: {
    overflow: 'visible',
    filter: 'drop-shadow(0px 2px 8px rgba(0,0,0,0.32))',
    mt: 1.5,
    '& .MuiAvatar-root': {
      width: 32,
      height: 32,
      ml: -0.5,
      mr: 1,
    },
    '&:before': {
      content: '""',
      display: 'block',
      position: 'absolute',
      top: 0,
```

```
      right: 14,
      width: 10,
      height: 10,
      bgcolor: 'background.paper',
      transform: 'translateY(-50%) rotate(45deg)',
      zIndex: 0,
    },
  },
};

const themeIcons = {
  light: <LightMode color="#887dca" />,
  dark: <DarkMode color="#887dca" />,
  system: <SettingsSuggest color="#887dca" />,
};

export default function Navbar(props) {
  const theme = useTheme();
  const [open, setOpen] = useState(false);
  const [anchorEl, setAnchorEl] = useState(null);
  const { user, setUser } = useContext(UserContext);
  const navigate = useNavigate();

  const handleDrawerOpen = () => {
    setOpen(true);
  };

  const handleDrawerClose = () => {
    setOpen(false);
  };

  const handleMenu = (event) => {
    setAnchorEl(event.currentTarget);
  };

  const handleClose = () => {
    setAnchorEl(null);
  };

  const handleLogout = async () => {
    await AuthService.logout();
    localStorage.removeItem('access-token');
    setUser(null);
    navigate('./');
  };

  const { openAuthModal } = React.useContext(UserContext);

  return (
    <Box sx={{ display: 'flex' }}>
      <CssBaseline />
      <AppBar position="fixed" open={open}>
        <Toolbar sx={{ justifyContent: 'space-between' }}>
          <Box sx={{ display: 'flex', alignItems: 'center' }}>
            <IconButton
              color="inherit"
              aria-label="open drawer"
              onClick={handleDrawerOpen}
              edge="start"
              sx={{ mr: 2, ...(open && { display: 'none' }) }}
            >
              <MenuIcon />
            </IconButton>
```

```
            <SvgIcon component={AlmaMater} inheritViewBox fontSize="large" />
            <Typography
              variant="h6"
              noWrap
              component={Link}
              to="/"
              sx={{ textDecoration: 'none', color: 'primary.contrastText' }}
            >
              AlmaMater
            </Typography>
          </Box>
          {user ? (
            <>
              <Box sx={{ display: 'flex', alignItems: 'center' }}>
                <IconButton
                  size="small"
                  aria-label="account of current user"
                  aria-controls="menu-appbar"
                  aria-haspopup="true"
                  onClick={handleMenu}
                  color="inherit"
                >
                  <Avatar src={configClient.API_URL + '/' + user.avatar}
alt={user.full_name} />
                </IconButton>
              </Box>
              <Menu
                id="menu-appbar"
                anchorEl={anchorEl}
                keepMounted
                transformOrigin={{
                  horizontal: 'right',
                  vertical: 'top',
                }}
                anchorOrigin={{
                  horizontal: 'right',
                  vertical: 'bottom',
                }}
                open={!!anchorEl}
                onClose={handleClose}
                PaperProps={paperProps}
              >
                <MenuItem onClick={handleClose} component={Link} to="/profile">
                  <ListItemIcon>
                    <Person fontSize="small" />
                  </ListItemIcon>
                  Profile
                </MenuItem>
                <Divider />
                <MenuItem
                  onClick={(e) => {
                    handleClose(e);
                    handleLogout(e);
                  }}
                >
                  <ListItemIcon>
                    <LogoutIcon fontSize="small" />
                  </ListItemIcon>
                  Logout
                </MenuItem>
              </Menu>
            </>
          ) : (
```

```
              <IconButton
                size="large"
                aria-label="account of current user"
                aria-controls="menu-appbar"
                aria-haspopup="true"
                onClick={openAuthModal}
                color="inherit"
              >
                <PersonIcon fontSize="large" />
              </IconButton>
            )}
          </Toolbar>
        </AppBar>
        <Drawer
          sx={{
            width: drawerWidth,
            flexShrink: 0,
            '& .MuiDrawer-paper': {
              width: drawerWidth,
              boxSizing: 'border-box',
            },
          }}
          variant="persistent"
          anchor="left"
          open={open}
        >
          <DrawerHeader>
            <IconButton onClick={handleDrawerClose}>
              {theme.direction === 'ltr' ? <ChevronLeftIcon /> : <ChevronRightIcon
/>}
            </IconButton>
          </DrawerHeader>
          <Divider />
          <List>
            <ListItem disablePadding>
              <ListItemButton component={Link} to="/">
                <ListItemIcon>
                  <Home />
                </ListItemIcon>
                <ListItemText primary={'Home'}></ListItemText>
              </ListItemButton>
            </ListItem>
            <Divider />
            <ListItem disablePadding>
              <ListItemButton component={Link} to="/profile">
                <ListItemIcon>
                  <PersonIcon />
                </ListItemIcon>
                <ListItemText primary={'Profile'}></ListItemText>
              </ListItemButton>
            </ListItem>
            <Divider />
            <ListItem disablePadding>
              <ListItemButton component={Link} to="/institutions">
                <ListItemIcon>
                  <SchoolIcon />
                </ListItemIcon>
                <ListItemText primary={'Institutions'}></ListItemText>
              </ListItemButton>
            </ListItem>
            <Divider />
          </List>
          <Box sx={{ justifyContent: 'center', mt: 2 }}>
```

```
          {Object.keys(themeIcons).map((key) => (
            <IconButton>{themeIcons[key]}</IconButton>
          ))}
        </Box>
      </Drawer>
      <Main open={open}>
        <DrawerHeader />
        <Container disableGutters component="main" sx={{ py: '20px' }}>
          {props.children}
        </Container>
      </Main>
    </Box>
  );
}
```

EditUserDataForm.js:

```
import { Box, Typography } from '@mui/material';
import { useContext } from 'react';
import * as yup from 'yup';
import UserService from '../../../services/UserService';
import UserContext from '../../Authorization/UserContext';
import StackForm from '../../Shared/StackForm';

const labels = {
  fullName: 'Full Name',
  email: 'Email',
  birthday: 'Choose birthday',
  avatar: 'Choose avatar',
};

const types = {
  fullName: 'text',
  email: 'email',
  birthday: 'date',
  avatar: 'file',
};

const validationSchema = yup.object({
  fullName: yup.string('Enter your full name').required('Full name is re-
quired'),
  email: yup.string('Enter your email').email('Enter a valid email').re-
quired('Email is required'),
  birthday: yup.string('Enter your birthday').required('Birthday is required'),
  avatar: yup.string('Enter your avatar').required('Avatar is required'),
});

function EditUserDataForm(props) {
  const { user, setUser } = useContext(UserContext);

  const initialValues = {
    fullName: user.fullName,
    email: user.email,
    birthday: user.birthday,
    avatar: user.avatar,
  };

  const onEditHandler = async (values) => {
    const formData = new FormData();
    for (let value in values) {
      formData.append(value, values[value]);
    }
```

```
    const response = await UserService.updatePerson(user.id, formData);
    setUser(response.data);
    props.setIsEditFormOpen(false);
  };

  return (
    <Box
      sx={{
        p: 2,
        bgcolor: '#fff',
        borderRadius: 2,
        boxShadow: '-5px 5px 7px rgb(208, 212, 222)',
      }}
    >
      <Typography variant="h5" gutterBottom color="#887dca">
        Edit profile
      </Typography>
      <StackForm
        initialValues={initialValues}
        validationSchema={validationSchema}
        onSubmit={onEditHandler}
        types={types}
        labels={labels}
      />
    </Box>
  );
}

export default EditUserDataForm;
```

UserData.js:

```
import { Avatar, Typography, Box, List, ListItem, ListItemIcon, ListItemText }
from '@mui/material';
import { Email, Cake } from '@mui/icons-material';
import configClient from '../../../configClient';

function UserData({ user }) {
  const IMG_URL = configClient.API_URL + '/' + user?.avatar;

  const icons = {
    email: <Email sx={{ color: '#fff' }} fontSize="small" />,
    birthday: <Cake sx={{ color: '#fff' }} fontSize="small" />,
  };

  return (
    <Box
      sx={{
        width: 280,
        maxHeight: 600,
        bgcolor: '#fff',
        display: 'flex',
        flexDirection: 'column',
        alignContent: 'center',
        justifyContent: 'center',
        alignItems: 'center',
        borderRadius: 3,
        boxShadow: '-5px 5px 7px rgb(208, 212, 222)',
      }}
    >
      <Box
        sx={{
```

```
          bgcolor: 'primary.main',
          height: 100,
          width: '100%',
          display: 'flex',
          justifyContent: 'center',
          borderRadius: '12px 12px 0 0',
        }}
      >
        <Avatar
          sx={{ width: 120, height: 120, top: 30, border: '3px solid #fff' }}
          alt={user?.fullName[0].toUpperCase()}
          src={IMG_URL}
        />
      </Box>

      <Box sx={{ mt: 8, mb: 2 }}>
        <Typography gutterBottom variant="h5" sx={{ color: '#887dca' }}>
          {user?.fullName}
        </Typography>
        <List>
          {Object.keys(icons).map((item) => (
            <ListItem disablePadding key={item} sx={{ mb: 1 }}>
              <ListItemIcon
                sx={{
                  justifyContent: 'center',
                  color: 'neutral.white',
                  minWidth: 20,
                  mr: 1,
                  p: 1,
                  borderRadius: 100,
                  bgcolor: '#d7e2f9',
                }}
              >
                {icons[item]}
              </ListItemIcon>
              <ListItemText primary={user?.[item] || '-'} sx={{ color: '#808080'
}} />
            </ListItem>
          ))}
        </List>
      </Box>

      <Box
        sx={{
          height: 50,
          width: '100%',
          display: 'flex',
          justifyContent: 'center',
          alignItems: 'center',
          borderRadius: '0px  0px 12px 12px',
          borderTop: '1px solid #e0e0e0',
        }}
      >
        <Typography variant="body1" color="#887dca">
          Created at - {new Date(user?.createdAt).toLocaleDateString()}
        </Typography>
      </Box>
    </Box>
  );
}

export default UserData;
```

ProfilePage.js:

```
import { Grid } from '@mui/material';
import { useContext } from 'react';
import EditUserDataForm from './UserData/EditUserDataForm';
import UserData from './UserData/UserData';
import UserContext from '../Authorization/UserContext';
import { UnauthorizedPage } from '../Shared/ErrorPages/UnauthorizedPage';

const ProfilePage = () => {
  const { user } = useContext(UserContext);

  if (!user) {
    return <UnauthorizedPage />;
  }

  return (
    <Grid container spacing={4} justifyContent="center">
      <Grid item>
        <UserData user={user} />
      </Grid>
      <Grid item xs={8}>
        <EditUserDataForm />
      </Grid>
    </Grid>
  );
};

export default ProfilePage;
```

DeleteModal.js:

```
import {
  Button,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogContentText,
  DialogActions,
  IconButton,
} from '@mui/material';
import { Delete } from '@mui/icons-material';

const DeleteModal = ({ open, onClose, onDelete, title, description }) => {
  return (
    <Dialog
      open={open}
      onClick={onClose}
      maxWidth="sm"
      PaperProps={{
        sx: { borderRadius: 2, bgcolor: '#fce4ec', px: 2, py: 2 },
      }}
    >
      <DialogTitle variant="h5" sx={{ display: 'flex', alignItems: 'center', px:
1 }}>
        <IconButton>
          <Delete fontSize="large" sx={{ color: '#d32f2f' }} />
        </IconButton>
        {title}
      </DialogTitle>
      <DialogContent>
        <DialogContentText sx={{ mb: 1 }}>
```

```
          {description}
          <strong> This process cannot be undone!</strong>
        </DialogContentText>
      </DialogContent>
      <DialogActions>
        <Button variant="outlined" onClick={onClose}>
          Cancel
        </Button>
        <Button variant="contained" sx={{ bgcolor: '#d32f2f' }} on-
Click={onDelete}>
          Delete
        </Button>
      </DialogActions>
    </Dialog>
  );
};

export default DeleteModal;
```

EmptyPage.js:

```
import { Typography, Box } from '@mui/material';
import EmptyPageImage from '../../../icons/EmptyPage.svg';

export const EmptyPage = () => {
  return (
    <Box
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
      }}
    >
      <Box sx={{ height: 480, ml: -7, mb: -2}}>
        <img src={EmptyPageImage} alt="Empty page" height="100%" width="100%" />
      </Box>
      <Typography variant="h5">Here is empty for now</Typography>
    </Box>
  );
};
```

Form.js:

```
import { useFormik } from 'formik';

export const Form = ({ id, initialValues, validationSchema, fieldsRenderers, on-
Submit, children }) => {
  const formik = useFormik({
    initialValues,
    validationSchema,
    onSubmit,
  });

  return (
    <form
      onSubmit={formik.handleSubmit}
      style={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'flex-start',
      }}
```

```
    >
      {Object.keys(fieldsRenderers).map((item) => {
        return fieldsRenderers[item]({
          id: id + '-' + item,
          key: id + '-' + item,
          name: item,
          fullWidth: true,
          margin: 'normal',
          value: formik.values[item] || '',
          onChange: formik.handleChange,
          error: formik.touched[item] && Boolean(formik.errors[item]),
          helperText: formik.touched[item] && formik.errors[item],
        });
      })}
      {children}
    </form>
  );
};
```

TabPanel.js:

```
import { Grid } from '@mui/material';
import PropTypes from 'prop-types';

export function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <Grid
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && <Grid sx={{ py: 3, display: 'flex', flexWrap: 'wrap'
}}>{children}</Grid>}
    </Grid>
  );
}

TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.number.isRequired,
  value: PropTypes.number.isRequired,
};

export function a11yProps(index) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  };
}
```

CourseContext.js:

```
import { createContext } from "react";

const CourseContext = createContext();

export default CourseContext;
```

ErrorPage.js:

```
import { Typography, Box } from '@mui/material';
import ErrorPageImage from '../../../icons/ErrorPage.svg';

export const ErrorPage = () => {
  return (
    <Box
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
      }}
    >
      <Box sx={{ height: 480, ml: -7, mb: -1 }}>
        <img src={ErrorPageImage} alt="Error page" height="100%" width="100%" />
      </Box>
      <Typography variant="h5" sx={{ ml: -6 }}>
        The page you were looking for does not exist!
      </Typography>
    </Box>
  );
};
```

UnauthorizedPage.js:

```
import { Typography, Box, Button } from '@mui/material';
import { useContext } from 'react';
import UserContext from '../../Authorization/UserContext';
import UnauthorizedPageImage from '../../../icons/UnauthorizedPage.svg';

export const UnauthorizedPage = () => {
  const { openAuthModal } = useContext(UserContext);

  return (
    <Box
      sx={{
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        justifyContent: 'center',
      }}
    >
      <Box sx={{ height: 480, ml: -7, mb: -2 }}>
        <img src={UnauthorizedPageImage} alt="Unauthorized page" height="100%"
width="100%" />
      </Box>
      <Typography variant="h5" sx={{ ml: -3 }}>
        You are not anauthorized yet!
      </Typography>
      <Button onClick={openAuthModal} sx={{ ml: -3, mt: 2, px: 5 }} vari-
ant="contained">
        Sign In
      </Button>
    </Box>
  );
};
```

DateField.js:

```
import { DatePicker } from '@mui/x-date-pickers/DatePicker';
```

```
const DateField = ({ onChange, ...props }) => {
  const handlerChange = (value) => {
    onChange({ currentTarget: { name: props.name, value: value } });
  };
  return <DatePicker label="Choose date" onChange={handlerChange} sx={{ my: 1 }}
/>;
};

export default DateField;
```

PasswordField.js:

```
import { TextField, IconButton, InputAdornment } from '@mui/material';
import { Visibility, VisibilityOff } from '@mui/icons-material';
import { useState } from 'react';

export const PasswordField = (props) => {
  const [showPassword, setShowPassword] = useState(false);

  const handleClickShowPassword = () => setShowPassword((show) => !show);

  const handleMouseDownPassword = (event) => {
    event.preventDefault();
  };

  return (
    <TextField
      type={showPassword ? 'text' : 'password'}
      label="Your password"
      InputProps={{
        endAdornment: (
          <InputAdornment position="end">
            <IconButton onClick={handleClickShowPassword} onMouseDown={handle-
MouseDownPassword}>
              {showPassword ? <Visibility /> : <VisibilityOff />}
            </IconButton>
          </InputAdornment>
        ),
      }}
      {...props}
    />
  );
};
```

TimeCustomField.js:

```
import { TimeField } from '@mui/x-date-pickers';

const TimeCustomField = ({ onChange, ...props }) => {
  const handlerChange = (value) => {
    onChange({ currentTarget: { name: props.name, value: value } });
  };
  return <TimeField label="Choose time" onChange={handlerChange} for-
mat="HH:mm:ss" sx={{ my: 1 }} />;
};

export default TimeCustomField;
```

Router.js:

```
import { useRoutes } from 'react-router-dom';
```

```
import ProfilePage from '../../ProfilePage/ProfilePage';
import InstitutionMainPage from '../../Institution/InstitutionMainPage';
import InstitutionOwnPage from '../../Institution/InstitutionOwnPage';
import CoursePage from '../../Course/CoursePage/CoursePage';
import { ErrorPage } from '../ErrorPages/ErrorPage';
import MainPage from '../../MainPage/MainPage';
import MaterialPage from '../../Course/Materials/MaterialPage';
import QuizPage from '../../Course/Quizzes/QuizPage';

export const Router = () => {
  let element = useRoutes([
    {
      path: '/',
      children: [
        {
          index: true,
          element: <MainPage />,
        },
        {
          path: 'profile/',
          element: <ProfilePage />,
        },
        {
          path: 'institutions/',
          children: [
            {
              index: true,
              element: <InstitutionMainPage />,
            },
            {
              path: ':institution',
              children: [
                {
                  index: true,
                  element: <InstitutionOwnPage />,
                },
                {
                  path: 'courses/',
                  children: [
                    {
                      path: ':course/',
                      children: [
                        {
                          index: true,
                          element: <CoursePage />,
                        },
                        {
                          path: ':material/',
                          children: [
                            {
                              index: true,
                              element: <CoursePage />,
                            },
                            {
                              path: ':materialId',
                              element: <MaterialPage />,
                            },
                          ],
                        },
                        {
                          path: 'quizzes/:quiz',
                          element: <QuizPage />,
                        },
```

```
                    ],
                  },
                ],
              },
            ],
          },
        ],
      },
      { path: '*', element: <ErrorPage /> },
    ]);

    return element;
};
```

CreationCard.js:

```
import { Card, CardContent, Typography, Button } from '@mui/material';
import { Add as AddIcon } from '@mui/icons-material';

function CreationCard({ actionTitle, actionDescription, onClick }) {
  return (
    <Card sx={{ maxWidth: 310, p: 2 }}>
      <CardContent>
        <Button
          variant="contained"
          startIcon={<AddIcon />}
          sx={{
            my: 2,
          }}
          onClick={onClick}
        >
          {actionTitle}
        </Button>
        <Typography variant="body2" color="text.secondary">
          {actionDescription}
        </Typography>
      </CardContent>
    </Card>
  );
}

export default CreationCard;
```

Modal.js:

```
import { Button, Dialog, DialogActions, DialogContent, DialogTitle } from
'@mui/material';

function Modal({ title, open, onClose, children }) {
  return (
    <Dialog
      open={open}
      onClose={onClose}
      aria-labelledby="contained-Dialog-title-vcenter"
      scroll={'paper'}
      maxWidth="xs"
      fullWidth
    >
      <DialogTitle id="contained-Dialog-title-vcenter">{title}</DialogTitle>
      <DialogContent>{children}</DialogContent>
```

```
      <DialogActions>
        <Button onClick={onClose}>Close</Button>
      </DialogActions>
    </Dialog>
  );
}

export default Modal;
```

StackForm.js:

```
import { useFormik } from 'formik';
import { Box, Button, Divider, IconButton, InputAdornment, Stack, TextField }
from '@mui/material';
import { useState } from 'react';
import { Visibility, VisibilityOff } from '@mui/icons-material';

function StackForm(props) {
  const { initialValues, validationSchema, onSubmit, types, labels, submitBut-
tonLabel = 'Submit', children } = props;

  const [showPassword, setShowPassword] = useState(false);

  const handleClickShowPassword = () => setShowPassword((show) => !show);

  const handleMouseDownPassword = (event) => {
    event.preventDefault();
  };

  const formik = useFormik({
    initialValues,
    validationSchema,
    onSubmit,
  });

  const getInput = (dataName, type) => {
    const helperTextInsteadofLabel = type === 'date' || type === 'file';
    const handleChange =
      type === 'file'
        ? (e) => {
            formik.handleChange(e);
            formik.setFieldValue(dataName, e.currentTarget.files[0]);
          }
        : formik.handleChange;

    return (
      <TextField
        key={dataName}
        fullWidth
        multiline={type === 'textarea'}
        rows={type === 'textarea' ? 17 : 1}
        id={dataName}
        name={dataName}
        label={helperTextInsteadofLabel ? undefined : labels[dataName]}
        type={type === 'password' ? showPassword ? 'text' : 'password' : type}
        variant="outlined"
        value={type === 'file' ? undefined : formik.values[dataName]}
        onChange={handleChange}
        error={formik.touched[dataName] && Boolean(formik.errors[dataName])}
        helperText={
          (formik.touched[dataName] && formik.errors[dataName]) || (helperTex-
tInsteadofLabel && labels[dataName])
```

```
        }
        InputProps={
          type === 'password' && {
            endAdornment: (
              <InputAdornment position="end">
                <IconButton onClick={handleClickShowPassword} on-
MouseDown={handleMouseDownPassword}>
                  {showPassword ? <Visibility /> : <VisibilityOff />}
                </IconButton>
              </InputAdornment>
            ),
          }
        }
      />
    );
  };

  const getInputs = () => {
    const inputs = [];
    for (let dataName in types) {
      inputs.push(getInput(dataName, types[dataName]));
    }
    return inputs;
  };

  return (
    <form onSubmit={formik.handleSubmit}>
      <Stack spacing={2}>
        <Divider hidden />
        {getInputs()}
        {children}
        <Box sx={{ display: 'flex', justifyContent: 'center' }}>
          <Button
            color="primary"
            variant="contained"
            type="submit"
            size="large"
            sx={{ m: 1, px: 10, borderRadius: '8px' }}
          >
            {submitButtonLabel}
          </Button>
        </Box>
      </Stack>
    </form>
  );
}

export default StackForm;
```

ThemeWrapper.js:

```
import { createTheme, ThemeProvider } from '@mui/material/styles';

const theme = createTheme({
  palette: {
    primary: {
      main: '#632bbb',
      contrastText: '#fff',
    },
    secondary: {
      light: '#ff7961',
      main: '#e0e0e0',
```

```
      dark: '#ba000d',
      contrastText: '#000',
    },
    neutral: {
      main: '#7e57c2',
    },
    text: {
      main: '#bdbdbd',
    },
    background: {
      default: 'rgb(231, 235, 246)',
    },
    divider: '#f1f1f1',
  },
});

function ThemeWrapper(props) {
  return <ThemeProvider theme={theme}>{props.children}</ThemeProvider>;
}

export default ThemeWrapper;
```

UserChip.js:

```
import { Avatar, Chip } from '@mui/material';
import configClient from '../../configClient';

function UserChip({ user, ...props }) {
  return (
    <Chip
      key={user.id}
      sx={{ margin: 0.5 }}
      color="primary"
      label={user.fullName}
      avatar={<Avatar src={configClient.API_URL + '/' + user.avatar} />}
      {...props}
    />
  );
}

export default UserChip;
```

UserLineCard.js:

```
import { Avatar, Divider, ListItem, ListItemAvatar, ListItemButton, Lis-
tItemText, Typography } from '@mui/material';
import configClient from '../../configClient';

function UserLineCard({ user, ...props }) {
  return (
    <>
      <ListItem disablePadding>
        <ListItemButton {...props}>
          <ListItemAvatar>
            <Avatar alt={user.fullName} src={configClient.API_URL + '/' +
user.avatar} />
          </ListItemAvatar>
          <ListItemText
            primary={user.fullName}
            secondary={
              <Typography sx={{ display: 'inline' }} component="span" vari-
ant="body2" color="text.primary">
```

```
                {user.email}
              </Typography>
            }
          />
        </ListItemButton>
      </ListItem>
      <Divider variant="inset" component="li" />
    </>
  );
}

export default UserLineCard;
```

UsersAutocomplete.js:

```
import { Autocomplete, TextField } from '@mui/material';
import { useState, useEffect } from 'react';
import UserService from '../../services/UserService';
import UserChip from './UserChip';
import UserLineCard from './UserLineCard';

const preventErase = (event) => {
  console.log(event.key);
  if (event.key === 'Backspace' || event.key === 'Delete') {
    event.stopPropagation();
  }
  console.log(event.isPropagationStopped());
};

function UsersAutocomplete({ people, setPeople, label }) {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    UserService.getAllPersons().then((people) => {
      setUsers(people.data);
    });
  }, []);

  return (
    <Autocomplete
      multiple
      freeSolo
      noOptionsText={`No ${label}`}
      limitTags={1}
      id="multiple-limit-tags"
      options={users}
      isOptionEqualToValue={(option, value) => option.id === value.id}
      filterSelectedOptions
      value={people}
      onChange={(event, value) => setPeople((prev) => value)}
      getOptionLabel={(user) => user.fullName}
      renderInput={(params) => (
        <TextField
          label={`Choose ${label}`}
          placeholder={label[0].toUpperCase() + label.slice(1)}
          onKeyDown={preventErase}
          {...params}
        />
      )}
      renderTags={(value, getTagProps) =>
        value.map((user, index) => <UserChip key={user.id} user={user}
{...getTagProps({ index })} />)
```

```
      }
      renderOption={(props, user, state) => <UserLineCard key={user.id}
user={user} {...props} />}
    />
  );
}

export default UsersAutocomplete;
```

ДОДАТОК Г

Апробація

Web-додаток підтримки дистанційного навчання

Подус К.О., *студент ІТ-91*; Нагорний В.В., *доцент*
Сумський державний університет, м. Суми, Україна

Актуальність. У сучасному світі люди намагаються спростити та оптимізувати всі робочі процеси, саме для чого, майже в будь-якій справі використовують інформаційні технології. Так як одним з важливих процесів для людини є освіта, web-додаток підтримки дистанційного навчання буде у нагоді всім, хто прагне мати змогу вчитися та навчати, незважаючи на відстань.

Постановка задачі. Розробити web-додаток підтримки дистанційного навчання для надання та засвоєння освітніх матеріалів.

Результати. Було розглянуто «Moodle» як аналог подібного додатку і виявлено, що цей продукт може бути складним у використанні для користувачів без попереднього досвіду, хоча має потужний функціонал. [1] Тому, додаток повинен мати легку для розуміння навігацію та сучасний дизайн. На основі проведеного аналізу, буде створено web-додаток, що містить такі основні функціональні вимоги:

– можливість створення та редагування власного аккаунту;

– створення будь-якого дистанційного освітнього закладу, наприклад, школи або університету, в якому адміністратор може створювати курси і навчальні класи;

– можливість адміністратору закладу закріпляти за певним курсом викладача та клас;

– можливість викладачу надавати освітні матеріали для курсу у вигляді лекцій або практик та зазначати терміни здачі робіт;

– можливість студентам переглядати освітні матеріали курсу та завантажувати роботи;

– можливість викладачу заносити оцінки до журналу для спільного перегляду успішності студентів;

– можливість створення та проходження тестів для контролю та засвоєння теоретичного матеріалу.

Висновки. Практичною цінністю даної розробки виступає легкий та необмежений доступ до всіх процесів навчання в будь-який час та з будь-якого місця.

1 Moodle.org. URL: https://moodle.org/ (дата звернення: 30.03.2023)