МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інтерактивний додаток підтримки вивчення вітряної віспи в педіатрії»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології роектування»

Виконавець роботи: студент групи ІТ.мз-01с Ткаченко Антон Юрійович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Баранова І.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет центр заочної, дистанційної та вечерної форм навчання
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедрою ІТ

_____ В. В. Шендрик
«___»_____2021 р.

З А В Д А Н Н Я

кваліфікаційну роботу магістра студентові

Ткаченко Антон Юрійович

(прізвище, ім'я, по батькові)

1 Тема проекту *Інтерактивний додаток підтримки вивчення вітряної віспи в педіатрії*

керівник роботи *Баранова Ірина Володимірівна, к.т.н., доцент*,

затверджені наказом по університету від «29» 10 2021 р. № 0786-IV

2 Термін здачі студентом закінченого проекту «10» грудня 2021 р.

3 Вхідні дані до проекту технічне завдання на розробку інтерактивного додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області інтерактивного додатку, функціональне і структурне проектування додатку, розробка інтерактивного додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, аналіз програмних продуктів-аналогів, мета та задача дипломного проекту, аналіз технологій реалізації, етапи розробки інтерактивного додатку

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	10.09.2021 – 01.10.2021	
1.1	Дослідження актуальності проблеми	10.09.2021 – 20.09.2021	
1.2	Огляд існуючих програмних продуктів	10.09.2021 – 01.10.2021	
2	Постановка задачі	02.10.2021 – 10.10.2021	
2.1	Мета і задачі проекту	02.10.2021 – 05.10.2021	
2.2	Технології та інструменти реалізації	06.10.2021 – 10.10.2021	
3	Моделювання роботи додатку	11.10.2021 – 31.10.2021	
3.1	Структурно-функціональне моделювання роботи веб-додатку	11.10.2021 – 17.10.2021	
3.2	Проектування моделі бази даних	18.10.2021 – 25.10.2021	
3.3	Моделювання варіантів використання	26.10.2021 – 31.10.2021	
4	Розробка інтерактивного веб-додатку	01.11.2021 – 10.12.2021	
4.1	Розробка бази даних	01.11.2021 – 12.11.2021	
4.2	Програмна реалізація	13.11.2021 – 05.12.2021	
4.3	Тестування веб-додатку	06.12.2021 – 10.12.2021	

Магістрант _____ Ткаченко А.Ю.

Керівник роботи _____ к.т.н., доц. Баранова І.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інтерактивний додаток підтримки вивчення вітряної віспи в педіатрії».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 27 найменувань, додатків. Загальний обсяг роботи – 89 сторінок, у тому числі 50 сторінок основного тексту, 3 сторінка списку використаних джерел, 36 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці веб-додатку для вивчення сиптоматики вітряної віспи в педіатрії.

У роботі проведено аналіз предметної області та моделювання інтерактивного додатку. Висвітлено дослідження актуальності проблеми, огляд існуючих програмних продуктів та поставленні задачі проекту.

У третьому розділі наведено структурно-функціональне моделювання роботи веб-додатку, проектування моделі бази даних та моделювання варіантів використання.

У четвертому розділі показано практичну реалізацію розробки веб-додатку: розроблено архітектуру, інтерфейс, представлено програмну реалізацію функцій додатку, та представлення кінцевого результату.

Результатом проведеної роботи є веб-додаток, який дає можливість користувачу перевірити знання в області педіатрії, а саме теми діагностики вітряної віспи у дітей. Після проходження тесту користувачу надається результат його вибору. Практична цінність веб-додатку полягає в підвищені якості знань студентів в області медицини.

Ключові слова: ТРЕНАЖЕР, ОСВІТА, ВЕБ-ДОДАТОК, ДІАГНОСТИКА, ВІТРЯНА ВІСПА, ПЕДІАТРІЯ, КОРИСТУВАЧ, БАЗА ДАНИХ.

ЗМІСТ

ВСТУП

На сьогоднішній день інформаційні технології застосовуються в усіх сферах життєдіяльності. Люди користуються ними, бо вони автоматизують багато необхідних процесів, що полегшує наше життя.

Освітня сфера не є виключенням. Велика кількість інформаційних систем у вигляді інтерактивних додатків значно покращують засвоюваність необхідного матеріалу тієї чи іншої галузі.

Тому створення такого додатку є актуальною задачею. А саме інтерактивного веб-додатку, спрямованого на вивчення студентами медичних спеціальностей теми інфекційних захворювань.

Метою дипломного проекту є розробка інтерактивного веб-додатку для вивчення вітряної віспи в педіатрії, завдяки якому студент зможе закріпити свої навички у визначенні захворювань за симптомами.

Для досягнення поставленої мети сформульовано перелік необхідних завдань:

- Аналіз предметної області та пошук інтерактивних додатків зі схожою тематикою й ознайомлення з ними.
- Вибір певного стеку сучасних технологій для реалізації веб-додатку.
- Моделювання структури веб-додатку.
- Створення прототипу, узгодженого з замовником.
- Розробка бази даних з необхідними таблицями для взаємодії з веб-додатком.
- Реалізація веб-додатку з необхідним функціоналом та певними ролями.
- Тестування веб-додатку.

Таким чином результатом дипломної роботи має бути тренажер у вигляді веб-додатку, в якому студент зможе пройти контроль засвоєного матеріалу у вигляді інтерактивного тесту та отримати оцінку за результатами своїх відповідей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Предметна область додатку є освітньою, і тому результат проекту орієнтується на студентів, які навчаються в медичних освітніх закладах.

В педіатрії існує безліч найменувань різноманітних інфекційних захворювань, які мають певні симптоми [1-2]. Іноді, не дуже досвідченим лікарям, буває складно одразу визначити, чим саме захворів пацієнт, тому що симптомів багато і вони бувають схожими між собою, але захворювання різні. Тому для правильної діагностики треба мати досвід у визначенні цих захворювань [3-4].

У наш час освітня сфера має потребу у використанні в навчальному процесі сучасних інтерактивних тренажерів [5-7], які завдяки візуальній складовій дозволяють краще засвоїти матеріал, та дещо розбавити сухий підхід до вивчення інформації. Для студентів це дуже актуально.

Таким чином, можна зробити висновок, що розроблюваний веб-додаток є актуальним та вирішує зазначені проблеми предметної області.

1.2 Огляд існуючих програмних продуктів

У наш час існує багато форм контролю і оцінювання знань в області медицини. Для того, щоб провести порівняльний аналіз, візьмемо до уваги три сучасних додатки з схожим направленням:

1. Для середовища IOS “Медичний тест” [8]. Додаток, що призначений для перевірки та закріплення знань в області медицини у студентів різних навчальних мед-закладів. Є безкоштовним.

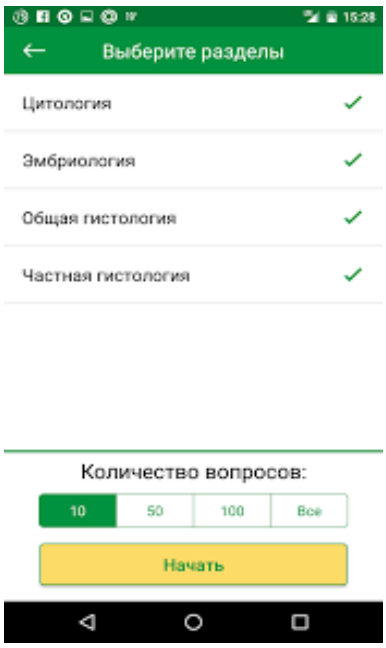

Рисунок 1.1 – Зображення головного меню додатку “Медичний тест”

2. Для середовища Android “Тест з фармакології” [9]. Додаток, що призначений для перевірки та закріплення знань в області фармацевтики, та також є актуальним для студентів. Є безкоштовним.

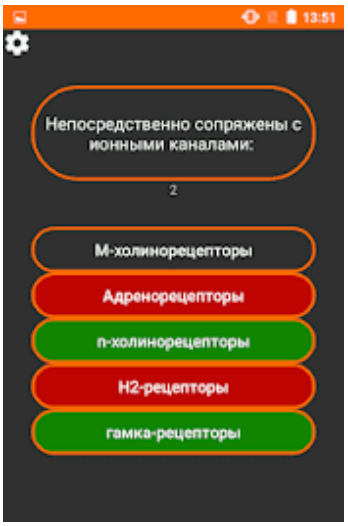

Рисунок 1.2 – Зображення головного меню додатку “ Тест з фармакології”

3. Для веб-середовища “Тест з патології інфекційних хвороб” [10]. Додаток, що призначений для перевірки та закріплення знань в області інфекційних захворювань для студентів. Є безкоштовним.

Рисунок 1.3 – Зображення головного меню додатку “ Тест з патології інфекційних хвороб ”

4. Для середовища Windows “Мед-тест” [11]. Додаток, що також призначений для перевірки та закріплення знань в області інфекційних захворювань в педіатрії. Є безкоштовним.

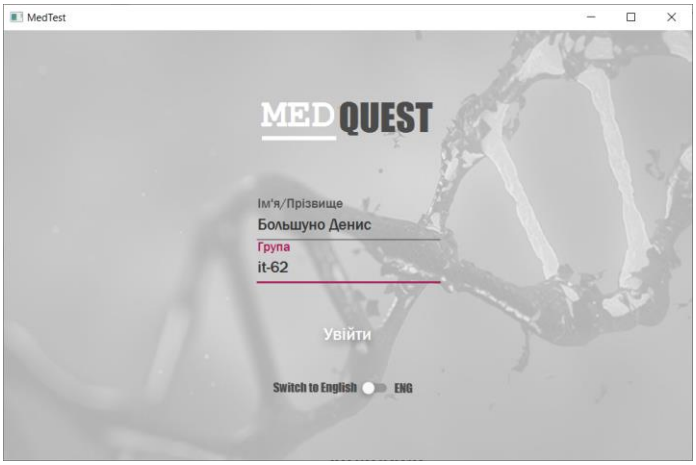

Рисунок 1.4 – Зображення головного меню додатку “ Мед-тест”

У результаті аналізу було створено порівняльну таблицю аналогів, в якій зазначено критерії порівняння та їх відповідність у розглянутих додатках.

Таблиця 1.1 - Порівняння аналогів

Характеристика	Медичний тест	Тест для фармакології	Загальна патологія інфекційних хвороб	Розроблюваний додаток
Реєстрація/ Авторизація	+	+	+	+
Підтримка різних мов	-	-	-	+
Кросплатформність	-	-	+	+
Зручний інтерфейс	+	+	+	+

Огляд аналогів дозволить реалізувати переваги існуючих додатків та уникнути їх недоліків. В результаті аналіза предметної області та існуючих аналогів можна зробити висновок, що розробка даного додатку є актуальною задачею.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета і задачі проекту

Мета дипломного проекту полягає в створенні інтерактивного веб-додатку, завдяки якому студент зможе вивчати інфекційні захворювання в педіатрії та проводити контроль своїх знань.

Для досягнення мети необхідно вирішити такі задачі:

- Провести аналіз серед вже реалізованих додатків зі схожою тематикою.
- Визначитись з певним стеком сучасних технологій для реалізації веб-додатку.
- Створити базу даних з необхідними таблицями.
- Спроектувати структуру веб-додатку та необхідний функціонал.
- Практично реалізувати тренажер, перевірити його працездатність.

Функціональні вимоги до створюваного додатку:

- Має бути дві ролі викладач і студент.
- Має бути реалізована авторизація і реєстрація з підтвердженням за сторони викладача.
- Студент повинен мати можливість проходження тесту та отримання результату його виборів.
- Має бути реалізований динамічний розвиток альтернатив відповідно обраним варіантам відповідей.
- Викладач має можливість передивлятися статистику по групах та додавати спроби повторного проходження тесту.

2.2 Технології та інструменти реалізації

Для реалізації веб-додатку планується розробити restful-api програмною мовою php та фронтенд за допомогою javascript бібліотеки React.

Restful-api [12] – це веб-сервіс, створений на основі rest технології. В свою чергу, rest – це інтернет-протокол, який задає певні правила роботи веб-додатку та відтворює його архітектуру. Розшифровується як «Representation State Transfer», тобто репрезентативна передача стану. Дана технологія реалізовує передачу даних між інформаційним сховищем та інтерфейсом користувача, завдяки запитам через протокол http. В додатку планується використовувати асинхронні http запити з методами POST і GET [13]. В сучасній веб-розробці такий підхід з використанням технології rest є розповсюдженим.

Технологія React [14] – це javascript бібліотека з відкритим кодом, яка використовується для створення інтерфейсів користувача в додатках. Головна задача бібліотеки – це зробити додаток швидким, гнучким та можливим к масштабуванню. Також перевага цієї технології в тому, що дані, які використовуються, що створюють стан додатку, змінюються без перезавантаження самої сторінки, завдяки компонентному підходу. Підхід самої бібліотеки співпадає з архітектурним шаблоном «Model-View-Controller», тому це робить її гнучкою у використанні з іншими технологіями. Особливо – це є актуальним у великих проектах. React створює власну віртуальну об’єктну модель документа, тому це дає можливість відслідковувати, які елементи було змінено і миттєво оновити їх стан. Також, зазвичай при використанні даної бібліотеки, файли компонент мають розширення .jsx.

Для написання стилів буде обрано фреймворк Material UI [15]. Це технологія, яка дозволяє створювати зручний та сучасний інтерфейс для додатку. Даний фреймворк працює з бібліотекою React і також має компонентний підхід. Розробник має можливість як перевизначити самі стилі компонент, так і створювати власну тему стилів. Для того, щоб перевизначити стилі за

замовчуванням, необхідно створити файл, який буде включати в себе стилі визначені користувачем, та підключити його через імпорт в файл з необхідною компонентою. Даний принцип перевизначення є не складним, тому часто застосовується на практиці.

Для керування станом веб-додатку буде обрано javascript бібліотеку Redux [16]. Redux – це javascript бібліотека з відкритим кодом, за допомогою якої проводиться керування станом додатка. Дана технологія зберігає стан всього додатку в об’єктному дереві, яке знаходиться в одному сховищі даних.

Redux має певний принцип роботи, а саме має об’єкт, який створює з’єднання між самими діями. До об’єкту станів можна отримати доступ через метод getState(), а змінити стан можна через функцію distpatch(), аргументом якої є певна дія.

Екшени в бібліотеці Redux – це певні структури, завдяки яким можна передати дані в стан, вони є єдиним способом передачі інформації. Редьюсерами називають функціїї, через які проходить попередній стан, і які обробляють цей стан певною логікою. Самі редьюсери можуть бути декомпозованими на дрібніші функції.

Бібліотека Redux гарно взаємодіє з react та відтворює зручну екосистему проекту. Дані приходять з серверу та завантажуються до сховища, і вже потім відображаються через компоненти.

Для керування базою даних планується використати СУБД MySQL [17]. Це реляційна система керування базами даних з відкритим кодом, яка використовується створення та керування базами даних. Вона є однією з найбільш використовуваних серед систем керування баз даних.

СУБД є крос-платформною, тому гнучка у застосуванні. Взаємодіє з багатьма мовами програмування, включаючи мову php, яку планується застосовувати в розробці веб-додатку.

MySQL має зручний користувацький інтерфейс, за допомогою, якого можна створювати таблиці та працювати з ними. Найчастіше даний продукт

використовують у веб-розробці, тому що показники швидкодії є високими на відміну від інших СУБД.

Також важливим критерієм вибору даної системи є безпека даних. MySQL має відмінну систему безпеки, що захищає дані від багатьох небезпек.

3 МОДЕЛЮВАННЯ РОБОТИ ДОДАТКУ

3.1 Структурно-функціональне моделювання роботи веб-додатку

Початок роботи веб-додатку – це вікно з авторизацією та реєстрацією. Функція авторизації має валідацію та верифікацію. Якщо користувача немає в базі даних, то він може зареєструватись. Після реєстрації акаунт користувача є неактивним. Активувати акаунт та надати спробу на тестування може викладач зі своєї сторінки. Після активації, користувач може авторизуватись.

Після успішної авторизації з бази даних завантажуються дані користувача та тестові дані. З’являється вікно з привітанням та можливістю розпочати тест. Функція початку тесту завантажує сторінку з тестом та дозволяє користувачу пройти тестування.

На контекстній діаграмі IDF0 та діаграмі декомпозиції продемонстровано функціональну взаємодію користувача з додатком (рис. 3.1 – 3.2).

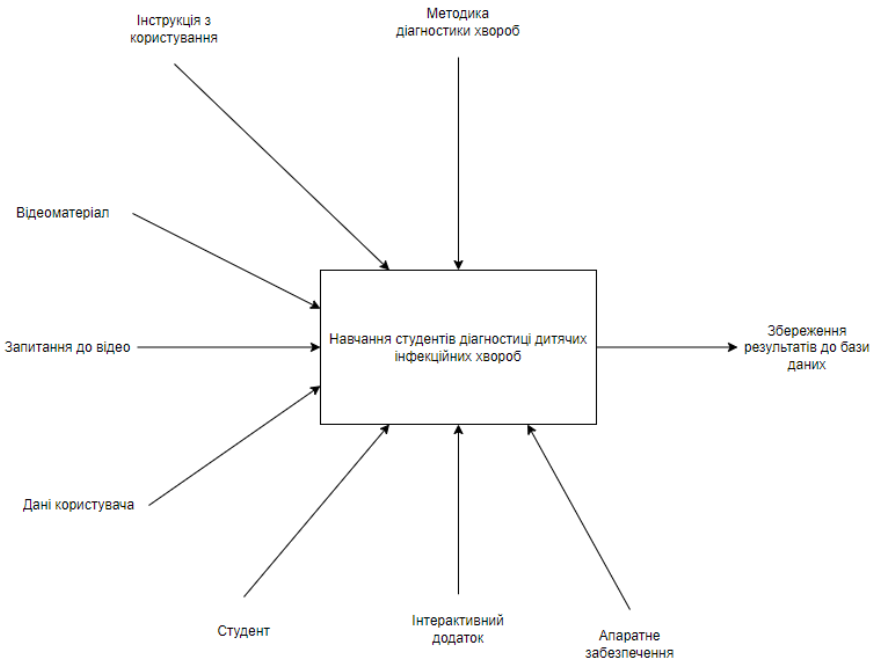

Рисунок 3.1 – Контекстна діаграма

Під час тестування кожна обрана відповідь завантажує відповідне наступне питання та веде до певної кінцівки. Разом з питанням та варіантами відповідей завантажується відповідне супроводжуюче відео.

Після тестування сторінка оновлюється та з’являється результат тесту. Користувач може переглянути результат та повернутися до вікна з привітанням.

Після тестування результат записується до бази даних, а спроба на проходження тест анулюється.

Після того, як користувач закінчив роботу з додатком, він може вийти зі свого акаунту. При виході з акаунту його сесія завершується.

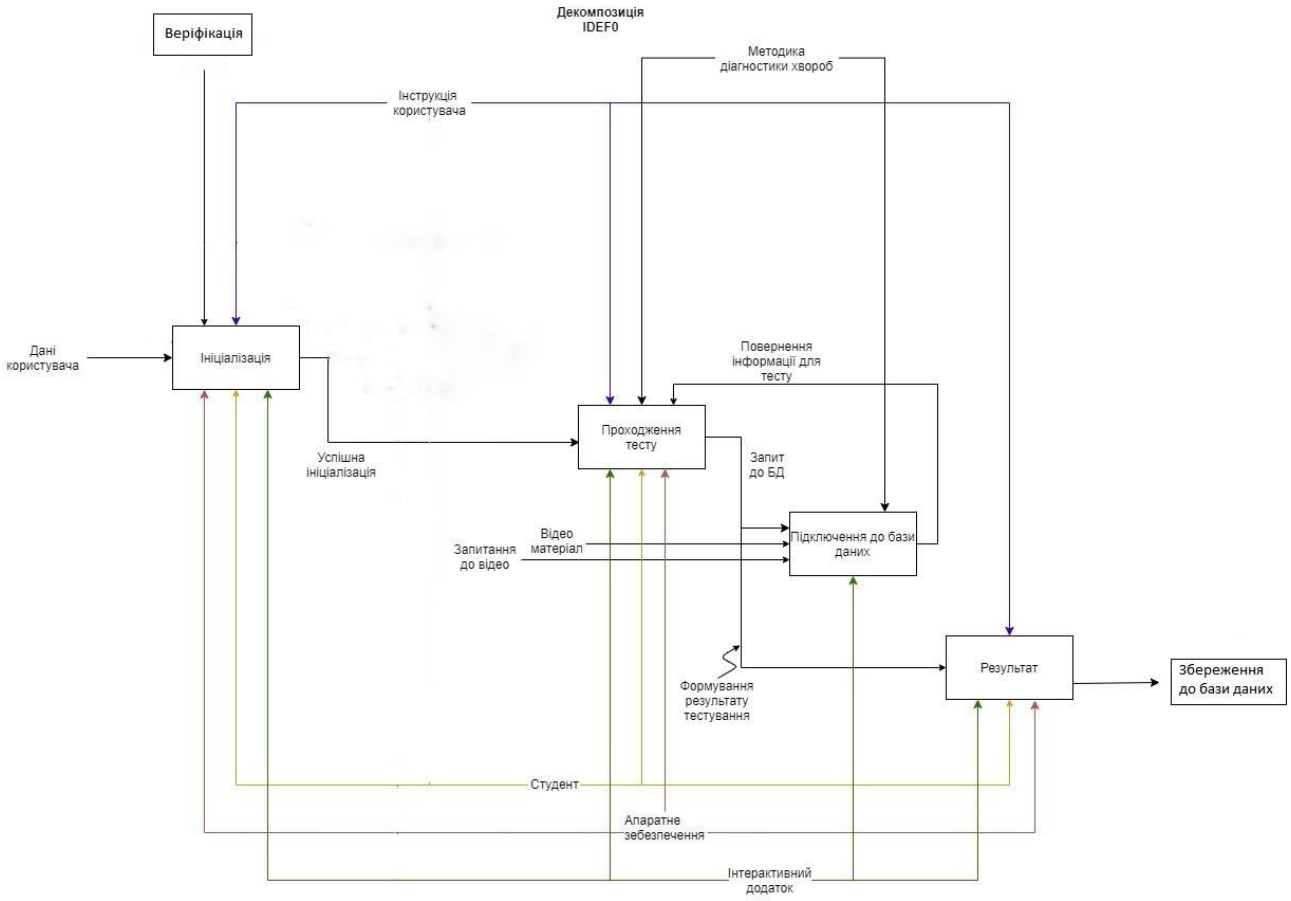

Рисунок 3.2 – Діаграма декомпозиції першого рівня

3.2 Проектування моделі бази даних

Веб-додаток має базу даних, яка складається з 7 пов'язаних між собою таблиць.

Таблиця «roles» містить поля з назвою ролей та їх описом, «users» - таблиця користувачів, яка включає дані користувачів такі, як логін, ім’я, прізвище, групу, пароль ідентифікатор ролі, наявність спроби проходження тесту та статус акаунту.

Таблиці «uk_stages», «uk_questions» та «uk_answers» містять дані тесту такі, як стадії, питання тесту та відповіді, таблиця «animations» включає назву файлу з відповідним відео. Всі дані таблиці пов’язані зовнішніми ключами.

Таблиця «history_results_test» містить в собі дані результату тестування, такі як час проходження тесту, дані користувача та сам результат. Вона також пов’язана зовнішніми ключами з іншими таблицями.

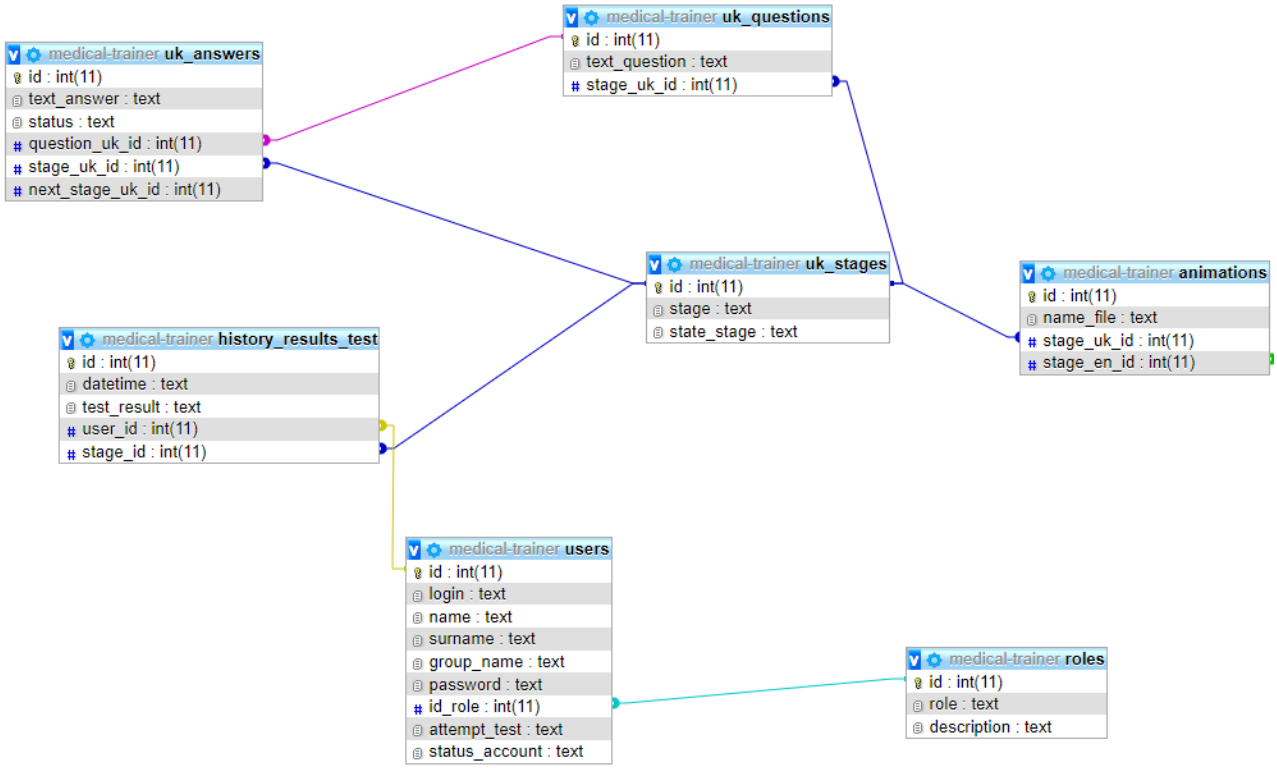

Рисунок 3.3 – Зв’язок таблиць бази даних

Опис полів таблиць бази даних та їх атрибутів наведено далі.

Таблиця 3.1 – Опис таблиць бази даних та їх атрибутів

Назва	Опис
users – таблиця в яку зберігаються користувачі веб-додатка	**id** – ідентифікатор поля; **login** – логін користувача; **name** – ім’я користувача; **surname** – прізвище користувача; **group_name** – назва групи користувача; **password** – пароль користувача; **id_role** – ідентифікатор ролі користувача; **attempt_test** – наявність спроби на проходження тесту; **status_account** – статус акаунту користувача (активований або не активований);
roles – таблиця, що зберігає ролі користувача	**id** – ідентифікатор поля; **role** – назва ролі; **description** – опис ролі;
history_results_test – таблиця в яку зберігаються результати проходження тесту	**id** – ідентифікатор поля; **datetime** –дата проходження тесту; **stage_id** – ідентифікатор стадії; **test_result** – результат тесту; **user_id** – ідентифікатор користувача;

Продовження табл. 3.1

Назва	Опис
uk_stages – таблиця, що зберігає стадії тесту	**id** – ідентифікатор поля; **stage** –назва стадії; **state_stage** – статус стадії;
uk_answers – таблиця, що зберігає варіанти відповідей тесту	**id** – ідентифікатор поля; **next_stage_uk_id** – ідентифікатор наступної стадії; **question_uk_id** – ідентифікатор питання **stage_uk_id** – ідентифікатор стадії; **status** – статус відповіді (правильне або ні); **text_answer** – текст відповіді;
uk_questions – таблиця, що зберігає питання тесту	**id** – ідентифікатор поля; **stage_uk_id** – ідентифікатор стадії; **text_question** – текст питання;
animations – таблиця, що зберігає назви файлів з анімацією певних стадій тесту	**id** – ідентифікатор поля; **name_file** –назва файлу анімації; **stage_uk_id** – ідентифікатор стадії;

3.3 Моделювання варіантів використання

Було створено діаграму варіантів використання для розуміння структури і веб-додатку, та взаємодії користувача з ним.

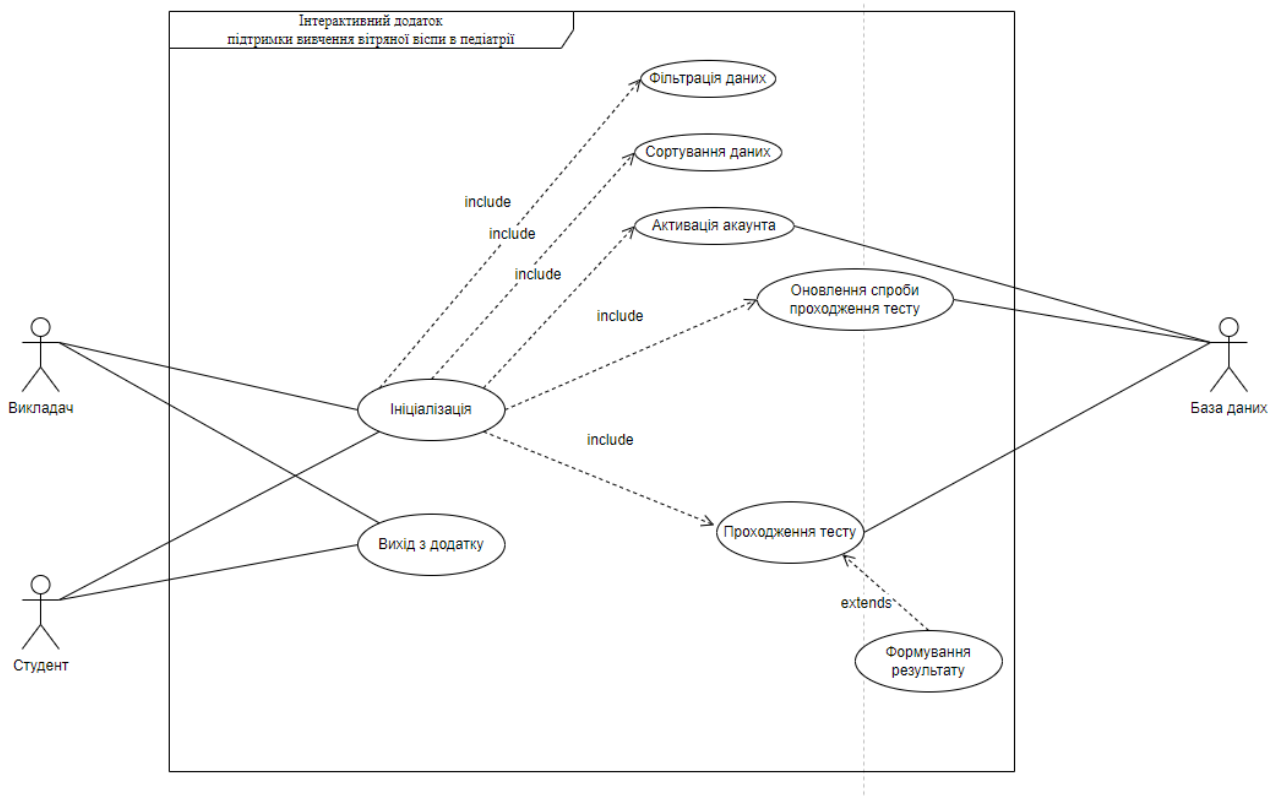

Рисунок 3.4 – Діаграма варіантів використання

На діаграмі варіантів використання відображено функціональну взаємодію користувача з додатком. Відображені такі функції як:

1) Ініціалізація користувача. Користувач проходить авторизацію.

Для ролі студента:

2) Функція тестування. Користувач проходить тестування.
3) Створення результату. Результат тесту зберігається до бази.
4) Вихід з додатку. Користувач завершує сеанс роботи.

Для ролі викладача:

5) Фільтрація даних. Користувач фільтрує дані в таблиці за певним критерієм.

6) Сортування даних. Користувач сортує дані в таблиці за певним стовпцем.
7) Активація акаунта. Користувач активує акаунт студента.
8) Оновлення спроби проходження тесту. Користувач оновлює спробу проходження тесту для певного студента.
9) Вихід з додатку. Користувач завершує сеанс роботи.

4 РОЗРОБКА ІНТЕРАКТИВНОГО ВЕБ-ДОДАТКУ

4.1 Розробка бази даних

Для створення та керування базою даних було обрано СУБД MySQL та веб-інтерфейс керування phpMyAdmin [18].

Веб-інтерфейс зображений на рисунку 4.1.

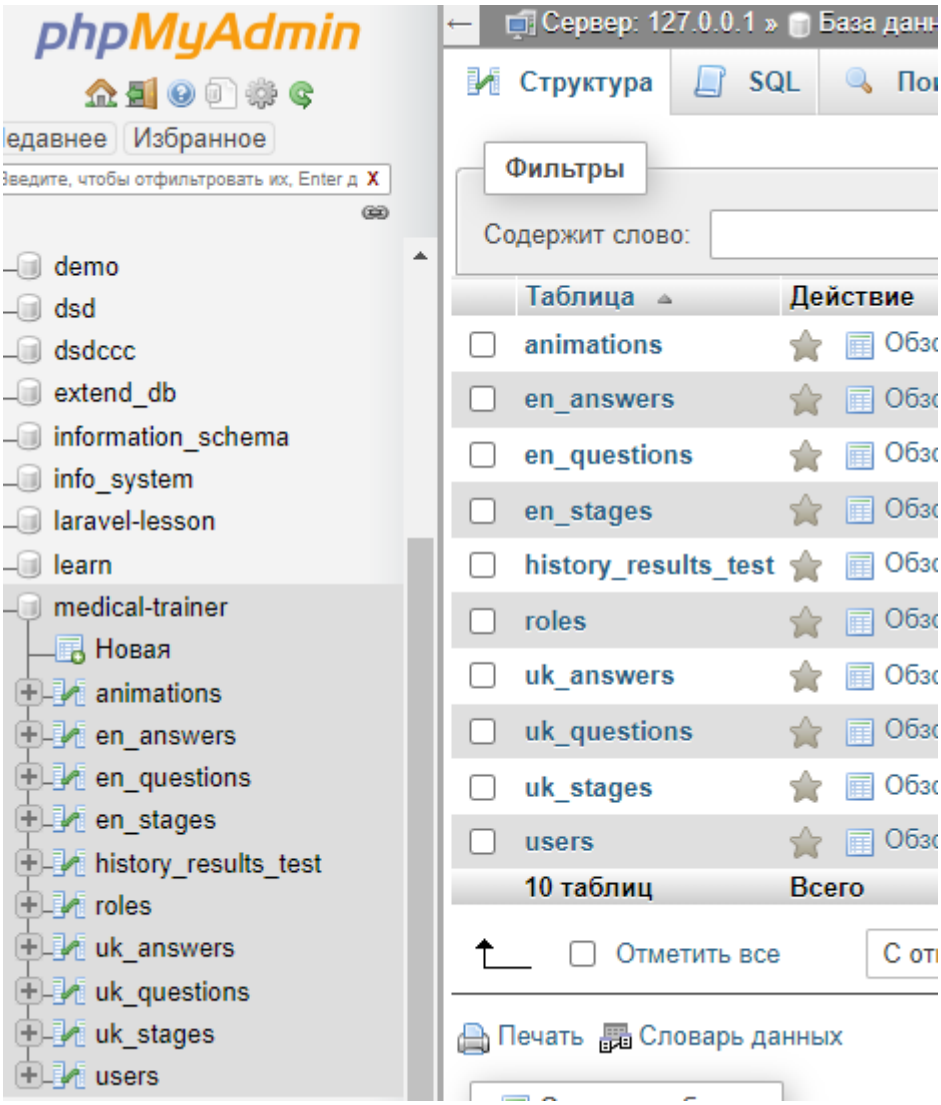

Рисунок 4.1 – Інтерфейс керування базою даних

Було створено базу даних «medical-trainer» та основні таблиці для взаємодії з додатком.

Рисунок 4.2 – Створені таблиці бази даних

Взаємодія веб-додатка з базою даних реалізована за допомогою restful-api, яке включає в собі всі необхідні запити:

1) Insert – занесення до бази даних;
2) Select – вибірка даних з бази;
3) Update – оновлення застарілих даних

```
$result = $mysqli->query( query: "SELECT * FROM users WHERE id_role = 3");
while($row = $result->fetch_assoc()){
    array_push( &array: $data, ['id' => $row['id'], 'login' => $row['login'], 'name' =>
                                'surname' => $row['surname'], 'group' => $row['group_
                                'attempt' => $row['attempt_test'], 'statusAccount' =>
}
$json = [
```

Рисунок 4.3 – Приклад взаємодії веб-додатку з базою даних

4.2 Програмна реалізація

Проект було реалізовано у вигляді веб-додатку за допомогою javascript бібліотеки react.js та створеному restful-api програмною мовою php.

Є клієнт і створене rest-api. Під час роботи додатка і певної дії, відбувається http запит до api, і через нього з бази даних передаються необхідні дані. На наступному рисунку продемонстрований даний підхід:

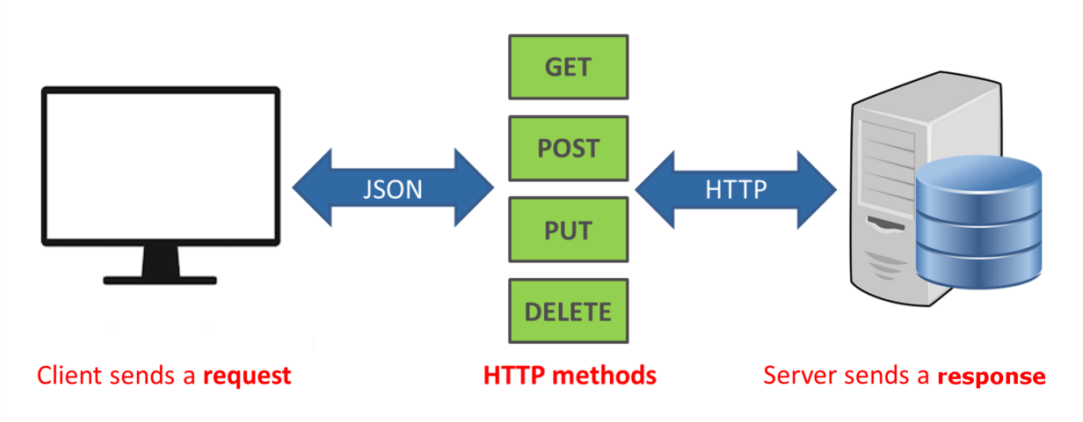

Рисунок 4.4– Демонстрація робота клієнт-сервера

Веб-додаток структурно складається з однієї сторінки, яка під час взаємодії з додатком оновлюється та завантажує відповідні компоненти.

Спочатку завантажується компонента з формою авторизації та реєстрації.

Після успішної авторизації студента, сторінка оновлюється і завантажується компонента зі статичною шапкою веб-додатка та текстом-привітанням з кнопкою початку тестування. Після переходу до тесту, сторінка завантажує сам тест, який також складається з компонент, таких, як питання, супроводжуюче відео та варіанти відповідей. Після завершення тесту сторінка завантажує результат тесту та кнопку, що дає можливість перейти до сторінки з привітанням.

Після авторизації викладача додаток оновлює сторінку та завантажує компоненту з бічним меню і відповідними таблицями. Викладач має можливість

переглядати та взаємодіяти з таблицею даних користувачів та таблицею з даними результатів користувачів.

На рисунку 4.5 продемонстровано перехід між компонентами в результаті взаємодії користувача з веб-додатком.

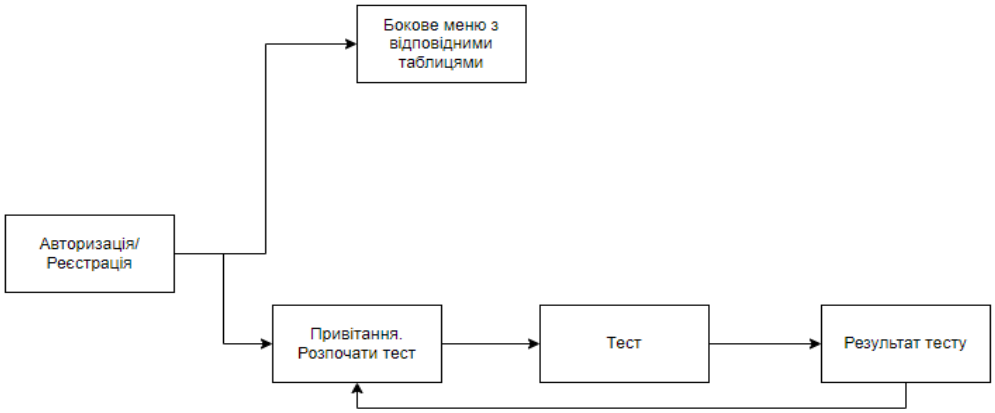

Рисунок 4.5 – Переходи між компонентами веб-додатку

Для створення додатку було обрано IDE PhpStorm [20].

Було встановлено пакетний менеджер npm node.js [21] з офіційного сайту, що дозволяє створити проект за допомогою бібліотеки react.js (рис. 4.6).

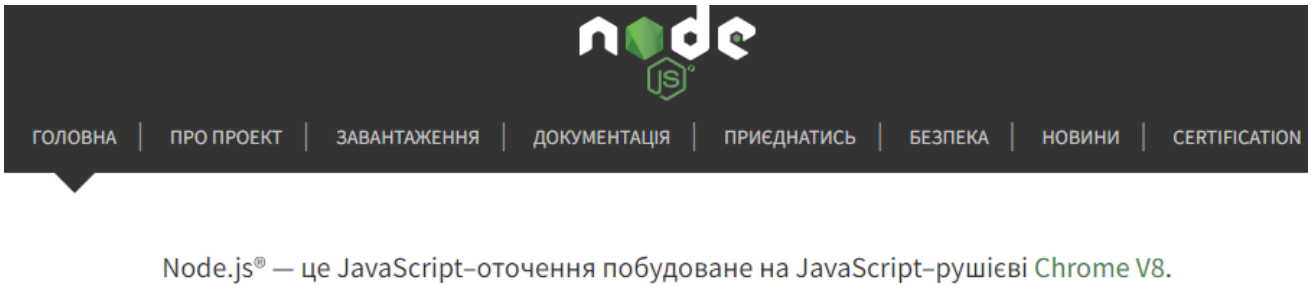

Рисунок 4.6– Завантаження пакетного менеджеру npm

Далі через командний рядок було створено сам проект за допомогою команди create react app:

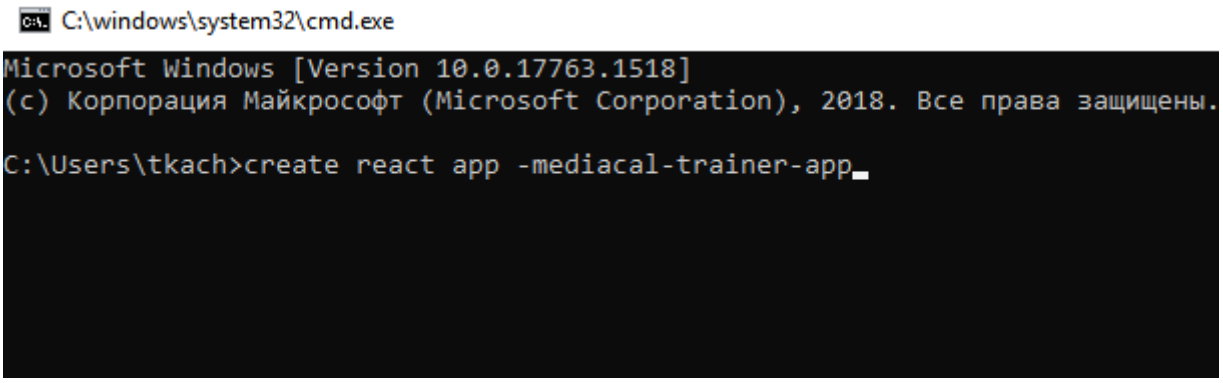

Рисунок 4.7 – Створення проекту додатку

Створений проект складається з внутрішніх файлів бібліотеки react.js в директорії node_modules, файлу manifest в директорії public та самих файлів проекту, що створює розробник, вони знаходяться в директорії src:

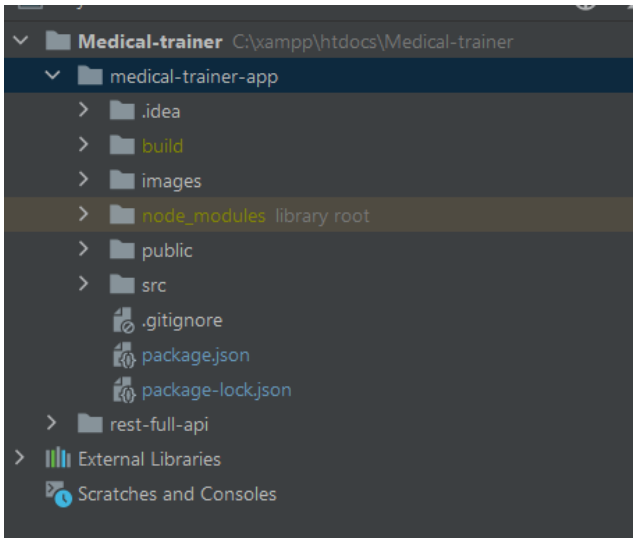

Рисунок 4.8 – Файлова структура проекту

Наступним кроком в проекті створено директорії компонентів. Там знаходяться всі файли-компоненти, які відповідно відображаються на сторінці веб-додатка під час взаємодії користувача з ним.

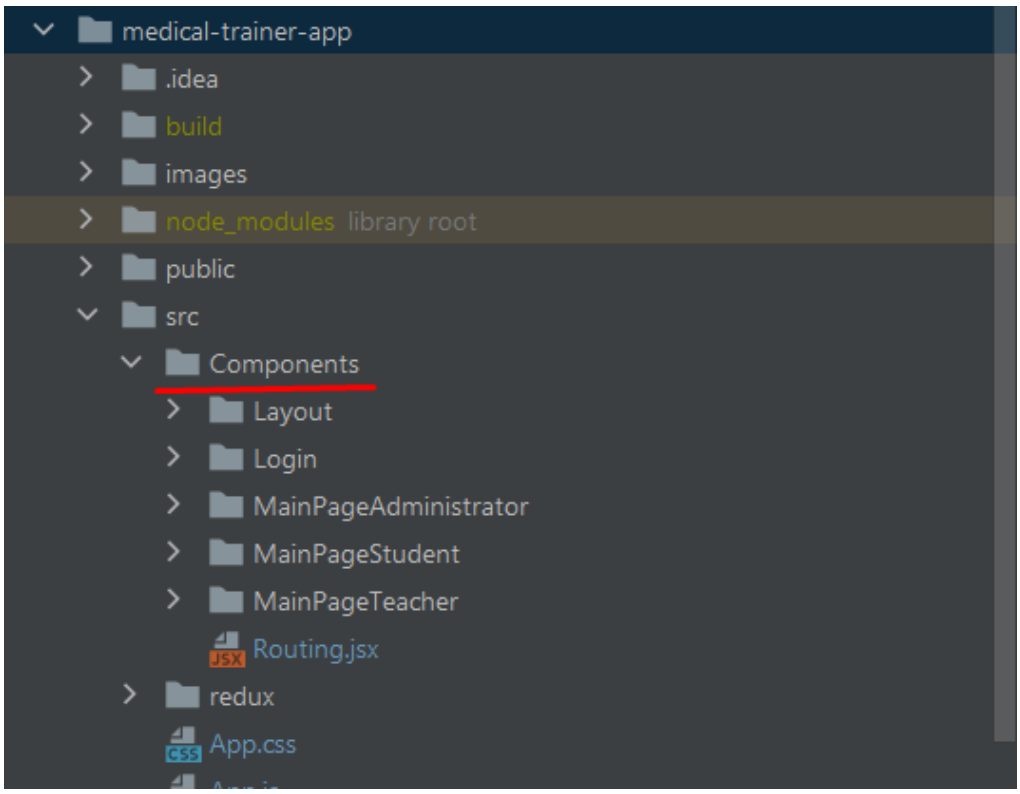

Рисунок 4.9 – Створена директорія компонентів

Першим було зроблено компоненти авторизації та реєстрації, які знаходяться в папці Login. Для створення сторінки авторизації та реєстрації було підключено відповідні компоненти фреймворку material-ui та саму бібліотеку react.js:

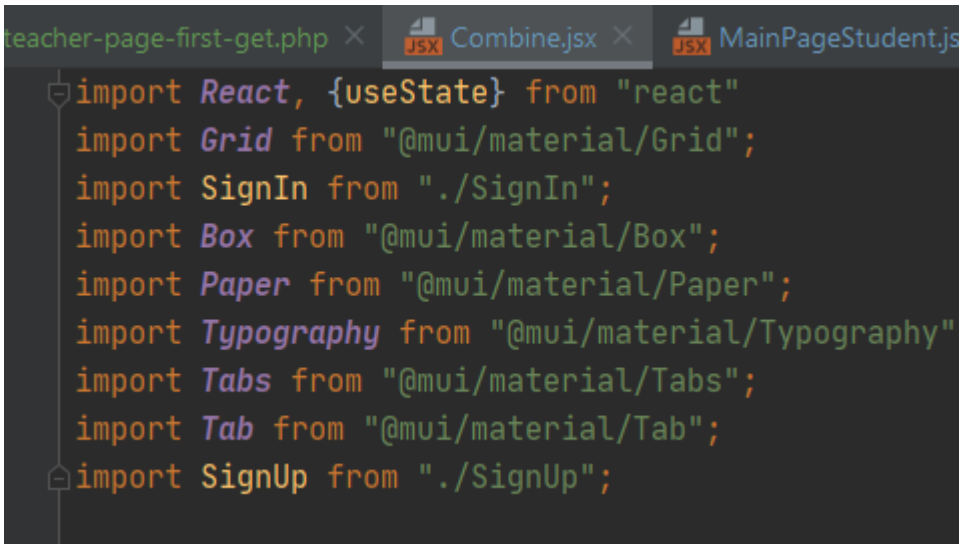

Рисунок 4.10 – Компоненти з авторизацією та реєстрацією

Далі створено папку з компонентами сторінки користувача з роллю студента. Вона включає в себе компоненту з текстом-привітанням, компоненту самого тесту та компоненту з результатом тесту.

В компоненту з тестом було підключено створені компоненти з відповідними частинами: анімацією, питанням та варіантами відповідей.

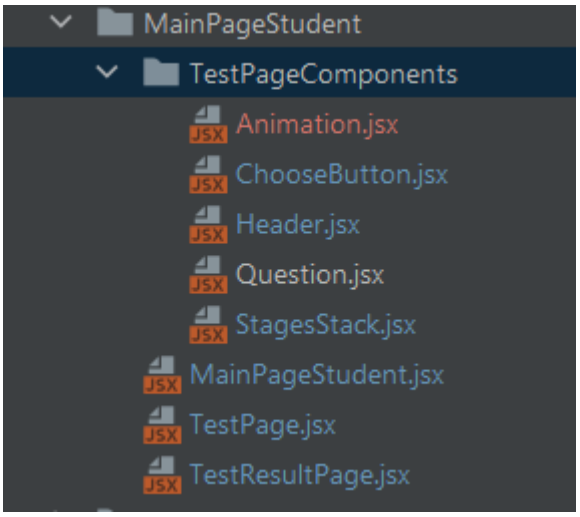

Рисунок 4.11 – Файли компонент ролі студента

Після цього аналогічним способом, було створено компоненти сторінки користувача з ролю викладача. Вона включає в собі дві таблиці та бокове меню.

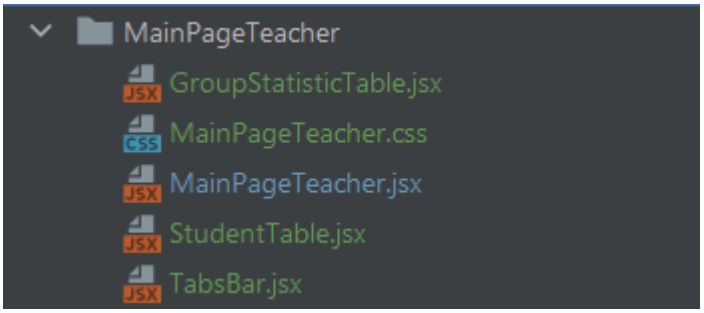

Рисунок 4.12 – Файли компонент ролі викладача

Далі було створено в папці layout та підключено статичну шапку для сторінки авторизованого користувача. Для обох ролей вона однакова.

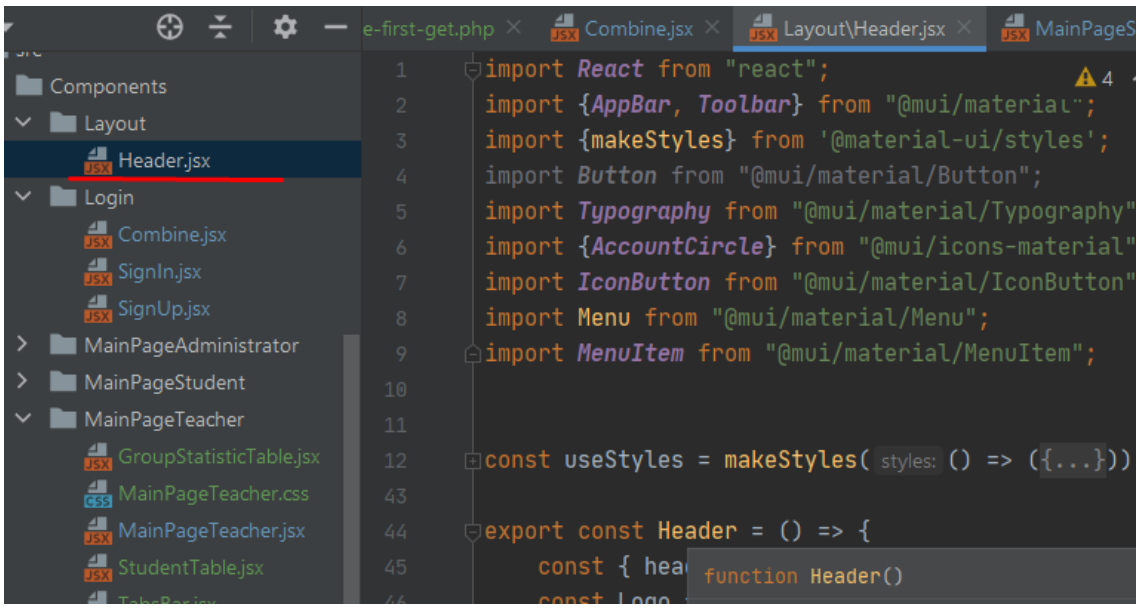

Рисунок 4.13 – Файл зі статичною шапкою

Результат коду компоненти статичної шапки представлено на наступному рисунку.

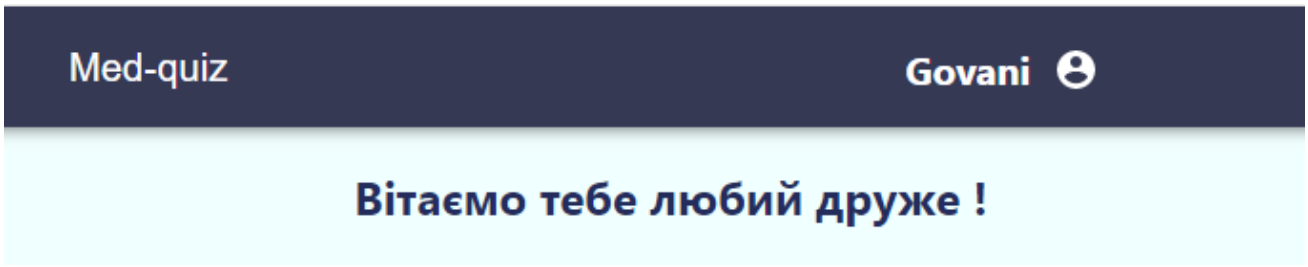

Рисунок 4.14 – Файл зі статичною шапкою

Хедер має меню користувача, де реалізовано вихід з акаунту і саму назву додатку.

Також для відображення необхідних компонент було створено файл маршрутизації, за допомогою якого вимальовуються компоненти потрібної ролі.

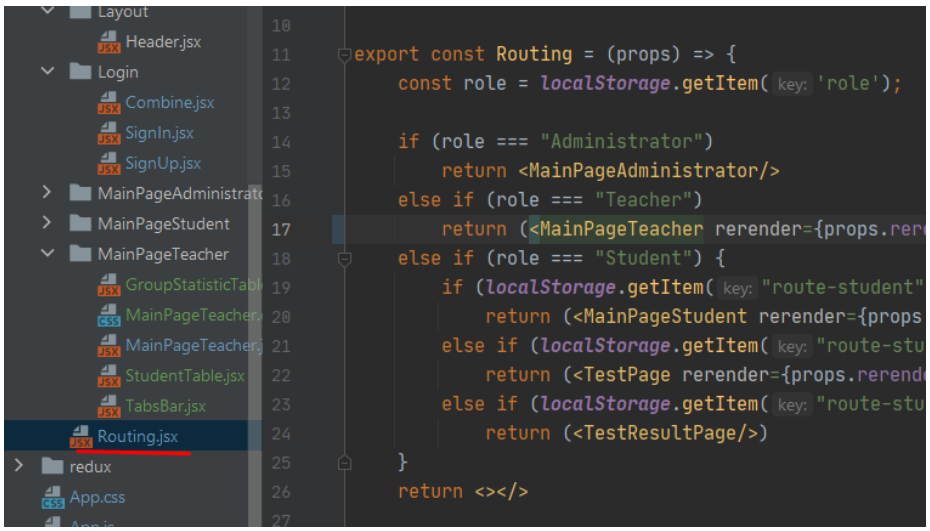

Рисунок 4.15 – Файл маршрутизації

Далі всі необхідні компоненти були підключені до файлу App.js, в якому також створена логіка відображення компонент.

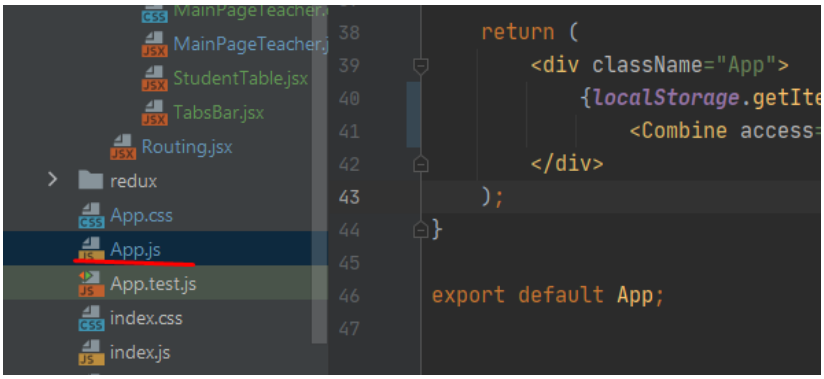

Рисунок 4.16 – Файл App.js

Для керування даними веб-додатку застосовано фреймворк redux. Для цього в терміналі було виконано команда npm install redux. Для зручної розробки та роботи з даними додатка було встановлено інструмент redux-toolkit [22].

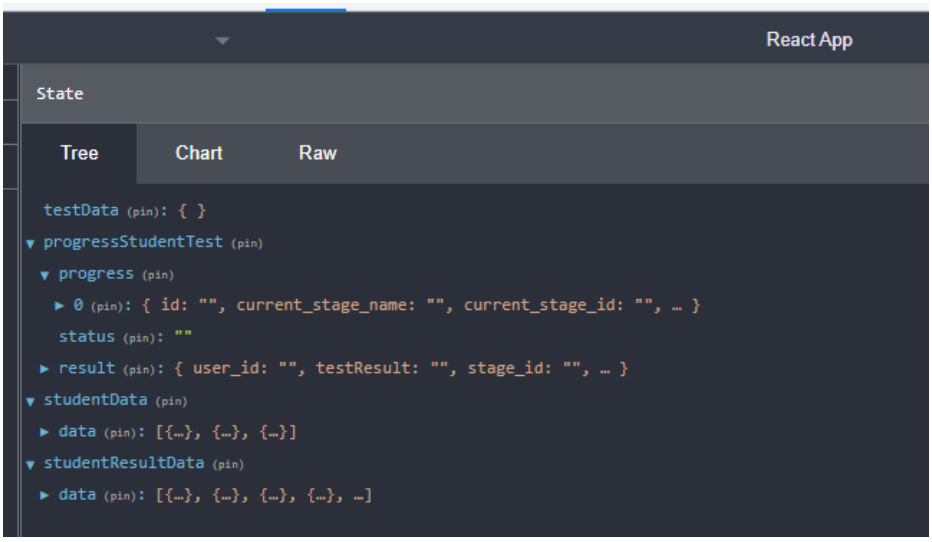

Рисунок 4.17 – Інструмент redux-tollkit

У файлах проекту, було створено директорію redux, а в ній папку reducers, яка включає в себе файли з даними і логікою їх обробки. Для створення функцій обробки даних було застосовано необхідні методи фреймворку redux.

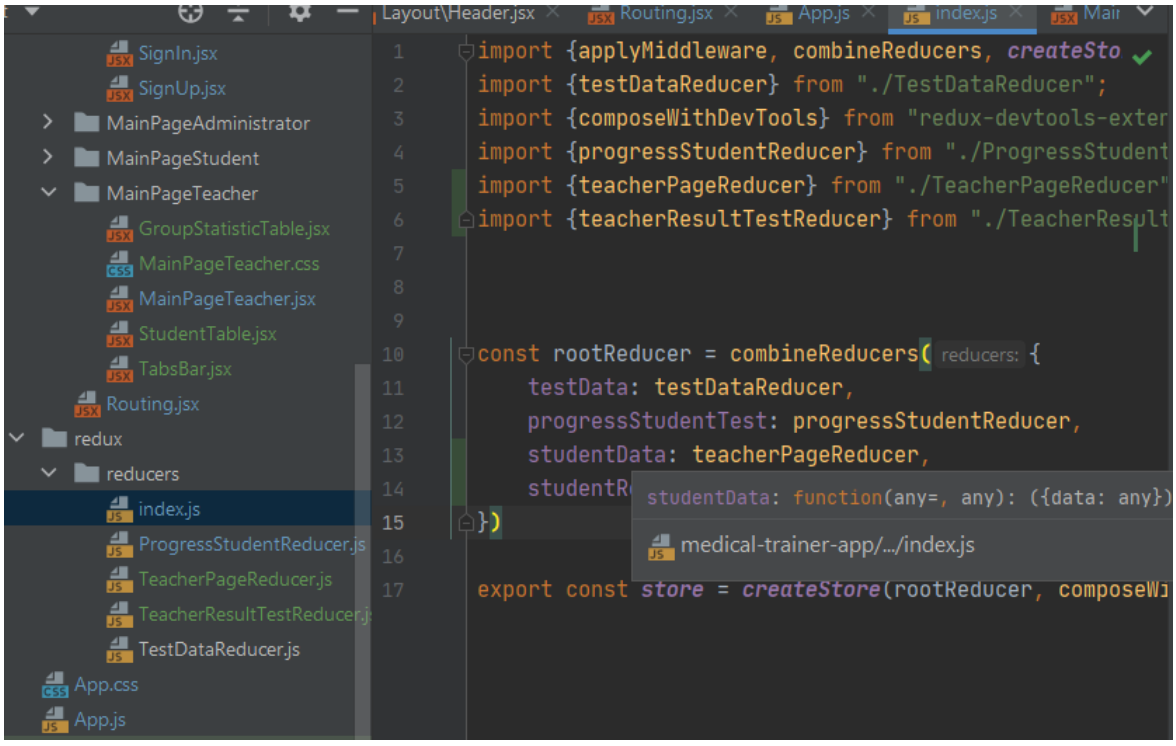

Рисунок 4.18 – Файли обробки та збереження даних

Для отримання даних з баз даних було створено restful-api програмною мовою php, до якого ми звертаємося через http запити [23]. Далі наведено приклад

http запиту з POST методом [24] за допомогою javascript функції fetch [25], який отримує файл типу json [26] та обробляє його:

```
const responseTeacherResultTestData = () => {
    return fetch(URL_GET_TEACHER_RESULT_TEST_DATA, init: {
        method: "POST",
        header: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: JSON.stringify( value: {action: 1})
    })
        .then((response : Response ) => {
            return response.json().then((data) => {
                return data;
            })
        })
}
```

Рисунок 4.19 – Приклад запиту до restful-api

Restful-api включає в себе файли: підключення до бази даних, реєстрація користувача, авторизація користувача, вибірка даних для сторінок студента та викладача.

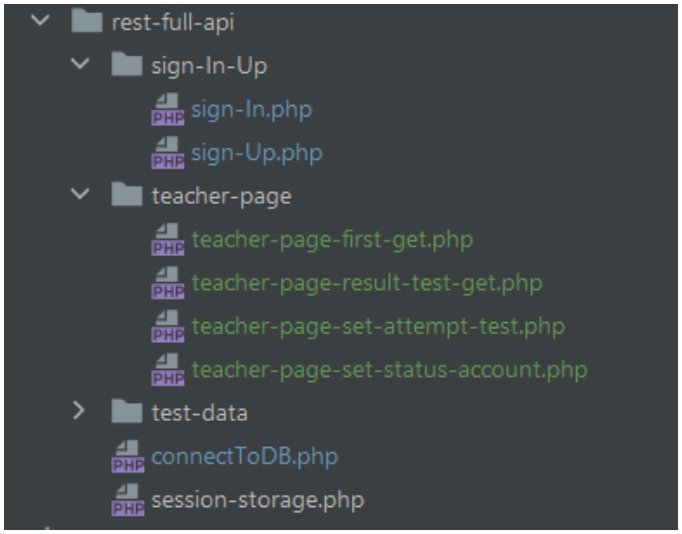

Рисунок 4.20 – Файлова структура restful-api

На рисунку 4.21 наведено приклад файлу авторизації користувача:

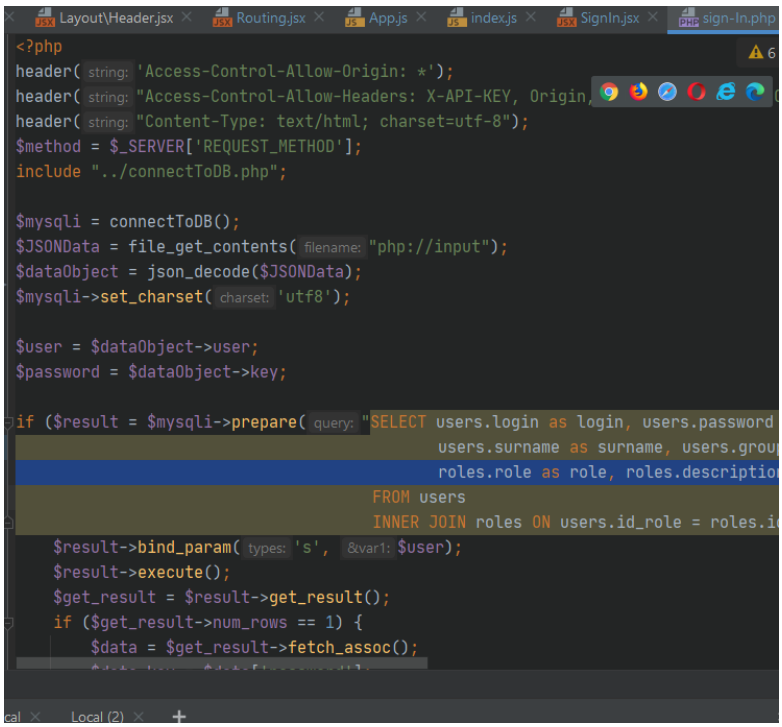

Рисунок 4.21 – Авторизація користувача через базу даних

Далі представлені альтернативні варіанти розвитку питань тесту залежно від обраних відповідей [11].

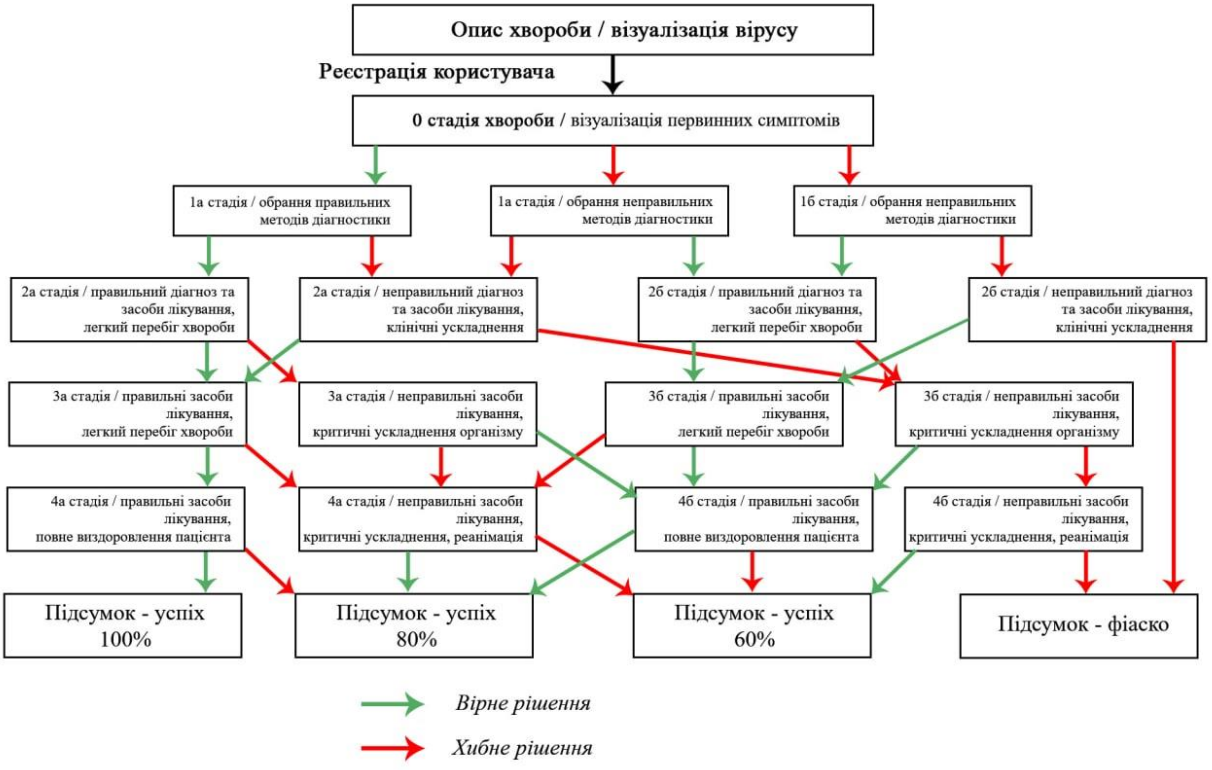

Рисунок 4.22 – Варіанти розвитку питань тесту

Всі стадії тесту були занесені в базу даних, включаючи підсумкову, де відображено відсоток правильних відповідей.

Для переходу між стадіями, в базі даних в таблиці uk_answer, було створено стовпець, який містить ідентифікатор наступної стадії.

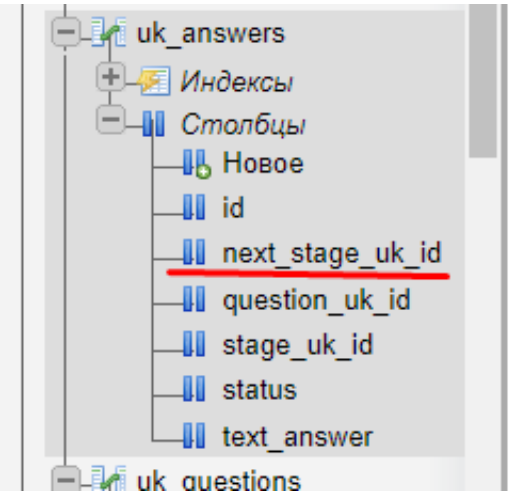

Рисунок 4.23 – Назва стовпців таблиці з питаннями

Структура сторінки самого теста має в собі такі компоненти, як анімацію, питання та варіанти відповідей на питання.

Кожна компонента була створена та підключена до основної компоненти TestPage.jsx.

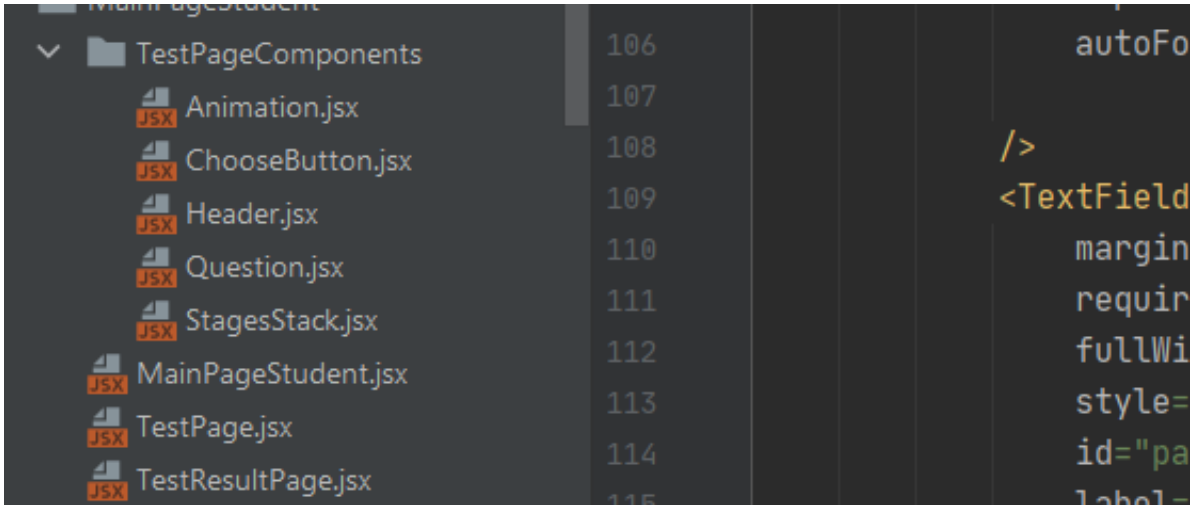

Рисунок 4.24 – Створення компонент

В компоненті ChooseButton.jsx було створено логіку переходу на наступну стадію. Після того, як користувач обрав певну відповідь, до сховища даних додатку заноситься ідентифікатор наступної альтернативної стадії з відповідним питанням, анімацією та варіантами відповідей. Після зміни стану, додаток оновлює об’єктне дерево елементів з новими даними і наступне питання предстає перед користувачем.

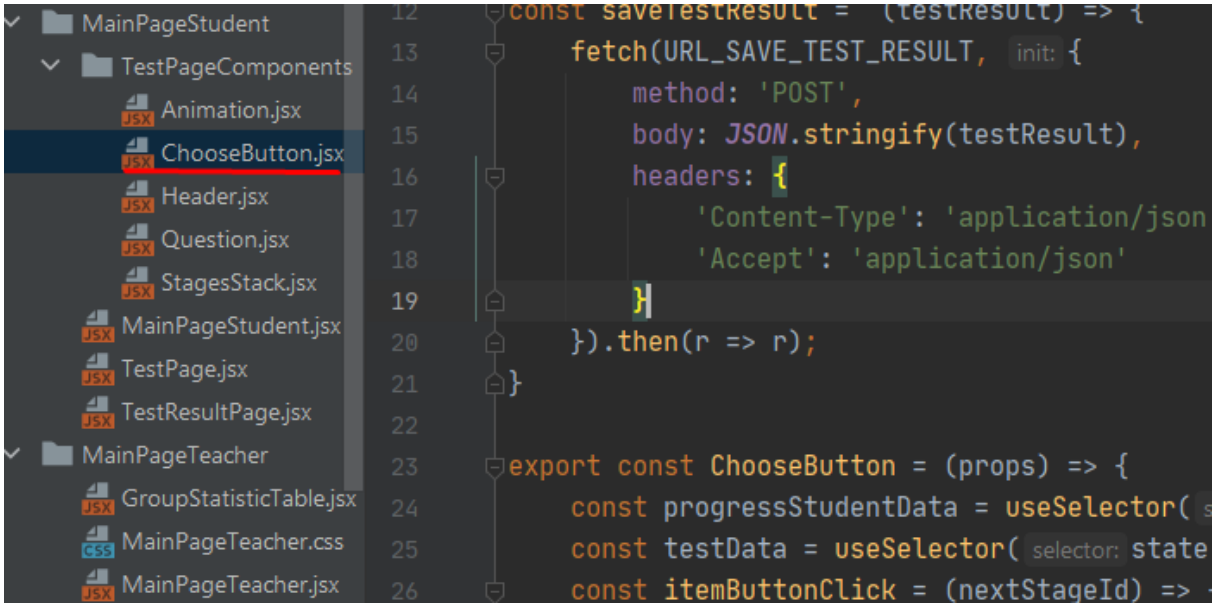

Рисунок 4.25 – Створення компоненти ChooseButton.jsx

Для зберігання анімацій та зображень створено папку images. В ній знаходяться файли з відео, які завантажуються під час тесту відповідно до певної стадії і питання.

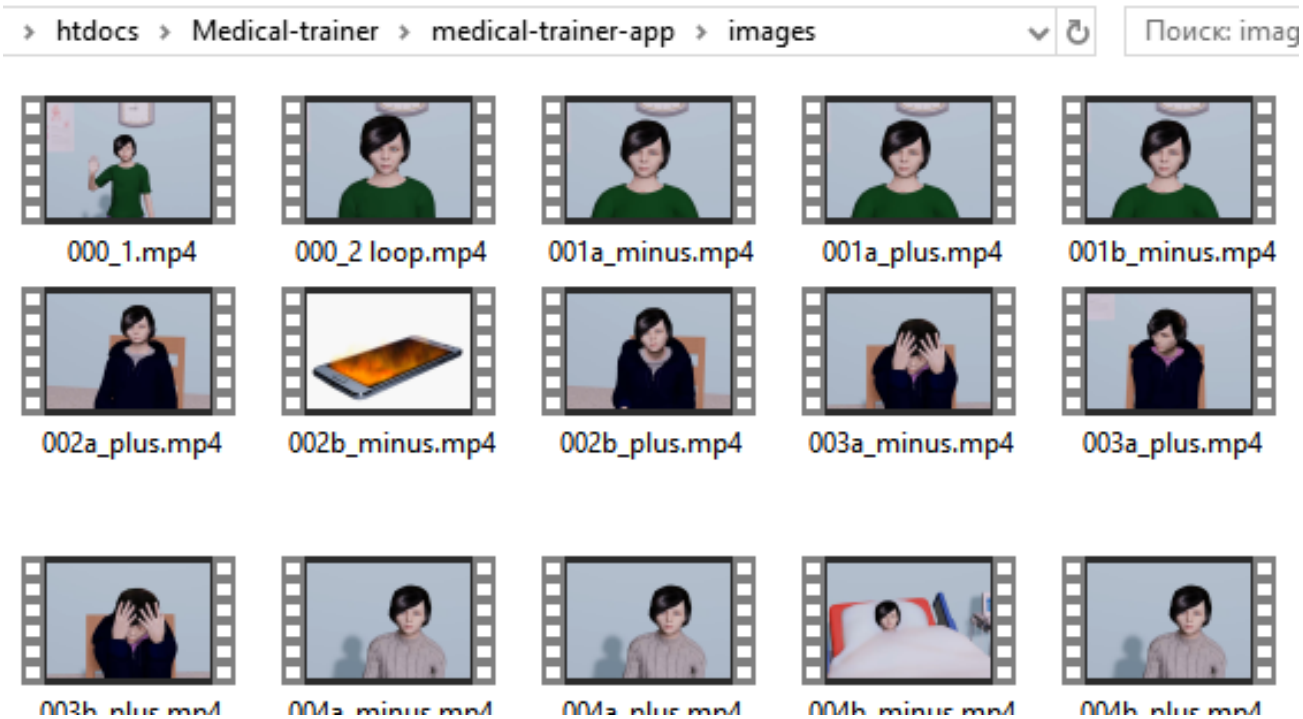

Рисунок 4.26 – Папка з файлами відео та зображення

Для відтворення анімації було створено компоненту animation.jsx, де залежно від ідентифікатору стадії оновлюється ім’я файлу. Для відео було використано тег «video» та «source», які мають певні атрибути, такі як адреса файлу відео, унікальний ключ, панель контролерів, зациклення та авто-відтворення. Тому користувач має можливість зупинити відтворення анімації в будь-який час.

Рисунок 4.27 – Створення компоненти анімації

Також було створено компоненту Question.jsx, в яку через стан додатку, передаються необхідні дані, а саме текст питання та ідентифікатор.

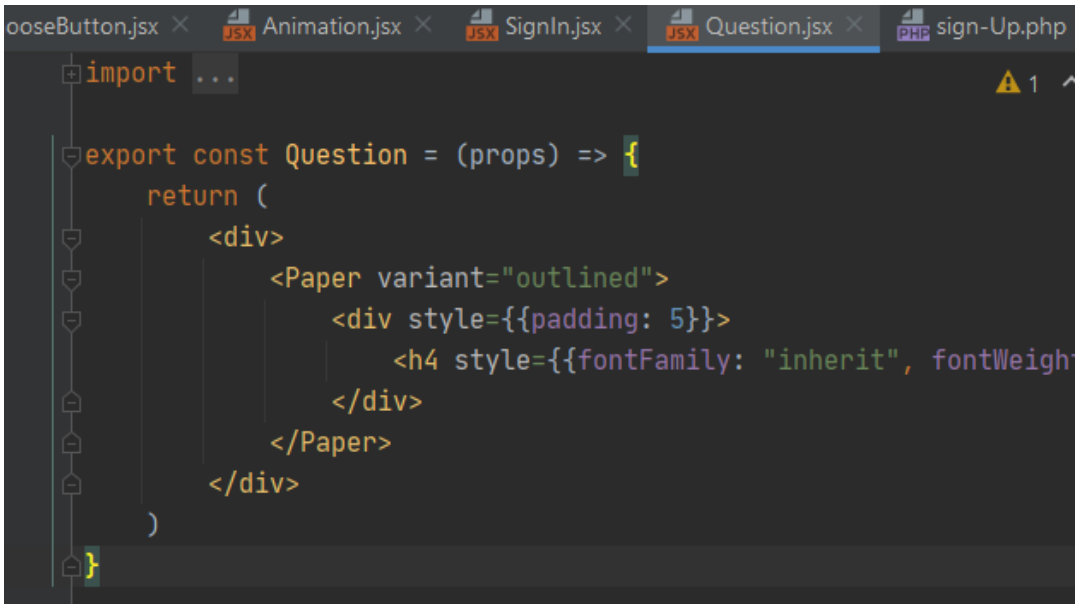

Рисунок 4.28 – Створення компоненти питання

Для акаунту викладача, було створено компоненти з таблицями, які містять в собі дані про студентів. Таблиці були реалізовані за допомогою компоненти «Datagrid», яка має певні атрибути, такі, як стовпці, рядки та локалізацію. Значення даних атрибутів береться із сховища даних додатку.

Рисунок 4.29 – Створення таблиць

Дизайн веб-додатку було створено у векторному графічному редакторі Figma [27].

Для відтворення запланованого дизайну веб-додатку використано фреймворк material-ui, який включає в собі компоненти з визначеними стилями.

Так виглядає дизайн основної сторінки з тестом:

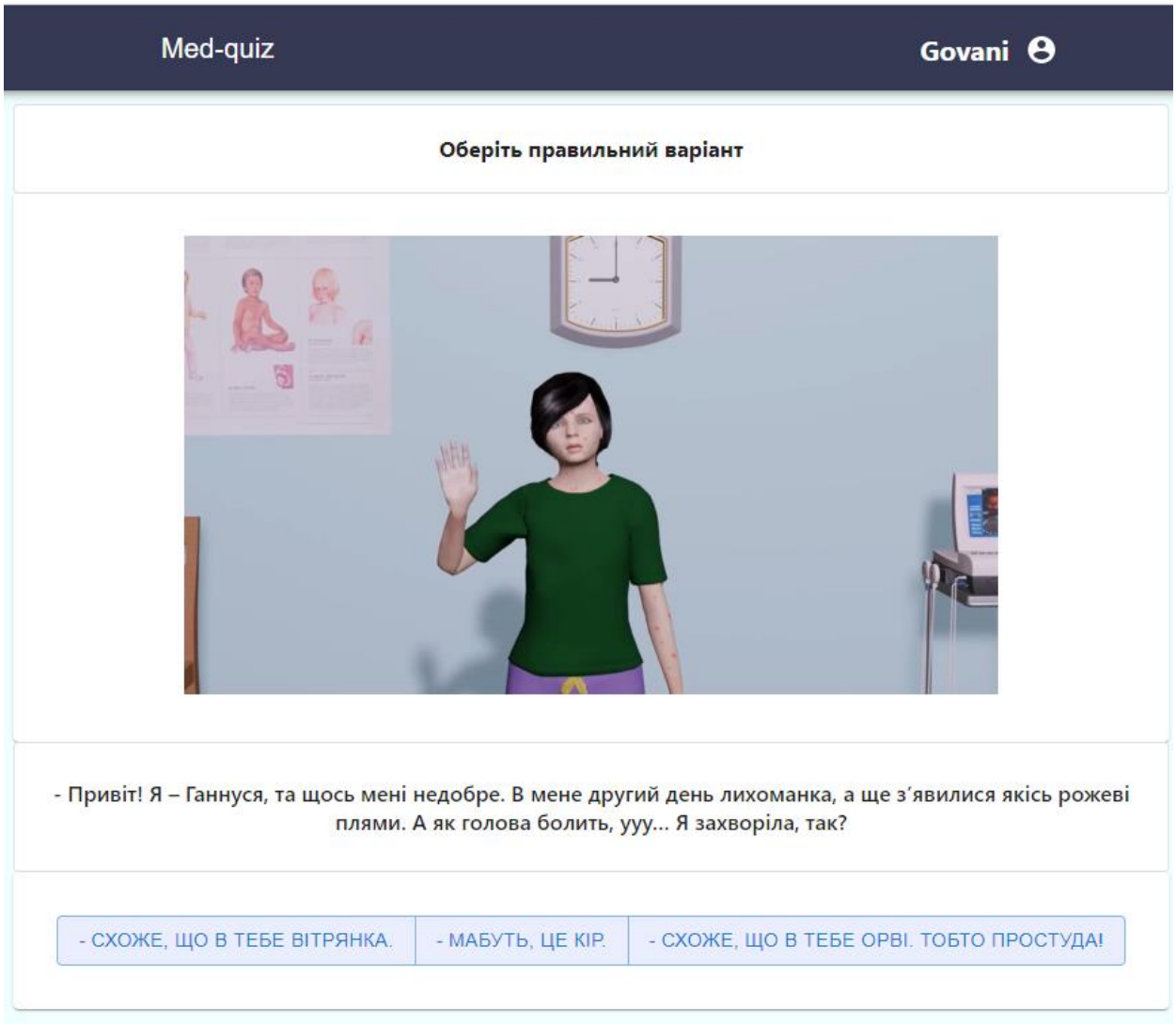

Рисунок 4.30 – Дизайн основної сторінки з тестом

Розроблений дизайн сторінки викладача представлений на наступному малюнку.

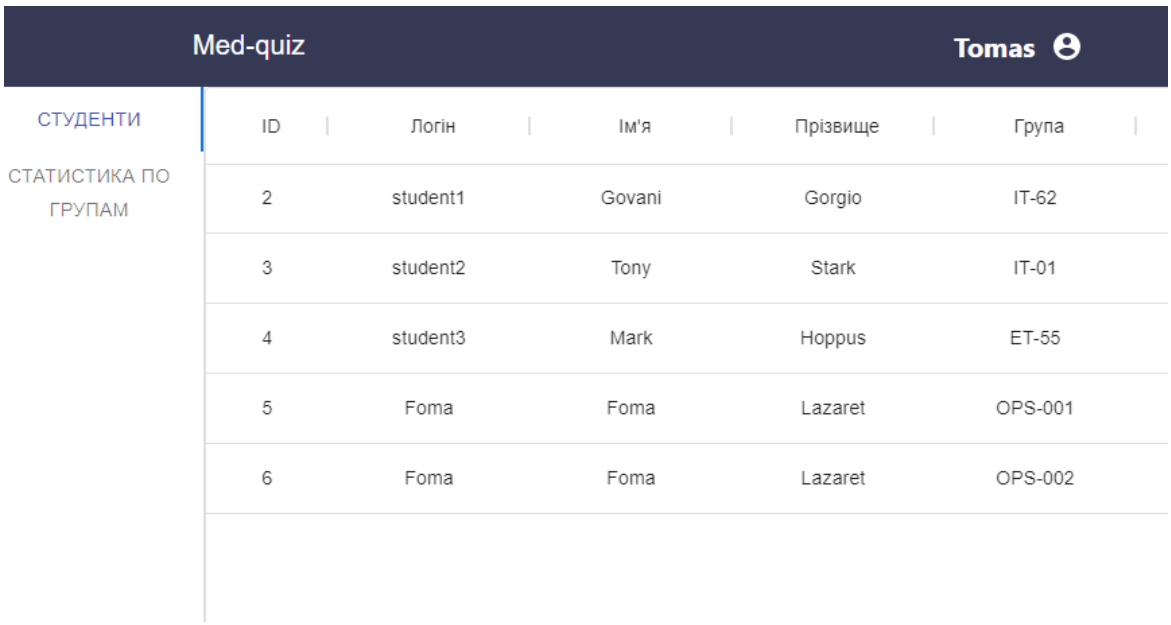

Рисунок 4.31 – Дизайн основної сторінки викладача

Також продемонструємо створення стартової сторінки з авторизацією та реєстрацією.

```
return (
    <Box component="form" sx={{mt: 1}}>
        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="login"
            label="Логін"
            name="login"
            autoComplete="login"
            inputRef={refUser}
            autoFocus

        />
        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="password"
            label="Пароль"
            name="password"
            autoComplete="password"
            type={"password"}
            inputRef={refKey}
            autoFocus
        />
```

Рисунок 4.32 – Створення форми авторизації

Для створення форм було застосовано теги TextField і Button з усіма необхідними атрибутами.

Результат коду представлений на наступному рисунку.

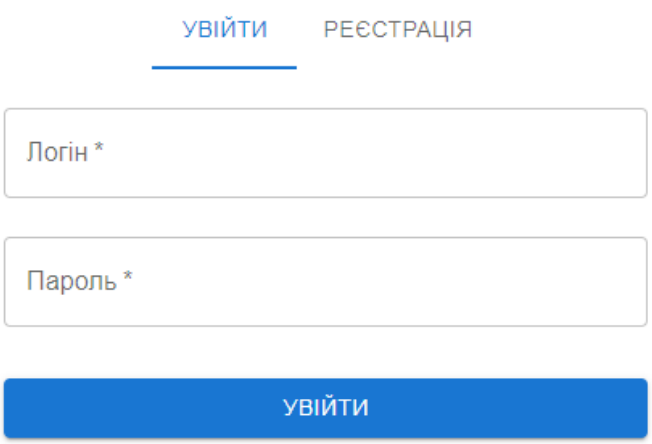

Рисунок 4.33 – Вигляд створеної форми авторизації

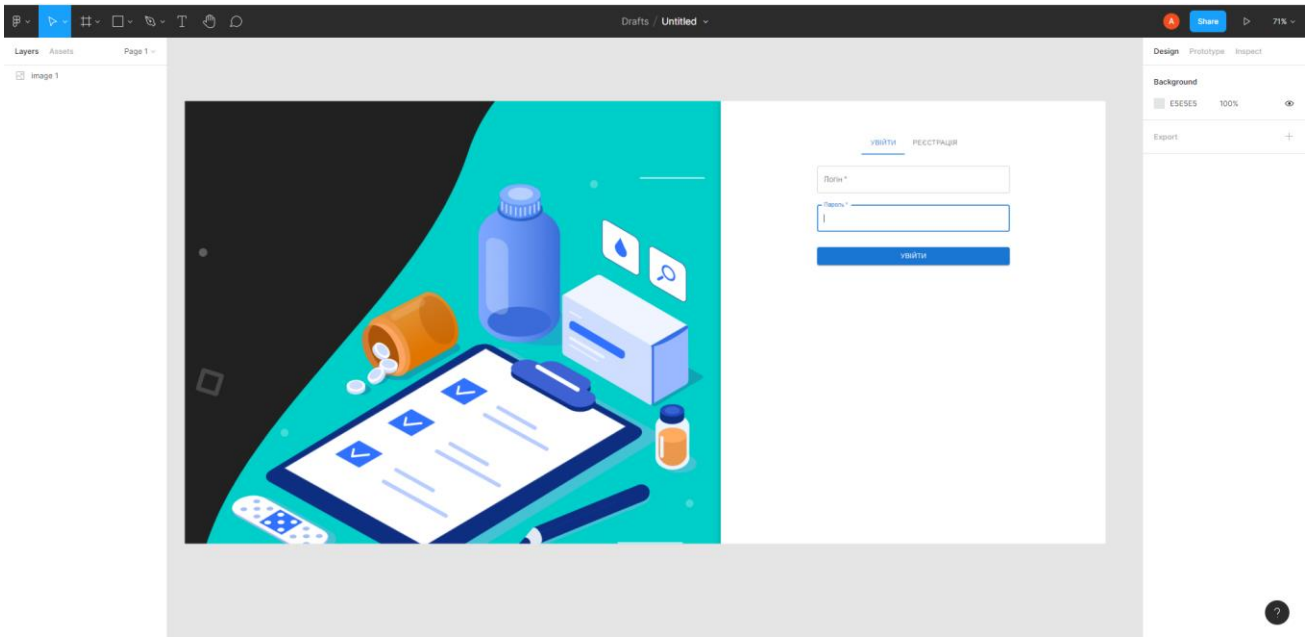

Рисунок 4.34 – Застосування Figma для розробки дизайну

Під час розробки для перевірки поточного результату, використовувалась команда npm start. Вона запускає локальний сервер і сам додаток.

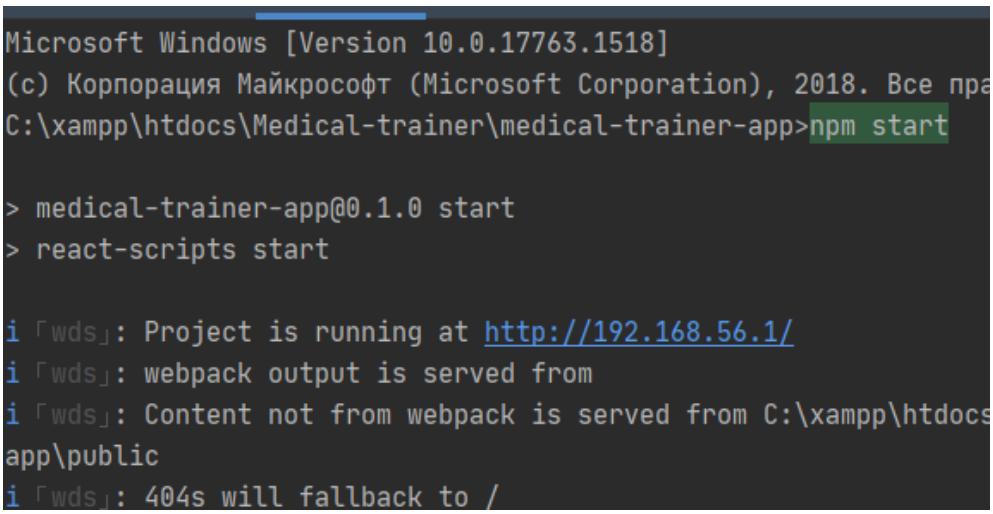

Рисунок 4.35 – Приклад запуску проекта

Після закінчення розробки веб-додатку, його було протестовано. За допомогою команди npm run build додаток було зібрано.

Код всієї програми наведено у додатку В.

4.3 Використання веб-додатку

Готовий веб-додаток завантажено на безкоштовний хостинг і там було продемонстровано його роботу:

1) Початок роботи веб-додатку – це сторінка з тематичним зображенням і авторизацією та реєстрацією:

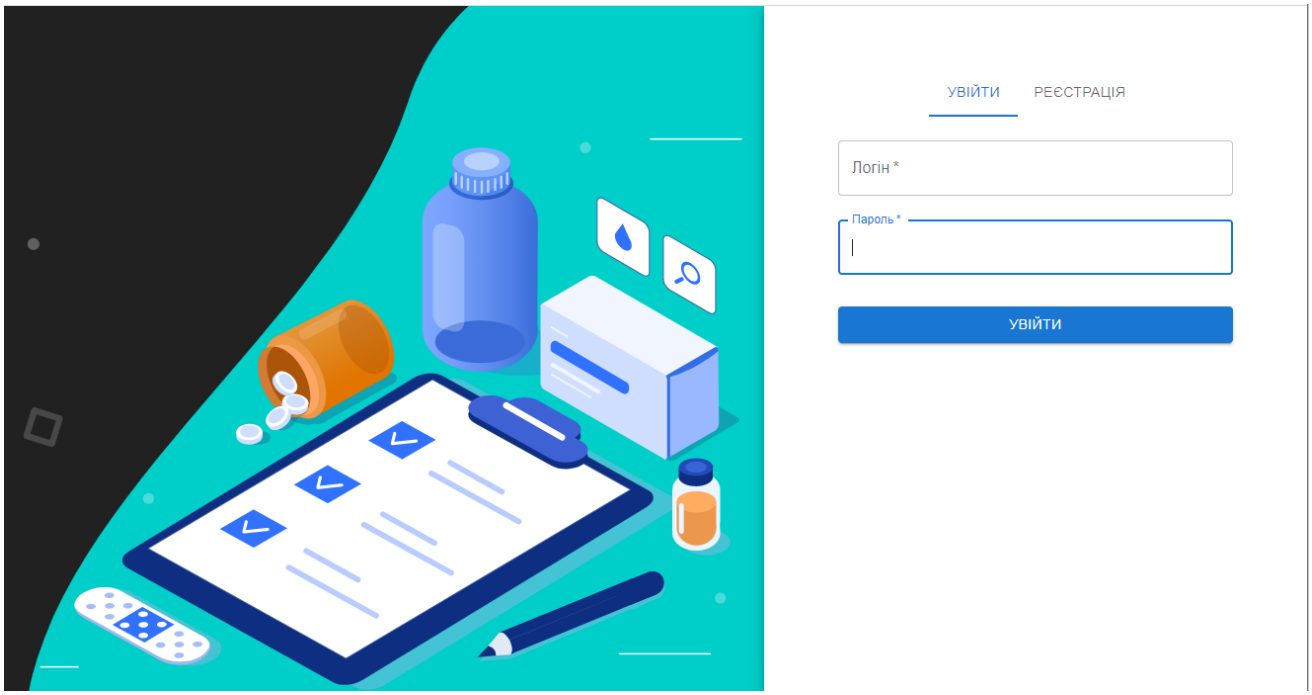

Рисунок 4.36 – Початкова сторінка веб-додатку

Користувач може авторизуватись або зареєструватись як студент.

2) Авторизуємося у веб-додатку як студент:

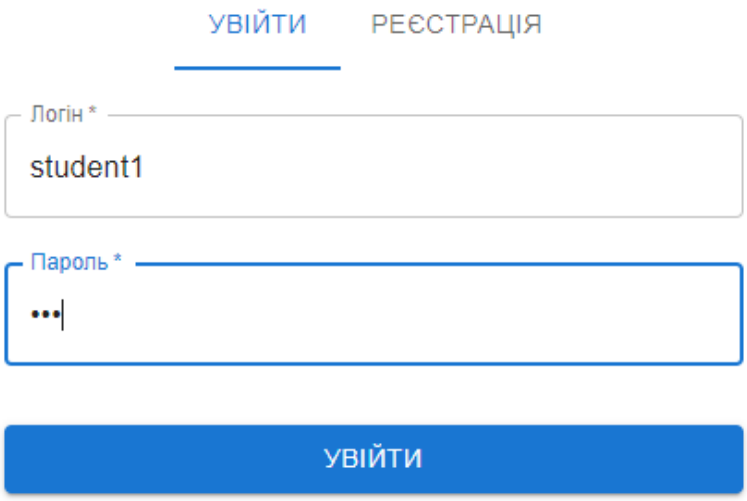

Рисунок 4.37 – Форма авторизації та реєстрації

Після авторизації сторінка оновлюється і завантажується привітання з можливістю розпочати тестування.

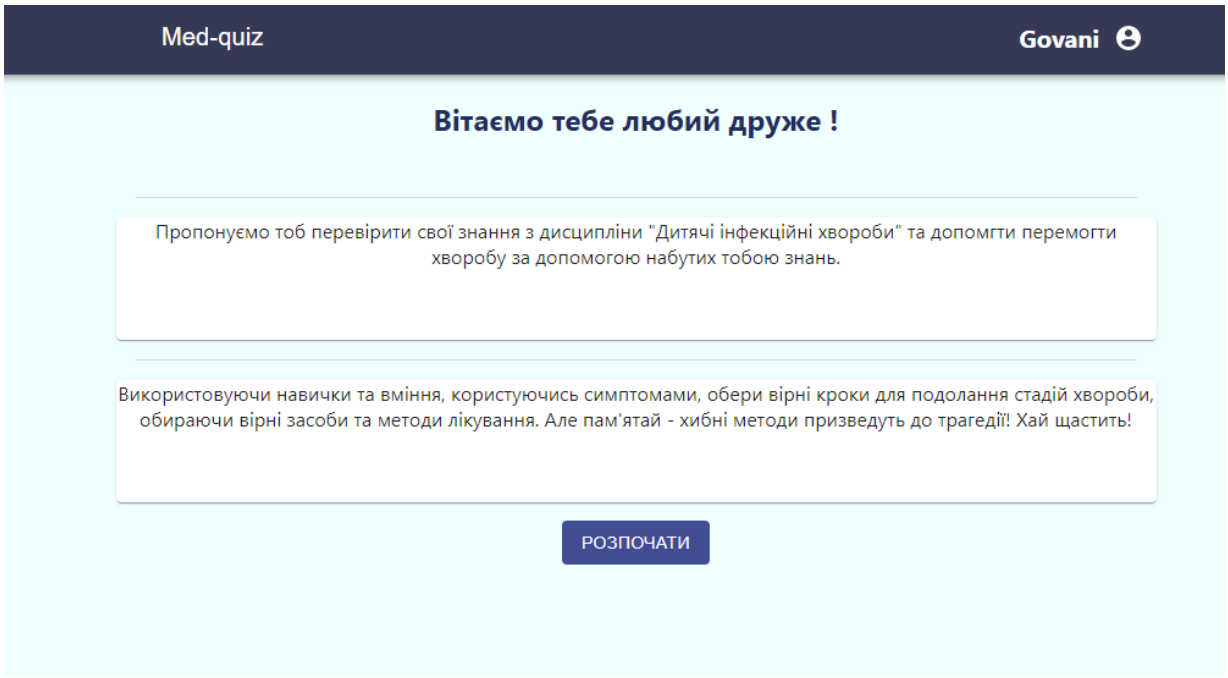

Рисунок 4.38 – Початкова сторінка ролі студент

3) Після того, як користувач натиснув кнопку розпочати тестування, відбувається перевірка на наявність спроб. Якщо у користувача немає спроб пройти тестування, то виводиться відповідне повідомлення:

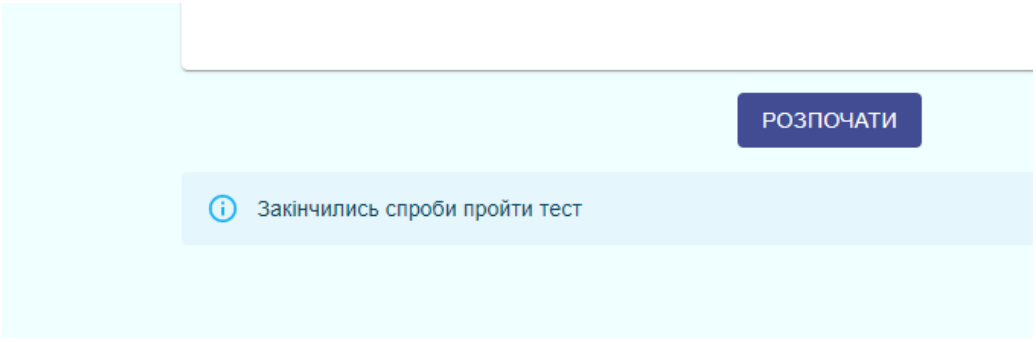

Рисунок 4.39 – Перевірка доступних спроб

Якщо у користувача є спроба на проходження тестування, то завантажується сам тест:

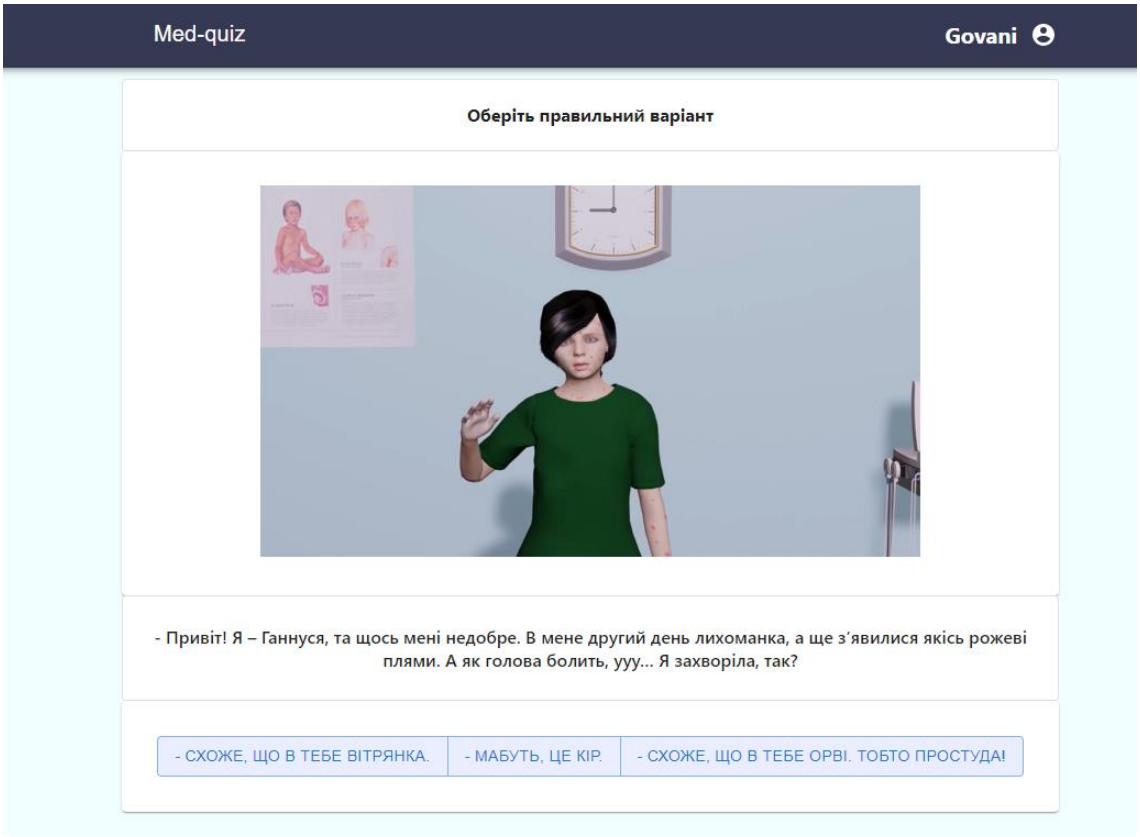

Рисунок 4.40 – Завантажений тест

Далі продемонстровано частину процесу тестування:

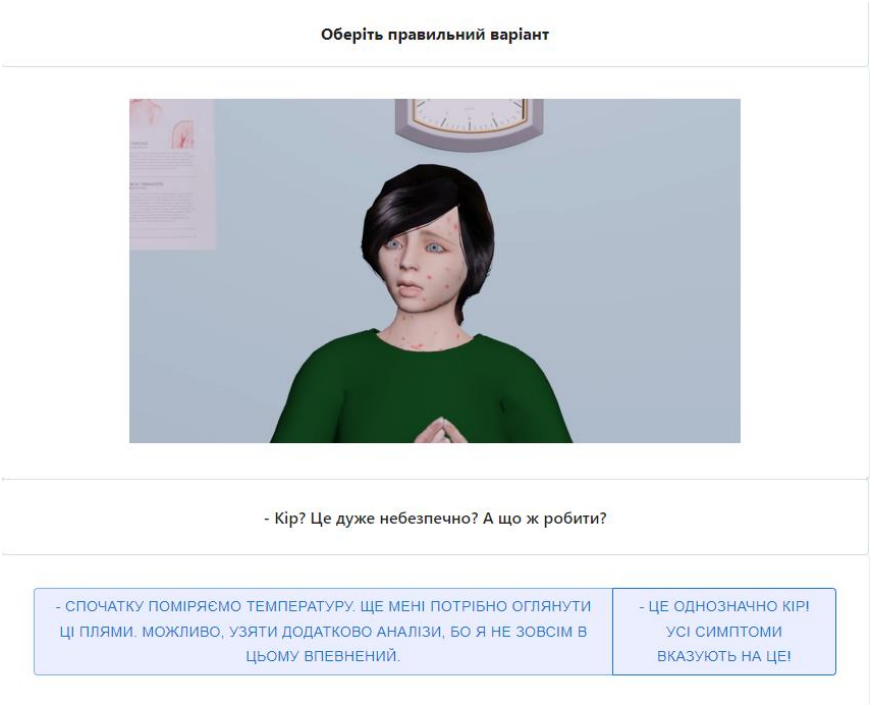

Рисунок 4.41 – Процес тестування

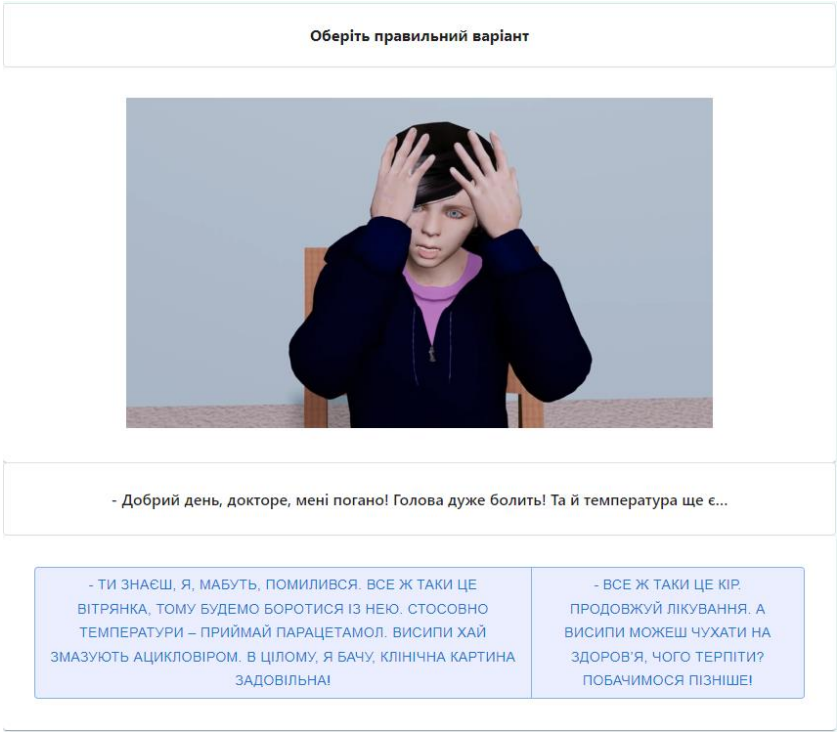

Рисунок 4.42 – Процес тестування

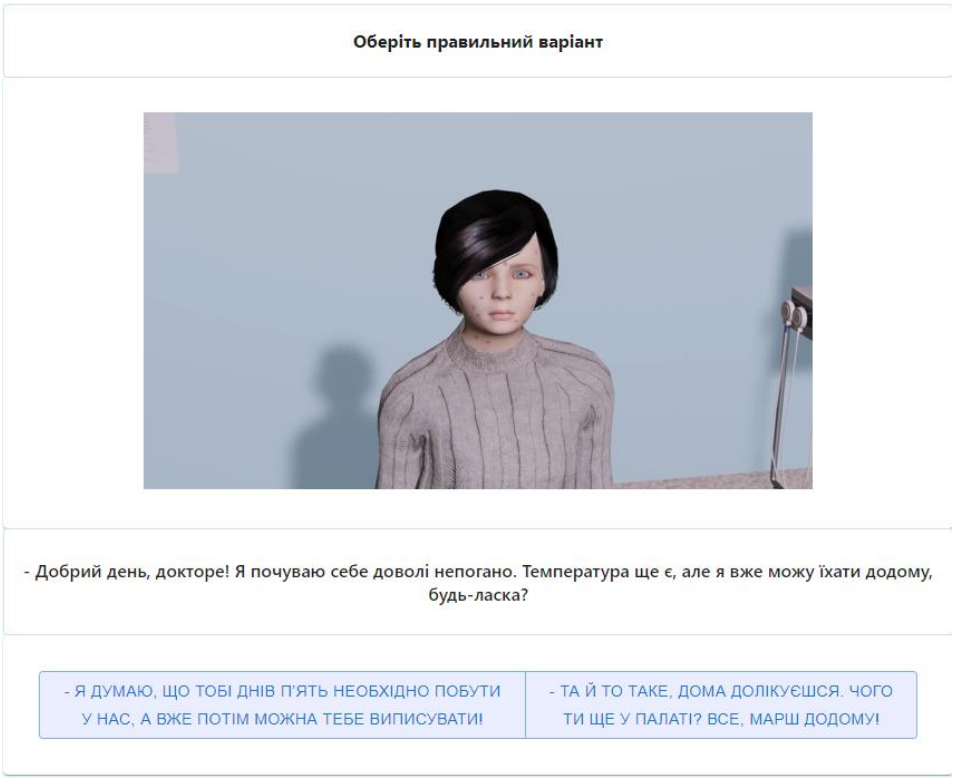

Рисунок 4.43– Процес тестування

4) Після проходження тесту користувачеві надається результат його відповідей та можливість повернутися на сторінку привітання:

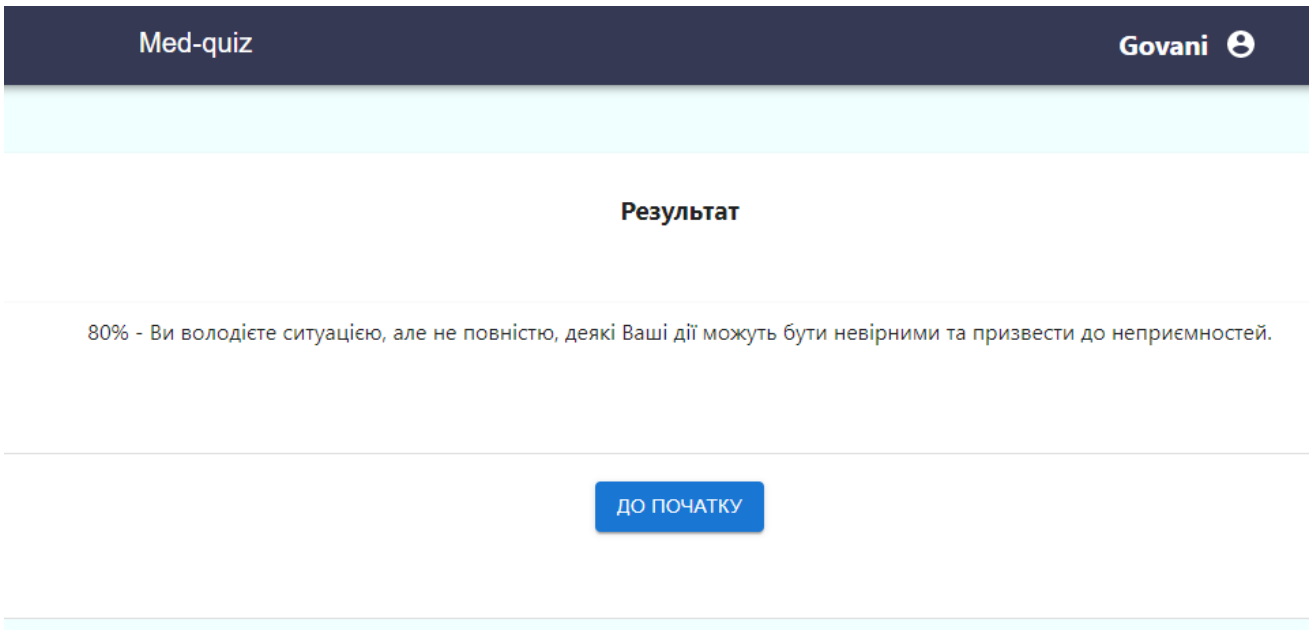

Рисунок 4.44 – Сторінка привітання

5) Далі вийдемо з акаунту студента і авторизуємося як викладач:

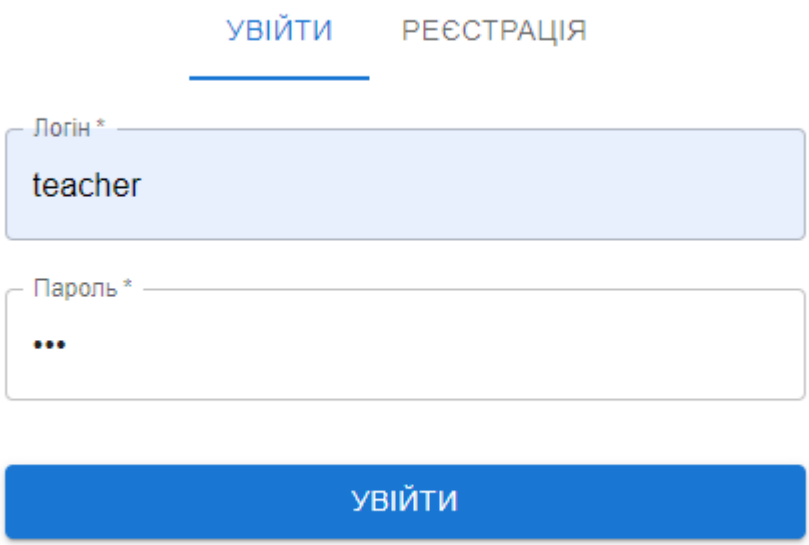

Рисунок 4.45 – Вхід до акаунту викладача

6) Після авторизації з’являється сторінка викладача з таблицями та відповідними функціональними можливостями:

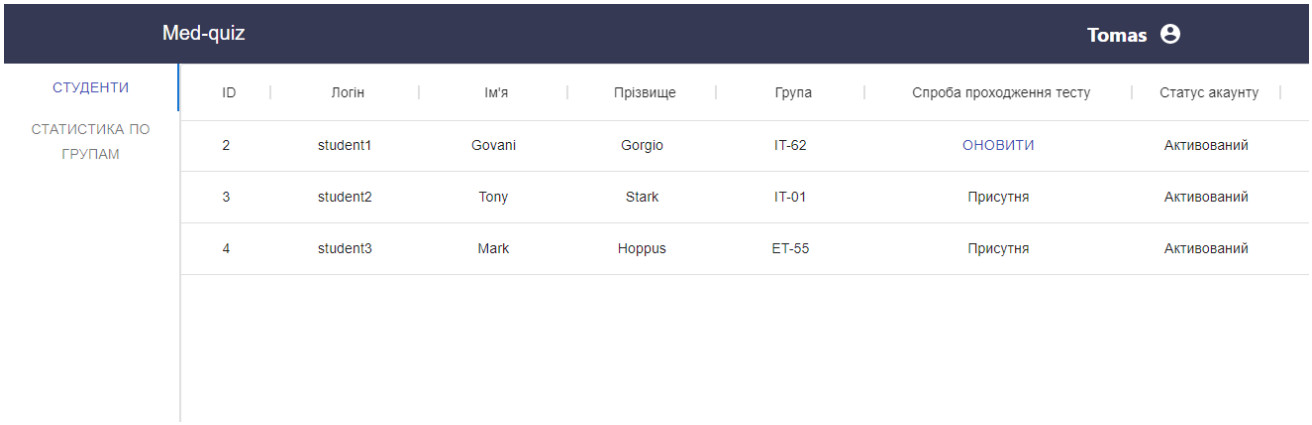

Рисунок 4.46 – Акаунт викладача

Викладач може переглядати інформацію про зареєстрованих користувачів та результатах тесту, оновлювати спроби проходження тестування та активувати акаунт студента, який зареєструвався.

7) Перейдемо до пункту «Статистика по групам»:

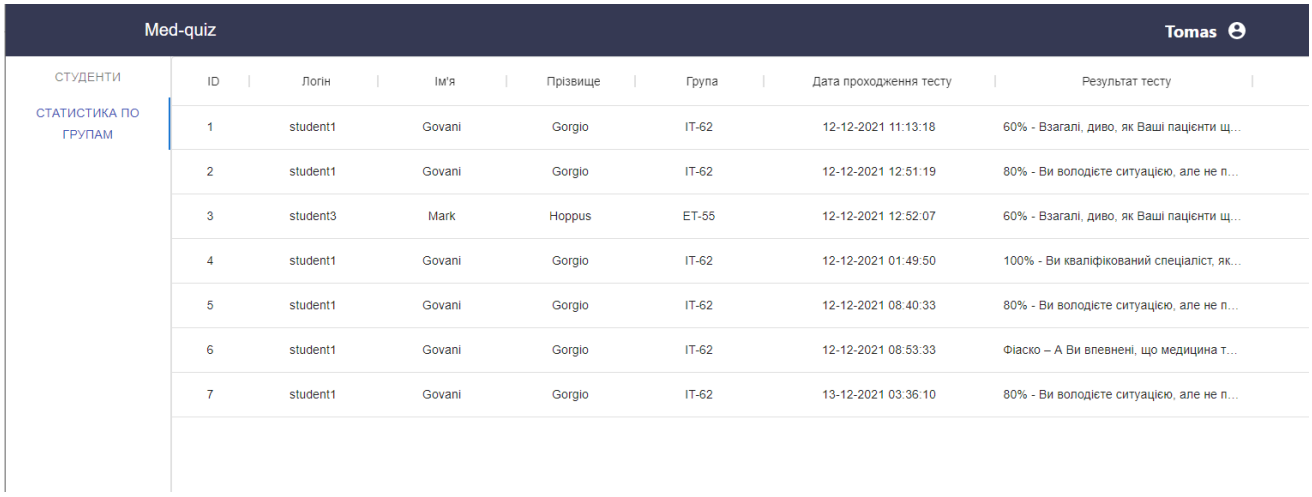

Рисунок 4.47 – Таблиці акаунту викладача

В даній таблиці представлені результати проходження тестування студентами.

Також можемо побачити результат тесту студента, з акаунту якого ми проходили контроль знань:

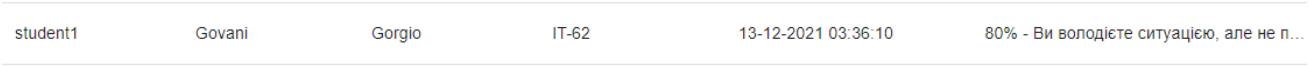

Рисунок 4.48 – Історія проходження тесту

При тестуванні роботи веб-додатку не було виявлено ніяких помилок.

ВИСНОВКИ

Кваліфікаційна робота магістра присвячена створенню інтерактивного додатку підтримки вивчення вітряної віспи в педіатрії.

Під час виконання було досліджено предметну область розроблюваного додатку, поставлені мета, чіткі задачі для її досягнення та сформульовано вимоги до розробки. Проведений аналіз серед аналогів показав, що розроблюваний додаток є актуальним і унікальним. Також було проведено аналіз необхідних функціональних потреб веб-додатку, та було обрано певний стек сучасних технологій для реалізації.

Проведено планування робіт: визначено та сформульовану мету проєкту методом SMART, розроблено структурну та організаційну діаграми, календарний план робіт, визначено можливі ризики виконання проєкту.

Розроблено структурно-функціональну модель додатку, спираючись на яку проведено подальшу практичну розробку додатка.

Було створено базу даних з необхідними таблицями полів та їх атрибутами.

Спроектовано структуру веб-додатку, розроблено дизайн сторінок.

Реалізовано практично необхідний функціонал інтерактивного додатку, проведена перевірка його працездатності.

Практичне значення розробленого проєкту полягає в тому, що він дає можливість користувачам закріпляти теоретичні знання в області педіатрії, а саме діагностики вітряної віспи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дитячі інфекційні хвороби: стаття. URL: https://spravochnick.ru/medicina/detskie_infekcionnye_bolezni/.

2. Інфекційні захворювання у дітей: діагностика, лікування та профілактика: стаття. URL: http://uldc.com.ua/ru/all-articles/189-infektsionnye-zabolevaniya-u-detej-diagnostika-lechenie-i-profilaktika.html.

3. Гуманітарні проблеми медицини та питання викладання у вищій медичній школі: стаття. URL: https://cyberleninka.ru/article/n/rozrobka-ta-vprovadzhennya-sistemi-elektronnogo-testuvannya-pri-vikladanni-medichnoyi-ta-biologichnoyi-fiziki/viewer (дата звернення: 19.10.2021).

4. Реформування системи медичної освіти в світлі концепції «суспільство знань»: наукова дискусія. URL: https://www.umj.com.ua/article/541/reformuvannya-sistemi-medichnoi-osviti-v-svitli-koncepcii-suspilstvo-znan (дата звернення: 15.11.2021).

5. Чернюк Н.В. Особливості застосування тестового контролю знань студентів при вивченні внутрішньої медицини, клінічної імунології та алергології у вищих медичних навчальних закладах: стаття. URL: https://cyberleninka.ru/article/n/osoblivosti-zastosuvannya-testovogo-kontrolyu-znan-studentiv-pri-vivchenni-vnutrishnoyi-meditsini-klinichnoyi-imunologiyi-ta/viewer.

6. Афанасьев А. Н. Модель и метод разработки и анализа компьютерных тренажеров [Текст] / А. Н. Афанасьев, Н.Н.Войт, Д. С. Канев // Автоматизация процессов управления. — 2015. — No 2. — С. 64–71.

7. Доценко Н. Застосування навчальних комп'ютерних інтерактивних тренажерів здобувачами вищої освіти інженерних спеціальностей в умовах інформаційно-освітнього середовища. // Педагогічні науки: теорія, історія, інноваційні технології, 2018, No 2 (76). URL: https://pedscience.sspu.sumy.ua/wp-content/uploads/2018/05/14-1.pdf.

8. Медичні тести – основні дисципліни: програмний додаток. URL: https://apps.apple.com/ru/app/%D0%BC%D0%B5%D0%B4%D0%B8%D1%86%D0

%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D1%8B/id1059865223 (дата звернення: 19.10.2021).

9. Тести з фармакології: програмний додаток. URL: https://play.google.com/store/apps/details?id=com.antomy.stomatologytest (дата звернення: 19.10.2021).

10. Загальна патологія інфекційних хвороб. Особливості інфекційного процесу: онлайн тест. URL: https://onlinetestpad.com/ru/testview/265270-obshhaya-patologiya-infekcionnykh-boleznej-osobennosti-infekcionnogo-proces (дата звернення: 19.10.2021).

11. Большунов, Д.М. Інтерактивний додаток для навчання студентів діагностиці дитячих інфекційних хвороб [Текст]: робота на здобуття кваліфікаційного рівня бакалавр; спец.: 122 – комп'ютерні науки, освітньо-професійна програма «Інформаційні технології проектування» / Д.М. Большунов; наук. кер. І.В. Баранова – Суми: СумДУ, 2020. – 56 с.

12. Restful-api. URL: https://habr.com/ru/post/483202

13. Personal Home Page Tools — scripting programming language. URL: ww.php.net/home (дата звернення: 19.10.2021).

14. React — open JavaScript library for creating user interfaces. URL: uk.reactjs.org (дата звернення: 19.10.2021).

15. Redux – open library to create the state of the application .URL: https://redux.js.org/info/14278 (дата звернення: 19.10.2021).

16. Material UI – open source framework for creating the interface and its styles. URL: https://mui.com/stack/14333 (дата звернення: 19.10.2021).

17. Mysql – СУБД. URL: https://mysql.com/info (дата звернення: 19.10.2021).

18. PhpMyadmin- web interface for database management. URL: https://php.myadmin (дата звернення: 19.10.2021).

19. Javascript – programming language. URL: https://javascript.com (дата звернення: 19.10.2021).

20. PhpStorm – development environment. URL: https://jetbrains.com (дата звернення: 19.10.2021).

21. Npm – package manager . URL: https://npmjs.com (дата звернення: 19.10.2021).

22. Redux-toolkit – web application status monitoring tool. URL: https://reduxtoolkit.com (дата звернення: 19.10.2021).

23. Http – internet protocol. URL: https://developer.mozila.org (дата звернення: 19.10.2021).

24. POST – method of data transfer between client and server. URL: https://developer.mozila.org (дата звернення: 19.10.2021).

25. Fetch – технологія створення http запитів. URL: https://javascript.learn.ru (дата звернення: 19.10.2021).

26. Json – тип файлу, що слугує для зберігання даних в форматі строки . URL: https://javascript.learn.ru (дата звернення: 19.10.2021).

27. Figma – векторний графічний редактор. URL: https://figma.com (дата звернення: 19.10.2021).

28. Цели по SMART: подробный обзор [Електронний ресурс] // PowerBranding.ru. – 2017. – URL: http://powerbranding.ru/marketing-strategy/smart-celi/#:~:text=SMART%20является%20аббревиатурой%2C%20расшифровка%20которой,критерий%20smart%20цели%20более%20подробно.

29. «Управління проєктами»: навч. посібн. до вивчення дисципліни для магістрів галузі знань 07 «Управління та адміністрування» спеціальності 073 «Менеджмент» спеціалізації: «Менеджмент і бізнес-адміністрування», «Менеджмент міжнародних проєктів», «Менеджмент інновацій», «Логістика»/ Уклад.: Л.Є. Довгань, Г.А.Мохонько, І.П Малик. – К.: КПІ ім. Ігоря Сікорського, 2017. – 420 с.

30. Єгорченков О. В. Азбука управління проєктами. Планування : навч. посіб. / О. В. Єгорченков, Н. Ю. Єгорченкова, Є. Ю. Катаєва. – Київ: КНУ ім.Т.Шевченка, 2017. – 117 с.

31. Методичні рекомендації до виконання лабораторних робіт з дисципліни «Управління ІТ-проєктами» студентами напряму підготовки 122 Комп'ютерні науки / І.Г. Гуліна, В.П. Козлов, О.С. Шевцова; М-во освіти і науки України, Нац. гірн. ун-т.– Дніпро: НГУ, 2017. – 42 с

32. План действий при управлении рисками проекта [Електронний ресурс] // Projectimo.ru. – 2020. – URL: http://projectimo.ru/upravlenie-riskami/riski-proekta.html.

33. Застосування UML для моделювання та проектування інформаційних систем. // Методичні вказівки до лабораторного практикуму з дисципліни „Об’єктно-орієнтований аналіз та проектування‖ для студентів напряму підготовки 123 -―Комп’ютерна інженерія‖./ Укл. А.М. Акименко, І.В. Богдан, А.С. Посадська ―Чернігів: ЧНТУ, 2018. ― 37 с.

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація мети IT-проекту

Деталізація мети проекту методом SMART [28]. Метою дипломної роботи є розробка інтерактивного додатка для підтримки вивчення вітряної віспи в педіатрії. Даний проєкт допоможе студентам медичних закладів оновити та перевірити свої знання в області дитячих хвороб.

S. Розробка структури інтерактивного тест додатку для вивчення теми з дисципліни «Інфекційні дитячі хвороби» в педіатрії.

M. Кінцевим результатом даного проекту повинен бути інтерактивний тренажер у вигляді веб-додатку. Результати проходження тренажеру зберігаються в pdf файл. Користувачу необхідно ідентифікуватися для подальшого проходження тесту. Додаток повинен містити тільки ту інформацію, яка відносяться безпосередньо до заданого завдання.

A. Проект буде реалізований у вигляді веб-додатку мовою Javascript. Також для збереження даних буде використана база даних MySQL.

R. Поставлену мету звичайно можна реалізувати тому, що на даний момент існує велика кількість ПЗ та ресурсів, які за допомогою своїх функцій скорочують час побудови, проектування, розробки проекту.

T. Обмеженість в часі зумовлена рішенням замовника, щоб якомога швидше отримати програмний продукт (додаток).

А.2 Планування змісту структури робіт ІТ-проекту

Структурна декомпозиція роботи **(структура розбиття роботи, WBS)** – це ієрархічна побудова роботи, побудована з метою логічного розподілу всіх служб для виконання проекту та подана у графічній формі [29-30]. Це поєднання декількох рівнів, кожен з яких в кінцевому рахунку розвивається між розподілом роботи попереднього рівня на його компоненти. Компонент найнижчого рівня - це склад роботи, або так званий робочий пакет.

Розроблена структура роботи представлена на рисунку А.1.

Наступним кроком створена OBS структура.

Організаційна структура проекту (OBS) – представляється графічним відображенням учасників проекту (фізичних та юридичних осіб) та їх відповідальних осіб, які беруть участь у здійсненні проекту. На найвищому рівні проекту OBS розташовані менеджер та команда з управління проектами; на наступному рівні виступають виконавці. Кінцевий рівень структури OBS представлений виконавцями. Не обов'язково бути менеджерами, а тими працівниками, яким безпосередньо доручено створити та відповідати підряднику для впровадження конкретного компонента структури WBS.

Розроблена організаційна структура представлена на рисунку А.2.

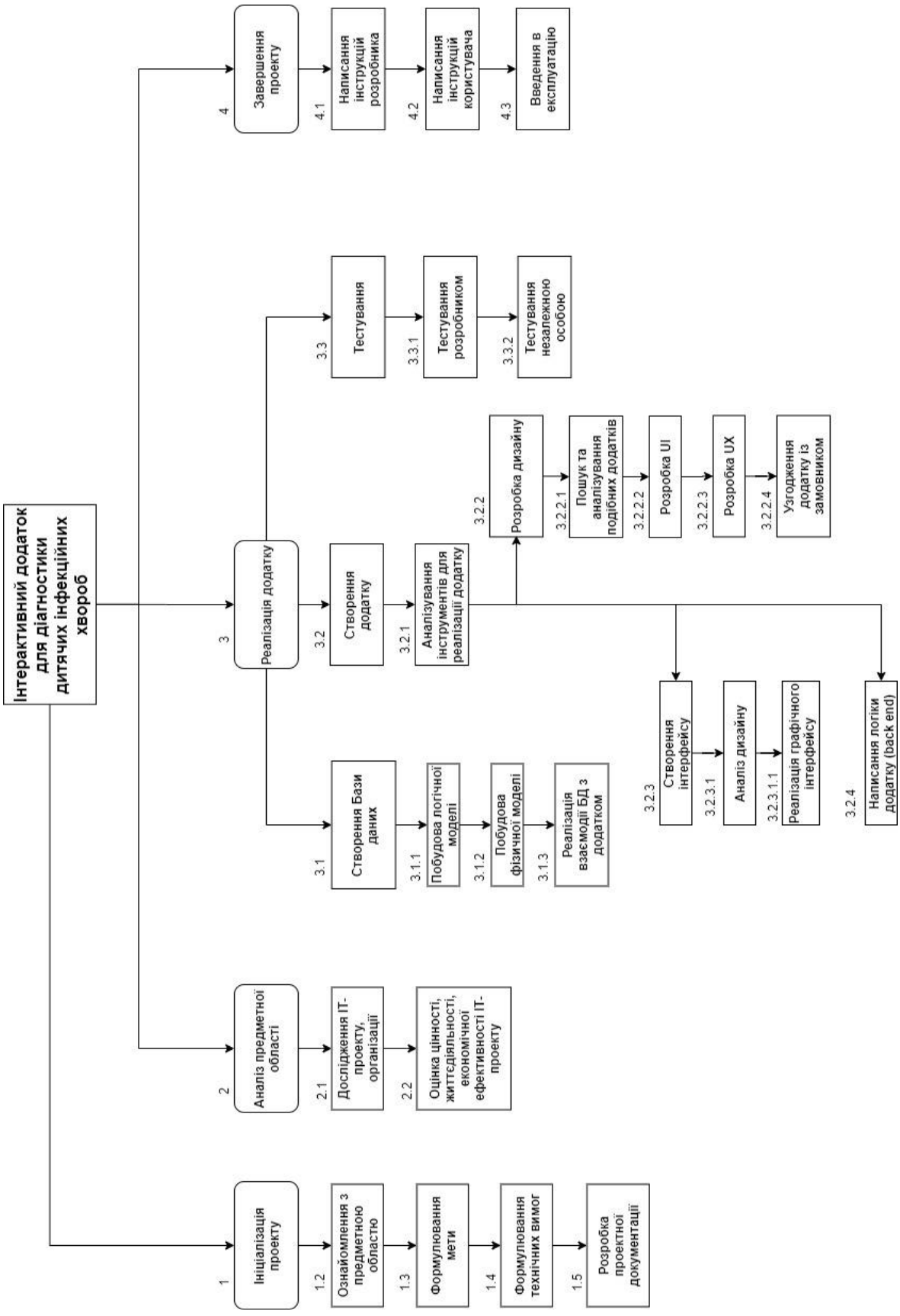

Рисунок А.1 - Структура проекту

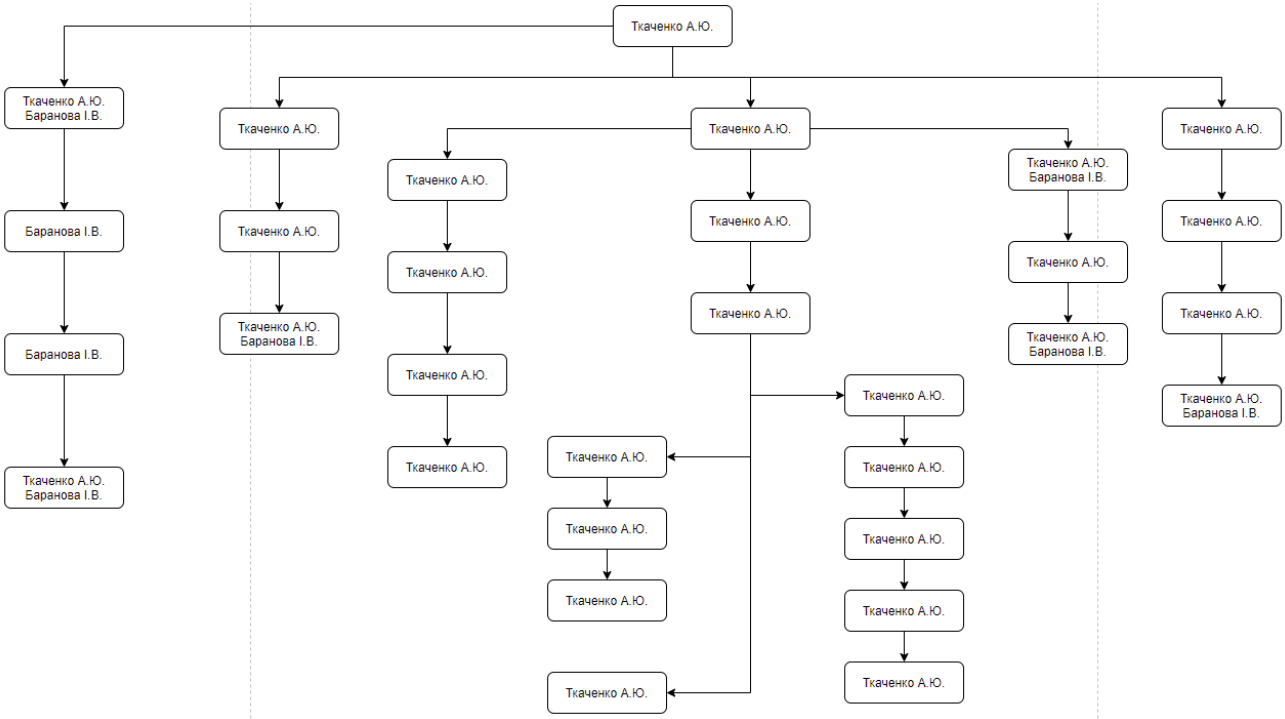

Рисунок А.2 - Організаційна структура проекту

А.3 Побудова календарного графіка виконання ІТ - проекту

Діаграма Ганта – це відома версія діаграми (придумана Генрі Ганттом), яка застосовується для планування та контролю за виконанням проекту [31-33]. Така інтерактивна мережева діаграма існує майже у всіх системах управління проектами. На схемі (структурі) відзначено завдання та стадії плану з урахуванням часу їх виконання. Завдання на діаграмі можуть існувати залежно одне від одного (наприклад, одне завдання може з’являтися лише в кінці другого). Крім того, може відображатися відсоток виконання будь-якого завдання та відповідальність за його виконання.

Розроблена діаграма Ганта дипломного проєкту представлена на рисунку А.3.

Имя задачи	Начало	Длительность	Готово
Інтерактивний додаток для діагностики інфекційних хвороб	20.10.2021	51	11.12.2021
Ініціалізація проекту	20.10.2021	4	24.10.2021
Ознайомлення з предметною областю	20.10.2021	1	21.10.2021
Формування мети	21.10.2021	1	22.10.2021
Формування технічних вимог	22.10.2021	1	23.10.2021
Розробка проектної документації	23.10.2021	1	24.10.2021
Аналіз предметної області	24.10.2021	3	28.10.2021
Дослідження IT-проекту, організації	24.10.2021	1	26.10.2021
Оцінка цінності, життєдіяльності, економічності ефективності IT-проекту	26.10.2021	2	28.10.2021
Реалізація додатку	28.10.2021	41	11.12.2021
Створення Бази даних	28.10.2021	8	13.11.2021
Побудова логічної моделі	05.11.2021	2	07.11.2021
Побудова фізичної моделі	07.11.2021	2	09.11.2021
Реалізаціявзаємодії БД з додатком	09.11.2021	4	13.11.2021
Створення додатку	13.11.2021	29	08.12.2021
Аналізування інструментів для реалізації додатку	13.11.2021	29	08.12.2021
Розробка дизайну	13.11.2021	11	25.11.2021
Пошук та аналізування подібних додатків	13.11.2021	1	14.11.2021
Розробка UI	14.11.2021	5	19.11.2021
Розробка UX	19.11.2021	5	24.11.2021
Узгодження додатку із замовником	24.11.2021	1	25.11.2021
Створення інтерфейсу	25.11.2021	4	29.11.2021
Аналіз дизайну	25.11.2021	1	26.11.2021
Реалізація графічного інтерфейсу	26.11.2021	3	29.11.2021
Написання логіки додатку	29.11.2021	9	08.12.2021
Тестування	08.12.2021	1	09.12.2021
Тестування незалежними особами	08.12.2021	1	09.12.2021
Завершення проекту	09.12.2021	3	11.12.2021
Написання інструкцій розробника	09.12.2021	1	10.12.2021
Написання інструкцій користувача	10.12.2021	1	11.12.2021
Введення в експлуатацію	11.12.2021	1	11.12.2021

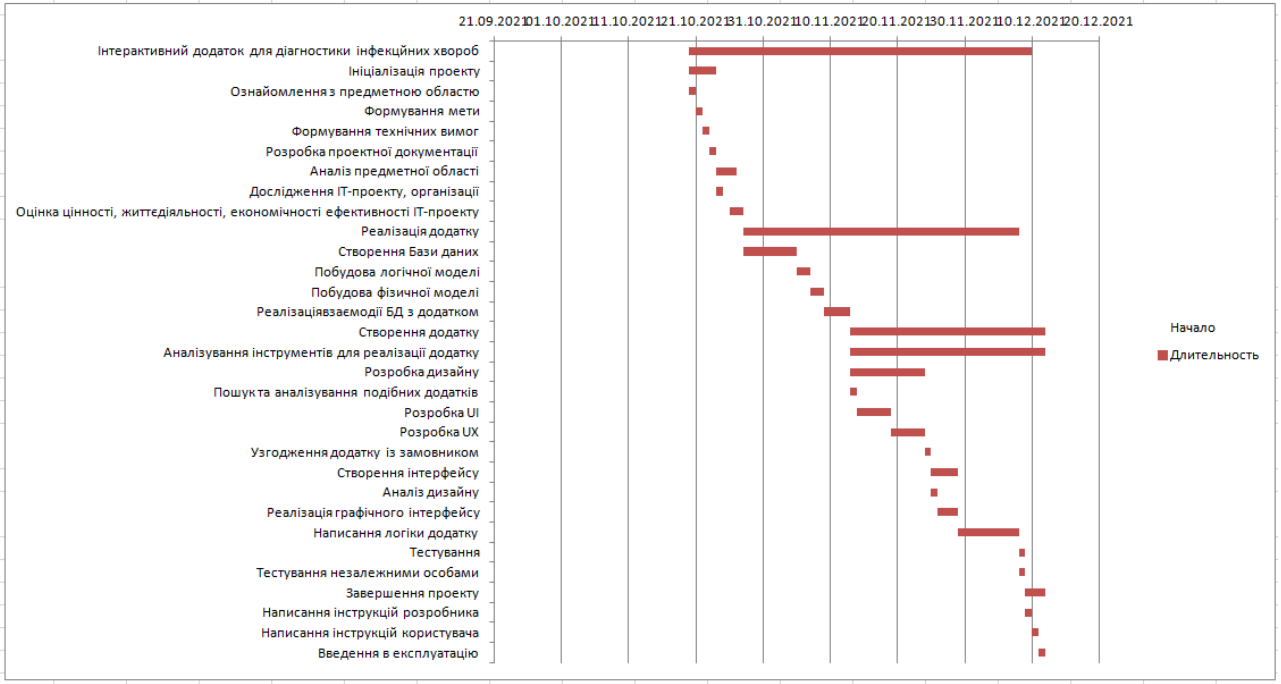

Рисунок А.3 - Діаграма Ганта

Побудова матриці відповідальності. Матриця відповідальності – це спеціальна методика пошуку функціональних зон, ключових сфер діяльності, критеріїв прийняття адміністративних рішень, де є неоднозначності. Усі розбіжності, що виникають у процесі передбаченого процесу, можуть бути винесені на загальний розгляд та згодом вирішені шляхом прийняття спільного рішення. У таблиці А.1 показано матрицю відповідальності проекту.

Таблиця А.1 – RAM

№	**Фази**	Ткаченко А.Ю.	Баранова І.В.
1)	Ідентифікація проекту	A	A
2)	Ознайомлення з предметною областю	R	C
3)	Формулювання мети	R	A
4)	Формулювання технічних вимог	R	C
5)	Розробка проектної документації	R	C
6)	Аналіз предметної області	R	C
7)	Дослідження ІТ-проекту, організації	R	
8)	Оцінка цінності, життєздатності, економічної ефективності ІТ-проекту		R
9)	Реалізація додатку	R	C
10)	Створення Бази Даних	R	C
11)	Побудова логічно моделі	R	C
12)	Побудова фізичної моделі	R	C
13)	Реалізація взаємодії БД із додатком	R	C
14)	Створення додатку	R	A
15)	Обирання мови програмування	R	
16)	Створення інтерфейсу	R	C
17)	Написання логіки додатку	R	
18)	Розробка дизайну	R	C
19)	Пошук та аналізування подібних додатків	R	
20)	Розробка UI	R	A
21)	Розробка UX	R	A
22)	Узгодження дизайну із замовником	R	C
23)	Тестування розробником	R	I
24)	Тестування незалежною особою	I	R
25)	Завершення проекту	R	C
26)	Написання інструкцій адміністратора	R	C
27)	Написання інструкцій для користувача	R	C
28)	Введення в експлуатацію	R	A

А.4 Планування ризиків проекту

Ризик – це можлива подія, яка у разі настання негативно або позитивно впливатиме на проект.

Процес управління ризиками включає наступні етапи:

1. ідентифікація
2. процес оцінювання ризиків, який включає в себе якісний, кількісний аналіз.
3. заходи реагування на ризики
4. моніторинг заходів і ризиків

Ризики проекту представлені у таблиці А.2.

Таблиця А.2 – Risk Register

№	Назва ризику	Ймовірність	Величина втрат
1	Замовник довго не відповідає	1	4
2	Неоптимальний розподіл часу	4	4
3	Неполадки з Інтернет зв’язком	4	5
4	Недостатній рівень знань розробника	1	3
5	Замовника не влаштовує кінцевий результат	2	2
6	Захворювання розробника	2	3
7	Проблеми із Базою Даних	2	3
8	Платне програмне забезпечення	4	3
9	Вихід з ладу апаратного забезпечення	5	2
10	Затримка надання матеріалу	2	4

1. За імовірністю виникнення:
   - слабо ймовірнісні - 1;
   - мало ймовірнісні - 2;
   - імовірні - 3;
   - досить імовірні - 4;

- майже імовірні - 5.

2. За величиною втрат:
   - Мінімальна - 1;
   - Низька - 2;
   - Середня - 3;
   - Висока - 4;
   - Максимальна - 5.

Таблиця А.3 - Матриця ризиків

5		9			
4			8	2	3
3					
2		**5**	**6, 7**	**10**	
1			**4**	**1**	
Величина втрат / Ймовірність	1	2	3	4	5

ДОДАТОК Б
ЛІСТИНГ КОДУ ВЕБ-ДОДАТКУ

Header.jsx

```
import React from "react";
import {AppBar, Toolbar} from "@mui/material";
import {makeStyles} from '@material-ui/styles';
import Button from "@mui/material/Button";
import Typography from "@mui/material/Typography";
import {AccountCircle} from "@mui/icons-material";
import IconButton from "@mui/material/IconButton";
import Menu from "@mui/material/Menu";
import MenuItem from "@mui/material/MenuItem";


const useStyles = makeStyles(() => ({
    header: {
        paddingRight: "79px",
        paddingLeft: "118px",
        "@media (max-width: 900px)": {
            paddingLeft: 0,
        },
        backgroundColor: "#353953" + " !important"

    },
    logo: {
        fontFamily: "Work Sans, sans-serif",
        fontWeight: 500,
        color: "#fff",
        textAlign: "left",
    },
    menuButton: {
        fontFamily: "Work Sans, sans-serif",
        fontWeight: 500,
        size: "18px",
        marginLeft: "38px",
        padding: "3px 8px !important"
    },

    toolbar: {
        display: "flex",
        justifyContent: "space-between",
        margin: "6px 20px 6px 20px",
    },

}))

export const Header = () => {
    const { header, logo, menuButton, toolbar} = useStyles();
    const Logo = (<Typography variant="h6" component="h1"
                              className={logo}>Med-quiz</Typography>);
    const Exit = () => {
        handleClose();
        localStorage.clear();
        window.location.reload();

    }
```

```
    const [auth, setAuth] = React.useState(true);
    const [anchorEl, setAnchorEl] = React.useState(null);

    const handleChange = (event) => {
        setAuth(event.target.checked);
    };

    const handleMenu = (event) => {
        setAnchorEl(event.currentTarget);
    };

    const handleClose = () => {
        setAnchorEl(null);
    };
    const name = localStorage.getItem("name") === null ? " " :
localStorage.getItem("name");
    const displayDesktop = () => {
        return(
            <Toolbar variant="h6" component="h1" className={toolbar}>
                <div>{Logo}</div>
                <div>
                    <text style={{fontSize: 20}}>{name}</text>
                    <IconButton
                        size="large"
                        aria-label="account of current user"
                        aria-controls="menu-appbar"
                        aria-haspopup="true"
                        onClick={handleMenu}
                        color="inherit"
                    >
                        <AccountCircle />
                    </IconButton>
                    <Menu
                        id="menu-appbar"
                        anchorEl={anchorEl}
                        anchorOrigin={{
                            vertical: 'top',
                            horizontal: 'right',
                        }}
                        keepMounted
                        transformOrigin={{
                            vertical: 'top',
                            horizontal: 'right',
                        }}
                        open={Boolean(anchorEl)}
                        onClose={handleClose}
                    >
                        <MenuItem onClick={Exit}>Exit</MenuItem>

                    </Menu>
                </div>
                {/*<Button className={menuButton} style={{backgroundColor:
"#fff"}} onClick={Exit}>Exit</Button>*/}
            </Toolbar>
        )
    }

    return(
            <AppBar position="static" className={header}>
                {displayDesktop()}
            </AppBar>
    )
}
```

Combine.jsx

```
import React, {useState} from "react"
import Grid from "@mui/material/Grid";
import SignIn from "./SignIn";
import Box from "@mui/material/Box";
import Paper from "@mui/material/Paper";
import Typography from "@mui/material/Typography";
import Tabs from "@mui/material/Tabs";
import Tab from "@mui/material/Tab";
import SignUp from "./SignUp";

export const Combine = (props) => {
    const [value, setValue] = useState(0)



    const handleChange = (event, newValue) => {
        setValue(newValue);
    };

    function TabPanel(props) {
        const {children, value, index, ...other} = props;

        return (
            <div
                role="tabpanel"
                hidden={value !== index}
                id={`simple-tabpanel-${index}`}
                aria-labelledby={`simple-tab-${index}`}
                {...other}
            >
                {value === index && (
                    <Box>
                        <Typography>{children}</Typography>
                    </Box>
                )}
            </div>
        );
    }

    return (
        <Grid container component="main" sx={{height: '100vh'}}>
            <Grid
                item
                xs={false}
                sm={4}
                md={7}
                sx={{
                    backgroundImage: 'url(http://u118049.test-
handyhost.ru/images/login-image.svg)',
                    backgroundRepeat: 'no-repeat',
                    backgroundColor: (t) =>
                        t.palette.mode === 'black' ? t.palette.grey[50] :
t.palette.grey[900],
                    backgroundSize: 'cover',
                    backgroundPosition: 'center',
                }}
            />
            <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6}
square>

                <Box
                    sx={{
```

```
                        my: 8,
                        mx: 4,
                        display: 'flex',
                        flexDirection: 'column',
                        alignItems: 'center',
                    }}
                >
                    <Tabs
                        value={value}
                        indicatorColor="primary"
                        textColor="primary"
                        onChange={handleChange}
                        aria-label="disabled tabs example"
                    >
                        <Tab label="Увійти"/>

                        <Tab label="Реєстрація"/>
                    </Tabs>
                    <TabPanel value={value} index={0}>
                        <SignIn handleChange={handleChange} access={props.access}
rerender={props.rerender}/>
                    </TabPanel>
                    <TabPanel value={value}  index={1}>
                        <SignUp rerender={props.rerender}/>
                    </TabPanel>


                </Box>
            </Grid>
        </Grid>
    )
}
```

SignIn.jsx

```
import * as React from 'react';
import {useRef, useState} from 'react';
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';
import Box from '@mui/material/Box';
import Alert from "@mui/material/Alert";
import {updateTeacherPage} from "../../redux/reducers/TeacherPageReducer";
import {useDispatch, useSelector} from "react-redux";
import {updateTeacherResultTest} from
"../../redux/reducers/TeacherResultTestReducer";

const URL_LOGIN = "http://localhost/Medical-trainer/rest-full-api/sign-In-
Up/sign-In.php";
const URL_GET_TEACHER_PAGE_DATA = "http://localhost/Medical-trainer/rest-full-
api/teacher-page/teacher-page-first-get.php";
const URL_GET_TEACHER_RESULT_TEST_DATA = "http://localhost/Medical-trainer/rest-
full-api/teacher-page/teacher-page-result-test-get.php";

const responseTeacherData = () => {
    return fetch(URL_GET_TEACHER_PAGE_DATA, {
        method: "POST",
        header: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: JSON.stringify({action: 1})
    })
        .then((response) => {
            return response.json().then((data) => {
```

```
                return data;
            })
        })
}

const responseTeacherResultTestData = () => {
    return fetch(URL_GET_TEACHER_RESULT_TEST_DATA, {
        method: "POST",
        header: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: JSON.stringify({action: 1})
    })
        .then((response) => {
            return response.json().then((data) => {
                return data;
            })
        })
}

const sendData = async (url, data) => {
    const response = await fetch(url, {
        method: 'POST',
        body: JSON.stringify(data),
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }
    });
    //console.log( response)
    return await response.json();
}

export default function SignIn(props) {

    const dispatch = useDispatch();

    const studentData = useSelector(state => state.studentData);
    const [error, setError] = useState(null);

    const refUser = useRef(null);
    const refKey = useRef(null);

    const handleSignIn = async () => {
        const data = {
            "user": refUser.current.value,
            "key": refKey.current.value
        };
        /*console.log(data)*/
        const requestJson = await sendData(URL_LOGIN, data).then();


        responseTeacherData().then((data) => {
            dispatch(updateTeacherPage(data.teacherFirstData));
            localStorage.setItem("teacherFirstData",
JSON.stringify(data.teacherFirstData));


        })
        responseTeacherResultTestData().then((data) => {
            dispatch(updateTeacherResultTest(data.teacherResultTestData));
            localStorage.setItem("teacherResultTestData",
JSON.stringify(data.teacherResultTestData));
            window.location.reload();
```

```
        })

        props.access(requestJson);
        setError(requestJson.error);

    }

    return (
        <Box component="form" sx={{mt: 1}}>
            <TextField
                margin="normal"
                required
                fullWidth
                style={{maxWidth: 400}}
                id="login"
                label="Логін"
                name="login"
                autoComplete="login"
                inputRef={refUser}
                autoFocus

            />
            <TextField
                margin="normal"
                required
                fullWidth
                style={{maxWidth: 400}}
                id="password"
                label="Пароль"
                name="password"
                autoComplete="password"
                type={"password"}
                inputRef={refKey}
                autoFocus
            />


            {error &&
            <Alert severity="error">{error}</Alert>}

            <Button
                /*type="submit"*/
                onClick={handleSignIn}
                fullWidth
                style={{maxWidth: 400}}
                variant="contained"
                sx={{mt: 3, mb: 2}}
            >
                Увійти
            </Button>
        </Box>

    );
}
```

SignUp.jsx

```
import * as React from 'react';
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import Box from "@mui/material/Box";
import {useRef, useState} from "react";
import Alert from "@mui/material/Alert";
```

```
const URL_LOGIN = "http://u118049.test-handyhost.ru/rest-full-api/sign-In-
Up/sign-Up.php";

const sendData = async (url, data) => {
    console.log(data)
    const response = await fetch(url, {
        method: 'POST',
        body: JSON.stringify(data),
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }
    });
    return await response.json();
}

export default function SignUp() {
    const [error, setError] = useState(null);
    const [success, setSuccess] = useState(null);

    const refLogin = useRef(null);
    const refName = useRef(null);
    const refSurname = useRef(null);
    const refGroup = useRef(null);
    const refKey= useRef(null);

    const handleSignUp = async () => {
        const data = {
            "login": refLogin.current.value,
            "name": refName.current.value,
            "surname": refSurname.current.value,
            "group": refGroup.current.value,
            "key": refKey.current.value,
        }

        const requestJson = await sendData(URL_LOGIN, data).then();

        refLogin.current.value = null;
        refName.current.value = null;
        refSurname.current.value = null;
        refGroup.current.value = null;
        refKey.current.value = null;
        setError(requestJson.error);
        setSuccess(requestJson.success)
    }

    return (
        <>
            <Box component="form" sx={{ mt: 1 }}>
                {success &&
                <Alert severity="success">{success}</Alert>}
                {error &&
                <Alert severity="error">{error}</Alert>}
                <TextField
                    margin="normal"
                    required
                    fullWidth
                    style={{maxWidth: 400}}
                    id="login"
                    label="Логін"
                    name="login"
                    autoComplete="login"
                    inputRef={refLogin}
```

```
            autoFocus
        />
        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="name"
            label="Ім'я"
            name="name"
            autoComplete="name"
            inputRef={refName}
            autoFocus
        />
        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="surname"
            label="Прізвище"
            name="surname"
            autoComplete="surname"
            inputRef={refSurname}
            autoFocus
        />
        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="group"
            label="Група"
            name="group"
            autoComplete="group"
            inputRef={refGroup}
            autoFocus
        />

        <TextField
            margin="normal"
            required
            fullWidth
            style={{maxWidth: 400}}
            id="password"
            label="Пароль"
            name="password"
            autoComplete="password"
            type={"password"}
            inputRef={refKey}
            autoFocus
        />
        <Button
            /*type="submit"*/
            onClick={handleSignUp}
            fullWidth
            style={{maxWidth: 400}}
            variant="contained"
            sx={{ mt: 3, mb: 2 }}
        >
            Зареєструватись
        </Button>
    </Box>
    </>
```

```
    );
}
```

Animation.jsx

```
import React from "react";
import Box from "@mui/material/Box";
import Divider from "@mui/material/Divider";
import Paper from "@mui/material/Paper";
import VideoPlayer from "react-video-js-player";



export const Animation = (props) => {

    const playAnimation = () => {

            return (
                <video key={props.animations[0].animation_nameFile} width="600"
height="400" controls="controls" loop autoPlay>
                    <source src={"http://u118049.test-handyhost.ru/images/" +
props.animations[0].animation_nameFile }/>
                </video>
            )
    }

    return (
        <div>
            <Paper id="paper-animation">
                {playAnimation()}
            </Paper>
        </div>)
}
```

ChooseButton.jsx

```
import React from "react";
import Divider from "@mui/material/Divider";
import Button from "@mui/material/Button";
import Box from "@mui/material/Box";
import ButtonGroup from "@mui/material/ButtonGroup";
import {useSelector} from "react-redux";
import moment from "moment";
import Paper from "@mui/material/Paper";

const URL_SAVE_TEST_RESULT = "http://u118049.test-handyhost.ru/rest-full-
api/test-data/save-test-result.php";

const saveTestResult =  (testResult) => {
    fetch(URL_SAVE_TEST_RESULT, {
        method: 'POST',
        body: JSON.stringify(testResult),
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }
    }).then(r => r);
}
```

```
export const ChooseButton = (props) => {
    const progressStudentData = useSelector(state => state.progressStudentTest);
    const testData = useSelector(state => state.testData);
    const itemButtonClick = (nextStageId) => {
        let nameStage = "";
        let nameQuestion = "";
        let idQuestion = "";
        let stageStatus = "";
        for (let i = 0; i < testData.data.stages.length; i++) {
            if (testData.data.stages[i].id === nextStageId) {
                nameStage = testData.data.stages[i].stage;
                stageStatus = testData.data.stages[i].state_stages;
                break;
            }
        }
        for (let i = 0; i < testData.data.questions.length; i++) {
            if (testData.data.questions[i].stage_uk_id === nextStageId) {
                nameQuestion = testData.data.questions[i].text_question;
                idQuestion = testData.data.questions[i].id;
                break;
            }
        }
        let progressCurrentStage = {
            id: progressStudentData.progress.length,
            current_stage_name: nameStage,
            current_stage_id: nextStageId,
            question_text: nameQuestion,
            question_id: idQuestion,
            answers: [],
            animations:[],
        }
        for(let i = 0; i < testData.data.animations.length; i++) {
            let animationsCurrentStage = {
                animation_id: "",
                animation_nameFile: "",
            }
            if(nextStageId === testData.data.animations[i].stage_uk_id) {
                animationsCurrentStage.animation_id =
testData.data.animations[i].id;
                animationsCurrentStage.animation_nameFile =
testData.data.animations[i].name_file;
                progressCurrentStage.animations.push(animationsCurrentStage);

            }

        }
        for (let i = 0; i < testData.data.answers.length; i++) {
            let answersCurrentStage = {
                answer_id: "",
                answer_text: "",
                answer_status: "",
                current_stage_id: "",
                next_stage_id: "",
                question_id: ""
            }
            if (testData.data.answers[i].stage uk id === nextStageId) {
                answersCurrentStage.answer_id = testData.data.answers[i].id;
                answersCurrentStage.answer_text =
testData.data.answers[i].text_answer;
                answersCurrentStage.answer_status =
testData.data.answers[i].status;
                answersCurrentStage.current_stage_id =
testData.data.answers[i].stage_uk_id;
                answersCurrentStage.next_stage_id =
```

```
testData.data.answers[i].next_stage_uk_id;
                answersCurrentStage.question_id =
testData.data.answers[i].question_uk_id;
                progressCurrentStage.answers.push(answersCurrentStage);
            }
        }
        progressStudentData.progress.push(progressCurrentStage);
        progressStudentData.status = stageStatus;
        localStorage.setItem('route-student', 'start-test');
        if (progressStudentData.status === "result") {
            progressStudentData.result.user_id = localStorage.getItem("user_id");
            progressStudentData.result.testResult =
progressStudentData.progress[progressStudentData.progress.length -
1].current_stage_name;
            progressStudentData.result.stage_id =
progressStudentData.progress[progressStudentData.progress.length -
1].current_stage_id;
            progressStudentData.result.datetime = moment().format("DD-MM-YYYY
hh:mm:ss");
            localStorage.setItem('route-student', 'test-result');
            console.log("ez");
            saveTestResult(progressStudentData.result);
        }
        props.rerender();
    }
    const itemButton = (id, text, nextStageId) => {
        return (<Button id={id} key={id} style={{backgroundColor: "#ebeeff"}}
onClick={itemButtonClick.bind(this, nextStageId)}>{text}</Button>)
    }
    const itemsButtonArray = props.progress.answers.map(button =>
itemButton(button.answer_id, button.answer_text, button.next_stage_id))

    return (
        <div>
            <Paper>
                <Box
                    sx={{
                        display: 'flex',
                        flexDirection: 'column',
                        padding: 3,
                        alignItems: 'center',
                        '& > *': {
                            m: 1,
                        },
                    }}
                >
                    <ButtonGroup  variant="outlined" aria-label="outlined primary
button group">
                        {itemsButtonArray}
                    </ButtonGroup>
                </Box>
            </Paper>
        </div>
    )
}
```

Header.jsx

```
import React from "react";
import Divider from "@mui/material/Divider";
import Paper from "@mui/material/Paper";

export const Header = () => {
```

```
    return (
        <div>
            <Paper variant="outlined" style={{margin: "10px 0 0 0"}}>
                <h4 >Оберіть правильний варіант</h4>
            </Paper>
        </div>
    )
}
```

Question.jsx

```
import React from "react";
import Divider from "@mui/material/Divider";
import Paper from "@mui/material/Paper";

export const Question = (props) => {
    return (
        <div>
            <Paper variant="outlined">
                <div style={{padding: 5}}>
                    <h4 style={{fontFamily: "inherit", fontWeight:
500}}>{props.questionText}</h4>
                </div>
            </Paper>
        </div>
    )
}
```

StageStack.jsx

```
import Paper from "@mui/material/Paper";
import Stack from "@mui/material/Stack";
import Divider from "@mui/material/Divider";
import React from "react";
import {makeStyles} from "@mui/styles";

const useStyles= makeStyles(() => ({
    stage: {
        padding: "6px 10px 6px 10px",
        margin: "8px 15px 8px 15px !important",

    },
    divider: {
        margin: "10px 0 15px 0"
    }
}))

export const StackStages = () => {
    const {stage, divider} = useStyles();
    return(
        <div className={divider}>
            <Divider variant="fullWidth"/>
            <Stack
                direction="row"
                divider={<Divider orientation="vertical" style={{margin: "0"}}
flexItem />}
                spacing={2}
```

```
            >
                <Paper className={stage}>Stage 1</Paper>
                <Paper className={stage}>Stage 2</Paper>
                <Paper className={stage}>Stage 3</Paper>
                <Paper className={stage}>Stage 4</Paper>
            </Stack>
            <Divider variant="fullWidth"/>
        </div>
    )
}
```

MainPageStudent.jsx

```
import React, {useState} from "react";
import {Container} from "@mui/material";
import Box from "@mui/material/Box";
import Button from "@mui/material/Button";
import Divider from "@mui/material/Divider";
import {useDispatch, useSelector} from "react-redux";
import {updateTestData} from "../../redux/reducers/TestDataReducer";
import Paper from "@mui/material/Paper";
import Alert from "@mui/material/Alert";

const URL_DATA_TEST_UK = "http://u118049.test-handyhost.ru/rest-full-api/test-
data/test-data-uk.php";
const URL_STATUS_ATTEMPT = "http://u118049.test-handyhost.ru/rest-full-api/test-
data/status-attempt.php";

const checkStatusAttempt = async (jsonId) => {
    const response = await fetch(URL_STATUS_ATTEMPT, {
        method: 'POST',
        body: JSON.stringify(jsonId),
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }
    });
    return await response.json();
}

export const MainPageStudent = (props) => {
    const dispatch = useDispatch();
    const [info, setInfo] = useState(null);
    const startTest = () => {
        const jsonId = {
            "user_id": localStorage.getItem("user_id")
        }
        let attempt_test = "false";
        checkStatusAttempt(jsonId).then(resp => {
            attempt_test = resp.attempt_test;
            if(attempt_test === "true"){
                localStorage.setItem('route-student', 'start-test');
                const responseDataTest = () => {
                    return fetch(URL_DATA_TEST_UK, {
                        method: "POST",
                        header: {
                            'Content-Type': 'application/x-www-form-urlencoded',
                        },
                        body: JSON.stringify({action: 1})
                    })
                        .then((response) => {
                            return response.json().then((data) => {
                                return data;
```

```
                            })
                        })
                }
                responseDataTest().then((data) => {
                    dispatch(updateTestData(data));
                    props.rerender();
                })
            }
            else if(attempt_test === "false") {
                setInfo(attempt_test);
            }
        });


    }

    return (
        <>
            <Container fixed>
                <Box sx={{height: '90vh'}}>
                    <Box sx={{height: '10vh'}} style={{minHeight: 80}}><h2
style={{
                        fontFamily: "system-ui",
                        color: "#242c5c"
                    }}>Вітаємо тебе любий друже !</h2></Box>
                    <Divider variant="middle"/>
                    <Paper style={{minHeight: 100}}>
                        <Box sx={{height: '10vh'}}><p>Пропонуємо тоб перевірити
свої
                            знання з дисципліни
                            "Дитячі інфекційні хвороби" та допомгти перемогти
хворобу за допомогою набутих тобою
                            знань.</p>
                        </Box>
                    </Paper>
                    <p/>
                    <Divider variant="middle"/>
                    <Paper style={{minHeight: 100}}>
                        <Box sx={{height: '10vh'}}><p>Використовуючи навички та
                            вміння, користуючись симптомами, обери вірні кроки
для подолання стадій
                            хвороби,
                            обираючи вірні засоби та методи лікування. Але
пам'ятай - хибні методи призведуть до
                            трагедії!
                            Хай
                            щастить!</p></Box>
                    </Paper>
                    <Button variant="contained" style={{margin: "15px 0 0 0 ",
backgroundColor: "#414c93"}}
                            onClick={startTest} disableElevation>
                        Розпочати
                    </Button><p/>

                    {info &&
                    <Alert severity="info">Закінчились спроби пройти
тест</Alert>}
                </Box>
            </Container>
        </>
    )
}
```

Test.jsx

```
import React from "react";
import {Container} from "@mui/material";
import {StackStages} from "./TestPageComponents/StagesStack";
import {ChooseButton} from "./TestPageComponents/ChooseButton";
import {Header} from "./TestPageComponents/Header";
import {Question} from "./TestPageComponents/Question";
import {Animation} from "./TestPageComponents/Animation";
import {useSelector} from "react-redux";


export const TestPage = (props) => {
    localStorage.setItem('route-student', 'greetings');
    const testData = useSelector(state => state.testData);
    const progressStudentData = useSelector(state => state.progressStudentTest);
    const setProgressData = () => {
        if(progressStudentData.progress.length === 1){
            for(let i = 0; i < progressStudentData.progress.length; i++){
                progressStudentData.progress[i].id = i + 1;
                progressStudentData.progress[i].current_stage_id =
testData.data.stages[i].id;
                progressStudentData.progress[i].current_stage_name =
testData.data.stages[i].stage;
                progressStudentData.progress[i].question id =
testData.data.questions[i].id;
                progressStudentData.progress[i].question_text =
testData.data.questions[i].text_question;
                for(let j = 0; j < testData.data.animations.length; j++) {
                    let animationsCurrentStage = {
                        animation_id: "",
                        animation_nameFile: "",
                    }
                    if(testData.data.stages[i].id ===
testData.data.animations[j].stage_uk_id) {
                        animationsCurrentStage.animation_id =
testData.data.animations[j].id;
                        animationsCurrentStage.animation_nameFile =
testData.data.animations[j].name_file;

progressStudentData.progress[i].animations.push(animationsCurrentStage);

                    }

                }
                for(let j = 0; j < testData.data.answers.length; j++){
                    let answersCurrentStage = {
                        answer_id: "",
                        answer_text: "",
                        answer_status: "",
                        current_stage_id: "",
                        next_stage_id: "",
                        question_id: ""
                    }
                    if(testData.data.stages[i].id ===
testData.data.answers[j].stage_uk_id){
                        answersCurrentStage.answer_id =
testData.data.answers[j].id;
                        answersCurrentStage.answer_text =
testData.data.answers[j].text_answer;
                        answersCurrentStage.answer_status =
testData.data.answers[j].status;
                        answersCurrentStage.current_stage_id =
testData.data.answers[j].stage_uk_id;
```

```
                        answersCurrentStage.next_stage_id =
testData.data.answers[j].next_stage_uk_id;
                        answersCurrentStage.question_id =
testData.data.answers[j].question_uk_id;

progressStudentData.progress[i].answers.push(answersCurrentStage);
                    } else break;

                }


            }
        }

        return progressStudentData.progress[progressStudentData.progress.length -
1];
    }
    const currentProgress = setProgressData();
    //console.log(currentProgress.animations);
    return (
        <Container fixed>
            {/*<StackStages />*/}
            <Header/>
            <Animation animations={currentProgress.animations}
rerender={props.rerender}/>
            <Question questionId={currentProgress.question_id}
questionText={currentProgress.question_text}/>
            <ChooseButton progress={currentProgress} rerender={props.rerender}
animations={currentProgress.animations}/>
        </Container>
    )
}
```

TestResultPage.jsx

```
import React from "react";
import {Button, Paper} from "@mui/material";
import {useSelector} from "react-redux";
import Box from "@mui/material/Box";

export const TestResultPage = () => {
    const progressStudent = useSelector(state => state.progressStudentTest);

    const toMainPage = () => {
        localStorage.setItem('route-student', 'greetings');
        window.location.reload();
    }
    return (
        <div style={{padding: "50px 0 0 0"}}>
            <Paper style={{minHeight: 100, padding: "10px 0 0 0"}}
variant="elevation">
                <Box><h3>Результат</h3></Box>
            </Paper>
            <Paper style={{minHeight: 100, padding: "10px 0 0 0"}}
variant="elevation">
                <Box>{progressStudent.result.testResult}</Box>
            </Paper>
            <Paper style={{minHeight: 100, padding: "20px 0 0 0"}}
variant="outlined"  >
                <Button onClick={toMainPage} variant="contained">До
початку</Button>
            </Paper>
        </div>
```

```
    )
}
```

GroupStatisticTable.jsx

```
import {React} from "react";
import {DataGrid, ukUA} from "@mui/x-data-grid";
import {useDispatch, useSelector} from "react-redux";
import {updateTeacherResultTest} from
"../../redux/reducers/TeacherResultTestReducer";


export const GroupStatisticTable = (props) => {
    const dispatch = useDispatch();
    const studentResultData = useSelector(state => state.studentResultData);
    const columns = [
        {field: 'id', headerName: 'ID', width: 90, headerAlign: 'center'},
        {field: 'login', headerName: 'Логін', width: 150, headerAlign: 'center'},
        {field: 'name', headerName: "Ім'я", width: 150, headerAlign: 'center'},
        {field: 'surname', headerName: "Прізвище", width: 150, headerAlign:
'center'},
        {field: 'group', headerName: "Група", width: 150, headerAlign: 'center'},
        {field: 'datetime', headerName: "Дата проходження тесту", width: 270,
headerAlign: 'center'},
        {field: 'resultTest', headerName: "Результат тесту", width: 300,
headerAlign: 'center'},
    ];

    if(Object.keys(studentResultData).length === 0) {

dispatch(updateTeacherResultTest(JSON.parse(localStorage.getItem("teacherResultTe
stData"))));

    }

    return (
        <div
            style={{
                height: "90vh",
            }}>
            <DataGrid
                localeText={ukUA.components.MuiDataGrid.defaultProps.localeText}
                rows={studentResultData.data}
                columns={columns}
                rowsPerPageOptions={[5]}
            />
        </div>
    )

}
```

MainPageTeacher.jsx

```
import React from "react";
import {TabsBar} from "./TabsBar";
import {Container} from "@mui/material";
import "./MainPageTeacher.css";

export const MainPageTeacher = (props) => {
```

```
    return(
        <TabsBar rerender={props.rerender}/>
    )
}
```

StudentTable.jsx

```
import {React} from "react";
import {DataGrid, ukUA} from "@mui/x-data-grid";
import {Button} from "@mui/material";
import {useDispatch, useSelector} from "react-redux";
import {updateTeacherPage} from "../../redux/reducers/TeacherPageReducer";

const URL_SET_STATUS_ACCOUNT = "http://u118049.test-handyhost.ru/rest-full-
api/teacher-page/teacher-page-set-status-account.php";
const URL_SET_ATTEMPT_TEST = "http://u118049.test-handyhost.ru/rest-full-
api/teacher-page/teacher-page-set-attempt-test.php";

export const StudentTable = (props) => {
    const dispatch = useDispatch();
    const studentData = useSelector(state => state.studentData);
    const columns = [
        {field: 'id', headerName: 'ID', width: 90, headerAlign: 'center'},
        {field: 'login', headerName: 'Логін', width: 150, headerAlign: 'center'},
        {field: 'name', headerName: "Ім'я", width: 150, headerAlign: 'center'},
        {field: 'surname', headerName: "Прізвище", width: 150, headerAlign:
'center'},
        {field: 'group', headerName: "Група", width: 150, headerAlign: 'center'},
        {
            field: 'attempt', headerName: "Спроба проходження тесту", width: 270,
headerAlign: 'center',
            renderCell: (params) => {
                if (params.formattedValue === "true")
                    return "Присутня";
                else
                    return <Button onClick={setAttemptTest.bind(this, params.id)}
style={{color: "#3f51b5"}}>Оновити</Button>;
            }
        },
        {
            field: 'statusAccount', headerName: "Статус акаунту", width: 150,
headerAlign: 'center',
            renderCell: (params) => {
                if (params.formattedValue === "true")
                    return "Активований";
                else
                    return <Button onClick={setStatusAccount.bind(this,
params.id)}
                                   style={{color:
"#3f51b5"}}>Активувати</Button>;
            }
        },

    ];
    const setStatusAccount = (id) => {
        const response = fetch(URL_SET_STATUS_ACCOUNT, {
            method: 'POST',
            body: JSON.stringify({user_id:  id}),
            headers: {
                'Content-Type': 'application/json',
                'Accept': 'application/json'
            }
```

```
        }).then((response) => {
            return response.json().then((data) => {
                return data;
            })
        });
        response.then((data)=>{
            localStorage.setItem("teacherFirstData",
JSON.stringify(data.teacherFirstData));

dispatch(updateTeacherPage(JSON.parse(localStorage.getItem("teacherFirstData"))))
;
            props.rerender();
        })
    }
    const setAttemptTest = (id) => {
        const response = fetch(URL_SET_ATTEMPT_TEST, {
            method: 'POST',
            body: JSON.stringify({user_id:  id}),
            headers: {
                'Content-Type': 'application/json',
                'Accept': 'application/json'
            }
        }).then((response) => {
            return response.json().then((data) => {
                return data;
            })
        });
        response.then((data)=>{
            localStorage.setItem("teacherFirstData",
JSON.stringify(data.teacherFirstData));

dispatch(updateTeacherPage(JSON.parse(localStorage.getItem("teacherFirstData"))))
;
            props.rerender();
        })
    }

    if(Object.keys(studentData).length === 0) {

dispatch(updateTeacherPage(JSON.parse(localStorage.getItem("teacherFirstData"))))
;
    }

    return (
        <div
            style={{
                height: "90vh",
            }}>
            <DataGrid
                localeText={ukUA.components.MuiDataGrid.defaultProps.localeText}
                rows={studentData.data}
                columns={columns}
                rowsPerPageOptions={[5]}
            />
        </div>
    )

}
```

TabsBar.jsx

```
import {React, useState} from "react";
import Box from "@mui/material/Box";
import Tabs from "@mui/material/Tabs";
import Tab from "@material-ui/core/Tab";
import Typography from "@mui/material/Typography";
import SignIn from "../Login/SignIn";
import SignUp from "../Login/SignUp";
import {GroupStatisticTable} from "./GroupStatisticTable";
import {StudentTable} from "./StudentTable";




export const TabsBar = (props) => {
    const [value, setValue] = useState(0)



    const handleChange = (event, newValue) => {
        setValue(newValue);
    };

    function TabPanel(props) {
        const {children, value, index, ...other} = props;

        return (
            <div
                role="tabpanel"
                hidden={value !== index}
                id={`simple-tabpanel-${index}`}
                aria-labelledby={`simple-tab-${index}`}
                {...other}
            >
                {value === index && (
                    <Box>
                        <Typography>{children}</Typography>
                    </Box>
                )}
            </div>
        );
    }
    return (
        <Box
            sx={{ flexGrow: 1, bgcolor: 'background.paper', display: 'flex',
minHeight: "80vh" }}
        >
            <Tabs
                value={value}
                indicatorColor="primary"
                textColor="primary"
                orientation={"vertical"}
                onChange={handleChange}
                aria-label="disabled tabs example"
                sx={{ borderRight: 1, borderColor: 'divider' }}
            >
                <Tab label="Студенти"/>

                <Tab label="Статистика по групам"/>
            </Tabs>
```

```
            <TabPanel value={value} index={0}>
                <StudentTable rerender={props.rerender}/>
            </TabPanel>
            <TabPanel value={value}  index={1}>
                <GroupStatisticTable />
            </TabPanel>
        </Box>
    )

}
```

Routing.jsx

```
import React from "react";
import {MainPageStudent} from "./MainPageStudent/MainPageStudent";
import {MainPageTeacher} from "./MainPageTeacher/MainPageTeacher";
import {Route, Routes} from "react-router";
import {TestPage} from "./MainPageStudent/TestPage";
import {MainPageAdministrator} from
"./MainPageAdministrator/MainPageAdministrator";
import {useSelector} from "react-redux";
import {TestResultPage} from "./MainPageStudent/TestResultPage";


export const Routing = (props) => {
    const role = localStorage.getItem('role');

    if (role === "Administrator")
        return <MainPageAdministrator/>
    else if (role === "Teacher")
        return (<MainPageTeacher rerender={props.rerender}/>)
    else if (role === "Student") {
        if (localStorage.getItem("route-student") === "greetings")
            return (<MainPageStudent rerender={props.rerender}/>)
        else if (localStorage.getItem("route-student") === "start-test")
            return (<TestPage rerender={props.rerender}/>);
        else if (localStorage.getItem("route-student") === "test-result")
            return (<TestResultPage/>)
    }
    return <></>

}
```

App.js

```
import './App.css';
import React, {useState} from "react";
import {Combine} from "./Components/Login/Combine";
import {Routing} from "./Components/Routing";
import {Header} from "./Components/Layout/Header";
import {updateTestData} from "./redux/reducers/TestDataReducer";



function App(props) {

    //const state = useSelector(state => state);

    const localStorageMethode = (state) => {
        const { user_id, login, name, surname, group, role, status_account} =
state;
```

```
        localStorage.setItem('user_id', user_id);
        localStorage.setItem('login', login);
        localStorage.setItem('name', name);
        localStorage.setItem('surname', surname);
        localStorage.setItem('group', group);
        localStorage.setItem('status_account', status_account);
        localStorage.setItem('role', role);
        localStorage.setItem('connect', 'true');
        localStorage.setItem('route-student', 'greetings');
    }



    const [connect, setConnect] = useState(false);
    const access = (state) => {
        if(state.connect === true)
        localStorageMethode(state);
        setConnect(state.connect);
        localStorage.setItem('teacherDataFirstGet', "true");
    }


    return (
        <div className="App">
            {localStorage.getItem("connect") === "true" ? <><Header/><Routing
rerender={props.rerender}/></> :
                <Combine access={access} rerender={props.rerender}/>}
        </div>
    );
}

export default App;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import {Provider} from "react-redux";
import {store} from "./redux/reducers";


const rerenderTree = () => {
    ReactDOM.render(
        <Provider store={store}>
                <App rerender={rerenderTree}/>
        </Provider>,
        document.getElementById('root')
    );
}

rerenderTree();
```

Index.js

```
import {applyMiddleware, combineReducers, createStore} from "redux";
import {testDataReducer} from "./TestDataReducer";
import {composeWithDevTools} from "redux-devtools-extension";
import {progressStudentReducer} from "./ProgressStudentReducer";
import {teacherPageReducer} from "./TeacherPageReducer";
```

```
import {teacherResultTestReducer} from "./TeacherResultTestReducer";



const rootReducer = combineReducers({
    testData: testDataReducer,
    progressStudentTest: progressStudentReducer,
    studentData: teacherPageReducer,
    studentResultData: teacherResultTestReducer,
})

export const store = createStore(rootReducer,
composeWithDevTools(applyMiddleware()))
```

ProgressStudentReducer.js

```
const UPDATE_PROGRESS_STUDENT = "UPDATE_PROGRESS_STUDENT";

const defaultState = {
    progress: [
        {
            id: "",
            current_stage_name: "",
            current_stage_id: "",
            question_text: "",
            question_id: "",
            animations:[],
            answers: []
        }
    ],
    status: "",
    result: {
        user id: "",
        testResult: "",
        stage_id: "",
        datetime: ""
    }
}

export const progressStudentReducer = (state = defaultState, action) => {
    switch (action.type) {
        case UPDATE_PROGRESS_STUDENT:
            return {
                ...state,
                data: action.payload
            };
        default:
            return state;
    }
}

export const updateProgressStudent = (data) => ({type: UPDATE_PROGRESS_STUDENT,
payload: data})
```

TeacherPageReducer.js

```
const UPDATE_STUDENT_DATA = "UPDATE_STUDENT_DATA";

const defaultState = {

}
```

```
export const teacherPageReducer = (state = defaultState, action) => {
    switch (action.type) {
        case UPDATE_STUDENT_DATA:
            return {
                ...state,
                data: action.payload
            };
        default:
            return state;
    }
}


export const updateTeacherPage = (data) => ({type: UPDATE_STUDENT_DATA, payload:
data})
```

TeacherResultTestReducer.js

```
const UPDATE_STUDENT_TEST_RESULT_DATA= "UPDATE_STUDENT_TEST_RESULT_DATA";

const defaultState = {

}


export const teacherResultTestReducer = (state = defaultState, action) => {
    switch (action.type) {
        case UPDATE_STUDENT_TEST_RESULT_DATA:
            return {
                ...state,
                data: action.payload
            };
        default:
            return state;
    }
}


export const updateTeacherResultTest = (data) => ({type:
UPDATE_STUDENT_TEST_RESULT_DATA, payload: data})
```

sign-In.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
include "../connectToDB.php";

$mysqli = connectToDB();
$JSONData = file_get_contents("php://input");
$dataObject = json_decode($JSONData);
$mysqli->set_charset('utf8');

$user = $dataObject->user;
$password = $dataObject->key;
```

```
if ($result = $mysqli->prepare("SELECT users.login as login, users.password as
password, users.name as name,
                                                users.surname as surname,
users.group_name as group_name, users.status_account as status_account, users.id
as user_id,
                                                roles.role as role,
roles.description as role_description, roles.id as role_id
                                            FROM users
                                            INNER JOIN roles ON users.id_role =
roles.id WHERE login = ?")) {
    $result->bind_param('s', $user);
    $result->execute();
    $get_result = $result->get_result();
    if ($get_result->num_rows == 1) {
        $data = $get_result->fetch_assoc();
        $data_key = $data['password'];
        if (password_verify($password, $data_key)) {
            if($data['status_account'] == "false"){
                echo json_encode(array('connect' => false, 'error' => 'Account
not activate!'));
            } else {
                echo json_encode(array('connect' => true, 'login' =>
$data['login'], 'name' => $data['name'],
                    'surname' => $data['surname'], 'group' =>
$data['group_name'], 'status_account' => $data['status_account'],
                    'user_id' => $data['user_id'], 'id_role' => $data['role_id'],
'role' => $data['role']));
            }

        } else {

            echo json_encode(array('connect' => false, 'error' => 'Wrong password
or login!'));
        }
    } else {
        echo json_encode(array('connect' => false, 'error' => 'Wrong password or
login!'));
    }
    $result->close();
} else {
    echo json_encode(array('connect' => false, 'error' => 'Error connect!'));
}
$mysqli->close();
```

sign-Up.php

```
<?php
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
include "../connectToDB.php";

$mysqli = connectToDB();
$JSONData = file_get_contents("php://input");
$dataObject = json_decode($JSONData);
$mysqli->set_charset('utf8');

$login = $dataObject->login;
$name = $dataObject->name;
$surname = $dataObject->surname;
$group = $dataObject->group;
$password = password_hash($dataObject->key, PASSWORD_DEFAULT);
```

```
if($login == null || $name == null || $surname == null || $group == null ||
$password == null)
    echo json_encode(array('error' => 'Incorrect data entered!'));
else {
    echo json_encode(array('success' => 'Success Sign-Up, Congrats!'));
    $query = "INSERT INTO users (login, name, surname, group_name, password,
id_role, attempt_test, status_account)
              VALUES ('$login', '$name', '$surname', '$group', '$password', 3,
'true', 'false')";
    $mysqli->query($query);
}
```

Teacher-page-1.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
include "../connectToDB.php";

$mysqli = connectToDB();

$mysqli->set_charset('utf8');


$data = array();


$result = $mysqli->query("SELECT * FROM users WHERE id_role = 3");
while($row = $result->fetch_assoc()){
    array_push($data, ['id' => $row['id'], 'login' => $row['login'], 'name' =>
$row['name'],
                                'surname' => $row['surname'], 'group' =>
$row['group_name'],
                                'attempt' => $row['attempt_test'],
'statusAccount' => $row['status_account']]);
}

$json = [
    "teacherFirstData" => $data,
];
echo json_encode($json);

$mysqli->close();
```

Teacher-page-2.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
```

```
include "../connectToDB.php";

$mysqli = connectToDB();

$mysqli->set_charset('utf8');


$data = array();


$result = $mysqli->query("SELECT h.id as id, u.login as login, u.name as name,
u.surname as surname,
                                        u.group_name, h.datetime as date,
h.test_result as test_result
                                        FROM users u JOIN history_results_test h
ON u.id = h.user_id WHERE u.id_role = 3");
while($row = $result->fetch_assoc()){
    array_push($data, ['id' => $row['id'], 'login' => $row['login'], 'name' =>
$row['name'],
        'surname' => $row['surname'], 'group' => $row['group_name'],
        'datetime' => $row['date'], 'resultTest' => $row['test_result']]);
}

$json = [
    "teacherResultTestData" => $data,
];
echo json_encode($json);

$mysqli->close();
```

Teacher-page-3.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
include "../connectToDB.php";

$mysqli = connectToDB();
$JSONData = file_get_contents("php://input");
$dataObject = json_decode($JSONData);
$mysqli->set_charset('utf8');

$user_id = $dataObject->user_id;
$data = array();

if($user_id != null){
    $query = "UPDATE `users` SET `attempt_test` = 'true' WHERE `users`.`id` =
'$user_id'";
    $mysqli->query($query);

    $result = $mysqli->query("SELECT * FROM users WHERE id_role = 3");
    while($row = $result->fetch_assoc()){
        array_push($data, ['id' => $row['id'], 'login' => $row['login'], 'name'
=> $row['name'],
            'surname' => $row['surname'], 'group' => $row['group_name'],
            'attempt' => $row['attempt_test'], 'statusAccount' =>
$row['status_account']]);
    }

    $json = [
```

```
        "teacherFirstData" => $data,
    ];
    echo json_encode($json);
}

$mysqli->close();
```

Teacher-page-4.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];
include "../connectToDB.php";

$mysqli = connectToDB();
$JSONData = file_get_contents("php://input");
$dataObject = json_decode($JSONData);
$mysqli->set_charset('utf8');

$user_id = $dataObject->user_id;
$data = array();

if($user_id != null){
    $query = "UPDATE `users` SET `status_account` = 'true' WHERE `users`.`id` =
'$user_id'";
    $mysqli->query($query);

    $result = $mysqli->query("SELECT * FROM users WHERE id_role = 3");
    while($row = $result->fetch_assoc()){
        array_push($data, ['id' => $row['id'], 'login' => $row['login'], 'name'
=> $row['name'],
            'surname' => $row['surname'], 'group' => $row['group_name'],
            'attempt' => $row['attempt_test'], 'statusAccount' =>
$row['status_account']]);
    }

    $json = [
        "teacherFirstData" => $data,
    ];
    echo json_encode($json);
}


$mysqli->close();
```

Connect.php

```
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-With,
Content-Type, Accept, Access-Control-Request-Method");
header("Content-Type: text/html; charset=utf-8");
$method = $_SERVER['REQUEST_METHOD'];

function connectToDB(){
```

```
  $host = "localhost";
  $user = "u118049_root";
  $password = "vCHpa6FrcSvxUR5";
  $bd = "u118049_medical-trainer";


    $connection = mysqli_connect($host, $user, $password,$bd);

        if($connection){
            echo "";
        }else{
            echo 'Error connect!';
        }

    return $connection;
}
```