МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
_____червня 2023 р._____

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: **«Веборієнтована інформаційна система організації шкільного освітнього процесу»**

здобувачки групи Кндн – 91км Козловської Ольги Василівни

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

_____ Ольга КОЗЛОВСЬКА
(підпис)

Керівник, доцент,
кандидат фізико-математичних наук Сергій ШАПОВАЛОВ _____
(підпис)

Суми – 2023

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»

здобувача групи Кндн – 91км Козловської Ольги Василівни

1. Тема роботи: «Веборієнтована інформаційна система організації шкільного освітнього процесу»

затверджую наказом по СумДУ від *«02» червня 2023 р. № 0620-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*

3. Вхідні дані до кваліфікаційної роботи *мова програмування Python, фреймворк Django*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області. 2) Аналіз можливих рішень. 3) Проєктування та розробка веборієнтованої інформаційної системи організації шкільного освітнього процесу. 4) Аналіз результатів проектування та розробки. 5) Тестування веборієнтованої інформаційної системи організації шкільного освітнього процесу.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Діаграма варіантів використання. Діаграма послідовності. Діаграма діяльності. Діаграма класів.*

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для вирішення завдань дослідження*		
3	*Розробка інформаційного та програмного забезпечення*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 71 стор., 32 рис., 2 табл., 1 додаток, 37 джерел.

Обґрунтування актуальності теми роботи – Процес навчання в сучасному світі став більш складним і вимагає від вчителів та учнів більш ефективного підходу до навчання та організації шкільного процесу взагалі. З метою полегшення цього процесу, веборієнтована інформаційна система може стати ефективним інструментом. Вона може забезпечити учням та вчителям легкий доступ до інформації, необхідної для навчання та проведення уроків. Крім того, така система може допомогти вчителям в організації уроків, розміщенні розкладів та контролю за успішністю кожного учня.

Об'єкт дослідження — веборієнтована інформаційна система організації шкільного освітнього процесу.

Мета роботи — створення веборієнтованої інформаційної системи, яка спростить організацію навчального процесу в школі.

Методи дослідження:

1. Аналіз літературних джерел і публікацій з питань організації шкільного освітнього процесу, вебдизайну, програмування та розробки вебсайтів.

2. Аналіз ринку існуючих вебсайтів для організації шкільного освітнього процесу, їх функціоналу, дизайну та технічних характеристик.

3. Вивчення вимог користувачів до інтерфейсу користувача, функціоналу та зручності використання системи.

4. Проведення тестування та валідації вебсайту на різних пристроях та в різних браузерах, для переконання в його коректності та зручності використання.

Результати — розроблено алгоритм та програмне забезпечення веборієнтованої інформаційної системи організації шкільного освітнього процесу. Розроблений алгоритм реалізовано у формі вебсайту, створеного за допомогою мови програмування Python з використанням фреймворку Django.

ВЕБОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА, ШКІЛЬНИЙ ОСВІТНІЙ ПРОЦЕС, ВЕБСАЙТ, PYTHON, DJANGO, HTML, CSS, BOOTSTRAP.

ЗМІСТ

ВСТУП5

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6

1.1 Актуальність проблеми 6

1.2 Аналіз існуючих аналогів 7

1.3 Постановка задачі12

2 АНАЛІЗ МОЖЛИВИХ РІШЕНЬ13

2.1 Вибір мови програмування 13

2.2 Вибір системи керування базами даних19

2.3 Вибір фреймворку 26

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ШКІЛЬНОГО ОСВІТНЬОГО ПРОЦЕСУ32

3.1 Методи та засоби об'єктно-орієнтованого аналізу і проектування 32

3.1.1 Діаграма варіантів використання 32

3.1.2 Діаграма послідовності (Sequencediagram) 34

3.1.3 Діаграма діяльності (Activitydiagram)35

3.1.4 Діаграма класів37

3.2 Дизайн вебсайту системи38

3.3 Архітектура вебсайту 43

3.4 Верстка вебсайту. Frontend-частина 46

3.5 Програмування серверної частини вебсайту. Backend-частина 53

3.6 Тестування вебсайту65

ВИСНОВКИ 70

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72

ДОДАТКИ 75

Додаток А. Лістинг models.py, views.py, urls.py 75

ВСТУП

У сучасному світі інформаційні технології є невід'ємною частиною нашого життя. Це стосується не тільки приватного сектору, але й сфери освіти. З кожним роком, використання інформаційних технологій у шкільному навчанні набуває все більшої популярності і актуальності.

Актуальність. Процес навчання в сучасному світі став більш складним і вимагає від вчителів та учнів більш ефективного підходу до навчання та організації шкільного процесу взагалі. З метою полегшення цього процесу, веборієнтована інформаційна система може стати ефективним інструментом. Вона може забезпечити учням та вчителям легкий доступ до інформації, необхідної для навчання та проведення уроків. Крім того, така система може допомогти вчителям в організації уроків, розміщенні розкладів та контролю за успішністю кожного учня.

Метою даної дипломної роботи є створення веборієнтованої інформаційної системи, яка спростить організацію навчального процесу в школі.

Цей проект є важливим, оскільки він відповідає сучасним вимогам до шкільної освіти та допоможе вчителям бути більш ефективними у своїй роботі.

Об'єктом дослідження є веборієнтована інформаційна система організації шкільного освітнього процесу.

Предметом дослідження є процес проектування та розробки вебсайту з метою покращення організації шкільного освітнього процесу.

Вебсайт буде розроблений на мові програмування Python з використанням фреймворку Django, що дозволить ефективно створити та управляти вебсайтом. Також буде створено дизайн та використано HTML, CSS та Bootstrap для створення зручного та привабливого інтерфейсу для користувачів.

Вебсайт буде зроблений адаптивним. Адаптивний дизайн вебсайту означає, що він автоматично адаптується до різних розмірів екранів пристроїв, на яких він відображається, забезпечуючи кращу читабельність та користувацький досвід.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Освітній процес в школі є однією з ключових галузей, що має велике значення для суспільства в цілому. У сучасних умовах, зокрема в епоху інформаційних технологій, виникає все більша потреба в автоматизації процесів, пов'язаних з організацією шкільної освіти.

Веборієнтовані інформаційні системи, що забезпечують доступ до інформації через Інтернет, вже зайняли важливу нішу в галузі освіти. Ці системи надають зручний і простий інтерфейс для взаємодії з користувачами, забезпечують швидкий доступ до необхідної інформації та збільшують продуктивність роботи педагогів і учнів.

Основна функціональність системи може включати в себе створення розкладу уроків, відображення інформації про вчителів та учнів та ведення обліку відвідування та оцінювання.

1.1 Актуальність проблеми.

Актуальність проблеми створення веборієнтованої інформаційної системи для організації шкільного освітнього процесу полягає в тому, що в сучасному світі зростає потреба в автоматизації процесів у всіх сферах життя, включаючи освіту. Шкільний освітній процес вимагає багато ресурсів та часу, а тому може бути покращений за допомогою використання інформаційних технологій.

Організація шкільного освітнього процесу вимагає багато зусиль від учнів, вчителів та адміністрації. Потрібна система, яка допоможе зібрати та аналізувати дані, спростити процеси планування та контролювання знань учнів, підвищити ефективність роботи вчителів та директорів шкіл.

Також, спочатку в умовах пандемії COVID-19, а зараз під час військового стану, коли вчителі та учні вчаться дистанційно, створення веборієнтованої інформаційної системи для організації шкільного освітнього процесу стає особливо актуальним. Така система може забезпечити віртуальне навчання та спілкування між учнями та вчителями, а також забезпечити збереження та обробку даних.

Отже, створення веборієнтованої інформаційної системи для організації шкільного освітнього процесу є актуальною проблемою, яка може покращити ефективність роботи шкіл та забезпечити якісну освіту для учнів.

1.2 Аналіз існуючих аналогів

Для успішної розробки веборієнтованої інформаційної системи організації шкільного освітнього процесу важливо проаналізувати існуючі аналоги. Це дозволить визначити їх переваги та недоліки, а також виявити можливі прогалини або функціональні можливості, які можна вдосконалити або включити в нову систему.

Одним з популярних аналогів є "Classroom" від Google.

"Classroom" від Google [1] є інструментом, спеціально розробленим для організації учбових процесів в цифровому форматі. Він надає вчителям і учням зручні можливості для ефективного взаємодії та спільної роботи над навчальними матеріалами (рис. 1.1).

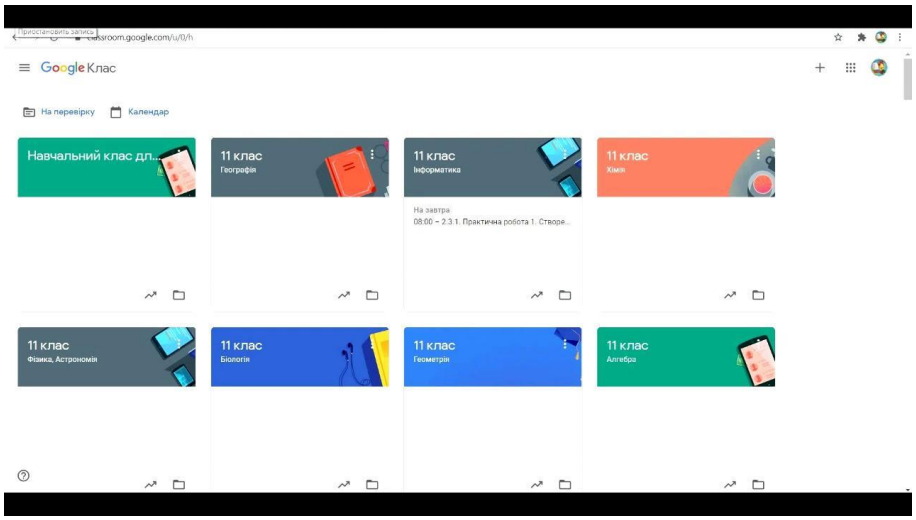


Рисунок 1.1 — Класи в Google Classroom

Основні функції "Classroom" включають:

1. ***Створення та спільне використання матеріалів***. Вчителі можуть завантажувати навчальні матеріали, такі як презентації, документи, відео або посилання, і надавати доступ до них учням. Учні можуть зручно переглядати та завантажувати ці матеріали для вивчення.

2. ***Ведення оцінок***. Вчителі можуть створювати завдання та оцінювати відповіді учнів. "Classroom" автоматично оброблює результати та зберігає оцінки, що дозволяє вчителям легко відстежувати успішність учнів та надавати зворотний зв'язок.

3. ***Надання завдань***. Вчителі можуть створювати завдання та встановлювати терміни їх виконання. Учні можуть зручно відправляти відповіді на завдання безпосередньо через "Classroom".

4. ***Зворотний зв'язок***. Учні можуть задавати питання та обговорювати матеріали чи завдання з вчителями та однокласниками через коментарі в "Classroom". Це сприяє активній комунікації та співпраці між учасниками навчального процесу.

5. ***Інтеграція з іншими сервісами Google***. "Classroom" має підтримку інтеграції з Google Диск, що дозволяє вчителям та учням легко додавати та зберігати файли. Крім того, він також інтегрується з Google Календарем.

Інший аналог – система "Єдина школа" є комплексною інформаційною системою, спеціально розробленою для управління закладами освіти [2]. Вона надає широкий спектр функціональності та інструментів для ефективної організації різних аспектів шкільного освітнього процесу (рис. 1.2).

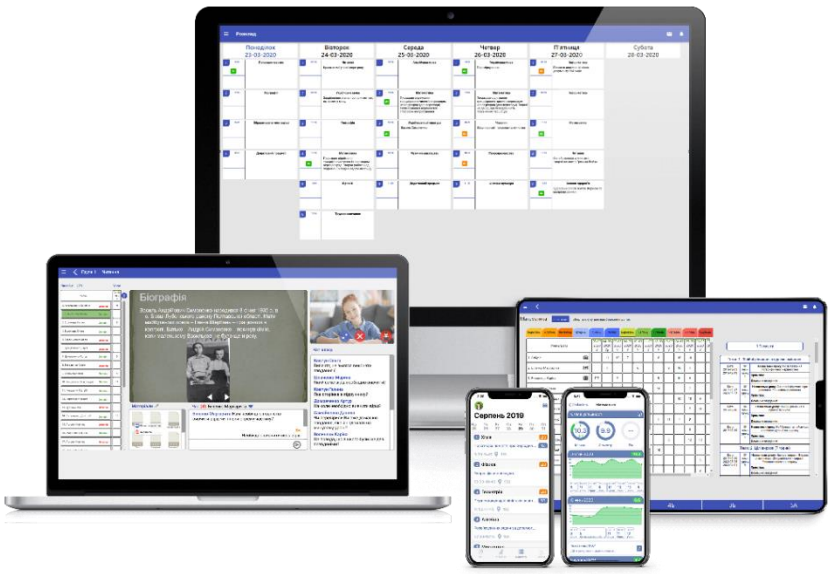


Рисунок 1.2 — Система "Єдина школа"

Основні можливості системи "Єдина школа" включають:

1. ***Автоматизоване ведення обліку***. Система дозволяє зручно зберігати та обробляти дані про учнів, вчителів, класи, розклади уроків, журнали відвідуваності, оцінки та інші важливі дані. Це спрощує процеси адміністрування та документообігу в школі.

2. ***Управління навчальним процесом***. Система надає засоби для створення та планування розкладу уроків, встановлення завдань та оцінювання. Вчителі можуть зручно вносити результати оцінювання, а учні можуть відстежувати свої успіхи та отримувати зворотний зв'язок.

3. ***Електронний журнал***. Система дозволяє вести електронний журнал, в якому можна відображати інформацію про відвідуваність учнів, оцінки, відсутності та інші аспекти стосовно академічної діяльності.

4. ***Комунікація та співпраця***. "Єдина школа" надає можливості для зручної комунікації між вчителями, учнями та батьками. Через систему можна відправляти повідомлення, надавати доступ до матеріалів, проводити збори, планувати події та співпрацювати над проектами.

Інший аналог – Moodle – система для управління навчанням (LMS - Learning Management System) [3], яка надає широкий спектр функціональності для організації інтерактивного та ефективного навчального процесу (рис. 1.3).

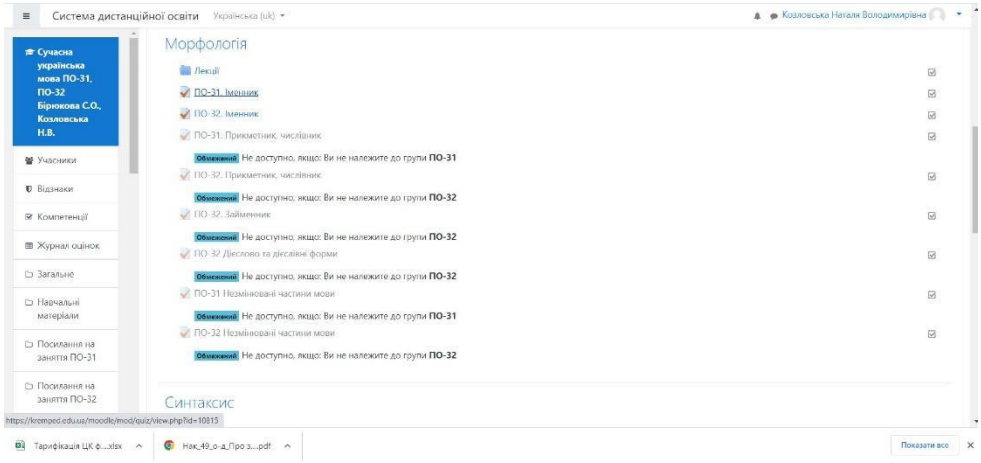


Рисунок 1.3 — Система для управління навчанням Moodle

Основні можливості Moodle включають:

1. ***Управління курсами***. Moodle дозволяє вчителям створювати та керувати курсами, включаючи завдання, матеріали, тести, форуми для обговорень,

відеоуроки та інші навчальні ресурси. Вчителі можуть створювати структуровані програми навчання, встановлювати терміни здачі завдань та вести оцінювання.

2. ***Взаємодія та спілкування***. Moodle надає інструменти для спілкування між вчителями та студентами. Вони можуть обмінюватися повідомленнями, обговорювати матеріали в форумах, спільно працювати над проектами та виконувати групові завдання.

3. ***Оцінювання та звітність***. Moodle має можливості для ведення оцінок та статистики успішності студентів. Вчителі можуть створювати тести, оцінювати роботи, встановлювати критерії оцінювання та надавати зворотний зв'язок студентам. Крім того, Moodle забезпечує генерацію звітів та аналітику для оцінки академічного прогресу та статистики навчання.

4. ***Доступність та гнучкість.*** Moodle є відкритою системою з відкритим вихідним кодом, що дозволяє вчителям та адміністраторам налаштовувати систему під свої потреби. Вона підтримує різноманітні формати навчальних матеріалів, інтегрується з іншими інструментами, наприклад відеоконференціями.

Система дистанційної освіти Сумського державного університету (СумДУ) є аналогом [4], спеціально розробленим для організації навчального процесу на віддаленій основі в рамках СумДУ (рис. 1.4).

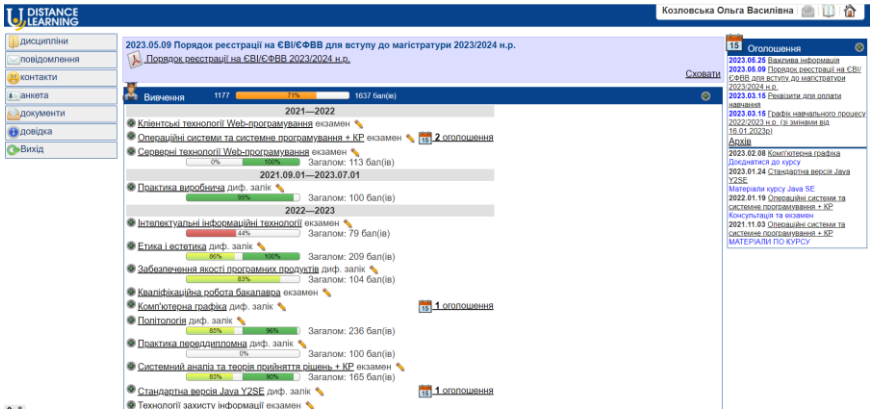


Рисунок 1.4 — Система дистанційної освіти СумДУ

Основні можливості системи дистанційної освіти СумДУ включають:

1. ***Електронні курси***. Система дозволяє створювати та вести електронні курси, які містять навчальні матеріали, завдання, тести та інші ресурси. Викладачі

можуть створювати структуровані програми навчання та визначати послідовність вивчення матеріалу.

2. ***Відеолекції та вебінари***. Система дозволяє проводити відеолекції та вебінари в режимі реального часу. Це дозволяє викладачам спілкуватися зі студентами, проводити онлайн-лекції, демонструвати матеріали та відповідати на питання.
3. ***Завдання та оцінювання***. Система надає можливість створювати завдання та оцінювати роботи студентів. Викладачі можуть встановлювати терміни здачі завдань, оцінювати їх та надавати зворотний зв'язок студентам.
4. ***Спілкування***. Система має функцію спілкування між студентами та викладачами. Це стимулює активну комунікацію та обмін ідеями.
5. ***Моніторинг та звітність***. Система дозволяє викладачам та адміністраторам моніторити академічний прогрес студентів, а також генерувати звіти та аналітику стосовно навчання.
6. ***Доступність та гнучкість***. Система дистанційної освіти СумДУ підтримує різні формати навчання.

Порівняльна таблиця, яка відображає основні можливості "Classroom" від Google, системи "Єдина школа", Moodle та системи дистанційної освіти Сумського державного університету (Таблиця 1.1)

Таблиця 1.1 Порівняльна таблиця аналогів

Особливість	**Classroom від Google**	**Єдина школа**	**Moodle**	**СумДУ**
Управління курсами	Так	Так	Так	Так
Управління завданнями	Так	Так	Так	Так
Обмін матеріалами	Так	Так	Так	Так
Система оцінювання	Так	Так	Так	Так
Засоби комунікації	Так	Так	Так	Так
Інтеграція з сервісами Google	Так	Ні	Ні	Ні
Відеоконференції	Інтеграція з Google Meet	Ні	Ні	Ні
Відкритий вихідний код	Ні	Ні	Так	Ні
Налаштування	Обмежено	Обмежено	Розширені	Обмежено

1.3 Постановка задачі

Мета дипломної роботи – створення веборієнтованої інформаційної системи, яка спростить організацію навчального процесу в школі.

Завдання дослідження:

- Проаналізувати існуючі вебсистеми, які використовуються для організації освітнього процесу.
- Визначити функціональні вимоги до системи
- Визначити мови програмування, фреймворк та інших технологій, які будуть використовуватись для розробки системи.
- Розробити структуру бази даних, яка буде використовуватись для зберігання інформації про курси, учнів, вчителів, оцінки та інші дані, необхідні для функціонування системи.
- Створити сучасний, адаптивний вебсайт.
- Реалізувати можливості створення, редагування та видалення класів та розкладів занять.
- Розробити функціонал для додавання та видалення учнів до класів, призначення ролей (студент, викладач, адміністратор) та відображення списків студентів в кожному курсі.
- Реалізувати систему оцінювання, де викладачі зможуть виставляти оцінки за завдання, а учні зможуть бачити свої оцінки.
- Забезпечити безпеку даних.

2 АНАЛІЗ МОЖЛИВИХ РІШЕНЬ

2.1 Вибір мови програмування

При виборі мови програмування для розробки веборієнтованої інформаційної системи для організації шкільного навчального процесу можна враховувати декілька факторів:

Потрібно обирати мову, яка добре підходить для веброзробки і має необхідні функції та бібліотеки для підтримки бажаної функціональності. Найпоширенішими мовами для веброзробки є Python, JavaScript, Ruby, PHP та Java.

Потрібно оцінити спільноту та підтримку, доступну для обраної мови програмування. Велика спільнота може надати корисні ресурси, бібліотеки та фреймворки, а також активні форуми для пошуку та усунення несправностей.

Також потрібно оцінити вимоги до продуктивності та масштабованості системи. Деякі мови, такі як Java або C++, можуть запропонувати кращу продуктивність для певних завдань, тоді як інші, такі як Python або Ruby, забезпечують швидку розробку та гнучкість.

Також необхідно звернути увагу на функції безпеки та найкращі практики, що підтримуються мовою програмування. Переконатися, що мова має надійні механізми безпеки для захисту від поширених вебвразливостей.

Python - високорівнева інтерпретована мова програмування, відома своєю простотою та читабельністю. Її створив Гвідо ван Россум і вперше випустив у 1991 році. Python робить акцент на читабельності коду і використовує значну кількість пробілів, що робить її легкою для розуміння і написання [5].

Python здобула популярність завдяки своїй універсальності і широко використовується в різних сферах, таких як веброзробка, аналіз даних, наукові обчислення, штучний інтелект, машинне навчання тощо. Він має велику стандартну бібліотеку та широку екосистему сторонніх бібліотек і фреймворків, які розширюють його можливості.

Однією з ключових переваг Python є простота використання, що робить її чудовим вибором для людей, які тільки починають програмувати. Вона підтримує декілька парадигм програмування, включаючи процедурний, об'єктно-орієнтований та функціональний стилі програмування.

Код на Python виконується за допомогою інтерпретатора, що дозволяє проводити інтерактивні експерименти та розробку. Інтерпретатор Python також може запускати скрипти, збережені у файлах .py, що робить його придатним як для невеликих скриптів, так і для великомасштабних додатків [6].

JavaScript - високорівнева інтерпретована мова програмування, яка використовується переважно для створення динамічного вебконтенту та сценаріїв на стороні клієнта. Вона була розроблена Бренданом Айхом у компанії Netscape Communications і вперше випущена в 1995 році [7].

JavaScript широко використовується для інтерфейсної веброзробки, де його застосовують для покращення інтерактивності та функціональності вебсайтів. Вона дозволяє розробникам маніпулювати елементами вебсторінок, обробляти події та створювати динамічний контент "на льоту". За допомогою JavaScript можна виконувати такі завдання, як перевірка форм, маніпуляції з DOM, анімація та асинхронний зв'язок з серверами за допомогою AJAX.

Окрім фронтенд-розробки, JavaScript поширився на інші сфери, включаючи серверну розробку (Node.js), розробку десктопних додатків (Electron) та розробку мобільних додатків (React Native, NativeScript).

JavaScript - це універсальна мова, яка підтримує як об'єктно-орієнтовану, так і функціональну парадигми програмування. Вона має багату екосистему бібліотек і фреймворків, таких як React, Angular, Vue.js і Express.js, які спрощують і прискорюють процес розробки.

JavaScript розвивався протягом багатьох років, з'являлися нові функції та вдосконалення. Специфікація ECMAScript, яка слугує стандартом для JavaScript, періодично представляє нові версії, причому ECMAScript 6 (ES6) є значною віхою, яка принесла багато вдосконалень у мову [8].

Ruby - це динамічна, об'єктно-орієнтована мова програмування, відома своєю простотою та читабельністю [9]. Її створив у середині 1990-х років Юкіхіро Мацумото. Ruby черпає натхнення з різних мов програмування, включаючи Perl, Smalltalk, Eiffel та Lisp.

Ruby створена для задоволення потреб програмістів і має на меті забезпечити приємний та інтуїтивно зрозумілий досвід програмування. Вона має чистий та елегантний синтаксис, який підкреслює читабельність і виражає концепції у спосіб, подібний до природної мови. Код на Ruby часто описують як "читабельний код" або "поезію".

Однією з визначальних особливостей Ruby є її орієнтація на об'єктно-орієнтоване програмування (ООП) [10]. У Ruby все є об'єктом, і ви можете визначати класи, створювати об'єкти, а також використовувати успадкування та інкапсуляцію для побудови складних систем. Ruby також підтримує інші парадигми програмування, такі як функціональне та процедурне програмування.

Ruby має всеосяжну стандартну бібліотеку, що пропонує широкий спектр функціональних можливостей без значної залежності від зовнішніх бібліотек. Крім того, спільнота розробників Ruby надає численні доповнення (бібліотеки), які розширюють її можливості, роблячи її придатною для різних сфер, включаючи веброзробку, автоматизацію, аналіз даних тощо.

PHP (Hypertext Preprocessor - препроцесор гіпертексту) - це серверна скриптова мова, призначена для веброзробки. Вона була створена Расмусом Лердорфом у 1994 році і відтоді стала однією з найпопулярніших мов для створення динамічних вебдодатків і вебсайтів [11].

PHP часто вбудовується безпосередньо в HTML-код, що дозволяє розробникам змішувати динамічний PHP-код зі статичним HTML-контентом. Вона надає широкий спектр можливостей і функцій, спеціально розроблених для веброзробки, включаючи інтеграцію з базами даних, управління сеансами, обробку форм, завантаження файлів і багато іншого.

PHP має відносно простий та інтуїтивно зрозумілий синтаксис, який легко вивчити, особливо початківцям. Він має C-подібний синтаксис з поєднанням знайомих конструкцій програмування, що робить його доступним для розробників з різним досвідом.

PHP має широку екосистему бібліотек і фреймворків, які надають готові функції та модулі для виконання поширених завдань, таких як підключення до баз даних (наприклад, PDO, MySQLi), розробка вебдодатків (наприклад, Laravel, Symfony), систем управління контентом (наприклад, WordPress, Drupal) та багато іншого.

PHP пропонує вбудовану підтримку різних баз даних, включаючи MySQL, PostgreSQL, SQLite та Oracle. Це дозволяє розробникам легко підключатися до баз даних, виконувати запити, отримувати дані та маніпулювати ними.

PHP підтримується більшістю операційних систем, включаючи Windows, macOS, Linux та Unix. Його можна використовувати з різними вебсерверами, такими як Apache, Nginx та Microsoft IIS.

PHP був спеціально розроблений для веброзробки, і він чудово справляється зі створенням динамічного вебконтенту. Він може легко інтегруватися з HTML, CSS, JavaScript та іншими вебтехнологіями для створення інтерактивних і цікавих вебдодатків.

Java - це високорівнева мова програмування загального призначення, яка була розроблена компанією Sun Microsystems (придбана корпорацією Oracle) в середині 1990-х років. Вона була розроблена як незалежна від платформи, забезпечуючи надійність та безпечність [12].

Java відома своїм принципом "напиши один раз, виконуй будь-де", що означає, що код Java може бути скомпільований у байт-код, який можна запустити на будь-якій платформі з віртуальною машиною Java (JVM). Це дозволяє виконувати програми на Java на різних операційних системах без необхідності перекомпіляції.

Java є повністю об'єктно-орієнтованою мовою програмування, де все є об'єктом. Вона підтримує такі поняття, як класи, успадкування, поліморфізм та інкапсуляція.

Програми на Java компілюються у байт-код, який можна виконати на будь-якій платформі із сумісною JVM. Це робить Java дуже портативною.

Java має вбудовані функції, які сприяють її надійності та безпеці. Це автоматичне керування пам'яттю (збір сміття), обробка винятків та сувора перевірка типів.

Java надає всеосяжну стандартну бібліотеку, яка пропонує широкий спектр готових класів і методів, що полегшує розробку додатків.

Java підтримує багатопоточність, що дозволяє розробникам створювати додатки, які можуть виконувати декілька потоків одночасно, підвищуючи продуктивність і швидкість реакції [13].

Java широко використовується в різних сферах, включаючи розробку корпоративного програмного забезпечення, розробку додатків для Android, веброзробку (з використанням фреймворків, таких як Spring та JavaServer Faces), наукові дослідження тощо. Вона має велику екосистему бібліотек, фреймворків та інструментів, які ще більше розширюють її можливості.

Таблиця 1.2 Порівняння Python, JavaScript, Ruby, PHP та Java

Мова програмування	**Використання**	**Синтаксис**	**Фреймворки та інструменти**	**Спільнота розробників**
Python	Веброзробка, наукові обчислення, штучний інтелект	Простий, логічний, читабельний	Django, Flask, Pyramid	Велика, активна
JavaScript	Веброзробка, фронтенд-розробка, розробка	Гнучкий, динамічний	React, Angular, Vue.js	Надзвичайно велика, активна

Мова програмування	Використання	Синтаксис	Фреймворки та інструменти	Спільнота розробників
	інтерактивних додатків			
Ruby	Веброзробка, стартапи, прототипування	Простий, елегантний	Ruby on Rails, Sinatra	Активна, але менша ніж у Python та JavaScript
PHP	Веброзробка, динамічні вебсайти	Легкий для вивчення, синтаксис схожий на C	Laravel, Symfony, CodeIgniter	Велика, але менш активна в порівнянні з Python та JavaScript
Java	Веброзробка, мобільна розробка, корпоративні додатки	Статичний, об'єктно-орієнтований	Spring, JavaServer Faces (JSF), Struts	Велика, особливо у сфері корпоративних додатків

Ми обираємо Python через його простоту вивчення, читабельний синтаксис та широкий спектр фреймворків, таких як Django і Flask, які полегшують розробку вебдодатків. Python також має велику та активну спільноту розробників, що означає наявність розширеного набору ресурсів та підтримки.

Ми обираємо JavaScript через його необхідність у веброзробці, зокрема для створення динамічних елементів та взаємодії з користувачем на стороні клієнта. Використання JavaScript дозволяє нам створювати багатофункціональні вебінтерфейси, реалізовувати анімацію, валідацію форм, обмін даними з сервером та багато іншого.

2.2 Вибір системи керування базами даних

При виборі системи керування базами даних (СКБД) для Python, є кілька варіантів. Вибір залежить від ваших конкретних вимог, потреб у масштабуванні, структури даних та вподобань. Розглянемо популярні варіанти СКБД для Python (рис. 2.1).

Рисунок 2.1 — СКБД для Python

SQLite є популярним вибором для системи керування базами даних (СКБД) на Python, особливо для малих та середніх додатків, які потребують легкого вбудованого рішення [14]. Оглянемо деякі ключові особливості SQLite:

SQLite розроблений, щоб бути легким і мінімалістичним. Вона реалізована як невелика, автономна бібліотека, яка вбудовується безпосередньо в додаток. Це означає, що вам не потрібно налаштовувати окремий сервер бази даних або встановлювати додаткове програмне забезпечення. Весь механізм бази даних міститься в одному файлі бібліотеки.

На відміну від традиційних клієнт-серверних СУБД, SQLite є безсерверною. Це означає, що немає окремого процесу сервера бази даних, який працює у фоновому режимі. Замість цього, доступ до бази даних здійснюється безпосередньо додатком. Це спрощує процес налаштування та конфігурації, а також зменшує накладні витрати, пов'язані з управлінням окремим сервером.

Бази даних SQLite зберігаються у вигляді одного файлу у файловій системі. Цей файл містить всю структуру бази даних, таблиці, індекси та дані. До цього файлу можна легко надавати спільний доступ або переміщати його між різними системами, що робить його дуже портативним. Файлова природа SQLite робить

його придатним для додатків, які потребують локального зберігання і перенесення даних.

SQLite надає простий і зручний інтерфейс для роботи з базами даних. Він підтримує SQL-запити і транзакції, дозволяючи виконувати різні операції з базами даних. Простота SQLite робить його чудовим вибором для розробників, які хочуть отримати безпроблемне вбудоване рішення для роботи з базами даних у своєму додатку.

SQLite є широко підтримуваною СКБД і має прив'язки для багатьох мов програмування, включаючи Python. Python включає модуль sqlite3 до своєї стандартної бібліотеки, який забезпечує зручний спосіб взаємодії з базами даних SQLite. Модуль пропонує високорівневий API для виконання SQL-запитів, управління транзакціями та отримання результатів.

Незважаючи на свою легкість, SQLite підтримує багато стандартних функцій SQL, включаючи поширені типи даних, індекси, тригери та представлення. Він також забезпечує підтримку транзакцій і властивостей ACID (атомарність, узгодженість, ізоляція, довговічність), забезпечуючи цілісність і надійність даних.

SQLite широко використовується в різних сценаріях, таких як локальне зберігання даних для десктопних додатків, розробка мобільних додатків, створення прототипів і тестування. Вона не призначена для заміни великих клієнт-серверних баз даних, але слугує ефективним і простим варіантом для невеликих додатків з помірними потребами у зберіганні даних.

MySQL - це популярна реляційна система управління базами даних з відкритим вихідним кодом, відома своєю продуктивністю, масштабованістю та надійністю [15]. Наведемо основні характеристики та особливості MySQL:

MySQL дотримується реляційної моделі, що означає, що вона організовує дані в таблиці з наперед визначеними схемами та забезпечує зв'язки між ними. Вона підтримує SQL (мова структурованих запитів) для запитів, маніпулювання та управління даними.

MySQL оптимізована для високошвидкісних операцій і може ефективно обробляти велику кількість паралельних транзакцій. Вона реалізує різні методи оптимізації продуктивності, такі як індексування, кешування та оптимізація запитів, щоб забезпечити швидкий і оперативний пошук даних.

MySQL добре масштабується, дозволяючи вам обробляти зростаючі обсяги даних і користувацького трафіку. Вона підтримує реплікацію, що дозволяє створювати кілька копій бази даних для підвищення продуктивності та відмовостійкості. Крім того, MySQL надає такі функції, як розбиття на розділи та шардинг для розподілу даних між декількома серверами в міру зростання вашого додатку.

MySQL забезпечує надійність даних завдяки таким функціям, як підтримка транзакцій і властивості ACID (атомарність, узгодженість, ізоляція, довговічність). Вона надає механізми для підтримки цілісності та відновлення даних, такі як відновлення після збоїв та автоматичне резервне копіювання.

MySQL - це СУБД з відкритим вихідним кодом, що означає, що вона вільно доступна для використання, модифікації та розповсюдження. Вона має велику і активну спільноту розробників і користувачів, які роблять свій внесок у її розвиток, надають підтримку і діляться ресурсами, що робить її надійним і добре підтримуваним варіантом.

MySQL доступна для багатьох операційних систем, включаючи Windows, macOS, Linux та Unix-подібні системи. Вона також сумісна з різними мовами програмування, включаючи Python. Існує кілька бібліотек Python для MySQL, таких як pymysql і mysql-connector-python, які надають зручні інтерфейси для взаємодії з базами даних MySQL.

MySQL пропонує широкий набір функцій, включаючи підтримку збережених процедур, тригерів, представлень, повнотекстового пошуку та просторових даних. Вона надає кілька механізмів зберігання даних, таких як InnoDB, MyISAM і MEMORY, кожен з яких має свої сильні сторони і

характеристики, що дозволяє вибрати найбільш підходящий механізм для вашого конкретного випадку використання.

MySQL добре інтегрується з іншими технологіями, які зазвичай використовуються у веброзробці. Її часто використовують разом з такими вебфреймворками, як Django та Flask, забезпечуючи надійну та масштабовану серверну частину бази даних для вебдодатків. Крім того, MySQL пропонує підтримку різних роз'ємів, API та протоколів, що полегшує інтеграцію з різними мовами програмування та платформами.

MySQL широко використовується в різних галузях і додатках, починаючи від невеликих проектів і закінчуючи великими корпоративними системами. Продуктивність, масштабованість, надійність і багатий набір функцій роблять її популярним вибором для розробників, які працюють з Python та іншими мовами програмування.

PostgreSQL, яку часто називають Postgres, - це багатофункціональна і високопродуктивна об'єктно-реляційна система управління базами даних (з відкритим вихідним кодом [16]. Нижче наведено основні відомості про PostgreSQL та її можливості:

PostgreSQL поєднує в собі традиційну функціональність реляційних баз даних з підтримкою розширених типів даних та об'єктно-орієнтованих функцій. Вона дозволяє визначати і використовувати складні структури даних, створювати користувацькі типи, а також використовувати успадкування і поліморфізм.

PostgreSQL відома своєю стійкістю та надійністю. У ній реалізована система управління багатоверсійним паралелізмом (MVCC), яка забезпечує високий рівень паралелізму і гарантує, що транзакції ізольовані одна від одної. Це запобігає таким проблемам, як брудні читання, і забезпечує послідовний і надійний доступ до даних.

PostgreSQL пропонує багатий набір розширених функцій, які виходять за рамки стандартних можливостей SQL. Вона підтримує складні запити, що

включають об'єднання, підзапити, загальні табличні вирази (CTE) і віконні функції. Крім того, він забезпечує підтримку повнотекстового пошуку, геопросторових даних, JSON, XML тощо.

PostgreSQL надає всебічну підтримку цілісності даних за допомогою обмежень, включаючи первинні ключі, зовнішні ключі, унікальні обмеження та обмеження на перевірку. Це дозволяє вам визначати правила, які регулюють цілісність і узгодженість ваших даних, забезпечуючи їх надійність.

PostgreSQL має високу розширюваність і дозволяє розробникам визначати власні типи даних, оператори, функції та процедурні мови. Ця розширюваність дозволяє адаптувати базу даних до ваших специфічних вимог і з легкістю створювати складні додатки.

PostgreSQL пропонує відмінні механізми контролю паралелізму, що дозволяють декільком клієнтам отримувати доступ до бази даних одночасно без конфліктів. Він підтримує різні рівні ізоляції, а система MVCC забезпечує послідовне читання навіть під час одночасних операцій запису. Крім того, PostgreSQL надає можливості для масштабування бази даних, такі як логічна реплікація та вбудована підтримка шардингу [17].

Розробники Python можуть взаємодіяти з PostgreSQL за допомогою бібліотеки psycopg2, яка забезпечує надійний та ефективний інтерфейс Python для PostgreSQL. psycopg2 дозволяє без проблем підключатися до бази даних, виконувати запити, керувати транзакціями та отримувати результати.

PostgreSQL має жваву та активну спільноту користувачів та розробників, що сприяє її постійному вдосконаленню та широкому впровадженню. Характер PostgreSQL, керований спільнотою, забезпечує регулярні оновлення, виправлення помилок і доступність додаткових розширень і плагінів для розширення функціональності.

PostgreSQL розроблена для роботи на різних операційних системах, включаючи Windows, macOS, Linux та Unix-подібні системи. Ця крос-

платформенна сумісність дозволяє розгортати PostgreSQL у бажаному середовищі без будь-яких обмежень.

PostgreSQL широко використовується в різних сферах, включаючи вебдодатки, сховища даних, геопросторові бази даних, наукові дослідження тощо. Її універсальність, розширені можливості та суворе дотримання стандартів роблять її привабливим вибором для проектів, які потребують потужної та масштабованої системи управління базами даних.

MongoDB - це широко використовувана система управління базами даних з відкритим вихідним кодом NoSQL (Not Only SQL), орієнтована на роботу з документами [18]. Вона відрізняється від традиційних баз даних SQL тим, що зберігає дані в гнучкому, безсхемному форматі, використовуючи модель документів на основі BSON (Binary JSON). Ось ключові деталі та особливості MongoDB:

MongoDB зберігає дані в колекціях, які є групами JSON-подібних документів. Кожен документ є самодостатньою одиницею, яка може містити різні типи даних, що робить його придатним для обробки неструктурованих або напівструктурованих даних. Документи в колекції можуть мати різну структуру, що дозволяє динамічно розвивати схему.

MongoDB використовує BSON, двійкове представлення JSON, для зберігання документів. BSON забезпечує підтримку різних типів даних, включаючи рядки, числа, дати, масиви, вбудовані документи тощо. Гнучка схема дозволяє легко зберігати та отримувати складні структури даних.

MongoDB розроблена для горизонтального масштабування на декількох серверах, забезпечуючи високу доступність і продуктивність. Вона підтримує шардінг, що дозволяє розподіляти дані між декількома машинами, забезпечуючи плавне масштабування по мірі зростання ваших даних і трафіку користувачів. Крім того, MongoDB використовує механізми кешування та індексування для оптимізації продуктивності запитів.

MongoDB підтримує потужні спеціальні запити, які дозволяють отримувати дані за допомогою гнучкої та виразної мови запитів. Ви можете виконувати складні запити з використанням умов, порівнянь, логічних операторів і проекцій, щоб отримати певні підмножини даних з ваших колекцій.

MongoDB містить потужний фреймворк агрегації, який дозволяє виконувати складні операції агрегації даних, такі як групування, фільтрація, сортування та перетворення даних. Фреймворк агрегації корисний для створення звітів, проведення аналітики та вилучення інсайтів з ваших даних [19].

MongoDB надає бібліотеку pymongo, яка пропонує комплексний інтерфейс Python для взаємодії з базами даних MongoDB. Це дозволяє вам підключатися до MongoDB, виконувати операції CRUD (створення, читання, оновлення, видалення), виконувати запити, а також безперешкодно працювати з індексами та агрегаціями у ваших додатках на Python.

MongoDB надає такі функції, як набори реплік, які є самовідновлюваними кластерами серверів баз даних, що забезпечують автоматичне обхід збоїв та резервування даних. Набори реплік забезпечують високу доступність і відмовостійкість, зберігаючи кілька копій даних на різних серверах.

MongoDB має велику спільноту розробників і користувачів, що сприяє її постійному вдосконаленню та зростанню екосистеми. Він пропонує широку документацію, навчальні посібники та ресурси підтримки спільноти, щоб допомогти розробникам розпочати роботу та вирішити проблеми.

MongoDB широко використовується в різних сферах застосування, таких як системи управління контентом, аналітика в реальному часі, додатки для соціальних мереж, зберігання даних IoT (Інтернет речей) тощо. Гнучкість, масштабованість та легка інтеграція з Python роблять її популярним вибором для роботи з різноманітними та мінливими вимогами до даних.

Після ретельного огляду обираємо SQLite як кращу систему управління базами даних для нашої вебсистеми на Python. Завдяки своїй легкій та вбудованій природі SQLite ідеально відповідає вимогам нашого проекту.

2.3 Вибір фреймворку

Python має багату екосистему вебфреймворків, які задовольняють різні потреби та уподобання. Розглянемо найпопулярніші вебфреймворки для Python (рис. 2.2).

Рисунок 2.2 – Найпопулярніші вебфреймворки для Python

Django - потужний і широко використовуваний вебфреймворк для Python, який слідує філософії "батарейки в комплекті", що означає, що він надає повний набір інструментів і функцій "з коробки" [20]. Ось ключові деталі та особливості Django:

Django - це високорівневий фреймворк, який абстрагується від багатьох низькорівневих деталей веброзробки, дозволяючи розробникам зосередитися на швидкому створенні додатків. Він надає багатий набір готових компонентів, включаючи ORM (об'єктно-реляційне відображення) для доступу до баз даних, механізм шаблонів для генерації HTML, систему маршрутизації URL-адрес, обробку форм, автентифікацію та багато іншого.

Django чудово підходить для створення вебдодатків на основі баз даних. Він включає потужний ORM, який абстрагує операції з базами даних, дозволяючи розробникам працювати з базами даних за допомогою коду на Python, а не писати SQL запити напряму. Django підтримує різні бекенди баз даних, включаючи PostgreSQL, MySQL, SQLite та Oracle.

Django підкреслює простоту і слідує принципу "не повторюйся" (DRY), зменшуючи потребу в шаблонному коді і сприяючи повторному використанню коду. Він забезпечує чітку і послідовну структуру для організації коду, розділення проблем і дотримання найкращих практик. Автоматичний інтерфейс адміністратора Django ще більше спрощує операції CRUD (створення, читання, оновлення, видалення), що дозволяє швидко розробляти та керувати контентом.

Django має вбудовані функції безпеки, які допомагають розробникам створювати безпечні додатки. Він забезпечує захист від поширених вебуразливостей, таких як міжсайтовий скриптинг (XSS), підробка міжсайтових запитів (CSRF) та SQL-ін'єкції. Система автентифікації та авторизації Django пропонує управління користувачами, обробку сеансів та детальний контроль доступу.

Django є універсальним і дозволяє розширювати його за допомогою повторно використовуваних компонентів. Він підтримує сторонні пакети і має багату екосистему багаторазових додатків, які можна інтегрувати в проекти. Модульний дизайн Django заохочує до створення та спільного використання багаторазових додатків, що може пришвидшити розробку та підвищити якість коду.

Django має велику і добре організовану документацію, що полегшує розробникам навчання і початок роботи [21]. Вона має активну спільноту розробників, які роблять свій внесок у її розвиток, діляться знаннями та надають підтримку. Спільнота Django пропонує такі ресурси, як навчальні посібники, форуми та конференції, що сприяють співпраці та зростанню.

Django легко інтегрується з іншими бібліотеками та фреймворками Python. Він добре працює з фронтенд-фреймворками, такими як React, Vue.js та Angular, що дозволяє розробляти сучасні та адаптивні вебінтерфейси. REST-фреймворк Django полегшує створення API і дозволяє створювати вебсервіси та мобільні додатки.

Django широко прийнятий і використовується багатьма організаціями та розробниками по всьому світу. Надійність, масштабованість, функції безпеки та акцент на продуктивність роблять його чудовим вибором для створення складних та багатофункціональних вебдодатків, особливо для великомасштабних проектів.

Flask - популярний вебфреймворк Python, відомий своєю простотою, гнучкістю та легкістю у використанні [22]. Ключові особливості Flask:

Flask розроблений як легкий фреймворк з мінімалістичним ядром. Він надає лише основні функції, необхідні для веброзробки, що дозволяє розробникам тримати свою кодову базу стрункою та сфокусованою. Ця простота робить Flask легким для розуміння і швидким для початку роботи.

Flask дотримується мікро-фреймворкового підходу, що означає, що він надає базовий набір інструментів і функцій, дозволяючи розробникам додавати функціональність за потребою. Цей підхід дає розробникам свободу вибору та інтеграції додаткових бібліотек і розширень відповідно до їхніх конкретних вимог, що робить Flask дуже гнучким у налаштуванні.

Flask надає просту та інтуїтивно зрозумілу систему маршрутизації, яка дозволяє розробникам визначати шаблони URL-адрес і зіставляти їх з відповідними функціями Python, відомими як функції перегляду. Це дозволяє обробляти різні методи HTTP (GET, POST тощо) та створювати динамічні та RESTful API.

Flask постачається з вбудованим движком шаблонів Jinja2, який дозволяє розробникам створювати та рендерити HTML-шаблони з динамічним контентом. Шаблонизація спрощує процес створення динамічних вебсторінок, відокремлюючи логіку представлення від логіки додатку.

Flask має багату екосистему розширень, які надають додаткову функціональність фреймворку. Ці розширення охоплюють широкий спектр областей, включаючи інтеграцію з базами даних, обробку форм, автентифікацію, кешування та багато іншого. Розробники можуть вибирати та інтегрувати

розширення, які найкраще відповідають вимогам їхнього проекту, розширюючи можливості Flask з легкістю.

Flask-SQLAlchemy - це розширення, яке інтегрує бібліотеку SQLAlchemy ORM (Object-Relational Mapping) з Flask. Воно забезпечує зручний спосіб взаємодії з базами даних, використовуючи потужні можливості SQLAlchemy, використовуючи при цьому простоту Flask. Flask-SQLAlchemy спрощує роботу з базами даних та покращує читабельність коду.

Flask має вбудований сервер розробки, який дозволяє розробникам запускати і тестувати свої додатки локально. Він також надає потужний налагоджувач, який допомагає виявляти та виправляти помилки під час процесу розробки. Ці інструменти спрощують робочий процес розробки та сприяють ефективному налагодженню.

Pyramid - універсальний вебфреймворк для Python, який робить акцент на гнучкості та кастомізації. Він дотримується мінімалістичної філософії, надаючи базовий набір інструментів і дозволяючи розробникам приймати рішення щодо архітектури та компонентів свого додатку [23]. Розглянемо ключові особливості та характеристики Pyramid:

Pyramid розроблена дуже гнучко, надаючи розробникам свободу вибору компонентів і бібліотек, які вони хочуть використовувати. Вона не нав'язує певну базу даних ORM (об'єктно-реляційне відображення) або механізм шаблонів, дозволяючи розробникам обирати ті, які найкраще відповідають потребам їхнього проекту. Така гнучкість робить Pyramid адаптивною до широкого спектру вимог додатків.

Pyramid добре підходить як для невеликих персональних проектів, так і для великих корпоративних систем. Вона добре масштабується завдяки модульному дизайну та поділу завдань. Розробники можуть створювати додатки таким чином, щоб забезпечити легке розширення та адаптацію в міру зростання складності та вимог проекту.

Pyramid пропонує потужну систему маршрутизації, яка зіставляє URL-адреси з функціями перегляду. Вона підтримує гнучкі шаблони URL-адрес і може обробляти складні сценарії маршрутизації. Розробники можуть визначати маршрути за допомогою простих шаблонів або більш просунутих методів, таких як регулярні вирази, що дозволяє легко створювати чисті та зручні для SEO URL-адреси.

Pyramid надає надійні функції безпеки, включаючи механізми автентифікації та авторизації. Він пропонує підтримку поширених схем автентифікації, таких як автентифікація на основі сеансу, автентифікація на основі токенів та інтеграцію зі сторонніми постачальниками автентифікації. Pyramid також включає в себе інструменти для управління ролями та дозволами користувачів, забезпечуючи безпечний контроль доступу до ресурсів програми.

Pyramid робить акцент на тестуванні і надає інструменти та утиліти для полегшення створення модульних та інтеграційних тестів. Він сприяє розробці на основі тестування та пропонує вичерпну документацію, щоб допомогти розробникам зрозуміти та ефективно використовувати фреймворк.

Pyramid має активну спільноту розробників, яка надає підтримку. Екосистема Pyramid включає широкий спектр розширень, бібліотек та плагінів, створених спільнотою, що дозволяє розробникам використовувати існуючі рішення та інтегруватися з популярними інструментами та технологіями.

CherryPy - це легкий і мінімалістичний вебфреймворк для Python, який робить акцент на простоті та продуктивності. Він надає необхідні інструменти та функції для створення вебдодатків, зберігаючи при цьому невеликий розмір і мінімальний набір залежностей [24]. Особливості CherryPy:

CherryPy дотримується філософії простоти, прагнучи зробити веброзробку простою та інтуїтивно зрозумілою. Його API розроблений таким чином, щоб бути чистим і легким для розуміння, що дозволяє розробникам швидко створювати вебдодатки без зайвої складності.

CherryPy використовує об'єктно-орієнтований підхід до проектування, що дозволяє розробникам організовувати свій код у класи та методи. Цей модульний підхід сприяє повторному використанню коду та його підтримці, що полегшує управління та розширення додатків у міру їх зростання.

CherryPy містить вбудований вебсервер, який дозволяє розробникам запускати свої додатки без необхідності використання додаткового програмного забезпечення вебсервера під час розробки. Цей вбудований сервер підходить для розробки та тестування і може обробляти декілька запитів одночасно.

CherryPy надає гнучкий механізм конфігурації, що дозволяє розробникам визначати налаштування та поведінку за допомогою конфігураційних файлів або програмно. Ця гнучкість забезпечує тонкий контроль над налаштуваннями додатків і дозволяє легко налаштовувати їх відповідно до конкретних вимог проекту.

CherryPy слідує традиційній моделі циклу запит-відповідь. Вона забезпечує інтуїтивно зрозумілий спосіб обробки вхідних запитів, перенаправлення їх до відповідних обробників та генерування відповідей на основі запитуваних ресурсів або дій. Розробники можуть визначати маршрути URL-адрес і пов'язувати їх з конкретними методами контролерів для обробки різних типів запитів.

CherryPy підтримує архітектуру плагінів, яка дозволяє розробникам розширювати функціональність фреймворку. Плагіни можуть додавати такі функції, як автентифікація, кешування, ведення журналів та інтеграція з базами даних. Така розширюваність дозволяє розробникам пристосовувати CherryPy до конкретних вимог їхніх додатків.

Після ретельного огляду було вирішено обрати Django як вебфреймворк для нашого проекту. Акцент Django на безпеці, масштабованості та найкращих практиках ідеально відповідає вимогам нашого проекту. Крім того, Django має велику спільноту, обширну документацію та велику екосистему сторонніх пакетів, які значно підтримають процес розробки.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ ШКІЛЬНОГО ОСВІТНЬОГО ПРОЦЕСУ

3.1 Методи та засоби об'єктно-орієнтованого аналізу і проектування

Уніфікована мова моделювання (UML) - це стандартизована мова візуального моделювання для специфікації, візуалізації, конструювання та документування артефактів програмної системи. UML надає набір нотацій та діаграм для представлення різних аспектів системи, включаючи діаграми класів, діаграми варіантів використання, діаграми послідовності тощо. UML широко прийнята і підтримується багатьма інструментами розробки програмного забезпечення [25].

3.1.1 Діаграма варіантів використання

Діаграма варіантів використання - це техніка, яка використовується для фіксації функціональних вимог до системи з точки зору її користувачів. Варіанти використання представляють конкретні взаємодії між акторами (користувачами або зовнішніми системами) і системою, описуючи цілі, яких необхідно досягти. Діаграми варіантів використання зазвичай використовуються для візуалізації та документування варіантів використання та їх взаємозв'язків [26].

На рисунку 3.1 зображено діаграму варіантів використання веборієнтованої інформаційної системи організації шкільного освітнього процесу.

Нижче наведена діаграма відображає варіанти використання для ролей "Admin" та "User" у системі. Адміністратор (Admin) має доступ до всіх операцій створення, редагування та видалення об'єктів, таких як Вчитель (Teacher), Клас (Class), Урок (Lesson), Розклад (Schedule) та Журнал (Journal). З іншого боку, користувач (User) має лише можливість переглядати ці об'єкти без зміни або видалення їх.

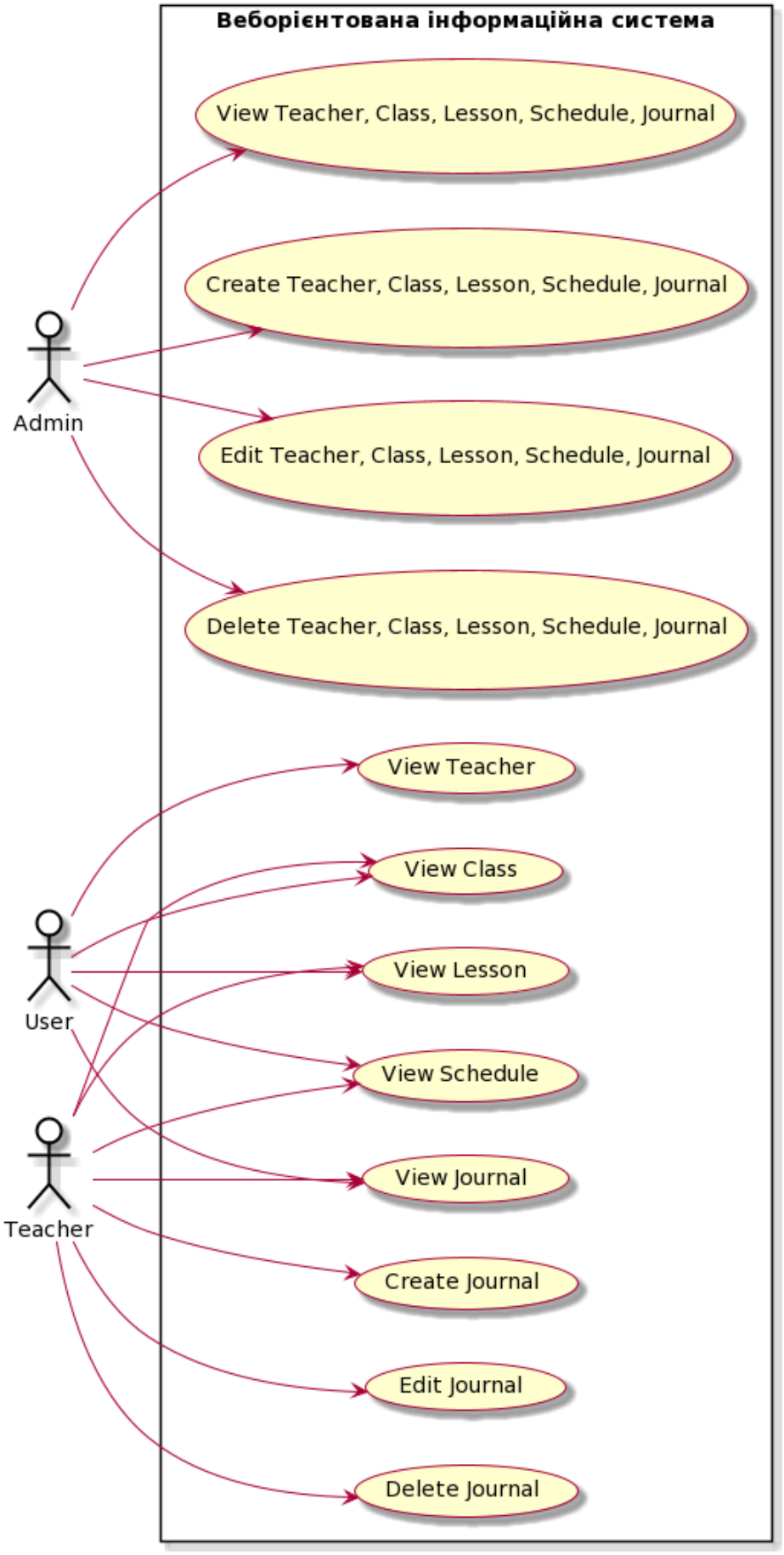


Рисунок 3.1 – Діаграма варіантів використання вебсистеми

3.1.2 Діаграма послідовності (Sequencediagram)

Діаграма послідовності (Sequencediagram) використовується для візуалізації взаємодії між об'єктами або компонентами системи у визначений часовий проміжок. Вона відображає послідовність повідомлень, які обмінюються між цими об'єктами під час виконання певного сценарію або функції [27].

Діаграма послідовності складається з вертикальних ліній, які представляють об'єкти або компоненти, і горизонтальних стрілок, які показують послідовність викликів або повідомлень між цими об'єктами. Вона дозволяє відстежувати порядок виконання операцій, передачу даних та повідомлень між об'єктами, а також визначити, які об'єкти залежать від інших і в якому порядку вони виконуються.

Діаграми послідовності є потужним інструментом для аналізу, проектування та моделювання взаємодії між об'єктами в програмних системах (рис. 3.2).

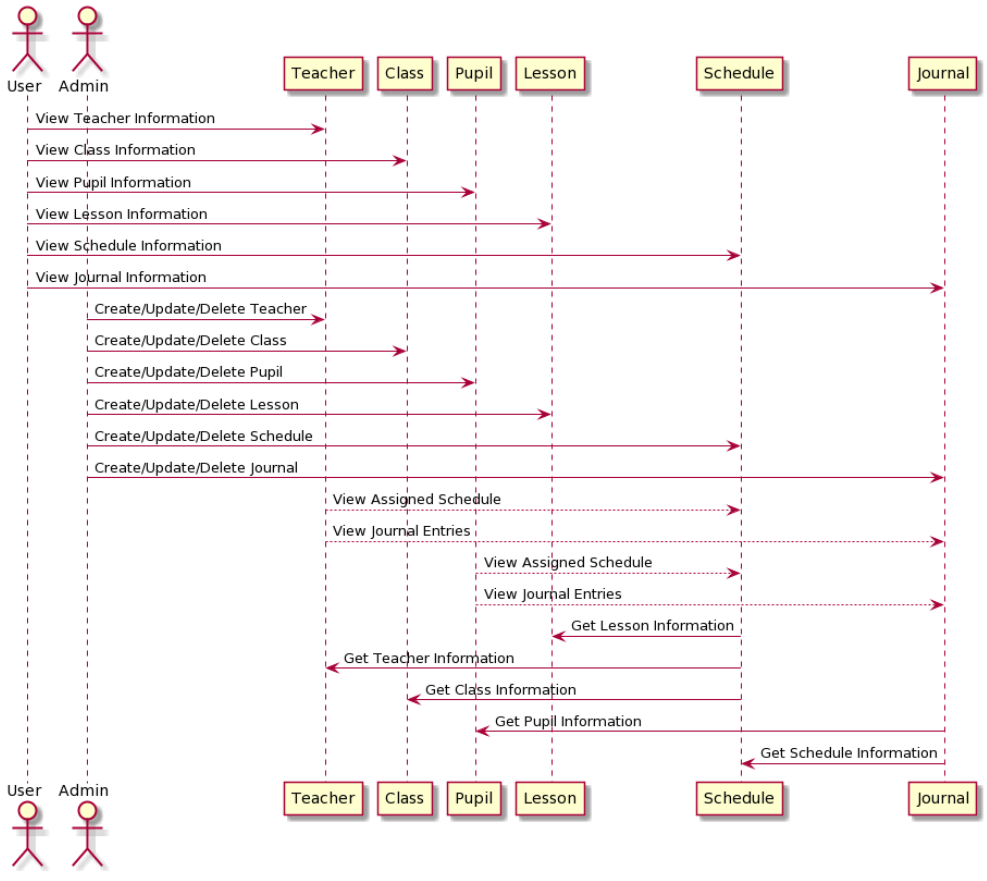


Рисунок 3.2 – Діаграма варіантів використання вебсистеми

Ця діаграма послідовності демонструє взаємодію акторів (User, Admin, Teacher, Pupil) з моделями даних (Teacher, Class, Pupil, Lesson, Schedule, Homework, Journal) у вашій системі. Актор User має обмежені права на перегляд інформації, тоді як Admin має повний доступ до створення, редагування та видалення записів. Учителі, учні та розклади взаємодіють з різними моделями даних у відповідних контекстах.

3.1.3 Діаграма діяльності (Activitydiagram)

Діаграма діяльності є одним із типів структурних діаграм, що використовується для моделювання процесів, потоків роботи та поведінки системи. Вона дозволяє представити послідовність дій, діаграмувати взаємодію між різними елементами системи та візуалізувати керування потоком в процесі виконання [28].

Діаграми діяльності складаються з активностей (дій), рішень, розділів та керуючих зв'язків. Активності представляють конкретні дії або процеси, що відбуваються в системі, а рішення визначають вибір або умову, яка впливає на потік виконання. Розділи використовуються для групування пов'язаних активностей, а керуючі зв'язки вказують послідовність та умови переходу між активностями.

Діаграми діяльності надають інтуїтивний спосіб моделювання процесів, відображення послідовності дій та визначення взаємодії між об'єктами.

Діаграма діяльності яка зображена на рисунку 3.3 відображає загальний потік дій для веб-орієнтованої інформаційної системи. Вона починається з початкової точки та вимагає введення даних. Потім вона перевіряє правильність авторизаційних даних. Якщо дані коректні, доступ дозволяється, і користувач може виконати операції в рамках основного функціоналу. Якщо операція успішна, результат відображається, в іншому випадку виводиться повідомлення про помилку. Якщо дані не коректні або доступ заборонено, робота системи зупиняється.

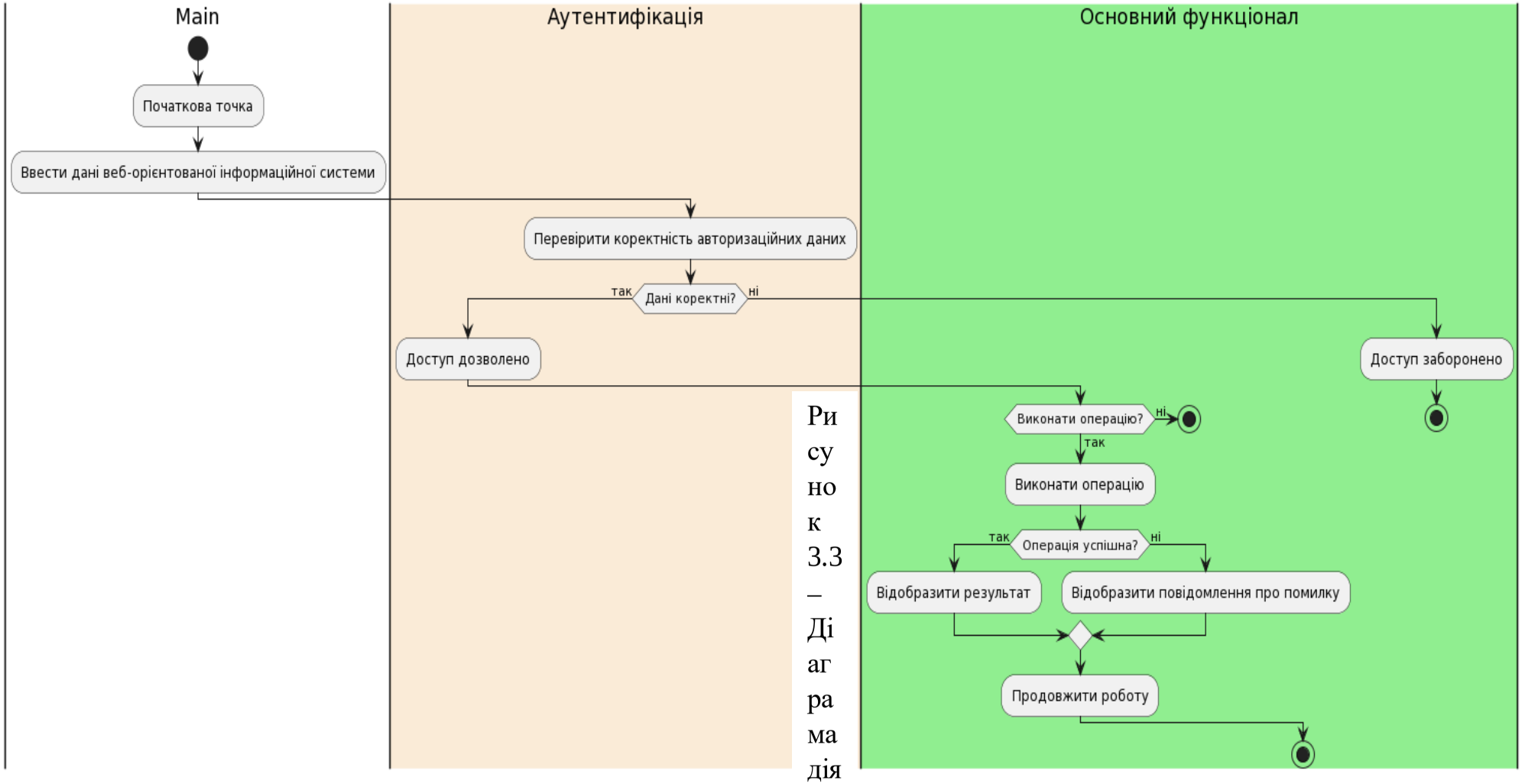


Рисунок 3.3 – Діаграма діяльності (Activityd

3.1.4 Діаграма класів

Діаграма класів - це структурна діаграма, яка відображає класи системи, їх атрибути, методи та взаємозв'язки між ними. Вона дозволяє візуалізувати структуру системи, показуючи класи і їхні залежності.

На діаграмі класів класи зображуються у вигляді прямокутників, які містять назви класів. Внутрішні атрибути та методи класу також можуть бути вказані у прямокутнику. Взаємозв'язки між класами показуються за допомогою стрілок, які вказують напрямок залежності [29].

Діаграма класів допомагає розуміти структуру системи, взаємозв'язки між класами, а також допомагає в процесі проектування, рефакторингу та аналізу програмного забезпечення (рис. 3.4). Вона є потужним інструментом для моделювання об'єктно-орієнтованих систем і допомагає зрозуміти взаємодію класів в системі.

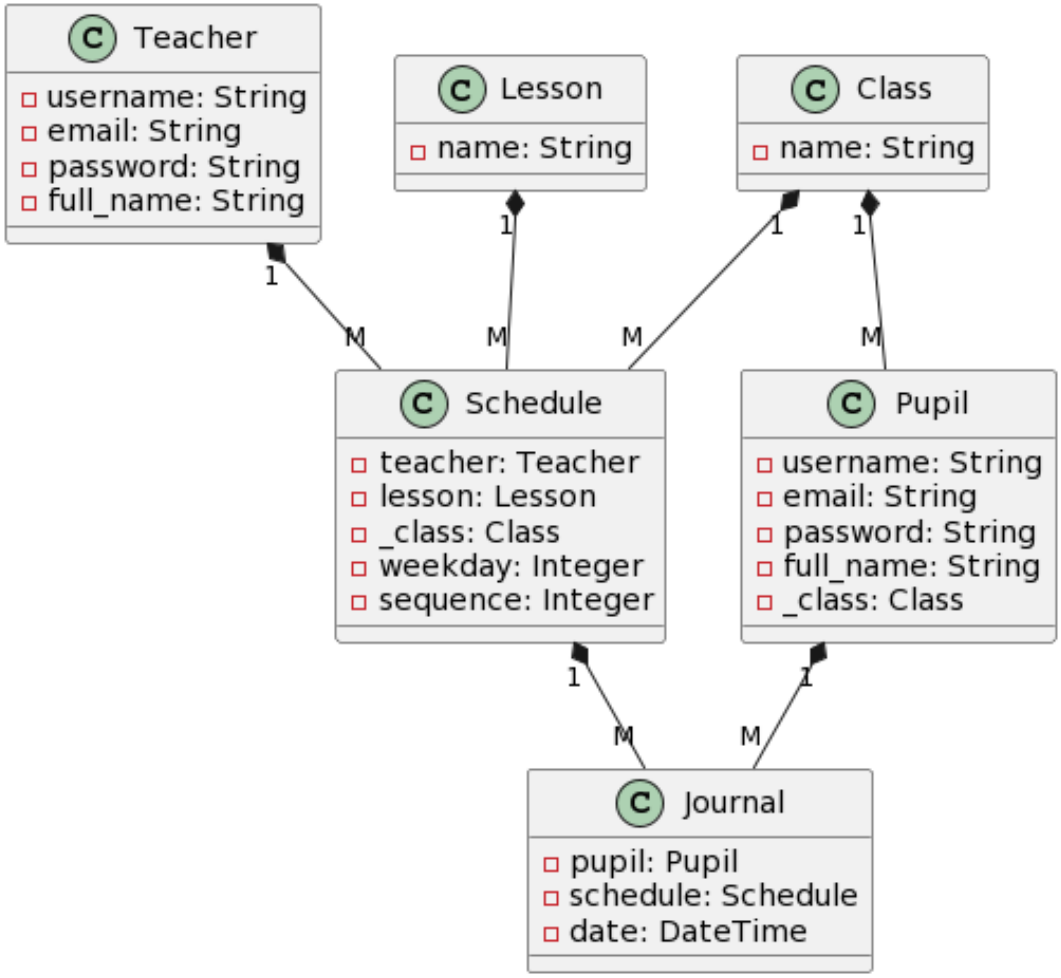


Рисунок 3.4 – Діаграма класів вебсистеми

Ця діаграма показує взаємозв'язки між класами та їхні атрибути. Наприклад, клас **Teacher** має зв'язок "1 до багатьох" з класом **Schedule**, оскільки один вчитель може мати багато розкладів. Так само, клас **Schedule** має зв'язки "1 до багатьох" з класом **Journal**, оскільки один розклад може мати багато домашніх завдань та записів в журналі.

3.2 Дизайн вебсайту системи

Щоб забезпечити безперебійний та ефективний процес веброзробки, важливо створити макет сайту. Для цього буде використано Figma, потужний інструмент для дизайну та створення прототипів. Figma пропонує широкий спектр функцій і можливостей, які дозволяють дизайнерам створювати комплексні та інтерактивні макети [30].

Деякі ключові особливості Figma:

– Figma надає надійний інтерфейс для проектування користувацьких інтерфейсів (UI). Він пропонує величезну колекцію елементів дизайну, включаючи кнопки, іконки, типографіку та багато іншого. Дизайнери можуть легко перетягувати ці елементи на полотно, щоб створити бажаний макет.

– Figma дозволяє дизайнерам створювати інтерактивні прототипи для імітації взаємодії та потоків користувачів. За допомогою гарячих точок та інтерактивних компонентів дизайнери можуть визначати дії, переходи та анімацію, надаючи зацікавленим сторонам реалістичний попередній перегляд кінцевого продукту.

– Figma підтримує принципи адаптивного дизайну, що дозволяє дизайнерам створювати макети, які адаптуються до різних розмірів екранів і пристроїв. Дизайнери можуть визначати обмеження і використовувати функції автоматичного макетування, щоб дизайн залишався послідовним і візуально привабливим на різних пристроях.

– Figma пропонує потужну систему компонентів і стилів, яка сприяє узгодженості дизайну та його багаторазовому використанню. Дизайнери можуть

створювати елементи дизайну та керувати ними як компонентами багаторазового використання, що дозволяє легко підтримувати та оновлювати систему дизайну протягом усього проекту.

Figma спрощує процес проектування, підвищує продуктивність і сприяє ефективній комунікації між членами команди, що призводить до створення візуально привабливих і зручних для користувача вебінтерфейсів.

Спочатку розробимо макет головної сторінки (рис. 3.5)

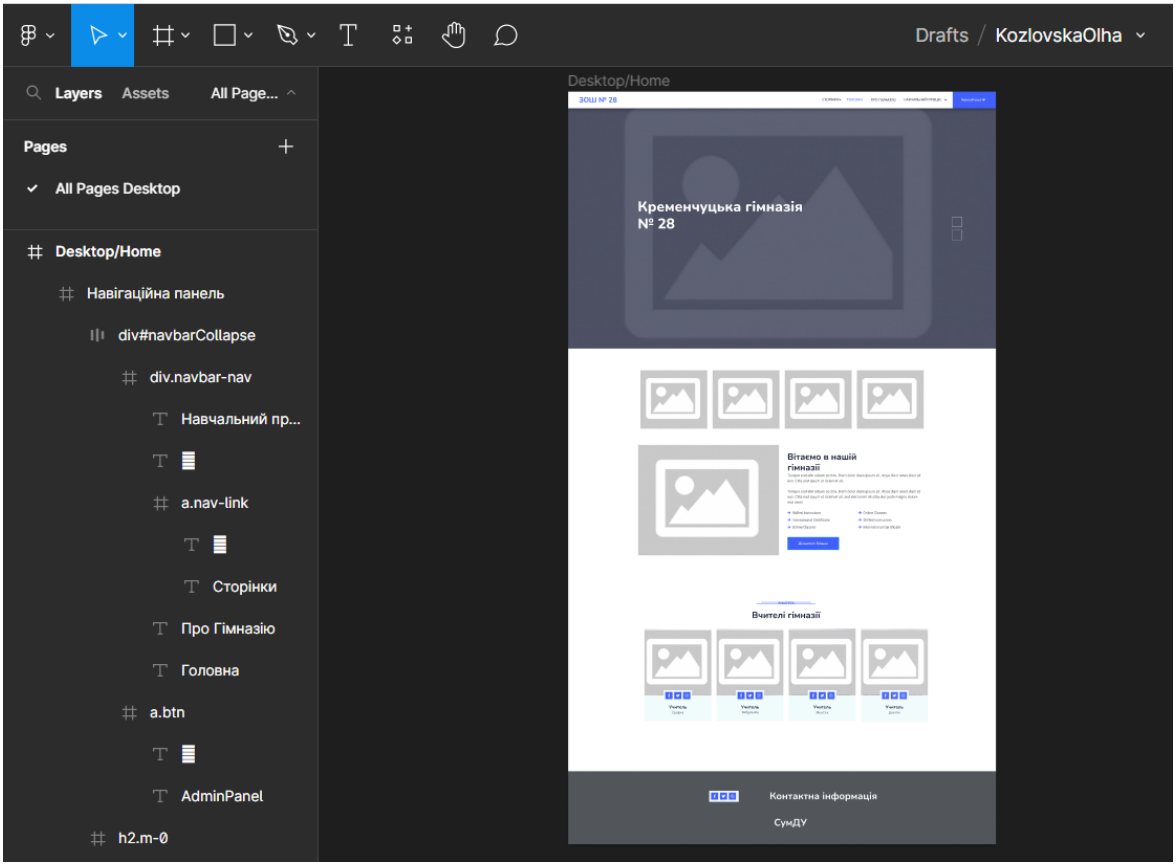


Рисунок 3.5 – Макет головної сторінки у Figma

Основні компоненти Figma [31], які були використані при створенні макета:

- ***Рамки***. Рамки слугують контейнерами для елементів вашого дизайну. Вони визначають межі вашого дизайну і дозволяють організувати та згрупувати пов'язані елементи разом. Ви можете створити кілька рамок для представлення різних екранів або розділів вашого дизайну.

● ***Фігури та векторні інструменти***. Figma надає різноманітні фігурні та векторні інструменти для створення та редагування базових геометричних фігур, піктограм та ілюстрацій. Ці інструменти включають прямокутники, кола, лінії та інструмент "Перо", що дозволяє створювати власні фігури та контури.

● ***Текстовий інструмент.*** Інструмент "Текст" у Figma дозволяє додавати і стилізувати текст у вашому дизайні. Ви можете встановити шрифт, розмір, колір, вирівнювання та інші властивості тексту. Figma підтримує як статичний, так і редагований текст, що дозволяє легко створювати заголовки, абзаци та мітки.

● ***Зображення***. Figma дозволяє імпортувати та розміщувати зображення у вашому дизайні. Ви можете змінювати розмір, обрізати та позиціонувати зображення в межах фреймів. Крім того, Figma надає можливості регулювати непрозорість, застосовувати маски і додавати ефекти до зображень.

● ***Компоненти.*** Компоненти - це багаторазові елементи дизайну, які можна створювати і використовувати впродовж усього проекту. Вони дають змогу підтримувати узгодженість і вносити глобальні зміни до кількох екземплярів компонента. Компоненти можуть включати кнопки, іконки, панелі навігації або будь-які інші елементи, які ви хочете використовувати повторно.

● ***Стилі***. Функція стилів Figma дозволяє вам визначати і застосовувати узгоджені стилі для кольорів, типографіки та ефектів. Створюючи та використовуючи стилі, ви можете легко оновлювати елементи дизайну в глобальному масштабі, забезпечуючи цілісний вигляд і відчуття в усьому вашому дизайні.

● ***Інструменти для створення прототипів***. Figma включає в себе можливості прототипування, які дозволяють створювати інтерактивні та анімовані прототипи. Ви можете визначати області, на які можна натиснути, створювати переходи та імітувати потоки користувачів, щоб продемонструвати інтерактивність вашого дизайну.

● ***Плагіни та інтеграції***. Figma підтримує широкий спектр плагінів та інтеграцій, які розширюють її функціональність. Плагіни дозволяють автоматизувати завдання, генерувати ресурси дизайну або інтегруватися з зовнішніми інструментами, покращуючи робочий процес проектування.

Далі розробимо макет сторінки «Про гімназію» (рис. 3.6)

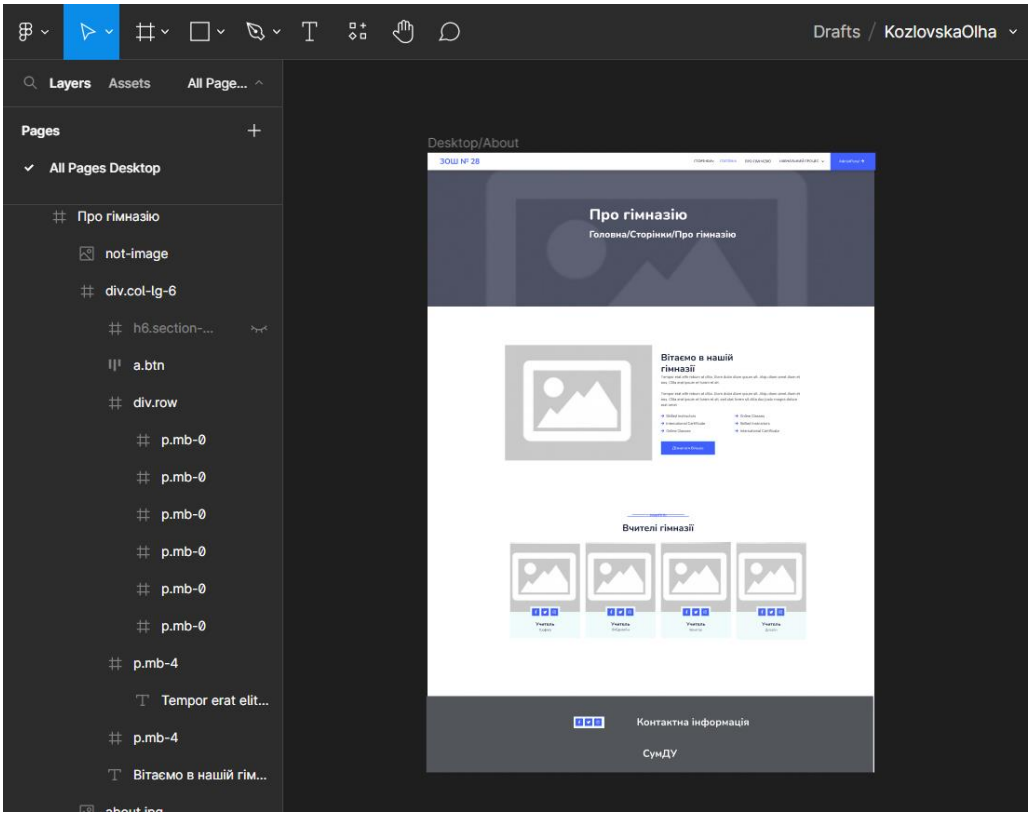


Рисунок 3.6 – Макет сторінки «Про гімназію» у Figma

Макет сторінок «Розклад занять» та «Журнал» створюємо пустими, бо наповнення буде відбуватися динамічно (рис. 3.7)

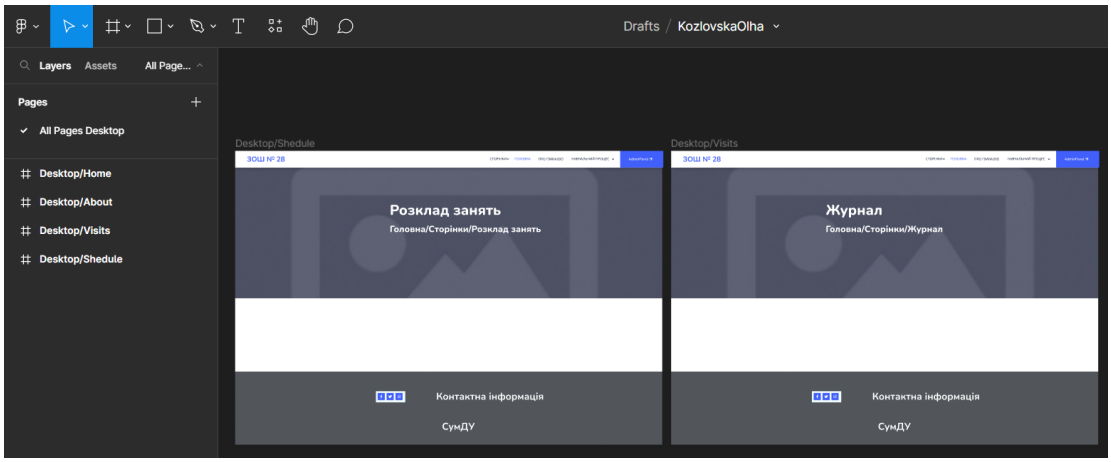
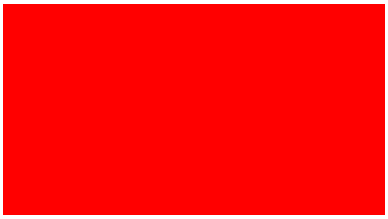


Рисунок 3.7 – Макети сторінок «Розклад занять» та «Журнал» у Figma

Таким чином, створення макета у Figma включає в себе створення фреймів, додавання елементів дизайну, застосування стилів, врахування адаптивності, створення прототипів. Figma надає повний набір інструментів і функцій для полегшення всього процесу створення макета, від створення каркасу до остаточної реалізації.

Для дизайну сайту гімназії будуть використані теплі кольори [32], щоб створити візуально привабливу та енергійну атмосферу. Теплі кольори відомі своїми асоціаціями з енергією, пристрастю та теплом. Вони можуть створити гостинне і стимулююче середовище, яке добре узгоджується з активним характером гімназії. Ось кілька прикладів теплих кольорів, які будуть використані в дизайні:



Вогненно-червоний (#ff0000). Яскравий та інтенсивний червоний колір можна використовувати, щоб привернути увагу і викликати почуття пристрасті та енергії. Його можна використовувати для кнопок із закликом до дії або виділення, щоб привернути увагу користувачів.

Енергійний помаранчевий (#ffa500). Помаранчевий - теплий і енергійний колір, який випромінює ентузіазм і життєву силу. Його можна використовувати для заголовків, акцентів або фонових елементів, щоб додати дизайну жвавості.

Підбадьорливий жовтий (#ffff00). Жовтий - яскравий і життєрадісний колір, який випромінює тепло й оптимізм. Його можна використовувати для таких елементів, як іконки, рамки або фони, щоб наповнити дизайн сонячним та енергійним настроєм.

Теплі земляні тони (зліва направо). Такі кольори, як теплий коричневий (#964b00), теракотовий (#e2725b) або вигоріла сієна (#e97451), можуть додати дизайну заземлений і природний відтінок. Їх можна використовувати для фону, текстур або допоміжних елементів, щоб створити відчуття тепла і затишку.

Пристрасний рожевий. Сміливий і яскравий відтінок рожевого може додати дизайну відтінок жіночності та грайливості. Його можна використовувати помірно для окремих елементів дизайну або як додатковий колір, щоб збалансувати загальну теплу кольорову гаму.

Окрім теплих кольорів основним кольором буде колір #415FFB який представляє собою відтінок синього, який можна описати як холодний, приємний для ока та заспокійливий.

3.3 Архітектура вебсайту

Архітектура вебсайту - це структурна організація, яка визначає компоненти, їх взаємозв'язки та способи взаємодії, які використовуються для

розробки вебсайту. Це включає в себе організацію файлів і каталогів, розмітку сторінок, базу даних, взаємодію з користувачем, логіку бізнес-процесів та інші складові [33].

Архітектура вебсайту визначається з урахуванням функціональних та нефункціональних вимог до сайту, таких як його мета, цільова аудиторія, обсяг інформації, безпека, масштабованість та продуктивність. Вона також включає в себе вибір відповідних технологій, платформи розробки, паттерни проектування та архітектурні шаблони.

Архітектура веборієнтованої інформаційної системи зображена на рисунку 3.8.

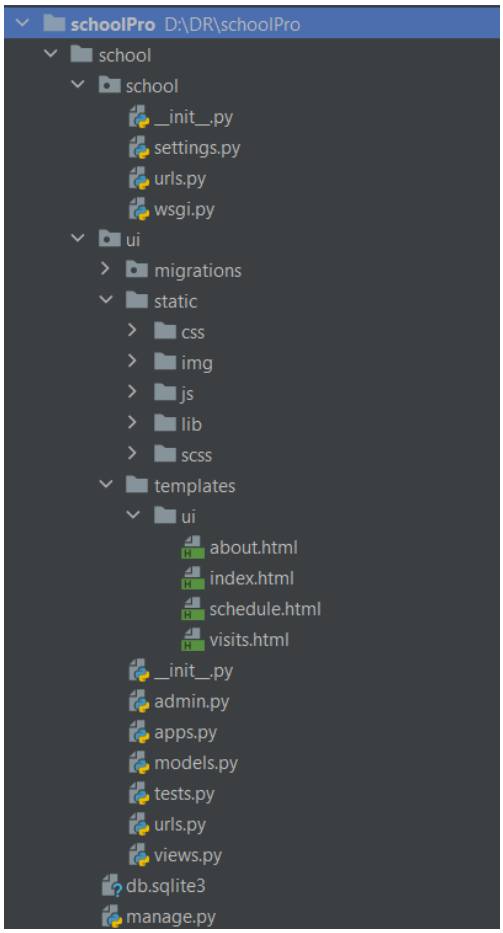


Рисунок 3.8 – Архітектура вебсайту

1. Файл db.sqlite3: Це файл бази даних SQLite, який використовується для зберігання даних, пов'язаних з вебсайтом.

2. Файл manage.py: Це файл, який використовується для керування вебсайтом, такий як запуск сервера розробки, виконання міграцій бази даних тощо.
3. Директорія "school": Ця директорія містить файл налаштувань settings.py, який визначає налаштування Django-проекту, такі як база даних, шляхи до статичних файлів, мова, часовий пояс тощо. Вона також містить файли urls.py та wsgi.py, які визначають шляхи та WSGI-сервер відповідно. Файл init.py використовується для позначення директорії як пакету Python.
4. Директорія "ui": Ця директорія містить файл admin.py, який використовується для налаштування адміністративного інтерфейсу Django. Файл apps.py визначає конфігурацію додатку "ui" в рамках Django-проекту. Файл models.py визначає моделі даних, пов'язані з додатком "ui". Файл tests.py містить тести для перевірки правильності роботи додатку. Файл urls.py визначає шляхи, пов'язані з додатком "ui". Файл views.py містить функції-контролери, які обробляють запити від користувачів. Файл init.py позначає директорію як пакет Python.
5. Директорія "ui/migrations": Ця директорія містить файли міграцій бази даних.
6. Директорія "ui/static": Ця директорія містить статичні ресурси, такі як CSS, зображення та JavaScript файлів, які використовуються на вебсайті. В директорії "css" знаходяться файли bootstrap.min.css та style.css, які визначають стилі для вебсайту. В директорії "img" знаходяться зображення, такі як about.jpg, carousel-1.jpg, team-1.jpg та інші, які використовуються на сторінках сайту. В директорії "js" знаходиться файл main.js, який містить JavaScript-код для вебсайту.
7. Директорія "ui/lib": Ця директорія містить різні бібліотеки та залежності, які використовуються на вебсайті. Вона містить піддиректорії "animate", "easing", "owlcarousel" та "wow". Кожна з цих піддиректорій містить

відповідні файли CSS та JavaScript, необхідні для візуальних ефектів та компонентів, які використовуються на вебсайті.

8. Директорія "ui/scss": Ця директорія містить файли SCSS, які використовуються для збірки стилів вебсайту. Вона містить файл bootstrap.scss, який містить SCSS-код для налаштування стилів Bootstrap.
9. Директорія "ui/templates/ui": Ця директорія містить шаблони HTML-сторінок для додатку "ui". Вона містить файли about.html, index.html, schedule.html та visits.html, які визначають вміст та розмітку сторінок сайту.

Ця архітектура вебсайту заснована на структурі Django-проекту та відповідає рекомендованій організації файлів і директорій для створення вебдодатків. Кожна директорія та файл мають свою роль і виконують певні функції у роботі вебсайту.

3.4 Верстка вебсайту. Frontend-частина

Верстка вебсайту - це процес створення структури, розміщення і стилізації елементів веб-сторінок з використанням мов розмітки, таких як HTML (HyperText Markup Language) і CSS (Cascading Style Sheets) та бібліотеки Bootstrap. Верстка веб-сайту визначає зовнішній вигляд і поведінку сторінок, а також їхню структуру і організацію [34].

Нижче описано верстку навігаційної панелі (navbar) головної сторінки index.html (рис.3.9)

Рисунок 3.9 – Верстка навігаційної панелі (navbar)

Основні компоненти цієї навігаційної панелі включають:

1. **<nav>**: Визначає контейнер для навігаційної панелі.
2. **.navbar**: Клас, що задає стилі для навігаційної панелі.
3. **.navbar-expand-lg**: Клас, що вказує на розширення навігаційної панелі для великих пристроїв.

4. **.bg-white**: Клас, що задає білий фон для навігаційної панелі.
5. **.navbar-light**: Клас, що встановлює світлу тему для навігаційної панелі.
6. **.shadow**: Клас, що додає тінь до навігаційної панелі.
7. **.sticky-top**: Клас, що фіксує навігаційну панель зверху сторінки.

Крім того, код містить такі елементи:

- **<a>**: Визначає посилання з логотипом на головну сторінку.
- **<button>**: Створює кнопку для розкриття/згортання меню на мобільних пристроях.
- ** **: Контейнер, що містить елементи навігаційного меню.
- ** **: Контейнер для посилань меню.
- ** **: Визначає пункт меню, який містить випадаюче меню.
- **<a class="nav-link dropdown-toggle" data-bs-toggle="dropdown">**: Посилання, що викликає випадаюче меню.
- ** **: Випадаюче меню, що містить посилання на сторінки.
- **<a class="nav-item nav-link">**: Посилання на сторінки без випадаючого меню.
- **<a class="btn btn-primary">**: Кнопка для доступу до адмін-панелі.

Цей код використовує HTML, CSS та фреймворк Bootstrap [35] для створення респонсивної та стильної навігаційної панелі.

Далі опишемо верстку каруселі (carousel) головної сторінки index.html (рис. 3.10)

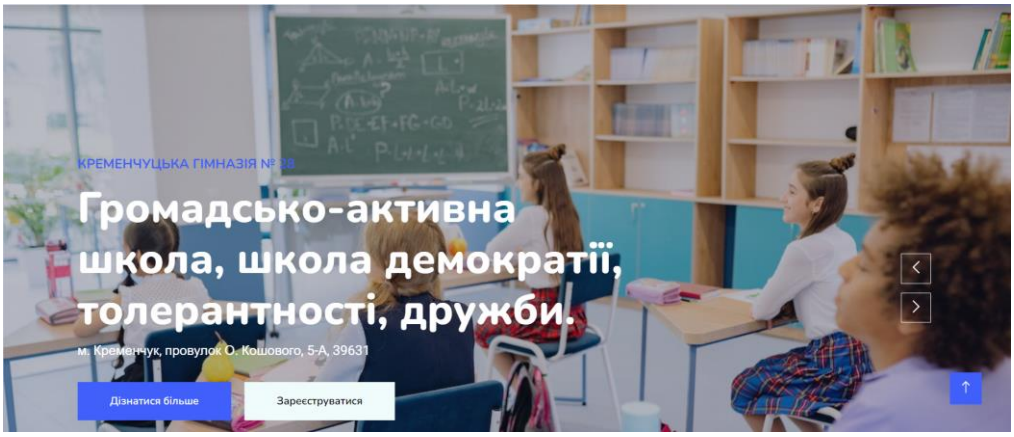


Рисунок 3.10 – Верстка каруселі (carousel)

Деталі верстки наступні:

1. ** **: Контейнер для каруселі, що розтягується на всю ширину екрану і має нульові відступи.
2. ** **: Контейнер каруселі, що використовує фреймворк Owl Carousel і має позицію relative.
3. ** **: Елемент каруселі, що має позицію relative.
4. **<img class="img-fluid" src="{% static 'img/carousel-1.jpg' %}" alt="">**: Зображення, що відображається в елементі каруселі.
5. ** **: Контейнер, що розміщується абсолютно внутрішньо елемента каруселі і має позицію absolute.
6. ** **: Контейнер внутрішнього контенту каруселі.
7. ** **: Рядок, що містить вміст контейнера.
8. ** **: Колонка, яка визначає ширину контенту в залежності від розміру екрану.
9. **<h5 class="text-primary text-uppercase mb-3 animated slideInDown">Кременчуцька гімназія № 28 </h5>**: Заголовок, що відображає назву гімназії.

10. **<h1 class="display-3 text-white animated slideInDown">**м. Кременчук, провулок О. Кошового, 5-А, 39631**</h1>**: Великий заголовок, що відображає адресу гімназії.
11. ** **Громадсько-активна школа, школа демократії, толерантності, дружби.** **: Параграф, що містить опис гімназії.
12. ****Зареєструватися****: Посилання на реєстрацію, яке представлено як кнопка зі світлим фоном.

Далі створимо і опишемо блок з сервісами гімназії (рис. 3.11)

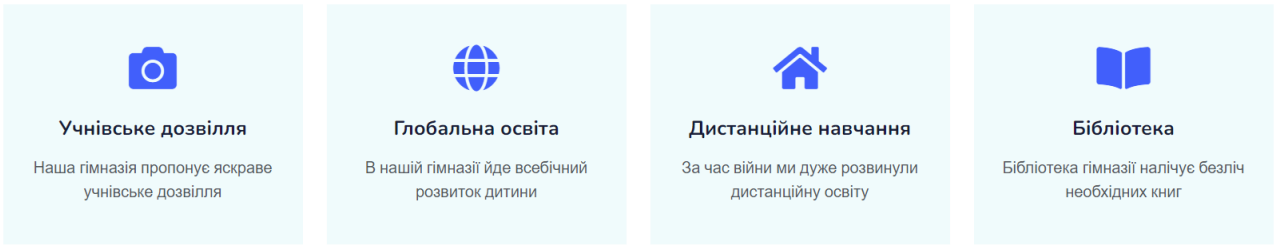


Рисунок 3.11 – Верстка блоку з сервісами гімназії

Основні елементи верстки включають:

- **.container-xxl**: Контейнер, який обмежує ширину блоку і створює простір по краях.
- **.container**: Внутрішній контейнер, який містить рядки та колонки.
- **.row**: Рядок, який містить колонки з послугами.
- **.col-lg-3**, **.col-sm-6**: Колонки, які визначають ширину блоків залежно від розміру екрану.
- **.wow**: Клас для анімації елементів з додатковими налаштуваннями.
- **.fadeInUp**: Ефект появи елементу знизу при прокручуванні сторінки.
- **.data-wow-delay**: Затримка анімації для кожного елементу.

Кожен елемент сервісу (**.service-item**) містить:

- **.p-4**: Внутрішній блок з відступами.
- **.fa-3x**: Розмір іконки Font Awesome.

- **.mb-4**: Відступ знизу для іконки.
- **.mb-3**: Відступ знизу для заголовка.
- **<h5>**: Заголовок послуги.
- **<p>**: Опис послуги.

Загальний результат - блок з чотирма послугами, кожна з яких має свою іконку, заголовок і опис. Блоки анімуються знизу при прокручуванні сторінки.

Наступний блок "Про гімназію" (рис. 3.12)

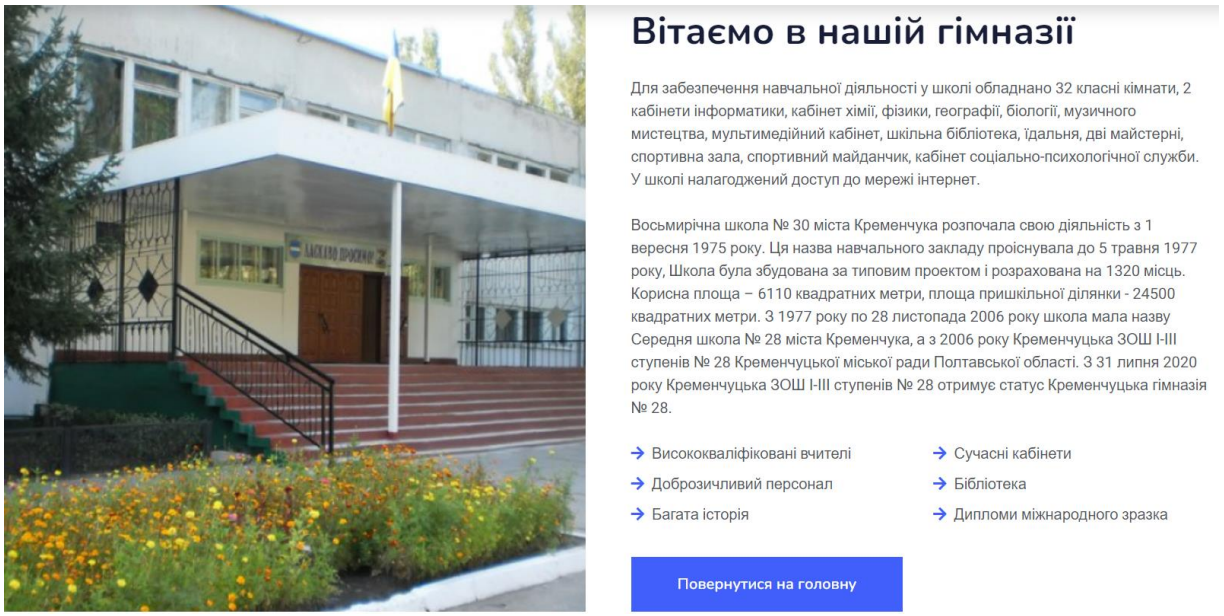


Рисунок 3.12 – Верстка блоку "Про гімназію"

Опис верстки коду блоку "Про гімназію":

1. ** **: Контейнер, який охоплює весь вміст розділу "Про гімназію". Має вертикальний відступ зверху і знизу (**py-5**).
2. ** **: Внутрішній контейнер, який обмежує ширину вмісту і розташовує його по центру.
3. ** **: Рядок, який містить два стовпці (**col-lg-6**), розташовані горизонтально із відступом між ними (**g-5**).
4. ** **: Лівий стовпець, який займає половину ширини на екранах планшетів і більших (**col-lg-6**). Містить зображення гімназії

(**<img>**) з ефектом анімації (**wow fadeInUp**) і затримкою (**data-wow-delay**), а також має мінімальну висоту 400 пікселів (**style="min-height: 400px;"**).

5. ** **: Правий стовпець, який також займає половину ширини на екранах планшетів і більших (**col-lg-6**). Містить заголовки (**<h6>**, **<h1>**) і параграфи (**<p>**), які описують гімназію. Також містить список елементів (** **) і кнопку "Повернутися на головну" (**<a class="btn btn-primary py-3 px-5 mt-2" href="/">**).

Цей код використовує фреймворк "Wow.js" для створення анімаційних ефектів під час завантаження сторінки.

Далі створимо блок з вчителями гімназії, де кожен вчитель представлений своїм зображенням, інформацією про посаду, предмет та прізвище, а також кнопками для соціальних мереж (рис. 3.13).

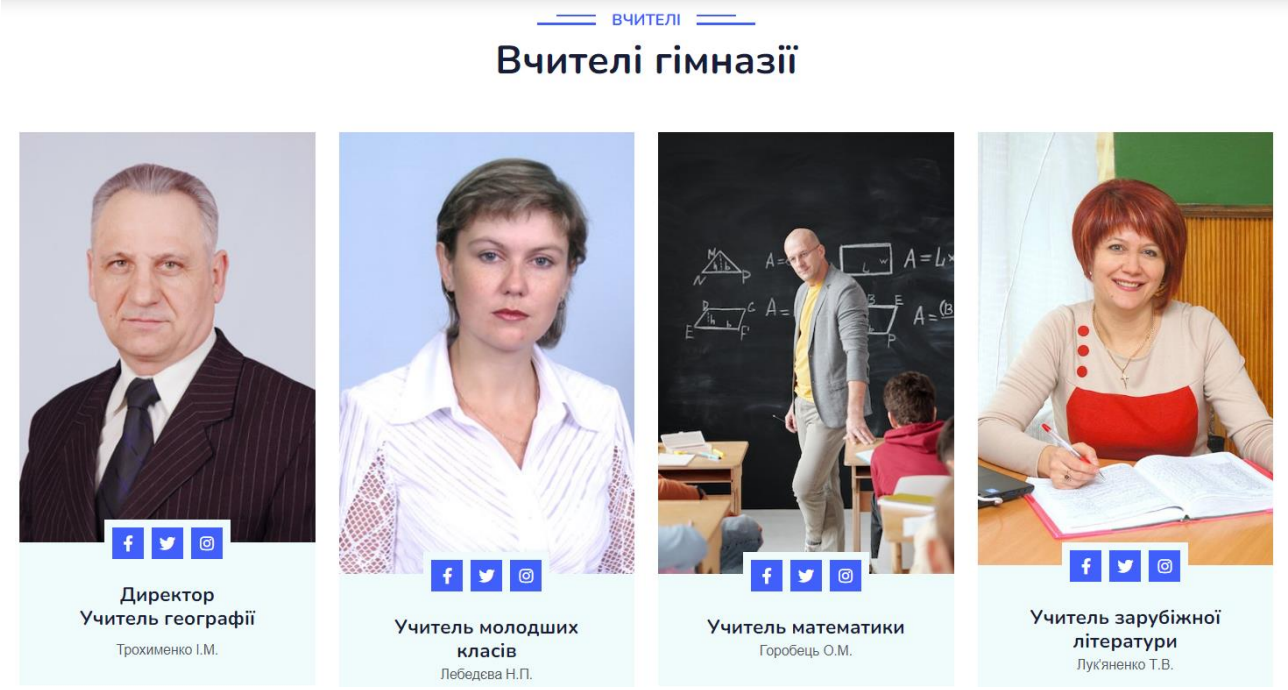


Рисунок 3.13 – Верстка блоку з вчителями гімназії

Опис верстки по коду:

1. ** **: Контейнер, який охоплює вміст сторінки. Встановлений великий розмір контейнера (**container-xxl**) та відступи по вертикалі (**py-5**).

2. Всередині контейнера ми маємо ще один контейнер (** **), який обмежує ширину вмісту та додає внутрішній відступ.
3. ** **: Контейнер з текстом, який вирівнюється по центру та анімується з ефектом з'явлення. Атрибут **data-wow-delay** встановлює затримку для анімації.
4. **<h6 class="section-title bg-white text-center text-primary px-3">Вчителі</h6>**: Заголовок розділу, який має фоновий колір (**bg-white**), текстовий колір (**text-primary**) та внутрішній відступи (**px-3**).
5. **<h1 class="mb-5">Вчителі гімназії</h1>**: Великий заголовок, який має внутрішній відступ знизу (**mb-5**).
6. ** **: Рядок, що містить елементи вчителів. Кожен елемент розміщений у відокремленому стовпчику (**col-lg-3 col-md-6**), який адаптивний для різних розмірів екрану. Відстань між елементами встановлена за допомогою класу **g-4**.
7. Для кожного вчителя створюється ** **. Елемент має світлий фон (**bg-light**).
8. Зображення вчителя представлене **<img>** з класом **img-fluid**, що забезпечує його адаптивність.
9. Для соціальних мереж вчителя використовуються **<a>** з класами **btn**, **btn-sm-square**, **btn-primary** та відповідними класами для іконок соціальних мереж (**fab fa-facebook-f**, **fab fa-twitter**, **fab fa-instagram**).
10. Інформація про вчителя міститься в ** **. Включаться назва посади вчителя **<h5 class="mb-0">**, назва предмета, який він викладає **<h5>**, і прізвище вчителя **<small>**.
11. Цей шаблон повторюється для кожного вчителя, змінюючи відповідні дані, такі як зображення, інформація про посаду, предмет та прізвище.

12. Загальний ефект анімації (**wow fadeInUp**) та затримка (**data-wow-delay**) використовуються для плавного з'явлення елементів під час прокрутки сторінки.

Далі опишемо верстку підвалу (footer) головної сторінки index.html (рис. 3.14)

Рисунок 3.14 – Верстка підвалу сайту (footer)

Підвал сайту (footer) має наступну структуру:

1. **<footer class="bg-dark text-center text-white">**: Елемент **<footer>** встановлює стилі для підвалу, включаючи темний фон та білий текст. Текст розташований по центру.
2. ** **: Контейнер з класом **.container** встановлює відступи для внутрішнього вмісту підвалу.
3. **<section class="mb-4">**: Розділ **<section>** встановлює відступи між кнопками соціальних мереж.
4. Кнопки соціальних мереж: Використовуються **<a>** елементи з класом **.btn**, **.btn-outline-light**, **.btn-floating** та класами іконок Font Awesome (наприклад, ****), щоб створити кнопки для посилань на соціальні мережі.
5. ** **: Другий ** ** елемент встановлює відступи, стиль фону та внутрішній вміст нижньої частини підвалу. Відображає копірайт і посилання на джерело.

Цей код використовує HTML, CSS та іконки Font Awesome [36] для створення підвалу з соціальними мережами та копірайтом.

3.5 Програмування серверної частини вебсайту. Backend-частина

Backend-частина вебсайту є важливою програмною складовою, яка виконується на сервері і відповідає за обробку запитів, взаємодію з базою даних та забезпечення логіки та функціональності системи. Основні завдання backend-частини включають:

1. Обробка запитів: Backend-частина отримує запити від фронтенд-частини (клієнта) і виконує необхідні дії відповідно до цих запитів. Вона може обробляти запити на створення, читання, оновлення та видалення даних, а також виконувати різні операції, які потрібні для функціонування вебсайту.
2. Взаємодія з базою даних: Backend-частина виконує запити до бази даних для отримання, збереження та оновлення інформації. Вона виконує операції, такі як вибірка даних з бази, додавання нових записів, оновлення існуючих даних та видалення записів з бази даних.
3. Логіка та функціональність: Backend-частина вебсайту містить бізнес-логіку та реалізує різні функціональні можливості системи. Це включає перевірку прав доступу користувачів, обробку форм, валідацію даних, генерацію відповідей та виконання різних операцій, необхідних для роботи вебсайту.
4. Комунікація з фронтенд-частиною: Backend-частина забезпечує комунікацію з фронтенд-частиною (клієнтом) через мережу Інтернет. Вона передає дані, отримані від фронтенду, обробляє їх і повертає відповіді, які відображаються на вебсайті.

Опишемо класи файлу **models.py** веборієнтованої інформаційної системи .

Клас **PersonalManager** є спеціальним менеджером користувачів, який використовується для створення різних типів користувачів в системі.

У цьому класі є три методи: **create_user**, **create_teacher** і **create_pupil**, які використовуються для створення користувачів з різними ролями (вчителі та учні).

1. Метод **create_user** викликається для створення звичайного користувача. Він приймає обов'язковий параметр **email** і необов'язковий параметр **password** для встановлення електронної адреси користувача і пароля відповідно. Якщо параметр **email** не вказаний, викидається виключення **ValueError**. Після перевірки, електронна адреса нормалізується (перетворюється в нижній регістр), і створюється новий користувач з вказаними параметрами. Пароль встановлюється за допомогою методу **set_password**, і користувач зберігається в базі даних.
2. Метод **create_teacher** викликається для створення користувача з роллю вчителя. Він приймає такі ж параметри, як метод **create_user**, але встановлює значення **is_teacher** в **True** у додаткових полях користувача. Потім він викликає метод **create_user**, передаючи йому всі вказані параметри, включаючи **is_teacher**, і повертає результат.
3. Метод **create_pupil** викликається для створення користувача з роллю учня. Він аналогічний до методу **create_teacher**, але встановлює значення **is_pupil** в **True** у додаткових полях користувача. Потім він викликає метод **create_user**, передаючи йому всі вказані параметри, включаючи **is_pupil**, і повертає результат.

Цей клас дозволяє зручно створювати користувачів з різними ролями, використовуючи спільний метод **create_user**, і встановлюючи відповідні значення для додаткових полів.

Клас **Personal** є моделлю користувача, який успадковує від абстрактного базового класу **AbstractBaseUser**. Цей клас визначає поля та функції, які використовуються для представлення користувачів в системі.

Основна логіка класу **Personal** включає наступне:

1. У класі оголошуються поля **is_pupil**, **is_teacher**, **is_staff**, **username**, **email** та **password**, які відповідають за збереження різних характеристик користувача, таких як роль (учень, вчитель), ідентифікатор, електронна

адреса та пароль. Ці поля використовуються для збереження даних користувача в базі даних.

2. Поле **USERNAME_FIELD** встановлює значення '**username**', що означає, що ідентифікатор користувача буде використовуватись як основне поле для автентифікації користувача.
3. Поле **objects** створює екземпляр класу **PersonalManager**, який відповідає за управління об'єктами цього класу, включаючи створення користувачів.
4. Метод **has_perm** перевіряє, чи має користувач задані дозволи (**perm**). У даному випадку, завжди повертається значення **True**, що означає, що користувач має всі необхідні дозволи.
5. Метод **has_module_perms** перевіряє, чи має користувач дозволи на доступ до заданого модулю (**app_label**). У даному випадку, завжди повертається значення **True**, що означає, що користувач має доступ до всіх модулів.

Цей клас використовується для представлення користувачів у системі і визначає їх характеристики та функції, пов'язані з автентифікацією та дозволами.

Далі в коді описані декілька класів, які моделюють певні об'єкти в системі в файлі ***models.py***.

```python
Olha Kozlovska
class Class(models.Model):
    name = models.CharField(max_length=255)


Olha Kozlovska
class Pupil(models.Model):
    personal = models.ForeignKey(to=Personal, on_delete=models.CASCADE, default=None)
    full_name = models.CharField(max_length=255)
    _class = models.ForeignKey(to=Class, on_delete=models.CASCADE)


Olha Kozlovska
class Lesson(models.Model):
    name = models.CharField(max_length=255)


Olha Kozlovska
class Schedule(models.Model):
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE)
    _class = models.ForeignKey(Class, on_delete=models.CASCADE)
    weekday = models.IntegerField()
    sequence = models.IntegerField()
```

Рисунок 3.15 – Деякі класи models.py

Опишемо ці класи:

1. Клас **Class**:
    - Має поле **name**, яке відображає назву класу.
    - Використовується для представлення класів, до яких належать учні та розклад занять.
2. Клас **Pupil**:
    - Має поля **personal**, **full_name** та **_class**.
    - **personal** є зовнішнім ключем, який посилається на об'єкт класу **Personal**, і вказує на особисті дані учня.
    - **full_name** відображає повне ім'я учня.
    - **_class** є зовнішнім ключем, який посилається на об'єкт класу **Class**, і вказує на клас, до якого належить учень.
3. Клас **Lesson**:
    - Має поле **name**, яке відображає назву уроку.

- Використовується для представлення уроків, які викладаються.

4. Клас **Schedule**:
    - Має поля **teacher**, **lesson**, **_class**, **weekday** та **sequence**.
    - **teacher** є зовнішнім ключем, який посилається на об'єкт класу **Teacher**, і вказує на вчителя, який проводить заняття.
    - **lesson** є зовнішнім ключем, який посилається на об'єкт класу **Lesson**, і вказує на урок, який проводиться.
    - **_class** є зовнішнім ключем, який посилається на об'єкт класу **Class**, і вказує на клас, для якого розклад створений.
    - **weekday** відображає день тижня, коли проводиться заняття.
    - **sequence** відображає послідовність занять у розкладі.

Ці класи використовуються для моделювання класів, учнів, уроків та розкладу занять в системі. Вони встановлюють відношення між цими об'єктами за допомогою зовнішніх ключів,

Далі опишемо функції-перегляди (views) файлу views.py.

1. **home(request)**: Ця функція-перегляд обробляє запити, які надходять на домашню сторінку. Вона виконує наступні дії:
    - Приймає об'єкт **request**, який містить інформацію про HTTP-запит.
    - Викликає функцію **render**, яка генерує відповідь з вказаним шаблоном **ui/index.html**.
    - Передає порожній словник **{}** як контекст шаблону, який міститиме дані, які можуть бути використані у шаблоні.
2. **about(request)**: Ця функція-перегляд обробляє запити, які надходять на сторінку "Про нас". Вона виконує наступні дії:
    - Приймає об'єкт **request**, який містить інформацію про HTTP-запит.
    - Викликає функцію **render**, яка генерує відповідь з вказаним шаблоном **ui/about.html**.

- Передає порожній словник **{}** як контекст шаблону, який міститиме дані, які можуть бути використані у шаблоні.

Ці функції-перегляди виконують рендеринг відповідних HTML-шаблонів та повертають згенеровану відповідь від сервера.

3. Функція **schedule(request)** в **views.py** відповідає за обробку запиту на сторінку розкладу. Основна логіка цієї функції наступна:

- Отримання списку всіх класів (**classes**) за допомогою запиту до моделі **Class**.
- Створення діапазонів для днів тижня (**weekdays**) і номерів пар (**sequences**).
- Ініціалізація порожнього словника **schedule_data**, який буде містити інформацію про розклад для кожного класу, дня тижня і пари.
- Проходження через кожен клас, день тижня і пару за допомогою вкладених циклів.
- У блоку спроби (**try**), спроба отримати об'єкт **Schedule** за допомогою методу **get()** з умовами належності до конкретного класу, дня тижня і пари.
- Якщо об'єкт **Schedule** знайдений, він додається до словника **schedule_data** за відповідними ключами.
- Якщо об'єкт **Schedule** не знайдений (виникає виключення **Schedule.DoesNotExist**), встановлюється значення **None** для відповідного ключа у словнику **schedule_data**.
- Створення словника **context**, який містить дані, що будуть передані в HTML-шаблон.
- Виклик функції **render**, яка генерує відповідь з вказаним шаблоном **ui/schedule.html** та переданим контекстом.
- Повернення згенерованої відповіді від сервера.

4. Функція **schedule_own(request)** в файлі **views.py** містить наступну логіку:

- Перевіряється, чи користувач, який зробив запит, є вчителем чи учнем. Якщо це вчитель, отримуються всі класи (**Class**) з бази даних за допомогою **Class.objects.all()**. Якщо це учень, отримується відповідний об'єкт учня

(**Pupil**) на основі **request.user** (поточного користувача) і отримується список класів, пов'язаних з цим учнем (**Class.objects.filter(id=pupil._class.id).all()**).

- Визначаються дні тижня (**weekdays**) та секвенції (**sequences**), які будуть використовуватись для створення розкладу.
- Ініціалізується словник **schedule_data**, який буде містити дані розкладу.
- Для кожного класу **_class** в списку класів:
    - Створюється ключ **_class.id** в словнику **schedule_data**.
    - Для кожного дня тижня (**weekday**) в списку днів тижня:
        - Створюється ключ **weekday** в словнику **schedule_data[_class.id]**.
        - Для кожної секвенції (**sequence**) в списку секвенцій:
            - Спроба отримати об'єкт розкладу (**Schedule**) на основі заданих параметрів, включаючи клас, день тижня, секвенцію. Якщо користувач є вчителем, то також виконується фільтрація розкладу за вчителем (**teacher=teacher**).
            - Якщо об'єкт розкладу знайдено, зберігається в словнику **schedule_data[_class.id][weekday][sequence]**. Якщо об'єкт розкладу не знайдено, зберігається значення **None**.
- Створюється словник **context**, який містить дані, що будуть передані у HTML-шаблон:
    - 'classes': список класів
    - 'weekdays': список днів тижня
    - 'sequences': список секвенцій
    - 'schedule_data': дані розкладу
- Викликається функція **render** аналогічно **home** та **about.**

5. Функція **get_working_days(teacher, _class)**, яка отримує викладача (**teacher**) та клас (**_class**) як параметри. Ця функція виконує наступні дії:
    - Створює порожній список **week_day**, який буде містити дні тижня, коли є розклад занять для вказаного викладача та класу.
    - Отримує всі розклади (**schedules**) за вказаного викладача та клас використовуючи фільтрацію на основі моделі **Schedule**.
    - Проходить через кожен розклад (**schedule**) і додає **weekday - 1** до списку **week_day**. Значення **weekday** представляє день тижня, де 0 - понеділок, 1 - вівторок, і так далі.
    - Створює порожній список **working_days**, який буде містити робочі дні (дати) на основі розкладу занять.
    - Встановлює поточну дату (**current_date**) як сьогоднішню дату.
    - Встановлює початкову дату (**start_date**) як початок навчального року (1 вересня) за вказаною датою.
    - Встановлює кінцеву дату (**end_date**) як поточну дату.
    - Створює об'єкт **delta**, який представляє різницю між датами в один день.
    - Виконує цикл, який триває, поки **end_date** більше або дорівнює **start_date**.
    - Перевіряє, чи поточний день тижня (**end_date.weekday()**) знаходиться в списку **week_day** (дні, коли є розклад занять). Якщо так, то додає дату у форматі "%Y-%m-%d" до списку **working_days**.
    - Зменшує **end_date** на один день використовуючи об'єкт **delta**.
    - Повертає список **working_days**, який містить робочі дні (дати) згідно з розкладом занять для вказаного викладача та класу.

Ця функція дозволяє отримати список робочих днів на основі розкладу занять для вказаного викладача та класу. Для цього вона перебирає всі розклади занять для вказаних параметрів, визначає дні тижня, коли є заняття, і потім

створює список робочих днів з урахуванням поточної дати і початку навчального року.

6. Функція **get_page(request)**, яка обробляє запити для отримання сторінки з даними журналу. Вона виконує наступні дії:

- Отримання параметрів запиту:
  - **page_number**: Номер сторінки, який передається в запиті.
  - **schedule_id**: Ідентифікатор розкладу, який передається в запиті.
- Запит до бази даних:
  - Фільтрує об'єкт **Schedule** за ідентифікатором **schedule_id** та отримує перший результат.
  - Фільтрує об'єкти **Pupil** за **_class**, яке відповідає **_class** з об'єкта **schedule**.
  - Викликає функцію **get_working_days** з аргументами **schedule.teacher** та **schedule._class**, яка повертає список робочих днів.
  - Створює об'єкт **Paginator** зі списком **all_working_days** та кількістю об'єктів на одну сторінку (10).
- Обробка сторінок та об'єктів:
  - Виконує перевірку на правильність переданого номеру сторінки.
  - Вибирає поточну сторінку з пагінатора або першу сторінку, якщо переданий номер некоректний або порожній.
  - Отримує список робочих днів на поточній сторінці.
  - Визначає загальну кількість сторінок у пагінаторі.
  - Створює список **journal**, який містить дані з моделі **Journal** для учнів, що належать до класу **pupils** та з датами зі списку **working_days**.
- Повернення відповіді:
  - Створює словник **response_data**, який містить дані для передачі у відповідь.

- Повертає відповідь у форматі JSON (**JsonResponse**) зі словником **response_data**.

Далі коротко опишемо ті функції, що залишилися у файлі **views.py**, які виконують наступні дії:

7. **journal_page(request, schedule_id)**: Відображає сторінку з журналом. Отримує ідентифікатор розкладу (**schedule_id**) та визначає тип користувача (вчитель або учень) на основі власностей користувача. Повертає відповідний шаблон **journal_page.html** з контекстом.
8. **add_mark(request, schedule_id)**: Додає нову оцінку до журналу. Отримує ідентифікатор учня (**pupil_id**), оцінку (**mark**) та дату (**date**) з POST-запиту. Створює новий об'єкт **Journal** з вказаними даними і повертає перенаправлення на сторінку журналу з ідентифікатором розкладу.
9. **edit_mark(request, schedule_id)**: Редагує існуючу оцінку в журналі. Отримує ідентифікатор учня (**pupil_id**), нову оцінку (**mark**) та дату (**date**) з POST-запиту. Оновлює відповідний запис у таблиці **Journal** з вказаними даними і повертає перенаправлення на сторінку журналу з ідентифікатором розкладу.
10. **delete_mark(request, schedule_id)**: Видаляє оцінку з журналу. Отримує ідентифікатор учня (**pupil_id**) та дату (**date**) з POST-запиту. Видаляє відповідний запис у таблиці **Journal** і повертає перенаправлення на сторінку журналу з ідентифікатором розкладу.
11. **sign_in(request)**: Відображає сторінку входу (логіну). Повертає шаблон **login.html**.
12. **auth(request)**: Аутентифікація користувача. Отримує ім'я користувача (**username**) та пароль (**password**) з POST-запиту. Перевіряє введені дані та, якщо користувач успішно аутентифікований.
13. **registration(request)**: Реєстрація нового користувача. Отримує електронну пошту (**email**), пароль (**password**), повне ім'я (**full_name**), ім'я

користувача (**username**), тип користувача (**user_type**) та ідентифікатор класу (**class_id**) з POST-запиту.

- Якщо тип користувача є "вчитель", створюється об'єкт **Personal** з властивістю **is_teacher=True** за допомогою методу **create_teacher** і об'єкт **Teacher** з вказаними даними.
- Якщо тип користувача є "учень", створюється об'єкт **Personal** з властивістю **is_pupil=True** за допомогою методу **create_pupil**, а також об'єкт **Pupil** з вказаними даними і зв'язком з класом.
- Після реєстрації користувача виконується аутентифікація за допомогою методу **authenticate**. Якщо користувач успішно аутентифікований, виконується вхід у систему і перенаправляється на домашню сторінку. У протилежному випадку відображається сторінка реєстрації знову.

14. **log_out(request)**: Вихід з системи. Виконує вихід користувача за допомогою методу **logout** і перенаправляє на домашню сторінку.

Файл urls.py містить маршрутизацію URL-адрес на відповідні функції у views.py (рис. 3.17).

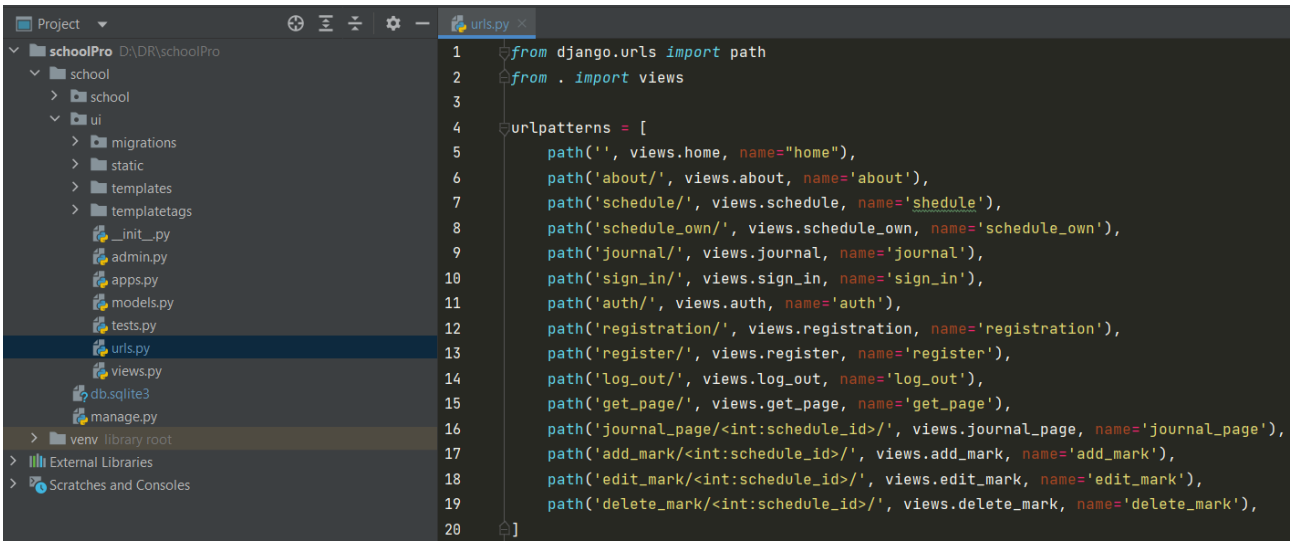


Рисунок 3.16 – Маршрутизація URL-адрес в urls.py

Опишемо кожен зазначений маршрут:

- '': Маршрут до головної сторінки (**home**), відображається при порожньому URL-адресі.
- '**about/**': Маршрут до сторінки "Про нас" (**about**).
- '**schedule/**': Маршрут до сторінки розкладу (**schedule**).
- '**schedule_own/**': Маршрут до сторінки власного розкладу (**schedule_own**).
- '**journal/**': Маршрут до сторінки журналу (**journal**).
- '**sign_in/**': Маршрут до сторінки входу в систему (**sign_in**).
- '**auth/**': Маршрут для аутентифікації користувача (**auth**).
- '**registration/**': Маршрут до сторінки реєстрації (**registration**).
- '**register/**': Маршрут до сторінки реєстрації нового користувача (**register**).
- '**log_out/**': Маршрут для виходу з системи (**log_out**).
- '**get_page/**': Маршрут для отримання сторінки з даними (**get_page**).
- '**journal_page/<int:schedule_id>/**': Маршрут до сторінки журналу з вказаним ідентифікатором розкладу (**journal_page**).
- '**add_mark/<int:schedule_id>/**': Маршрут для додавання оцінки в журнал з вказаним ідентифікатором розкладу (**add_mark**).
- '**edit_mark/<int:schedule_id>/**': Маршрут для редагування оцінки в журналі з вказаним ідентифікатором розкладу (**edit_mark**).
- '**delete_mark/<int:schedule_id>/**': Маршрут для видалення оцінки з журналу з вказаним ідентифікатором розкладу (**delete_mark**).

Кожен маршрут вказує на відповідну функцію у **views.py**, яка буде виконана при отриманні запиту за відповідним URL-адресом.

Лістинг коду файлів models.py, views.py, urls.py наведений в **Додатку А**, а весь код програми завантажено на GitHub [37].

3.6 Тестування вебсайту

Під час тестування проводилося ручне тестування всіх сторінок сайту з метою перевірки їх коректності та відповідності вимогам. Виконано наступні дії:

1. Головна сторінка (home). Перевірено, що головна сторінка відображається коректно і містить необхідну інформацію (рис. 3.17)

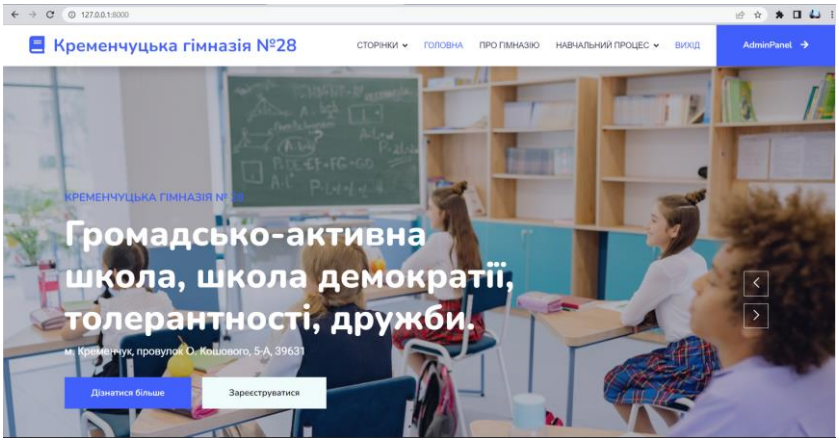


Рисунок 3.17 – Тестування відображення головної сторінки

2. Сторінка "Про гімназію" (about). Перевірено, що сторінка "Про гімназію" відображається правильно і містить відповідну інформацію про сайт або компанію (рис.3.18).

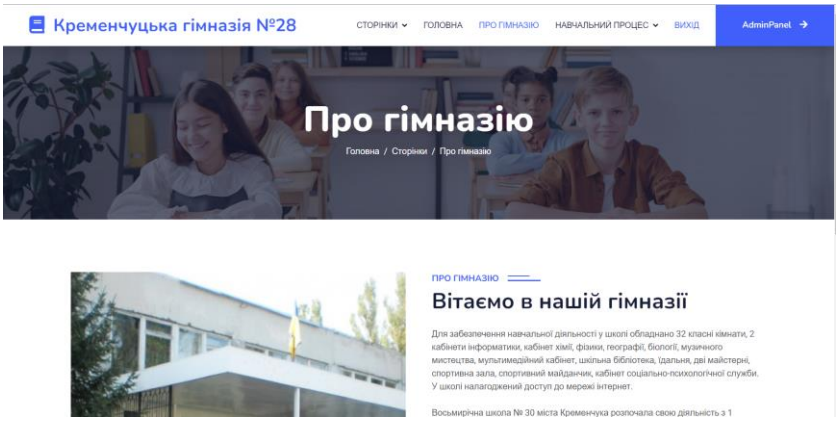


Рисунок 3.18 – Тестування відображення головної сторінки

3. Сторінка розкладу (schedule). Перевірено, що сторінка розкладу відображається інформативно та правильно відображає розклад з відповідними даними (рис.3.19).

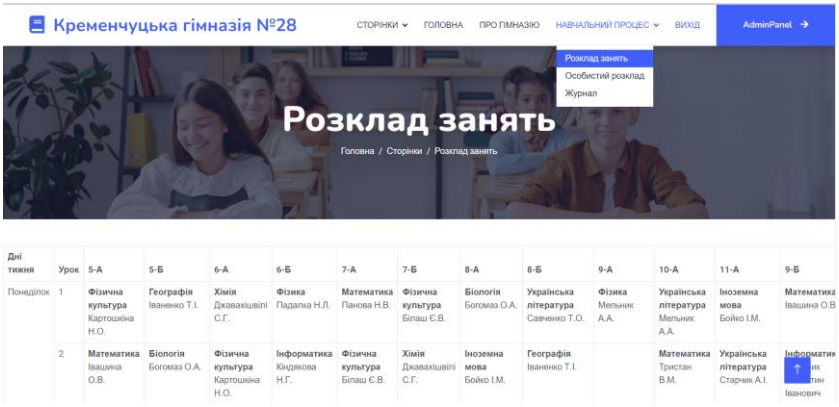


Рисунок 3.19 – Тестування відображення розкладу

4. Сторінка власного розкладу (schedule_own). Перевірено, що сторінка власного розкладу відображається інформативно та правильно відображає розклад з відповідними даними користувача (рис. 3.20).

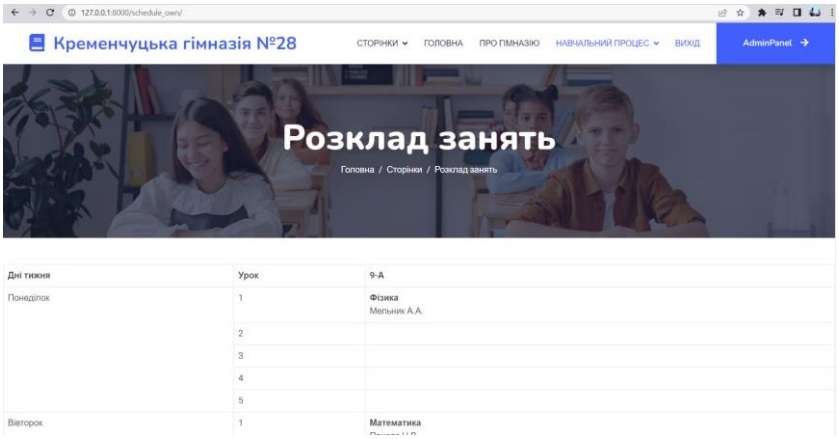


Рисунок 3.20 – Тестування відображення власного розкладу

5. Сторінка входу в систему (sign_in). Перевірено, що сторінка входу в систему відображається коректно і містить відповідні поля для введення даних користувача (рис. 3.21).

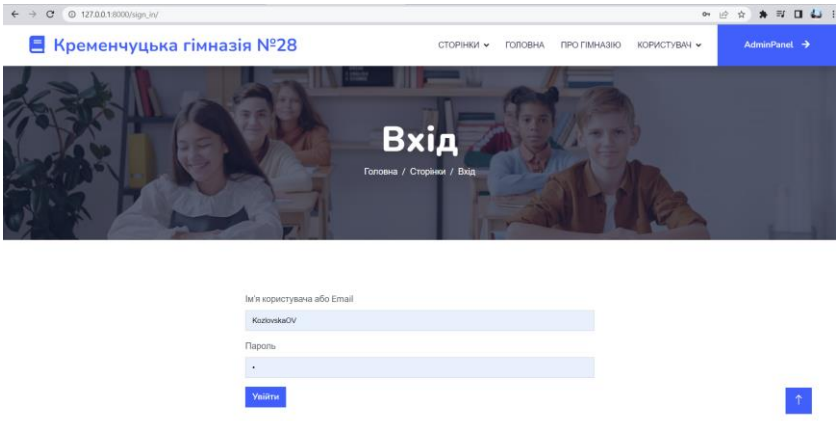


Рисунок 3.21 – Тестування відображення сторінки входу в систему

6. Сторінка реєстрації (register). Перевірено, що сторінка реєстрації відображається правильно і містить відповідні поля для введення даних користувача (3.22).

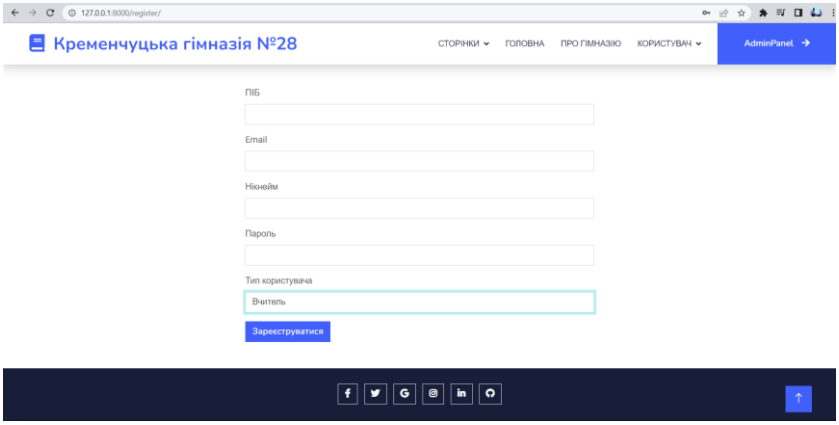


Рисунок 3.22 – Тестування відображення сторінки реєстрації

7. Сторінка журналу (journal). Перевірено, що сторінка журналу відображається інформативно та правильно відображає журнал з відповідними даними (рис. 3.23).

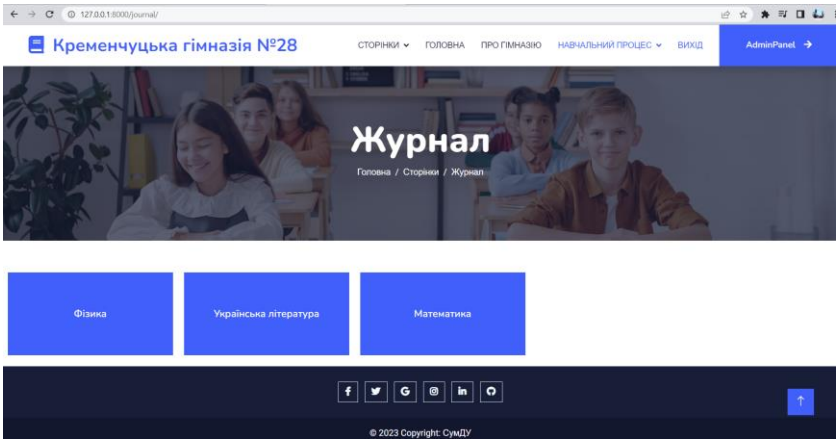


Рисунок 3.23 – Тестування відображення сторінки журналу

8. Сторінка журналу з вказаним ідентифікатором розкладу (journal_page). Перевірено, що сторінка журналу з вказаним ідентифікатором розкладу відображається правильно і містить відповідні дані журналу (3.24).

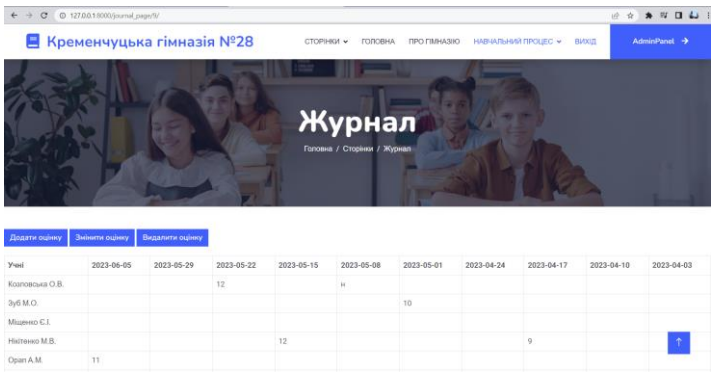


Рисунок 3.24 – Тестування відображення журналу з ідентифікатором розкладу

9. Додавання оцінки (add_mark). Перевірено, що після введення необхідних даних (ідентифікатор учня, оцінка, дата) і відправлення форми, оцінка успішно додається до журналу (рис. 3.25).

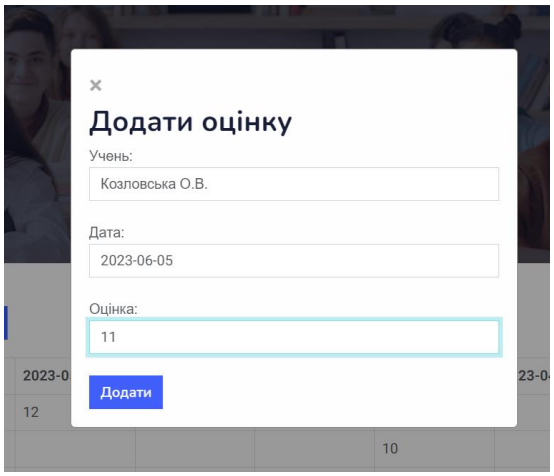


Рисунок 3.24 – Тестування функціоналу додавання оцінки

10. Редагування оцінки (edit_mark). Перевірено, що після введення необхідних даних (ідентифікатор учня, оцінка, дата) і відправлення форми, відповідна оцінка у журналі успішно оновлюється (рис.3.25).

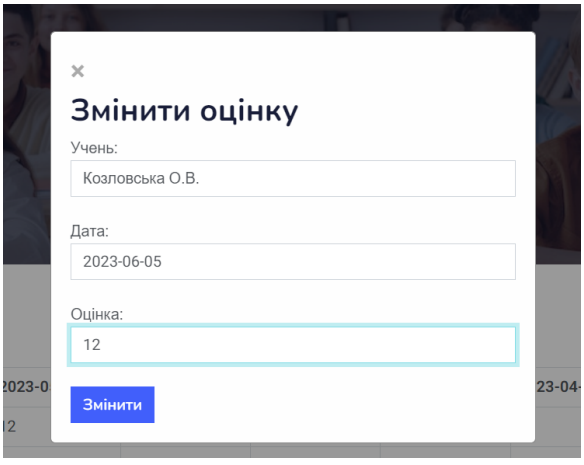


Рисунок 3.25 – Тестування функціоналу редагування оцінки

11. Видалення оцінки (delete_mark). Перевірено, що після введення необхідних даних (ідентифікатор учня, дата) і відправлення форми, відповідна оцінка з журналу успішно видаляється (рис. 3.26).

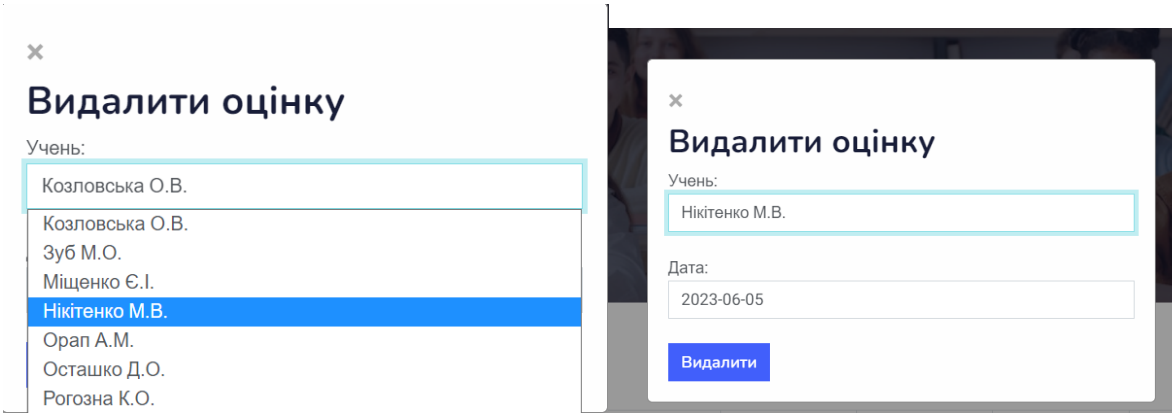


Рисунок 3.26 – Тестування функціоналу видалення оцінки

12. Вихід з системи (log_out). Перевірено, що після натискання кнопки виходу з системи користувач успішно виходить і перенаправляється на головну сторінку (рис. 3.27).

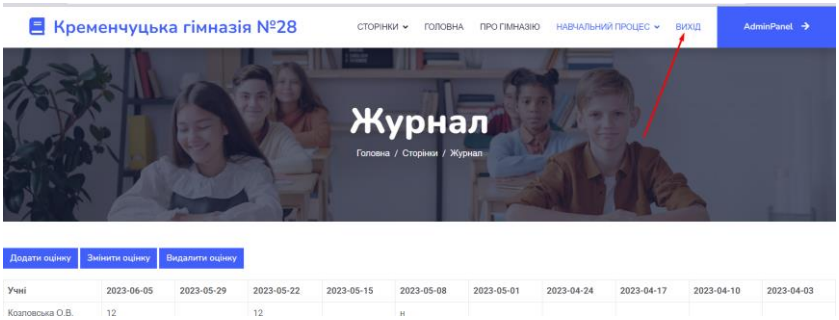


Рисунок 3.26 – Тестування функціоналу виходу з системи

У процесі ручного тестування перевірялася коректність відображення сторінок, правильність роботи форм та взаємодії з базою даних. Також перевірялася зручність використання сайту, введення даних та взаємодія з функціоналом. У разі виявлення помилок або некоректностей відбувалася їх виправлення та повторне тестування.

ВИСНОВКИ

В дипломній роботі проведений аналіз предметної області, визначена актуальність проблеми та проведений огляд існуючих аналогів. Поставлено задачі для подальшого розроблення веборієнтованої інформаційної системи організації шкільного освітнього процесу.

У другому розділі роботи проведений аналіз можливих рішень, включаючи вибір мови програмування, системи керування базами даних та фреймворку для реалізації системи. Для кожного з аспектів враховані важливі фактори, такі як продуктивність, зручність використання, підтримка спільнотою та інші.

У третьому розділі описано процес проектування та розробки веборієнтованої інформаційної системи організації шкільного освітнього процесу. Застосовані методи та засоби об'єктно-орієнтованого аналізу і проектування, такі як діаграми варіантів використання, послідовності, діяльності та класів. Також описано дизайн вебсайту та архітектуру системи, включаючи розподілення функціональності між frontend- та backend-частинами.

У розділі "Програмування серверної частини вебсайту" розглянуто деталі реалізації backend-частини системи. Описано логіку роботи класів, зокрема PersonalManager, Personal, Class, Pupil, Lesson і Schedule. Також наведено короткий опис функцій у файлі views.py, які відповідають за обробку запитів та взаємодію з базою даних.

У розділі "Тестування вебсайту" описано процес ручного тестування всіх сторінок сайту. Перевірено коректність відображення, роботу форм та взаємодію базою даних.

Система, розроблена в цій дипломній роботі буде корисною як для вчителів, так і для учнів. Вчителям надається зручна платформа для управління розкладом уроків, ведення журналу оцінок та взаємодії з учнями. Вони зможуть легко створювати розклади, додавати, редагувати та видаляти оцінки, а також відстежувати академічний прогрес кожного учня.

Учні також отримують багато переваг від цієї системи. Вони зможуть переглядати свій розклад, бачити оцінки за кожний урок і відстежувати свій академічний прогрес. Вчасно отримувати інформацію про домашні завдання та події, пов'язані з навчанням. Взаємодія з вчителями стає більш зручною, оскільки учні можуть звертатися до них через систему, ставити запитання та отримувати додаткову допомогу.

Загалом, ця робота сприяє автоматизації процесів навчання і полегшує взаємодію між вчителями і учнями. Вона допомагає вчителям більш ефективно організувати свою роботу, а учням отримати доступ до необхідної інформації та контролювати свої академічні досягнення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Classroom Management Tools & Resources - Google for Education. *Google for Education*. URL: https://classroom.google.com/ (дата звернення: 15.04.2023).
2. Єдина Школа. *Єдина Школа*. URL: https://eschool-ua.com/ (дата звернення: 16.04.2023).
3. Moodle - Open-source learning platform | *Moodle.org*. URL: https://moodle.org/?lang=uk (дата звернення:17.04.2023).
4. Сумський державний університет. *eLearning@SumyStateUniversity*. URL: https://dl.sumdu.edu.ua/uk/ (дата звернення: 18.04.2023).
5. Welcome to Python.org. *Python.org*. URL: https://www.python.org/ (дата звернення: 19.06.2023).
6. Підручник з Python. *Python documentation*. URL: https://docs.python.org/uk/3/tutorial/index.html (дата звернення: 21.04.2023).
7. Сучасний підручник з JavaScript. *JavaScript*. URL: https://uk.javascript.info/ (дата звернення: 23.04.2023).
8. ECMAScript 6: New Features: Overview and Comparison. *ECMAScript 6*. URL: http://es6-features.org/#Constants (дата звернення: 25.04.2023).
9. Ruby Programming Language. *Ruby*. URL: https://www.ruby-lang.org/en/ (дата звернення: 27.04.2023).
10. Частина 1. Моя перша програма на Ruby. *Codeguida*. URL: https://codeguida.com/post/219 (дата звернення: 29.04.2023).
11. PHP: Hypertext Preprocessor. *PHP*. URL: https://www.php.net/ (дата звернення: 01.05.2023).
12. Java. *java.com*. URL: https://www.java.com/ (дата звернення: 04.06.2023).
13. Багатопотоковість на Java - Підручник із прикладами - Інший. *Огляди 2023*. URL: https://uk.myservername.com/multithreading-java-tutorial-with-examples (дата звернення: 02.05.2023).
14. SQLite Home Page. *SQLite*. URL: https://www.sqlite.org/index.html (дата звернення: 04.05.2023).
15. База даних MySQL. *MySQL*. URL: https://www.mysql.com/ (дата звернення: 07.05.2023).
16. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (дата звернення: 09.05.2023).

17. POSTGRESQL. *Створення і розробка інтернет-магазину під ключ - Brander*. URL: https://brander.ua/technologies/postgresql (дата звернення: 10.05.2023).
18. MongoDB: The Developer Data Platform. *MongoDB*. URL: https://www.mongodb.com/ (дата звернення: 12.05.2023).
19. Alex cultofdigits. MongoDB Aggregation Framework, 2015. *YouTube*. URL: https://www.youtube.com/watch?v=rnenE1NT33E (дата звернення: 14.05.2023).
20. Django. *Django Project*. URL: https://www.djangoproject.com/ (дата звернення: 15.05.2023).
21. Django. *Django DOCS*. URL: https://docs.djangoproject.com/en/4.2/ (дата звернення: 16.05.2023).
22. Welcome to Flask – Flask Documentation (2.3.x). *Welcome to Flask*. URL: https://flask.palletsprojects.com/en/2.3.x/ (дата звернення: 17.05.2023).
23. Welcome to Pyramid, a Python Web Framework. *Pyramid Framework*. URL: https://trypyramid.com/ (дата звернення: 18.05.2023).
24. CherryPy – A Minimalist Python Web Framework – CherryPy 18.8.1. documentation. URL: https://docs.cherrypy.dev/en/latest/ (дата звернення: 19.05.2023).
25. UML. *KDE Documentation*.URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html (дата звернення: 21.05.2023).
26. Yura Shtyfurak. UML 3. Діаграма варіантів використання, 2020. *YouTube*. URL: https://www.youtube.com/watch?v=7lVeW7BAB2g (дата звернення: 22.05.2023).
27. Застосування UML (частина 2). Діаграма послідовності - Sequence Diagram :: Державний університет телекомунікацій. *Головна :: Державний університет телекомунікацій*. URL: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 24.05.2023).
28. Учасники проектів Вікімедіа. Діаграма діяльності – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/ (дата звернення: 25.05.2023).
29. Застосування UML (частина 3). Діаграма класів - Class Diagram :: Державний університет телекомунікацій. *Головна :: Державний*

університет телекомунікацій. URL: https://dut.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 26.05.2023).

30. Figma: the collaborative interface design tool. *Figma*. URL: https://www.figma.com/ (дата звернення: 27.05.2023).
31. Figma для початківців - огляд популярного інструменту для дизайнерів. *IT курси в Одесі з працевлаштуванням | Академія IT STEP*. URL: https://od.itstep.org/blog_3/figma-is-a-basic-tool-for-designers (дата звернення: 29.05.2023).
32. Теорія кольору і значення кольору в дизайні - Блог Depositphotos. *Блог Depositphotos*. URL: https://blog.depositphotos.com/ua/teoriya-koloru-i-znachennya-koloru-v-dyzajni.html (дата звернення: 30.05.2023).
33. Архітектура веб-додатків - як вибрати?: стаття з блогу IT-школи Hillel. *Корисні матеріали: Статті та новини IT-індустрії | Комп'ютерна школа Hillel*. URL: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення: 01.06.2023).
34. HTML і CSS довідник українською. *Html CSS довідник*. URL: https://html-css.co.ua/ (дата звернення: 02.06.2023).
35. Bootstrap. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/ (дата звернення: 03.06.2023).
36. Font Awesome. *Font Awesome*. URL: https://fontawesome.com/ (дата звернення: 04.06.2023).
37. GitLab. *GitLab*. URL: https://gitlab.com/oliakozlovska/school (дата звернення: 07.06.2023)

ДОДАТКИ

Додаток А. Лістинг models.py, views.py, urls.py

models.py

```
import datetime

from django.contrib.auth.base_user import AbstractBaseUser, BaseUserManager
from django.contrib.auth.models import AbstractUser, Group, Permission
from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage
from django.db import models


class PersonalManager(BaseUserManager):
    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('The Email field must be set')

        email = self.normalize_email(email)
        user = self.model(email=email, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_teacher(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_teacher', True)
        return self.create_user(email, password, **extra_fields)

    def create_pupil(self, email, password=None, **extra_fields):
        extra_fields.setdefault('is_pupil', True)
        return self.create_user(email, password, **extra_fields)


class Personal(AbstractBaseUser):
    is_pupil = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=False)
    username = models.CharField(max_length=150, unique=True)
    email = models.EmailField(max_length=150)
    password = models.CharField(max_length=128)
    USERNAME_FIELD = 'username'
    objects = PersonalManager()

    def has_perm(self, perm, obj=None):
        return True

    def has_module_perms(self, app_label):
        return True


class Teacher(models.Model):
    personal = models.ForeignKey(to=Personal, on_delete=models.CASCADE,
default=None)
    full_name = models.CharField(max_length=255)


class Class(models.Model):
    name = models.CharField(max_length=255)


class Pupil(models.Model):
```

```
    personal = models.ForeignKey(to=Personal, on_delete=models.CASCADE,
default=None)
    full_name = models.CharField(max_length=255)
    _class = models.ForeignKey(to=Class, on_delete=models.CASCADE)


class Lesson(models.Model):
    name = models.CharField(max_length=255)


class Schedule(models.Model):
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE)
    _class = models.ForeignKey(Class, on_delete=models.CASCADE)
    weekday = models.IntegerField()
    sequence = models.IntegerField()


class Journal(models.Model):
    pupil = models.ForeignKey(Pupil, on_delete=models.CASCADE)
    schedule = models.ForeignKey(Schedule, on_delete=models.CASCADE)
    date = models.CharField(max_length=10)
    mark = models.CharField(max_length=2)
```

views.py

```
import datetime

from django.contrib.auth import authenticate, login, logout
from django.core import serializers
from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage
from django.http import JsonResponse
from django.shortcuts import render, redirect

from ui.models import Pupil, Teacher, Personal, Class, Schedule, Journal


def home(request):
   return render(request, 'ui/index.html', {})


def about(request):
   return render(request, 'ui/about.html', {})

def schedule(request):
   classes = Class.objects.all()
   weekdays = range(1, 6)
   sequences = range(1, 6)
   schedule_data = {}

   for _class in classes:
      schedule_data[_class.id] = {}

      for weekday in weekdays:
         schedule_data[_class.id][weekday] = {}

         for sequence in sequences:
            try:
               schedule = Schedule.objects.get(_class=_class, weekday=weekday,
sequence=sequence)
               schedule_data[_class.id][weekday][sequence] = schedule
```

```
            except Schedule.DoesNotExist:
                schedule_data[_class.id][weekday][sequence] = None

    context = {
        'classes': classes,
        'weekdays': weekdays,
        'sequences': sequences,
        'schedule_data': schedule_data
    }

    return render(request, 'ui/schedule.html', context)


def schedule_own(request):
    if request.user.is_teacher:
        classes = Class.objects.all()
    else:
        pupil = Pupil.objects.filter(personal=request.user).first()
        classes = Class.objects.filter(id=pupil._class.id).all()
    weekdays = range(1, 6)
    sequences = range(1, 6)
    schedule_data = {}

    for _class in classes:
        schedule_data[_class.id] = {}

        for weekday in weekdays:
            schedule_data[_class.id][weekday] = {}

            for sequence in sequences:
                try:
                    if request.user.is_teacher:
                        teacher =
Teacher.objects.filter(personal=request.user).first()
                        schedule = Schedule.objects.get(_class=_class,
weekday=weekday, sequence=sequence, teacher=teacher)
                    else:
                        schedule = Schedule.objects.get(_class=_class,
weekday=weekday, sequence=sequence)
                    schedule_data[_class.id][weekday][sequence] = schedule
                except Schedule.DoesNotExist:
                    schedule_data[_class.id][weekday][sequence] = None

    context = {
        'classes': classes,
        'weekdays': weekdays,
        'sequences': sequences,
        'schedule_data': schedule_data
    }

    return render(request, 'ui/schedule_own.html', context)


def get_working_days(teacher, _class):
    week_day = []
    schedules = Schedule.objects.filter(teacher=teacher, _class=_class)
    for schedule in schedules:
        week_day.append(schedule.weekday - 1)
    working_days = []
    current_date = datetime.date.today()
    start_date = datetime.date(2022, 9, 1)  # Початок навчального року (1
вересня)
    end_date = current_date
```

```
    delta = datetime.timedelta(days=1)
    while end_date >= start_date:
        if end_date.weekday() in week_day:  # 0 - понеділок, 2 - середа
            working_days.append(end_date.strftime("%Y-%m-%d"))
        end_date -= delta

    return working_days

def get_page(request):
    page_number = request.GET.get('page')
    schedule_id = request.GET.get('schedule_id')
    schedule = Schedule.objects.filter(id=schedule_id).first()
    pupils = Pupil.objects.filter(_class=schedule._class).all()
    all_working_days = get_working_days(schedule.teacher, schedule._class)
    paginator = Paginator(all_working_days, 10)
    try:
        current_page = paginator.page(page_number)
    except PageNotAnInteger:
        current_page = paginator.page(1)
    except EmptyPage:
        current_page = paginator.page(paginator.num_pages)

    working_days = current_page.object_list
    total_pages = paginator.num_pages
    journal = list(Journal.objects.filter(pupil__in=pupils,
date__in=working_days).values())
    response_data = {
        'all_working_days': all_working_days,
        'working_days': working_days,
        'total_pages': total_pages,
        'pupils': list(pupils.values()),
        'journal': journal
    }
    return JsonResponse(response_data)

def journal(request):
    if request.user.is_teacher:
        schedules = Schedule.objects.filter(teacher__personal=request.user)
    else:
        _class = Pupil.objects.filter(personal=request.user).first()._class
        schedules = Schedule.objects.filter(_class=_class)
    context = {
        'schedules': schedules,
        'type': 'teacher' if request.user.is_teacher else 'pupil'
    }
    return render(request, 'ui/journal.html', context)


def journal_page(request, schedule_id):
    context = {
        'schedule_id': schedule_id,
        'type': 'teacher' if request.user.is_teacher else 'pupil'
    }
    return render(request, 'ui/journal_page.html', context)


def add_mark(request, schedule_id):
    pupil_id = request.POST.get('pupil_id')
    mark = request.POST.get('mark')
    date = request.POST.get('date')
    Journal.objects.create(pupil_id=pupil_id, mark=mark, date=date,
schedule_id=schedule_id)
    return redirect('journal_page', schedule_id=schedule_id)
```

```
def edit_mark(request, schedule_id):
    pupil_id = request.POST.get('edit_pupil_id')
    mark = request.POST.get('edit_mark')
    date = request.POST.get('edit_date')
    Journal.objects.filter(pupil_id=pupil_id, date=date,
schedule_id=schedule_id).update(mark=mark)
    return redirect('journal_page', schedule_id=schedule_id)


def delete_mark(request, schedule_id):
    pupil_id = request.POST.get('delete_pupil_id')
    date = request.POST.get('delete_date')
    Journal.objects.filter(pupil_id=pupil_id, date=date,
schedule_id=schedule_id).delete()
    return redirect('journal_page', schedule_id=schedule_id)


def sign_in(request):
    return render(request, 'ui/login.html', {})


def auth(request):
    username = request.POST.get('username')
    password = request.POST.get('password')
    user = authenticate(request, username=username, password=password)
    if user is not None:
        login(request, user)
        return redirect('home')
    else:
        return render(request, 'ui/login.html', {})

def register(request):
    classes = Class.objects.all()
    return render(request, 'ui/register.html', {'classes': classes})

def registration(request):
    email = request.POST.get('email')
    password = request.POST.get('password')
    full_name = request.POST.get('full_name')
    username = request.POST.get('username')
    user_type = request.POST.get('user_type')
    class_id = request.POST.get('class_id')
    if user_type == 'teacher':
        personal = Personal.objects.create_teacher(is_teacher=True, email=email,
password=password, username=username)
        Teacher.objects.create(full_name=full_name, personal=personal)
    else:
        personal = Personal.objects.create_pupil(is_pupil=True, email=email,
password=password, username=username)
        _class = Class.objects.filter(id=class_id).first()
        Pupil.objects.create(full_name=full_name, personal=personal,
_class=_class)
    user = authenticate(request, username=username, password=password)
    if user is not None:
        login(request, user)
        return redirect('home')
    else:
        return render(request, 'ui/register.html', {})


def log_out(request):
    logout(request)
    return redirect('home')
```

urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name="home"),
    path('about/', views.about, name='about'),
    path('schedule/', views.schedule, name='shedule'),
    path('schedule_own/', views.schedule_own, name='schedule_own'),
    path('journal/', views.journal, name='journal'),
    path('sign_in/', views.sign_in, name='sign_in'),
    path('auth/', views.auth, name='auth'),
    path('registration/', views.registration, name='registration'),
    path('register/', views.register, name='register'),
    path('log_out/', views.log_out, name='log_out'),
    path('get_page/', views.get_page, name='get_page'),
    path('journal_page/<int:schedule_id>/', views.journal_page,
name='journal_page'),
    path('add_mark/<int:schedule_id>/', views.add_mark, name='add_mark'),
    path('edit_mark/<int:schedule_id>/', views.edit_mark, name='edit_mark'),
    path('delete_mark/<int:schedule_id>/', views.delete_mark,
name='delete_mark'),
]
```