ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
Навчально-науковий інститут енергетичної, інформаційної
та транспортної інфраструктури

Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Створення вебсистеми для онлайн-навчання»

Виконав: студент 4 курсу, групи КН 2021-1
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)

Микола МІРОШНИЧЕНКО
(ім'я та прізвище)

Керівник: Наталія СІЗОВА
(ім'я та прізвище)

Рецензент Микола КАРПЕНКО
(ім'я та прізвище)

м. Харків – 2025 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Марина НОВОЖИЛОВА
«_24___» ___06_____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мірошниченка Миколи Миколайовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Створення веб-системи для онлайн-навчання»

керівник роботи Сізова Наталія Дмитріївна, докт.фіз.-мат. наук, проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 09 травня 2025 р. №341-03
2. Термін подання студентом роботи 20 червня 2025 року
3. Вихідні дані до роботи: проєктування і реалізація веб-системи для онлайн-навчання з застосуванням Python, Django, JavaScript ES6, React
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналітична частина, що включає опис процесу діяльності та функціональної моделі з визначенням ролей, огляд аналогів платформ для дистанційного навчання
2. Інформаційно-математична частина: визначення вхідних та вихідних даних, проєктування бази даних з використанням сервісу dbdiagram.io для візуалізації взаємозв'язків, огляд класів з побудовою UML-діаграм за допомогою сервісу PlantUML Web Server, математичне та алгоритмічне забезпечення
3. Програмно-технічна частина, яка включає обґрунтування обраних засобів розробки, опис програмної реалізації, зокрема бекенд- та фронтенд-частин, демонстрування дієздатності вебсистеми для онлайн-навчання з урахуванням ролей викладача та студента, яка успішно працює у локальному середовищі й може бути розгорнута на будь-якому сервері з підтримкою Python
4. Охорона праці
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Презентація – 15 слайдів

6. Консультанти розділів роботи

Розділ	Ім’я та Прізвище, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Наталія СІЗОВА, д.ф.-м.н., професор каф КНтаІТ	10.05.2025	11.05.2025
2	Наталія СІЗОВА, д.ф.-м.н., професор каф КНтаІТ	15.05.2025	17.05.2025
3	Наталія СІЗОВА, д.ф.-м.н., професор каф КНтаІТ	30.05.2025	21.05.2025
4	Вікторія МАЛИШЕВА, к. т. н., доцент кафедри ОП та БЖ	15.06.2025	27.05.2025

7. Дата видачі завдання 03.05.2025.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми дипломної роботи	03.05.2025	виконано
2	Затвердження тем, наукових керівників, завдань та календарного плану підготовки кваліфікаційної роботи	05.05.2025	виконано
3	Написання І розділу	10.05.2025	виконано
4	Написання ІІ розділу	15.05.2025	виконано
5	Написання ІІІ розділу	20.05.2025	виконано
6	Написання IV розділу	30.05.2025	виконано
7	Подання дипломної роботи керівнику	05.06.2025	виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	10.06.2025	виконано
9	Подання доопрацьованого варіанту роботи керівнику	15.06.2025	виконано
10	Захист матеріалів дипломної роботи на засіданні кафедри	18.06.2025	виконано
11	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	22.06.2025	виконано

Студент _____ Мірошниченко М. М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сізова Н. Д.

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи бакалавра студента групи КН 2021-1 спеціальності 122 Комп'ютерні науки Мірошниченка Миколи Миколайовича та темою «Створення вебсистеми для онлайн-навчання» складається з 4 розділів, містить 43 рисунки, 19 джерел.

Кваліфікаційну роботу бакалавра присвячено розробці програмного забезпечення для організації дистанційного навчання зі звучною взаємодією між викладачами та студентами. Розроблено інформаційне та програмне забезпечення розв'язання цієї задачі.

У розділі «Загальні положення» наведено опис предметного середовища, пов'язаного з темою кваліфікаційної роботи, розглянуто аналоги та визначено задачі дослідження.

Розділ «Інформаційне та математичне забезпечення» присвячений розгляду вхідних та вихідних даних, проєктуванню бази даних та об'єктно-орієнтованої моделі і визначенню математичного та алгоритмічного забезпечення.

У розділі «Програмне та технічне забезпечення» наведено опис програмних засобів та керівництво користувача. Програмне забезпечення включає розглянуто з боку фронтенд- та бекенд-розробки.

У розділі охорони праці визначені вимоги до організації робочого місця у навчальних закладах із урахуванням шкідливих та небезпечних виробничих факторів.

Ключові слова: ВЕБ-СИСТЕМА, ОНЛАЙН-НАВЧАННЯ, ОСВІТНЯ ПЛАТФОРМА, УПРАВЛІННЯ КУРСАМИ, ТЕСТУВАННЯ, КОРИСТУВАЧІ, ХМАРНА ІНФРАСТРУКТУРА.

ANNOTATION

Structure and scope of work. Explanatory note of the bachelor's qualification work of the student of the group KN 2021-1, specialty 122 Computer Science, Mykola Mykolaiovych Miroshnychenko, on the topic “Creating a web-based system for online learning” consists of 4 sections, contains 43 figures, 19 sources.

The bachelor's qualification work is devoted to the development of software for organizing distance learning with sound interaction between teachers and students. Information and software for solving this problem has been developed.

The “General Provisions” section describes the subject environment related to the topic of the qualification work, analogs are considered, and research objectives are defined.

The section “Information and Mathematical Support” is devoted to the consideration of input and output data, the design of a database and an object-oriented model, and the definition of mathematical and algorithmic support.

The Software and Hardware section provides a description of software tools and a user manual. The software includes consideration of front-end and back-end development.

The section on labor protection defines the requirements for the organization of the workplace in educational institutions, taking into account harmful and dangerous production factors.

Keywords: WEB-BASED SYSTEM, ONLINE LEARNING, EDUCATIONAL PLATFORM, COURSE MANAGEMENT, TESTING, USERS, CLOUD INFRASTRUCTURE.

ЗМІСТ

ВСТУП

У сучасному світі інформаційних технологій та цифрової трансформації освіти створення ефективних веб-систем для онлайн-навчання набуває все більшої важливості. Зростаюча роль дистанційних та змішаних форм навчання в освітніх закладах і корпоративному секторі вимагає розробки сучасних інструментів, які б забезпечували не лише доступ до навчальних матеріалів, а й повноцінне управління навчальним процесом.

Актуальність створення таких веб-систем зумовлена широким поширенням дистанційного та змішаного навчання, що стало особливо помітним у зв’язку зі змінами у форматі освіти. Сучасні системи онлайн-навчання повинні не лише надавати навчальну інформацію, а й забезпечувати зручне управління процесом взаємодії між викладачами та студентами, підтримувати організацію курсів, завдань, тестувань і відображення прогресу. Це робить розробку комплексних веб-рішень важливим завданням, що сприяє підвищенню якості освіти та її доступності.

Метою дипломної роботи є проєктування та реалізація веб-системи, яка дозволяє організовувати повноцінний навчальний процес онлайн, забезпечуючи базовий функціонал для керування курсами, користувачами, завданнями, тестуванням, комунікацією між викладачами та студентами, а також відображенням прогресу навчання.

Об’єкт дослідження – цифровий освітній процес у закладах освіти або корпоративному середовищі.

Предмет дослідження – програмна архітектура та технологічні рішення для створення веб-систем дистанційного навчання.

Для досягнення поставленої мети у роботі передбачено виконання низки завдань:

- аналіз предметної області онлайн-освіти та існуючих рішень;
- проєктування інформаційної системи;

– вибір інструментарію та методів реалізації програмного продукту;
– проведення тестування та оцінки працездатності створеної системи.

У дослідженні застосовуються методи системного аналізу, об’єктно-орієнтованого проєктування, а також сучасні технології веб-розробки, що забезпечують надійність і зручність використання кінцевого продукту.

Теоретична значимість роботи полягає у формуванні системного підходу до проєктування веб-систем дистанційного навчання з урахуванням особливостей освітнього процесу та потреб користувачів. Практична значущість отриманих результатів полягає у створенні програмного продукту, що може бути впроваджений у навчальні заклади та корпоративні структури для підвищення ефективності навчального процесу.

Результатом роботи буде розроблена веб-система для онлайн-навчання, яка забезпечує інтеграцію основних функціональних модулів, таких як управління курсами, користувачами, завданнями, тестуванням та комунікацією. Запропоноване рішення сприятиме підвищенню якості дистанційної освіти, покращенню взаємодії між учасниками навчального процесу та створенню комфортних умов для організації навчання в різних освітніх і корпоративних середовищах. Отже, результати дослідження можуть бути використані як основа для подальшого розвитку і вдосконалення цифрових освітніх платформ.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.1.1 Опис процесу діяльності

Розвиток цифрових технологій значно вплинув на трансформацію традиційних підходів до навчання, сформувавши нову галузь – онлайн-освіту. Веб-системи дистанційного навчання поступово стали не лише доповненням до класичної форми викладання, а й повноцінною альтернативою, що здатна забезпечити якісну організацію освітнього процесу без необхідності фізичної присутності учасників. Такий формат особливо актуальний у контексті глобалізаційних процесів, поширення моделей змішаного навчання, підвищення попиту на самостійне та гнучке засвоєння знань.

Актуальність створення веб-системи для онлайн-навчання зумовлена потребою в доступних, масштабованих і інтерактивних інструментах, які забезпечують ефективну комунікацію між викладачами та здобувачами освіти. Сучасні освітні установи стикаються з викликами, пов’язаними з необхідністю адаптації до нових умов, забезпечення безперервності навчального процесу, а також інтеграції цифрових технологій у щоденну практику. Створення власної веб-платформи дозволяє врахувати специфіку цільової аудиторії, реалізувати індивідуальні освітні траєкторії, забезпечити високий рівень інтерактивності та контролю засвоєння матеріалу, що є ключовими чинниками успішного навчання в умовах цифрової трансформації.

Предметне середовище онлайн-навчання охоплює комплексну взаємодію між студентом, викладачем та цифровою системою, що слугує основною платформою для управління навчальними матеріалами, моніторингу успішності, зворотного зв’язку й аналітики освітнього процесу. На відміну від офлайн-середовища, де переважають особисті комунікації та

фізичний доступ до ресурсів, веб-система повинна створювати віртуальну екосистему, в якій ці функції реалізуються технологічно – через інтерфейс, базу даних, механізми комунікації та контролю.

Ключовими вимогами до такого середовища є:

- наявність особистих кабінетів користувачів, із розмежуванням прав доступу (студенти, викладачі, адміністратори);
- можливість створення та редагування навчального контенту: завантаження лекцій, презентацій, відео, тестів, практичних завдань;
- підтримка асинхронного навчання, коли студент може самостійно обирати час і темп опрацювання матеріалу;
- функціонал для проведення тестування та оцінювання, що включає автоперевірку відповідей або перевірку викладачем;
- інструменти комунікації – повідомлення, коментарі, чати, форуми;
- аналітика та моніторинг успішності, яка дозволяє викладачам бачити прогрес студентів, а студентам – відслідковувати власні досягнення.

У рамках навчального процесу така система також виконує функції цифрового архіву: вона фіксує усі взаємодії між користувачем та контентом, зберігає результати, історію навчання, документи, звіти тощо. Це дає змогу повернутися до матеріалів, проаналізувати хід навчання та забезпечити прозорість оцінювання.

Іншою важливою складовою предметного середовища є інфраструктурний рівень: система має бути стабільною, масштабованою, безпечною, доступною цілодобово та незалежною від конкретного пристрою користувача. Для цього в сучасних рішеннях широко застосовуються хмарні технології: зберігання файлів на AWS, використання хмарних СУБД (наприклад, PostgreSQL на Neon), а також автоматизоване розгортання на хостингу типу Render.com.

З урахуванням цього, предметне середовище веб-системи для онлайн-навчання не обмежується лише інтерфейсом або базою знань – воно охоплює цілу інфраструктуру навчання, адаптовану до цифрового простору. Така

система повинна бути водночас функціональною, стійкою до навантаження, захищеною, легкою в адмініструванні та зручною для різних категорій користувачів. Саме ці принципи ляжуть в основу наступних етапів проєктування та реалізації програмного продукту.

1.1.2 Опис функціональної моделі

У веб-системі для онлайн-навчання беруть участь три основні категорії користувачів: студент, викладач та адміністратор. Кожна з ролей має визначений набір функціональних можливостей, які відповідають їх завданням у межах освітнього процесу. Спільними для всіх користувачів є дії, пов’язані з аутентифікацією, доступом до особистого кабінету та базовою навігацією в системі. Однак функціональні обов’язки та рівень доступу кожного актора суттєво різняться.

Адміністратор системи відповідає за налаштування середовища, розгортання додатку, управління базою даних і контроль прав доступу на рівні системи. Він не має окремого веб-інтерфейсу в рамках реалізованого проєкту, однак здійснює технічне обслуговування, додає або редагує користувачів безпосередньо через адміністративну консоль Django або засоби керування базою даних. Адміністратор також забезпечує збереження даних, моніторинг працездатності сервера та безперебійну роботу системи в локальній мережі.

Студент взаємодіє з курсами, до яких його додано викладачем. У межах кожного курсу він бачить завдання, виконує їх, слідкує за дедлайнами та отримує коментарі й оцінки від викладача. Особистий кабінет студента дозволяє переглядати активні та завершені задачі з фільтрацією, підсвічуванням прострочених завдань, а також бачити оцінки та коментарі до кожного завдання.

Основні функції студента включають:

- авторизація в системі;
- перегляд особистого кабінету;
- перегляд призначених курсів і завдань;

– зміна статусу завдань (перетягування між колонками: «нові», «у роботі», «здані»);

– завантаження файлів для виконання завдання;

– перегляд дедлайнів і коментарів викладача;

– ознайомлення з оцінками.

Основні сценарії взаємодії студента з системою наведено на діаграмі використання (рис. 1.1).

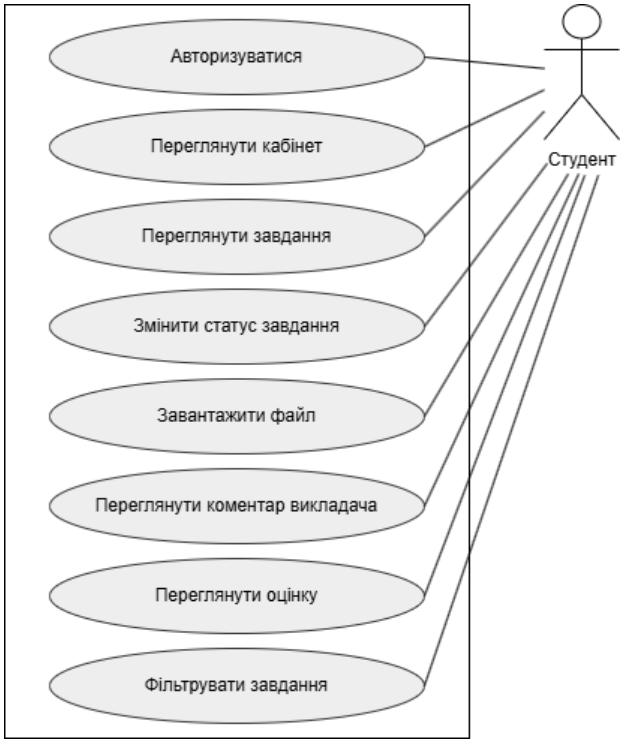

Рисунок 1.1 – Діаграма використання для ролі студента

Викладач створює курси, формує завдання з параметрами (тема, опис, дедлайн, тип задачі, прикріплені файли) та додає до них студентів. Він бачить усі задачі у вигляді дошки з колонками за статусом, може коментувати роботи студентів, перевіряти результати, змінювати статуси й ставити оцінки. Інтерфейс викладача оптимізований для швидкого перегляду активності всіх студентів у межах курсу.

Основні функції викладача включають:

– авторизація в системі;

- створення курсів;
- додавання опису, учасників і матеріалів до курсу;
- створення завдань (з темою, описом, дедлайном, файлами);
- перегляд дошки завдань усіх студентів;
- коментування задач;
- виставлення оцінок;
- фільтрація задач за статусом або датою.

Сценарії діяльності викладача в системі відображено на діаграмі використання (рис. 1.2).

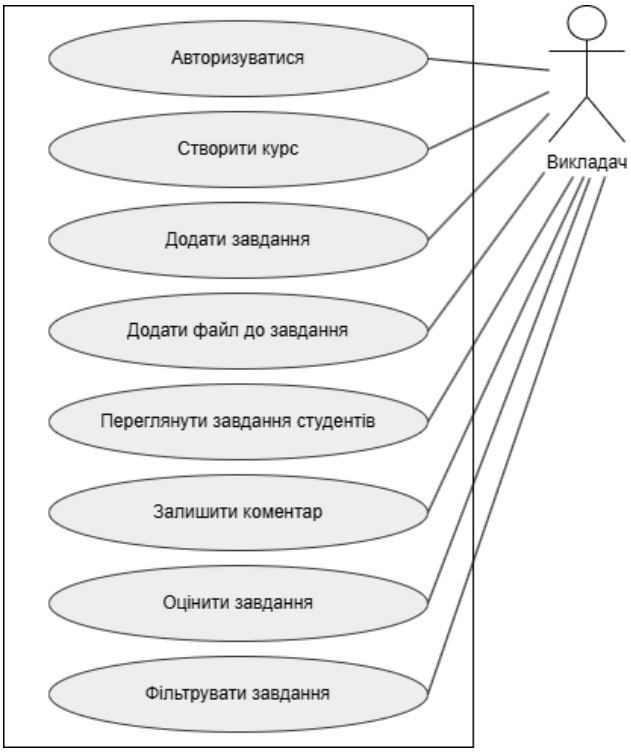

Рисунок 1.2 – Діаграма використання для ролі викладача

1.2 Огляд наявних аналогів

Серед великої кількості сучасних інструментів для організації дистанційного та змішаного навчання особливу увагу привертають платформи, які вже довели свою ефективність у масштабах усього світу. Розглянуто найпоширеніші освітні системи та сервіси, зокрема Udemy,

Coursera, Canvas LMS, Google Classroom та Moodle. Аналіз їхніх можливостей, функціоналу та підходів до взаємодії зі студентами дозволить виділити сильні сторони, обмеження та актуальні рішення, які можуть бути враховані під час розробки власної системи для онлайн-навчання.

Moodle [1] – це одна з найстаріших і наймасштабніших систем управління навчанням (LMS), що розповсюджується як проєкт із відкритим кодом. Вона підтримує повний набір функцій: курси, тести, обговорення, аналітику, доступ через браузер і мобільний додаток. Moodle часто використовується в університетах завдяки широким можливостям кастомізації та великій кількості плагінів. Основний недолік – складність налаштування та перевантажений інтерфейс для звичайного користувача.

Система Moodle є однією з найпоширеніших у світі платформ для дистанційного навчання. Згідно з інформацією, наведеною на офіційному сайті (рис. 1.3), вона охоплює понад 51 мільйон курсів, майже 150 тисяч сайтів, розгорнутих у понад 237 країнах.

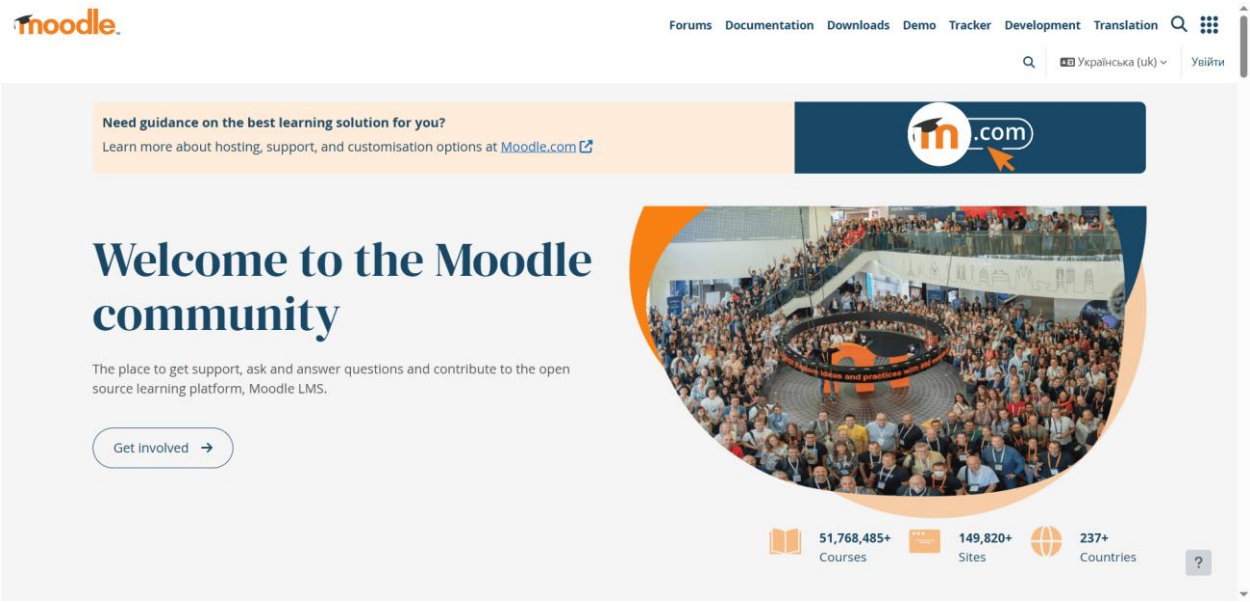

Рисунок 1.3 – Головна сторінка платформи Moodle

Google Classroom [2] – це хмарне освітнє рішення, інтегроване з Google Workspace, що дозволяє організовувати навчальний процес у межах

екосистеми Google. Його головна перевага полягає у простоті використання, зручному доступі до навчальних матеріалів, автоматичній інтеграції з Google Диском, Gmail та Календарем, а також у мінімальному порозі входу для вчителів і учнів. Класична структура курсів, можливість публікації завдань, коментування та виставлення оцінок робить сервіс зручним для базового дистанційного навчання. Водночас система поступається повнофункціональним LMS у гнучкості налаштувань, підтримці аналітики, модульності курсів і механізмах керування доступом. На рис. 1.4 зображено головну сторінку платформи Google Classroom.

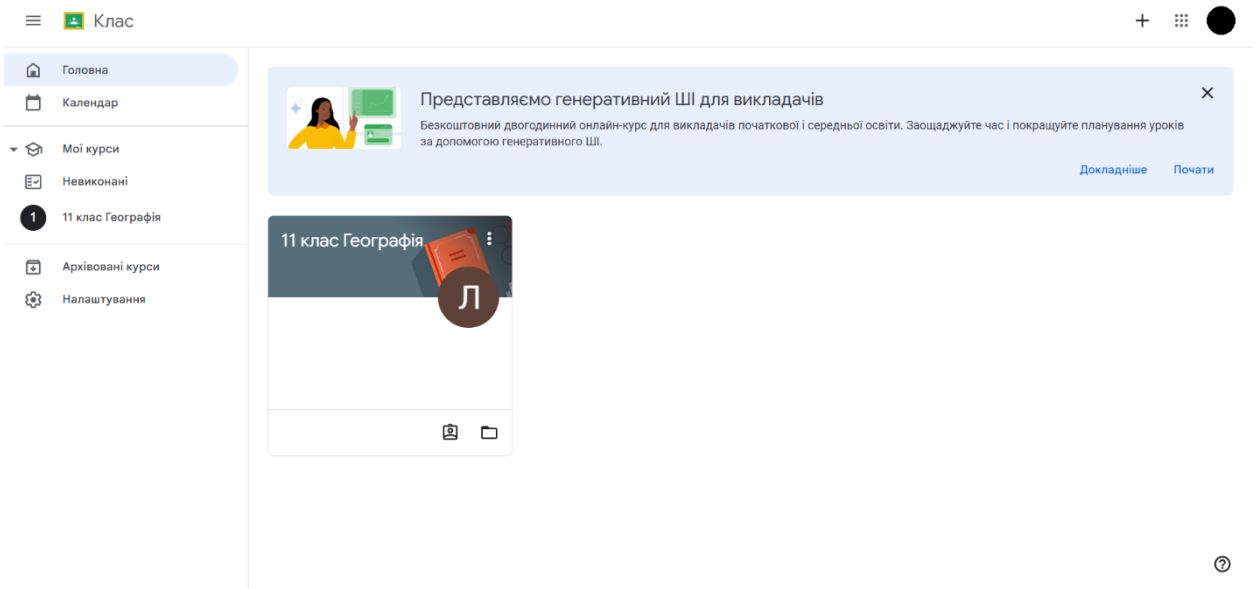

Рисунок 1.4 – Головна сторінка платформи Google Classroom

Canvas LMS [3] – потужна платформа, орієнтована на університетський сектор. Вона відзначається гнучким API, зручним адаптивним дизайном, високим рівнем безпеки й масштабованістю. Використовується такими навчальними закладами, як Stanford University, MIT. Має платну бізнес-модель, тому менш поширена у відкритому середовищі.

Головна сторінка Canvas LMS демонструє інтуїтивно зрозумілий інтерфейс та основні можливості платформи для організації навчального процесу (рис. 1.5).

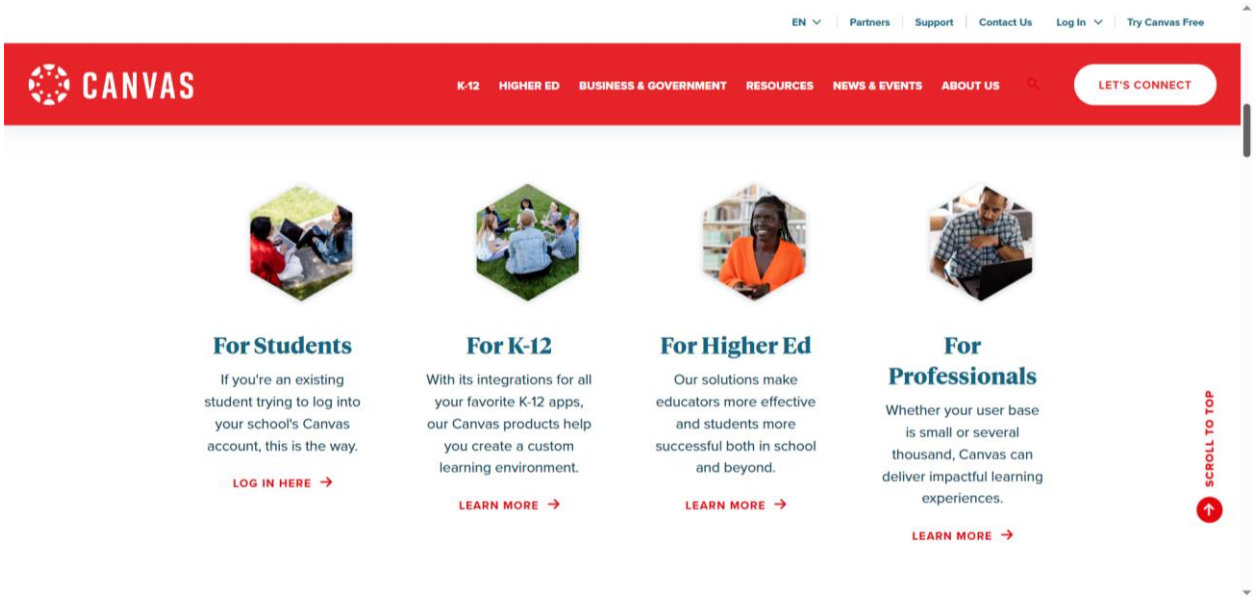

Рисунок 1.5 – Головна сторінка платформи Canvas LMS

Udemy [4] та Coursera [5] – це масові платформи онлайн-курсів. Вони мають повну інфраструктуру для викладача (створення курсів, додавання відео, тестів, сертифікатів) та для студента (оплата, відгуки, рейтинг, прогрес). Але ці сервіси не є універсальними LMS – вони не призначені для внутрішнього використання в організації або ВНЗ без обмежень.

Таким чином, існуючі платформи демонструють різні підходи до організації дистанційного навчання: одні орієнтовані на простоту й зручність, інші – на гнучкість і модульність, а ще частина – на обслуговування великого комерційного сегменту. Це ілюструє різноманітність рішень на ринку, кожне з яких відповідає певним потребам користувачів і завданням навчального процесу.

Щоб наочно оцінити масштаби популярності платформ, розглянемо графік, створений на основі наявної інформації [1-5], із кількістю щомісячних активних користувачів (MAU) протягом 2024 року, де дані усереднено поквартально (рис. 1.6):

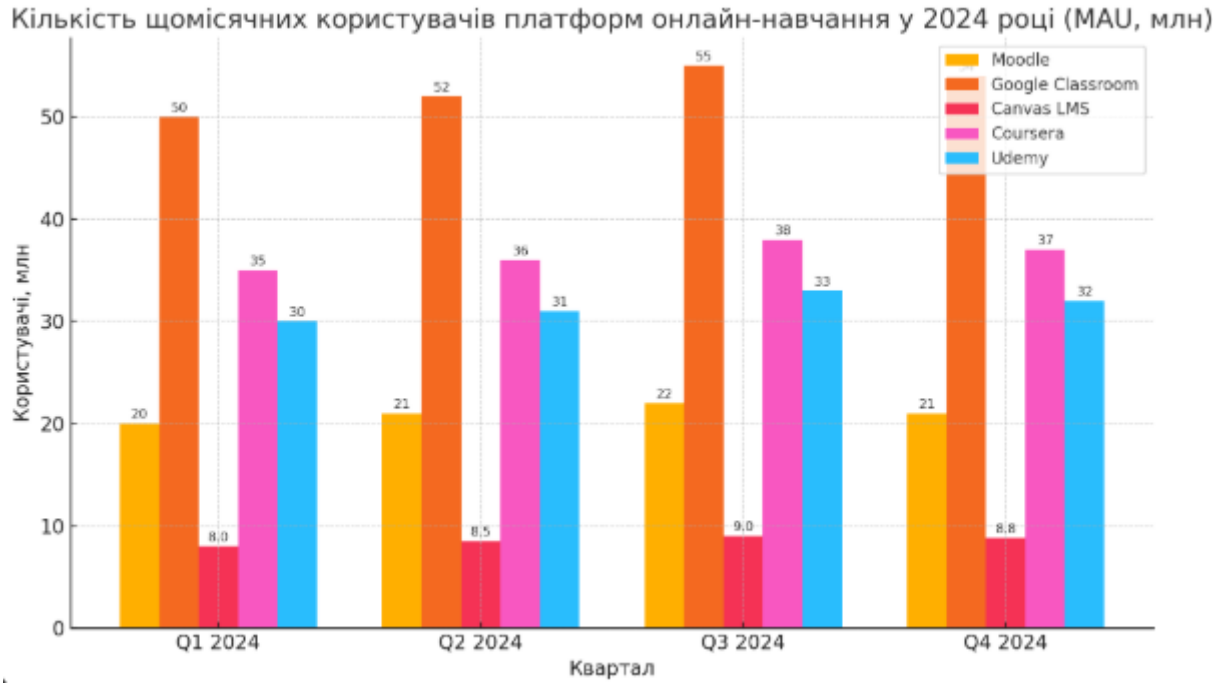

Рисунок 1.6 – MAU представників предметної області за 2024рік (поквартально)

Існуючі рішення підтверджують високий попит на подібні платформи, однак жодна з них не є універсальною. Деякі з них орієнтовані на простоту, інші – на гнучкість і модульність, а решта – на масовий комерційний ринок. Це відкриває можливість для створення вузькоспеціалізованого або адаптованого під конкретні вимоги рішення – наприклад, для невеликих навчальних центрів або авторських курсів із сучасним інтерфейсом, простим деплоєм, хмарною інфраструктурою та чітко структурованим функціоналом.

1.3 Постановка задачі

Метою дипломного проєкту є розробка локального веб-додатку для онлайн-навчання, який, окрім базових освітніх функцій, реалізує зручний інтерфейс управління завданнями у вигляді дошки – подібно до принципів, що застосовуються в системах на кшталт Jira або Trello. Такий підхід дозволяє організувати навчальний процес не лише як набір матеріалів, а як

послідовність задач із дедлайнами, статусами та відповідальними користувачами, що значно полегшує комунікацію та контроль прогресу.

Додаток створюється з використанням фреймворку Django, який забезпечує повноцінну серверну логіку, зв’язок із локальною базою даних та інтеграцію з HTML-шаблонами для побудови інтерфейсу. Вся система функціонуватиме в локальному середовищі без залучення зовнішніх сервісів, що робить її придатною для використання як навчального або демонстраційного інструменту.

Головним завдання є створення системи, яка міститиме такі функціональні компоненти:

- реєстрація та авторизація користувачів із розмежуванням прав доступу (викладач та студент);
- створення курсів викладачем, додавання описів, матеріалів, списку учасників;
- додавання завдань до курсу з такими полями: тема, опис, дедлайн, прикріплені файли, тип задачі;
- відображення завдань у вигляді дошки, розділеної на статуси: нові, у роботі, здані, перевірені;
- можливість перетягування завдань між колонками (drag-and-drop), редагування статусу, коментарі викладача до кожної задачі;
- підсвічування задач із простроченим терміном, фільтрація за датою або статусом;
- особистий кабінет студента з відображенням активних і завершених задач, оцінок та коментарів;
- сторінка викладача з повним оглядом задач усіх студентів, сортуванням і швидкою перевіркою результатів.

Для збереження даних буде використано SQLite або PostgreSQL, залежно від середовища запуску. Усі компоненти додатку працюватимуть в рамках локального веб-сервера Django, що дозволяє легко тестувати, змінювати і демонструвати проєкт без потреби у зовнішньому хостингу.

Система орієнтована на інтуїтивний інтерфейс і чітку логіку взаємодії, що робить її зручною як для студентів, так і для викладачів. Окрема увага приділяється візуалізації дедлайнів і управлінню навантаженням, що особливо актуально в умовах самостійного навчання.

Таким чином, проєкт має на меті створення повнофункціонального освітнього середовища з елементами таск-менеджменту, яке може бути використане в локальній мережі, у навчальних закладах або як основа для подальшої розробки більш масштабованої системи.

Висновки до розділу

У розділі було детально описано предметне середовище веб-системи для онлайн-навчання, що охоплює ключові процеси, ролі користувачів та функціональні можливості системи. Особлива увага приділена структурі взаємодії між студентами, викладачами та адміністраторами, а також технічним вимогам до платформи, що забезпечують її стабільність, безпеку та масштабованість. Визначено, що сучасна онлайн-система навчання повинна не лише надавати доступ до освітнього контенту, а й забезпечувати інтерактивність, індивідуалізацію навчального процесу, а також ефективні засоби контролю та аналітики.

Також у розділі проведено огляд існуючих аналогів – відомих LMS та освітніх платформ, таких як Moodle, Google Classroom, Canvas LMS, Udemy та Coursera. Проведений аналіз їхніх функціональних можливостей і обмежень дозволив виділити ключові переваги та недоліки, що можуть бути враховані при розробці власної системи. Це закладає міцну основу для подальшого проєктування та реалізації веб-системи, яка максимально відповідає сучасним потребам цифрової освіти.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

В умовах цифровізації освіти та впровадження гібридних і дистанційних форм навчання особливої актуальності набуває проблема ефективної організації навчального процесу та взаємодії між учасниками освітнього середовища – учнями та викладачами. Традиційні інструменти управління завданнями, як от зошити, щоденники, електронні таблиці, месенджери та листування в електронній пошті, не забезпечують повноцінного контролю за процесом навчання та значною мірою ускладнюють аналітику результатів.

Таким чином, проблематика предметної області охоплює кілька важливих аспектів:

– відсутність централізованого простору для управління завданнями, де учень не має єдиного інтерфейсу для перегляду всіх задач по предметах, а викладач – не бачить повної картини виконання завдань своїми учнями;

– неможливість ефективного контролю за строками виконання включно з виявленням прострочених завдань та формуванням списків задач, термін яких спливає найближчим часом;

– відсутність персоналізованої статистики результативності учня в розрізі предметів та загального рівня прогресу за дисциплінами викладача;

– необхідність у підтримці ролей і прав доступу, де учень повинен мати доступ лише до своїх задач, з можливістю відмічати виконання, переглядати дедлайни, коментувати, а вчитель – створювати та оцінювати завдання, переглядати статуси виконання учнів, додавати предмети, налаштовувати дошки;

– потреба у логічній структурованості завдань за предметами, дошками та колонками а також статусами «To Do», «In Progress», «Done»;

– комунікація та зворотний зв’язок у вигляді коментарів.

2.1.1 Вхідні дані

Для повноцінної роботи системи управління навчальними завданнями необхідно враховувати широкий спектр вхідної інформації. Всі ці дані надходять як вручну від користувачів, так і ззовні – з адміністративних інформаційних систем шкіл. Вхідні дані мають бути структурованими, достовірними й придатними до автоматизованої обробки.

До основних категорій вхідних даних належать: користувацька інформація, навчальна структура, завдання процес виконання та системна інфоомація.

Користувацька інформація включає:

– дані облікових записів (логін, пароль, роль – учень або вчитель);
– прив’язка користувача до класу та навчального закладу;
– особисті профілі з контактною інформацією.

Навчальна структура налічує такі елементи:

– список навчальних предметів;
– перелік тем у межах кожного предмета;
– календар навчального процесу (семестри, дні занять);
– дані про класи та класні групи.

До завдань належать:

– назва, опис, тип завдання;
– дата створення, дедлайн виконання;
– прикріплені файли або посилання;
– прив’язка до предмета і теми;
– вказівки щодо виконання та критерії оцінювання.

Процес виконання включає:

– статуси завдань (нове, у процесі, здано, перевірено);
– коментарі учнів і вчителів;
– оцінки та рецензії;

– час подачі та перевірки.

До системної інформації належать:

– журнали дій користувачів (login, дії із завданнями);

– дані для аналітики активності;

– імпортовані записи з електронних журналів або таблиць.

Завдяки такому комплексному підходу до структури вхідних даних, система здатна адаптуватися до потреб кожної ролі користувача й забезпечити високий рівень інформативності та функціональності.

2.1.2 Вихідні дані

Вихідні дані у системі керування навчальними завданнями – це результат обробки введеної інформації та дій користувачів, який надається в зрозумілій формі для кожного типу користувача. Вони відображають стан процесу навчання, результати виконання завдань та взаємодії між учасниками освітнього процесу. Ключові типи вихідної інформації можна розподілити залежно від визначених ролей, тобто для учнів та вчителів.

Для учнів вихідними даними є:

– списки активних і завершених завдань із дедлайнами;

– статуси виконання завдань (наприклад: «не розпочато», «виконується», «на перевірці», «перевірено»);

– отримані оцінки та коментарі від викладачів;

– повідомлення про зміни в завданнях або нові завдання;

– інтерфейс зворотного зв’язку з викладачем.

Вихідні дані для вчителів включають:

– панель завдань для кожного класу або предмету;

– статистика виконання завдань учнями;

– журнал оцінювання з можливістю фільтрації й аналітики;

– можливість перегляду коментарів і історії взаємодії із завданням;

– повідомлення про здані роботи або завершення дедлайнів.

Всі ці вихідні дані повинні бути структурованими відповідно до ролей користувачів і виводитися через інтуїтивно зрозумілий інтерфейс. Вони дозволяють кожному учаснику навчального процесу ефективно взаємодіяти із системою, приймати зважені рішення та своєчасно реагувати на події в освітньому середовищі.

2.2 Проєктування системи

2.2.1 Проєктування бази даних

База даних – одна з ключових складових системи, яка забезпечує зберігання та взаємозв’язок усіх елементів: користувачів, навчального контенту, дошок, завдань, коментарів тощо. Під час проєктування було використано реляційну модель, яка найкраще підходить для структурованих даних із чіткими зв’язками між сутностями.

Таблицями бази даних є Users, Boards, Columns, Tasks, Subjects, Comments, BoardMembers.

Users зберігає інформацію про всіх користувачів системи, тобто здобувачів освіти та наставників, вона використовується для автентифікації та авторизації. Полями Users є:

- id – унікальний ідентифікатор користувача;
- username – ім’я користувача;
- password – пароль користувача;
- is_teacher – булевий прапорець, що визначає, чи є користувач наставником.

Boards містить дані про навчальні дошки, які створюють користувачі, вона поєднана зв’язком один користувач – багато дошок та налічує поля:

- id – унікальний ідентифікатор дошки;
- name – назва дошки;
- owner_id – зовнішній ключ, що вказує на користувача-власника.

Columns відповідає за колонки в межах кожної дошки, які використовуються для організації завдань (наприклад, «До виконання», «У процесі», «Завершено»), вона має зв’язок одна дошка – багато колонок та містить поля:

– id – унікальний ідентифікатор колонки;

– name – назва колонки;

– board_id – зовнішній ключ, що вказує на дошку;

– position – позиція для сортування.

Tasks зберігає самі завдання, які прив’язані до колонок, дошок та предметів, налічує поля:

– id – унікальний ідентифікатор завдання;

– title – назва завдання;

– description – опис завдання;

– due_date – термін виконання;

– completed – булеве значення, що вказує на факт виконання;

– score – оцінка за виконання завдання;

– subject_id (зв’язок один предмет – багато завдань) – зовнішній ключ, що вказує на предмет;

– column_id (зв’язок одна колонка – багато завдань) – зовнішній ключ, що вказує на колонку;

– board_owner (зв’язок один здобувач освіти – багато завдань) – зовнішній ключ, що вказує на автора дошки.

Subjects відповідає за навчальні дисципліни або напрями, має поля id (унікальний ідентифікатор предмета), name (назва предмета) та зв’язок один предмет – багато завдань.

Comments – коментарі до кожного завдання від наставника або здобувача освіти, поєднується зв’язком одне завдання – багато коментарів, налічує поля:

– id – унікальний ідентифікатор коментаря;

– content – текст коментаря;

– created_at – дата й час створення коментаря;

– task_id – зовнішній ключ, що вказує на завдання;

– user_id – зовнішній ключ, що вказує на користувача-автора.

BoardMembers – проміжна таблиця, що реалізує зв’язок багато користувачів – багато дошок, має поля board_id (зовнішній ключ, що вказує на дошку), user_id (зовнішній ключ, що вказує на користувача).

Таблиці, поля та їх взаємозв’язки продемонстровано на рис. 2.1.

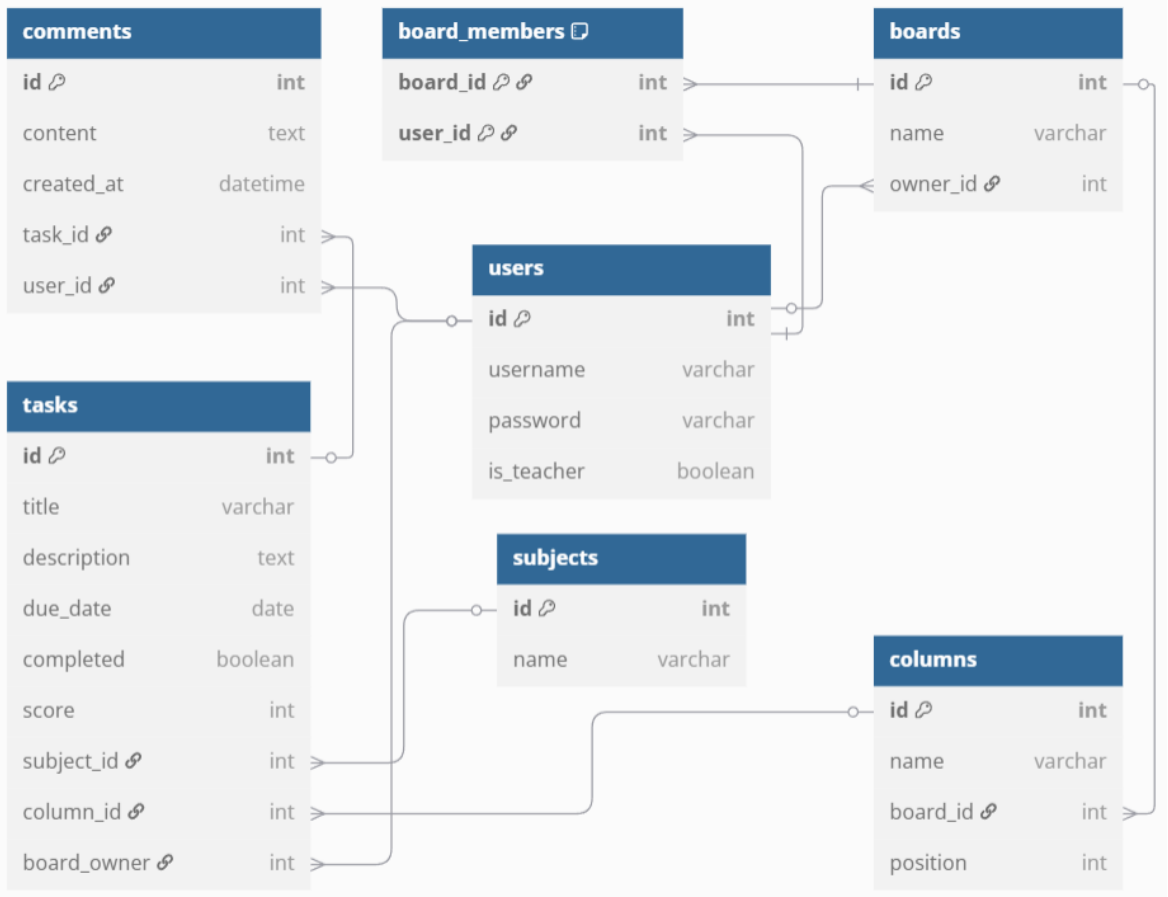

Рисунок 2.1 – ER-діаграма зв’язків між таблицями бази даних

2.2.2 Побудова об’єктно-орієнтованої моделі

Система реалізується з використанням об’єктно-орієнтованого підходу, що передбачає виділення логічних сутностей у вигляді класів, до яких

належать User, Task, Column, Board, Subject, Comment, вони разом з методами та атрибутами відображені на рис. 2.2.

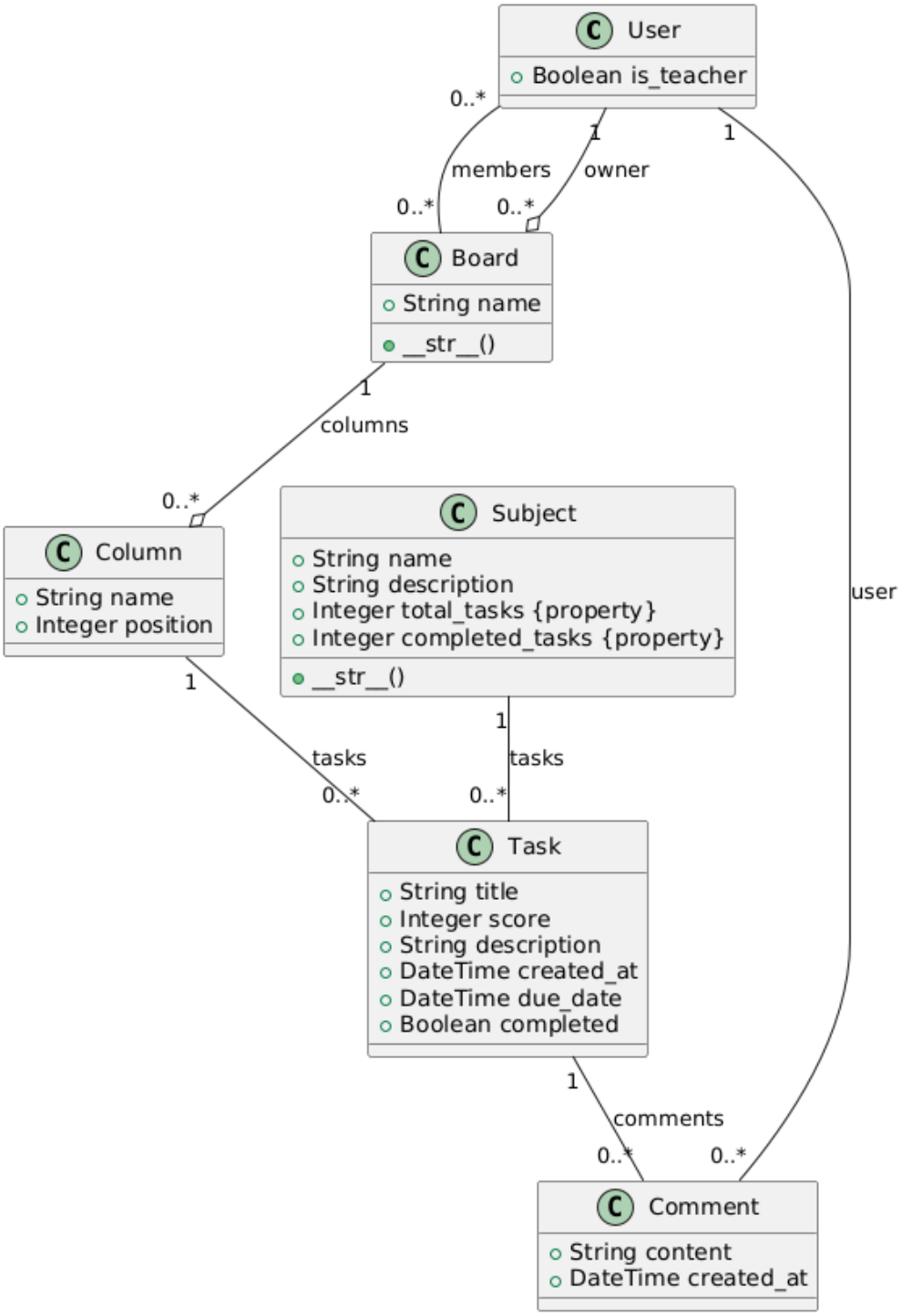

Рисунок 2.2 – UML-діаграма класів моделей

Класи моделей відповідають за збереження і структурування даних у базі, а також за логіку бізнес-процесів, пов’язаних з цими даними:

– User – розширена модель користувача, що додає поле is_teacher для позначення ролі користувача (викладач чи здобувач освіти), відповідає за збереження інформації про користувачів системи;

– Board – модель навчальної дошки, яка містить назву, власника (користувача, що створив дошку) та учасників, відповідає за організацію робочого простору, де розміщуються колонки та завдання;

– Column – модель колонки в межах дошки, яка організовує завдання у певному порядку (наприклад, «До виконання», «В процесі», «Завершено»), відповідає за структуризацію завдань усередині дошки;

– Subject – модель навчального предмета або напряму, що має назву та опис, містить властивості для підрахунку загальної кількості завдань та завершених завдань у межах предмета;

– Task – модель завдання, що належить певній колонці і предмету, має заголовок, опис, дату створення, термін виконання, статус виконання та бал, відповідає за відображення конкретної роботи чи завдання, яке необхідно виконати;

– Comment – модель коментаря до завдання, що містить текст коментаря, автора та дату створення, відповідає за додавання нотаток, зауважень або обговорень до завдань.

У проєкті використовуються й класи представлень, мета яких – реалізувати логіку обробки HTTP-запитів та забезпечити взаємодію між користувачем і сервером через REST API. Завдяки цим класам користувачі можуть реєструватися, автентифікуватися, працювати з дошками, колонками, завданнями, предметами й коментарями. Такими класами є:

– RegisterView – відповідає за реєстрацію нового користувача в системі, перевіряє наявність імені користувача та пароля, створює нового користувача з можливістю вказати, чи є він викладачем (is_teacher);

– CustomLoginView – забезпечує автентифікацію користувача та повертає токен доступу разом із додатковою інформацією (ідентифікатор, ім’я користувача, статус викладача);

– BoardViewSet – клас для роботи з дошками, який дозволяє створювати, переглядати, редагувати та видаляти дошки, використовує методи add_member і remove_member для керування учасниками дошки;

– ColumnViewSet – забезпечує створення та керування колонками в межах дошки, після створення колонки їй автоматично присвоюється позиція;

– TaskViewSet – відповідає за перегляд, створення, редагування й видалення завдань, кожне з яких пов’язане з колонкою та предметом;

– SubjectViewSet – надає інтерфейс для роботи з навчальними предметами;

– UserViewSet – дозволяє працювати з інформацією про користувачів;

– CommentViewSet – відповідає за додавання та перегляд коментарів до завдань, дає можливість фільтрації коментарів за ідентифікатором завдання.

Класи представлень відображені на рис. 2.3.

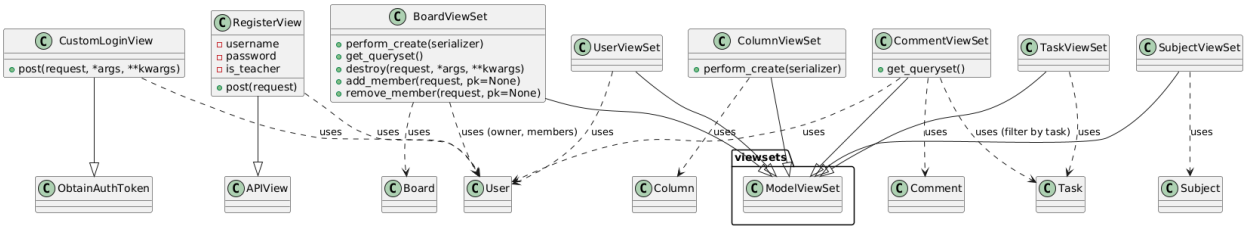

Рисунок 2.3 – UML-діаграма класів представлень

Окрім згаданих класів у проєкті наявні класи-серіалізатори, призначення яких – забезпечити перетворення даних між моделями та зовнішнім представленням у форматі JSON. Вони також відповідають за валідацію вхідних даних та (у деяких випадках) створення або обробку пов’язаних об’єктів. До таких класів належать:

– TaskSerializer – серіалізує модель Task, а також додає поле board_owner, що повертає ID власника дошки, до якої належить завдання;

– ColumnSerializer – серіалізує модель Column і включає пов’язані завдання через вкладений TaskSerializer;

– SubjectSerializer – серіалізує модель Subject, додаючи статистичні поля total_tasks і completed_tasks для відображення кількості всіх і завершених завдань відповідно;

– UserSerializer – серіалізує модель User, хешує пароль перед збереженням і дозволяє створювати користувачів із вказуванням, чи є вони викладачами (is_teacher);

– BoardSerializer – серіалізує модель Board, додає список колонок (через метод get_columns), список учасників дошки (members) та ім’я власника (owner_username);

– CommentSerializer – серіалізує модель Comment, забезпечує лише читання для автора коментаря та дати створення, а також відображає користувача у вигляді рядка.

Діаграма на рис. 2.4 ілюструє структуру класів, атрибути, методи та зв’язки з відповідними моделями.

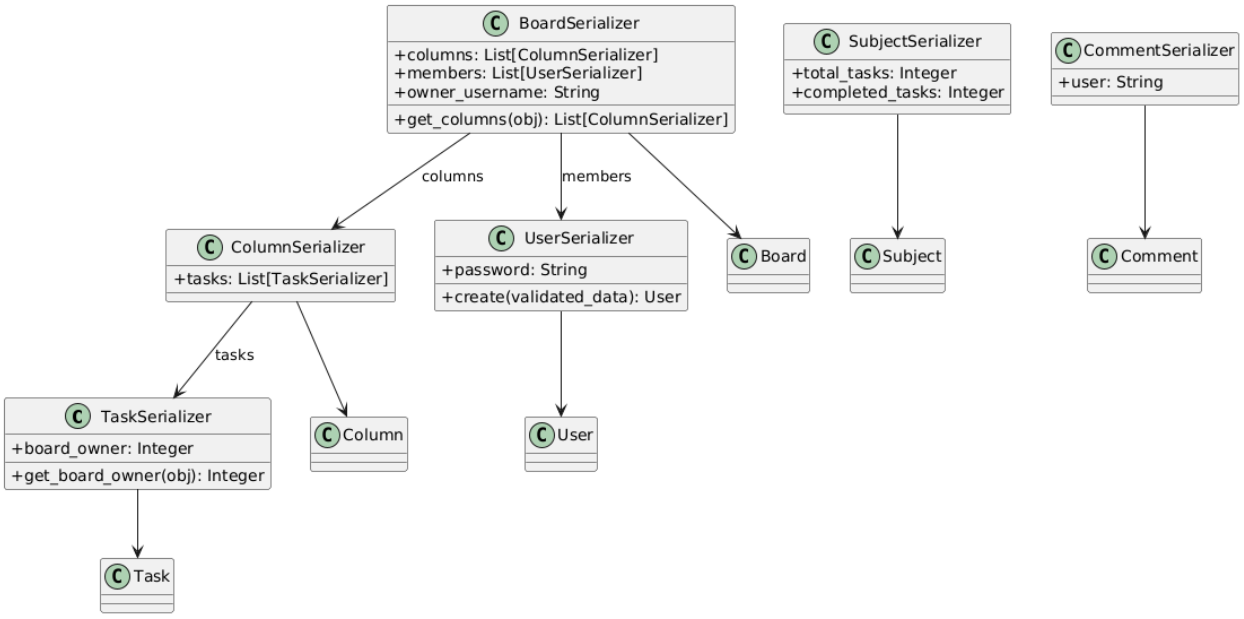

Рисунок 2.4 – UML-діаграма класів-серіалізаторів

У системі передбачено два основних класи користувачів: наставники (педагоги) та здобувачі освіти (учні). Ролі визначають набір доступних функцій у застосунку та логіку взаємодії з даними.

Наставник:

- має право створювати нові дошки, які виступають як робочі середовища для групи учасників;
- може додавати до дошки інших користувачів, призначаючи їх учасниками;
- визначає набір предметів (дисциплін) для систематизації завдань;
- створює колонки (наприклад, «To Do», «In Progress», «Done»), як інструмент візуального планування;
- додає завдання в обрані колонки, встановлює терміни виконання, предмет і очікуваний результат;
- має можливість оцінювати виконані завдання від 1 до 12 балів;
- переглядає статистику здобувачів освіти – кількість виконаних завдань, середній бал, активність за предметами.

Здобувач освіти:

- має доступ лише до дошок, учасником яких є;
- бачить створені наставником завдання, відфільтровані за колонками, термінами та предметами;
- моє можливість переміщувати завдання між колонками, відмічаючи етап виконання;
- може залишати коментарі до завдань, уточнюючи деталі або надаючи відповідь наставнику;
- має окрему сторінку «Мої завдання» з фільтрами, які включають виконані, прострочені, з наближеним дедлайном (до 14 днів) та за предметом;
- може переглядати свою статистику в профілі: кількість завдань, рівень виконання, середній бал за дисциплінами.

Рольова логіка побудована наступним чином:

- при реєстрації користувач обирає свою роль (через чекбокс «Teacher»);
- система на основі поля is_teacher обмежує доступ до функціоналу;
- наставник не може бачити особисті сторінки «Мої завдання» або залишати оцінки собі;

– здобувач освіти не має доступу до налаштувань дошок, створення предметів, редагування інших користувачів.

Такий розподіл ролей гарантує безпеку даних, гнучкість керування контентом та індивідуальний підхід до кожного типу користувача.

2.3 Математичне та алгоритмічне забезпечення

Оцінка прогресу здобувача освіти в межах кожного навчального предмета здійснюється шляхом визначення відношення кількості виконаних задач до загальної кількості задач, пов’язаних із цим предметом, що можна відобразити у вигляді (2.1):

$$P = \frac{K}{N} \cdot 100\% \qquad (2.1)$$

де P – оцінка прогресу здобувача освіти (%);

K – кількість задач, які вже позначені як виконані;

N – загальна кількість задач, призначених здобувачу освіти з певного предмета.

Алгоритм обчислення прогресу виконання задач алгоритм реалізовується у вигляді циклу фільтрації задач за ідентифікатором користувача та предмета з подальшим підрахунком загальної кількості (N) та виконаних (K). У разі відсутності задач (N = 0) прогрес відображається у вигляді «–».

Дані про прогрес містяться у таблицях на сторінці профілю кожного здобувача освіти, а також доступні для перегляду його викладачем.

Система також дозволяє обчислювати середній бал здобувача освіти з кожного навчального предмета, що дає змогу викладачу оцінити рівень засвоєння матеріалу та вчасно виявити труднощі у навчанні.

Середній бал обчислюється за формулою (2.2):

$$\bar{S} = \frac{1}{n}\sum_{i=1}^{n} s_i \qquad (2.2)$$

де $\bar{S}$ – середній бал здобувача освіти з певного навчального предмета;

n – кількість задач, які мають оцінку;

i – індекс задачі;

s_i – оцінка за і-ту задачу, що належить до вибраного предмета та має числове значення;

$1 \leq i \leq n$.

Для задач без оцінки ці значення не враховуються в обчисленнях.

Цей алгоритм реалізується через фільтрацію задач за ідентифікатором предмета і користувача, завдяки відбору лише тих задач, які мають числове значення поля score, і подальше обчислення середнього арифметичного.

Середній бал відображається в окремій колонці таблиці статистики на сторінці профілю та оновлюється в режимі реального часу при зміні оцінок.

Окрім оцінки прогресу та підрахунку середнього балу, важливим алгоритмічним елементом системи є класифікація навчальних задач за строками виконання з урахуванням ступеня терміновості.

Система передбачає чотири режими фільтрації:

– усі задачі – відображаються всі наявні задачі користувача, незалежно від статусу чи строку виконання;

– виконані задачі – задачі, які позначено як виконані;

– прострочені задачі – не завершені задачі з датою дедлайну, яка вже минула;

– задачі з наближеним дедлайном – не виконані задачі, термін яких настає в межах наступних 14 днів.

Така класифікація передбачає аналіз кожного завдання за двома критеріями:

– значення поля завершеності (тобто статусу виконання);

– календарна різниця між поточною датою та зазначеним терміном виконання.

Реалізація алгоритму фільтрації завдань дозволяє здобувачам освіти фокусуватись на найбільш актуальних задачах, а наставникам – швидко виявляти проблеми з термінами здачі.

Для оцінки успішності передбачено створення системи побудови статистичних звітів на основі задач, предметів і результатів виконання, які дозволять отримати узагальнену інформацію про успішність як окремого здобувача освіти, так і цілої групи.

Основні елементи алгоритму включають:

– розрахунок прогресу по кожному предмету – співвідношення виконаних задач до загальної кількості (2.1);

– побудова таблиць і графіків – статистичні таблиці з колонками предмет, кількість задач, кількість виконаних завдань, відсоток виконання, середній бал (2.2);

– групування по здобувачах освіти – для викладача доступна вибірка по кожному користувачу на основі зв’язаних з ним задач.

Кінцеві результати звітності відображаються у двох форматах:

– індивідуальні звіти – у профілі кожного здобувача освіти;

– групові звіти – на сторінці викладача для вибраного користувача.

Висновки до розділу

У розділі було проведено комплексний аналіз предметної області управління навчальними завданнями в умовах цифрового освітнього середовища. Визначено ключові проблеми, що виникають у процесі взаємодії між учнями та викладачами: відсутність єдиного середовища для перегляду та оцінювання задач, нестача аналітичних інструментів, а також потреба у структурованості й ролевій моделі доступу. Узагальнення вимог дозволило

сформувати перелік необхідних вхідних та вихідних даних, які враховують потреби обох типів користувачів – здобувача освіти та наставника.

Було спроєктовано базу даних із використанням реляційної моделі, яка забезпечує логічне зберігання та взаємозв’язок основних об’єктів системи: користувачів, дошок, колонок, завдань, предметів, коментарів і членства у дошках. Також реалізовано об’єктно-орієнтовану модель із відповідними класами, які відображають логіку бізнес-процесів і дозволяють взаємодіяти із системою. У межах цієї структури чітко визначено функціональність кожної ролі користувача, що гарантує безпечний і персоналізований доступ до даних.

У межах математичного та алгоритмічного забезпечення описано основні процедури, що забезпечують функціональність системи: розрахунок прогресу виконання задач, визначення середнього балу, класифікація задач за строками виконання та генерація статистичних звітів. Алгоритми базуються на фільтрації даних, агрегації значень і формальних обчисленнях, що дозволяє об’єктивно оцінювати успішність здобувачів освіти та ефективно організовувати навчальний процес у цифровому середовищі.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Під час створення вебзастосунку для організації навчального процесу було обрано сучасний технологічний стек, який забезпечує гнучкість, масштабованість і зручність у підтримці та розширенні системи. Вибір мови програмування та фреймворків обумовлений специфікою задач, роллю клієнтської і серверної частин, вимогами до інтерактивності, безпеки й збереження даних.

Мовою програмування для бекенду було обрано Python [6], оскільки вона є однією з найпопулярніших у сфері веброзробки, машинного навчання та наукових обчислень. Її перевагами є простий синтаксис, велика спільнота, численні бібліотеки і фреймворки, які роблять її ідеальним вибором для швидкої розробки.

Для бекенду було використано – Django [7] – високорівневий фреймворк для Python, який дозволяє швидко створювати безпечні й масштабовані вебзастосунки. Він забезпечує:

- систему аутентифікації;
- адміністративну панель;
- ORM для взаємодії з базою даних;
- захист від типових атак (XSS, CSRF тощо).

Ці особливості дозволяють сконцентруватися на логіці застосунку, а не на низькорівневій реалізації стандартних функцій.

Мовою програмування для фронтенду стала JavaScript (ES6) [8], яка є стандартом для створення динамічного контенту у вебзастосунках. Вона дозволяє реалізувати інтерактивні інтерфейси, обробку подій та зміну DOM без перезавантаження сторінки.

Бібліотекою для фронтенду обрано React [9] – це популярна бібліотека JavaScript для побудови UI-компонентів. Її основні переваги включають:

– компонентну архітектуру;
– високу продуктивність завдяки віртуальному DOM;
– підтримку односторінкових застосунків (SPA);
– великий екосистемний стек і розширення.

У проєкті React забезпечує реалізацію клієнтської частини, включаючи роботу з формами, навігацію, інтеграцію з API.

Для обміну даними між клієнтом і сервером використовується REST API [10]. Django автоматично генерує ендпоінти завдяки бібліотеці Django REST Framework, що забезпечує чітку структуру запитів/відповідей і дозволяє масштабувати застосунок без порушення логіки взаємодії.

Додаткові інструменти налічують:

– Axios для виконання HTTP-запитів на фронтенді;
– React Router для маршрутизації сторінок у SPA;
– CSS (ручне оформлення) для стилізації компонентів.

На рис. 3.1 відображено загальну схему технологічного стеку.

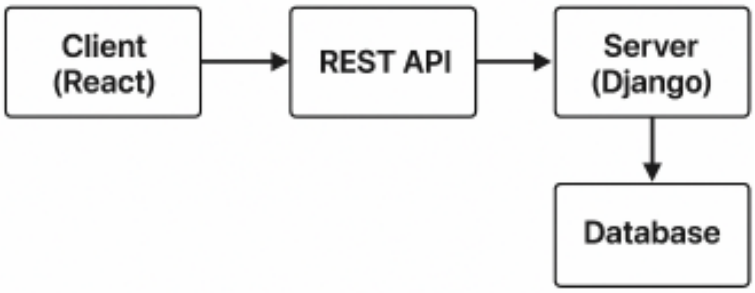

Рисунок 3.1 – Схема технологічного стеку

У проєкті використовуються база даних SQLite [11] – легка, кросплатформна та вбудована система керування базами даних, що дозволяє зберігати всі дані в одному файлі без потреби в окремому сервері.

Середовище розробки – PyCharm [12] – інтегроване середовище розробки, яке підтримує підсвітку синтаксису, автодоповнення, роботу з віртуальними середовищами та інтеграцію з системами контролю версій.

3.2 Вимоги до технічного та програмного забезпечення

Системне ПЗ:

- операційна система (Windows, macOS, Linux);
- Python 3.11 або вище;
- Node.js з npm для роботи фронтенд-складової.

Інструментальне ПЗ:

- віртуальне середовище Python;
- середовище розробки PyCharm або інший текстовий редактор з підтримкою Python;
- інструменти командного рядка PowerShell або Bash для запуску серверних і клієнтських сервісів.

Функціональне ПЗ:

- Django як бекенд-фреймворк;
- Django REST Framework для реалізації API;
- React для клієнтської частини;
- Axios для здійснення HTTP-запитів на фронтенді;
- React Router для маршрутизації;
- SQLite як система управління базами даних.

Серверна частина реалізована на фреймворку Django та розгортається у локальному середовищі розробника з використанням вбудованого серверу runserver. У перспективі можливе розгортання на будь-якому хостингу, що підтримує Python.

Мінімальні технічні вимоги до запуску серверної частини:

- операційна система: Windows/Linux/macOS;

- встановлений Python 3.11+;
- встановлений Node.js та npm (для фронтенду);
- RAM: мінімум 4 ГБ (рекомендовано 8 ГБ);
- CPU: від 2 ядер;
- вільне місце на диску: 500 МБ для застосунку та залежностей;
- порти: 8000 для бекенду, 3000 для фронтенду (налаштовуються вручну).

3.3 Опис програмної реалізації

Програмна система складається з двох основних частин: бекенда, реалізованого на Django, та фронтенда, побудованого на бібліотеці React. Проєкт має чітко структуровану ієрархію директорій, що забезпечує зручність розробки, тестування й масштабування. Основна директорія проєкту має назву task-manager.

3.3.1 Бекенд-частина проєкту

У кореневій директорії task-manager (рис. 3.2) розташовані такі важливі компоненти:

- manage.py – стандартний керуючий файл Django, що використовується для запуску сервера, міграцій, створення суперкористувача тощо;
- db.sqlite3 – файл бази даних SQLite, який використовується як сховище у поточній версії;
- requirements.txt – список усіх залежностей Python, потрібних для запуску проєкту;
- templates/ – директорія для HTML-шаблонів, якщо такі використовуються сервером;
- task_manager/ – головний конфігураційний модуль Django-проєкту.

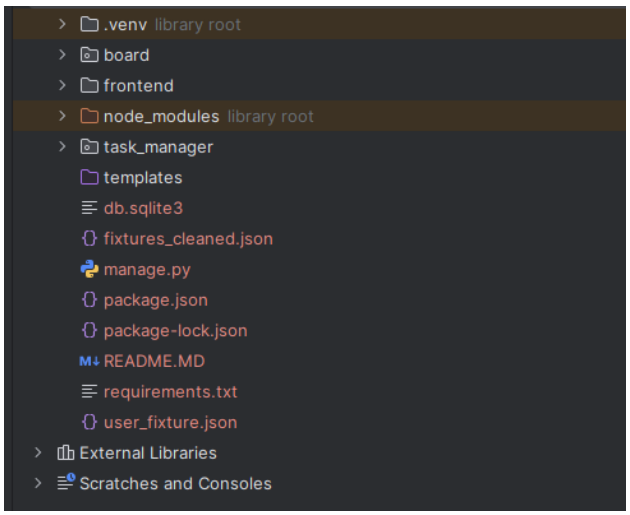

Рисунок 3.2 – Структура кореня проєкту

Папка task_manager (рис. 3.3) містить налаштування Django-проєкту:

– __init__.py – позначає папку як Python-пакет;

– settings.py – головний файл конфігурації: підключення додатків, бази даних, параметри безпеки тощо;

– urls.py – файл маршрутизації, де задаються основні URL-адреси;

– wsgi.py / asgi.py – файли для запуску серверів WSGI/ASGI відповідно.

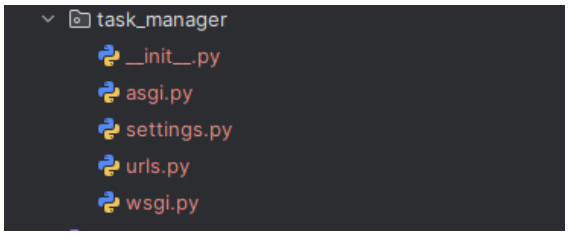

Рисунок 3.3 – Структура файлів Django-проєкту в папці task_manager

Модуль board (рис. 3.4) реалізує функціональність управління завданнями, дошками, користувачами та коментарями. Основні файли включають:

- models.py – моделі бази даних;
- serializers.py – серіалізатори для перетворення моделей у формат JSON;
- views.py – реалізація API-логіки через Django REST Framework;
- urls.py – маршрути для ендпоінтів цього модуля;

- admin.py – реєстрація моделей у Django Admin;
- tests.py – модульні тести;
- apps.py – конфігурація додатку.

Папка migrations/ містить всі автоматично згенеровані або ручні міграції для бази даних.

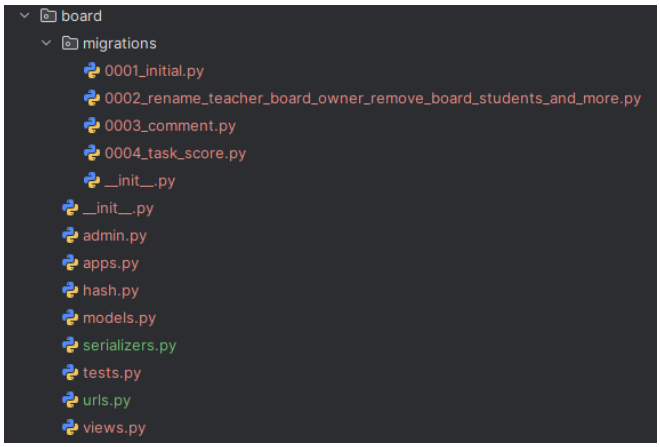

Рисунок 3.4 – Структура додатку board з файлами моделей, в'юшок, серіалізаторів, міграцій тощо

У створеному вебзастосунку для управління навчальними завданнями основою взаємодії з базою даних є набір моделей, реалізованих за допомогою ORM фреймворку Django. Кожна модель відображає реальну сутність, необхідну для організації освітнього процесу в рамках цифрового середовища.

Моделі структуровані в модулі board/models.py, забезпечуючи чітку логіку зв'язків, обмежень та ієрархії об'єктів у застосунку.

Однією з моделей є User (Користувач) (рис. 3.5) – стандартна модель Django AbstractUser, яка була розширена для додаткової функціональності.

Кожен користувач має такі характеристики:

– username, password – стандартні поля автентифікації;

– is_teacher – булевий прапорець, що визначає роль користувача (викладач чи здобувач освіти);

– email, first_name, last_name – необов'язкові поля для додаткової ідентифікації.

Користувач може бути:

– власником дошки (створює й адмініструє її);

– учасником інших дошок (додається до команд для виконання завдань);

– автором коментарів;

– виконавцем завдань.

```
class User(AbstractUser):
    is_teacher = models.BooleanField(default=False)
```

Рисунок 3.5 – Програмна реалізація моделі User

Board (Дошка) – логічна структура для групування завдань, яка має такі атрибути:

– name – назва дошки (наприклад: «Історія – 9-А клас»);

– owner – FK до User, який є адміністратором цієї дошки;

– members – many-to-many-зв’язок із користувачами, які можуть працювати з цією дошкою;

– created_at, updated_at – мітки часу.

Це дозволяє реалізувати розділення задач за предметами, курсами або групами. Програмна реалізація моделі подана на рис. 3.6.

```
class Board(models.Model):
    name = models.CharField(max_length=255)
    owner = models.ForeignKey(User, on_delete=models.CASCADE, related_name='owned_boards')
    members = models.ManyToManyField(User, related_name='boards', blank=True)

    def __str__(self):
        return self.name
```

Рисунок 3.6 – Програмна реалізація моделі Board

Column (Колонка) (рис. 3.7) – це стан, в який може потрапити завдання, зокрема: «Заплановано», «В роботі», «Завершено» тощо. Поля включають:

– name – назва колонки;

– position – позиція для сортування;

– board – FK до Board;

– created_at – дата створення.

```
class Column(models.Model):
    board = models.ForeignKey(Board, on_delete=models.CASCADE, related_name='columns')
    name = models.CharField(max_length=255)
    position = models.PositiveIntegerField(default=0)
```

Рисунок 3.7 – Програмна реалізація моделі Column

Task (Завдання) (рис. 3.8) – центральна модель застосунку, яка містить навчальні або організаційні завдання, її поля:

- title, description – текстове наповнення;
- due_date – крайній термін виконання;
- completed – булеве поле, чи виконано завдання;
- score – оцінка за завдання (1–12 балів);
- column – FK до Column;
- subject – FK до Subject;
- board_owner – id власника дошки (використовується для фільтрації статистики);
- created_at, updated_at.

```
class Task(models.Model):
    column = models.ForeignKey(Column, on_delete=models.CASCADE, related_name='tasks')
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE, related_name='tasks')
    title = models.CharField(max_length=255)
    score = models.PositiveIntegerField(null=True, blank=True)
    description = models.TextField(blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    due_date = models.DateTimeField(null=True, blank=True)
    completed = models.BooleanField(default=False)
```

Рисунок 3.8 – Програмна реалізація моделі Task

Subject (Предмет) – проста модель, яка дозволяє класифікувати завдання за навчальними напрямами:

- name – назва предмету (наприклад: «Математика», «Фізика»);

- created_at.

Це забезпечує фільтрацію та аналіз ефективності роботи за напрямами. Програмна реалізація моделі подана на рис. 3.9.

```
class Subject(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.name

    @property
    def total_tasks(self):
        return self.tasks.count()

    @property
    def completed_tasks(self):
        return self.tasks.filter(completed=True).count()
```

Рисунок 3.9 – Програмна реалізація моделі Subject

Comment (Коментар) – дозволяє зберігати зворотній зв’язок:

- task – FK до Task;
- user – FK до User, автор коментаря;
- content – текст коментаря;
- created_at.

Коментарі можуть залишати викладачі або здобувачі освіти залежно від прав доступу. Програмна реалізація моделі наведена на рис. 3.10.

```
class Comment(models.Model):
    task = models.ForeignKey('Task', related_name='comments', on_delete=models.CASCADE)
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
```

Рисунок 3.10 – Програмна реалізація моделі Comment

Взаємозв’язки між моделями подано на рис. 3.11.

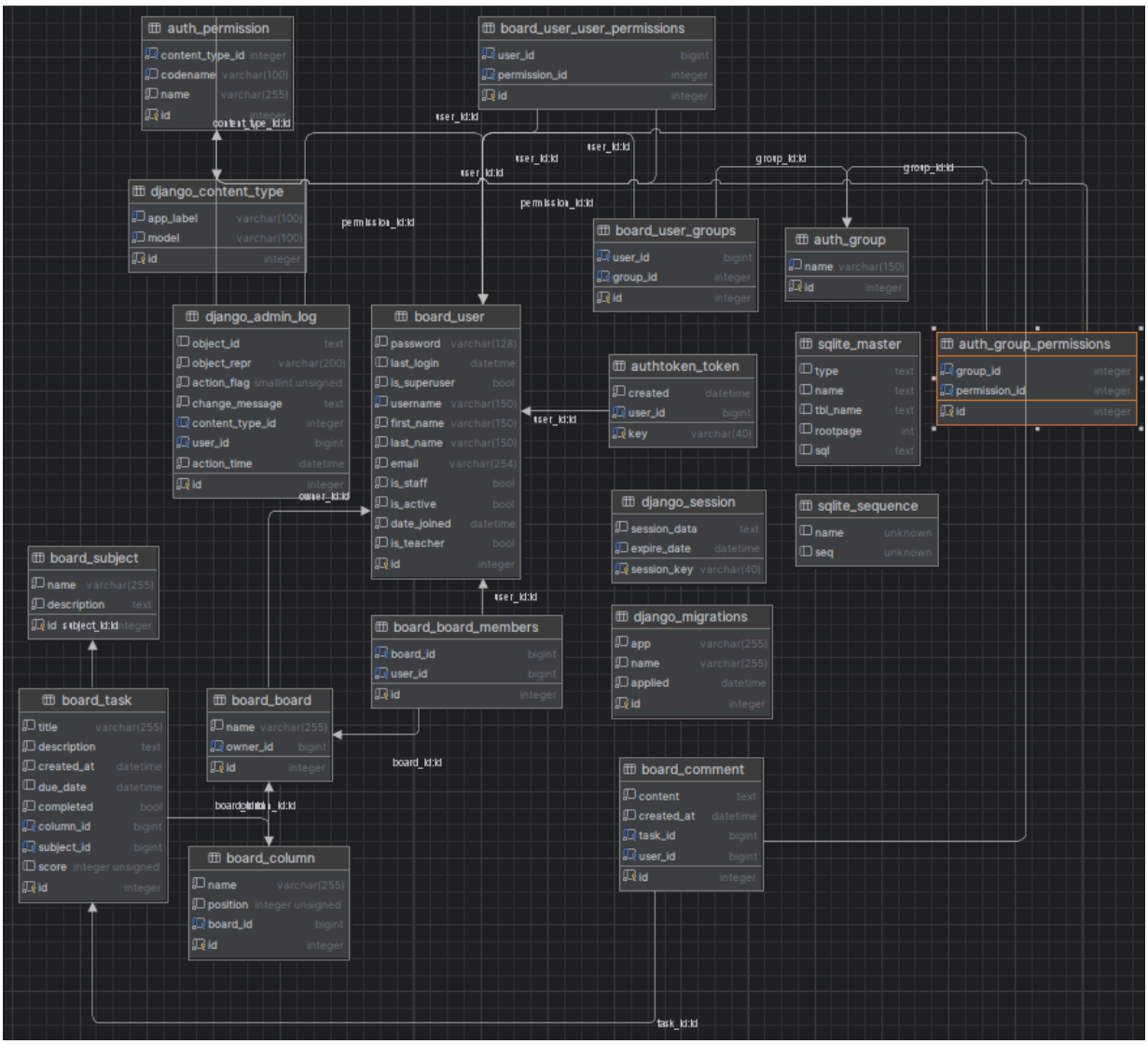

Рисунок 3.11 – Взаємозв’язки між моделями

Серверна частина побудована на основі Django REST Framework (DRF). Це дозволяє реалізувати RESTful API для взаємодії клієнта та сервера через HTTP-запити.

Для кожної моделі створено відповідний серіалізатор у serializers.py, який визначає, як об’єкти конвертуються у формат JSON для клієнта, і назад – у Python-об’єкти при отриманні даних. Серіалізатор користувача продемонстровано на рис. 3.12. Інші серіалізатори, до числа яких входять: BoardSerializer, ColumnSerializer, SubjectSerializer, CommentSerializer, UserSerializer – аналогічні.

```python
class UserSerializer(serializers.ModelSerializer):  3 usages  new *
    password = serializers.CharField(write_only=True)

    class Meta:  new *
        model = User
        fields = ['id', 'username', 'password', 'is_teacher']

    def create(self, validated_data):  new *
        print(f"{make_password('1')}")
        user = User(
            username=validated_data['username'],
            is_teacher=validated_data.get('is_teacher', False)
        )
        user.set_password(validated_data['password'])
        user.save()
        return user
```

Рисунок 3.12 – Програмна реалізація UserSerializer

Усі API-ендпоїнти реалізовані через class-based views (APIView, ModelViewSet) або через ViewSet. Наприклад, перегляд і створення задач має такий вигляд (рис. 3.13):

```python
/api/tasks
class TaskViewSet(viewsets.ModelViewSet):  2 usages
    queryset = Task.objects.all()
    serializer_class = TaskSerializer
```

Рисунок 3.13 – Програмна реалізація TaskViewSet

Для нестандартних дій – створено окремі APIView, наприклад, додавання учасника до дошки або логін.

Файл urls.py містить маршрути до всіх API (рис. 3.14).

```python
/api/
router = DefaultRouter()
/.../
router.register( prefix: r'boards', BoardViewSet)
/api/columns
router.register( prefix: r'columns', ColumnViewSet)
/api/tasks
router.register( prefix: r'tasks', TaskViewSet)
/api/subjects
router.register( prefix: r'subjects', SubjectViewSet)
/api/users
router.register( prefix: r'users', UserViewSet)
/api/comments
router.register( prefix: r'comments', CommentViewSet)

/api/
urlpatterns = [
    /api/
    path( route: '', include(router.urls)),
    /api/register/
    path( route: 'register/', RegisterView.as_view(), name='register'),
    /api/login/
    path( route: 'login/', CustomLoginView.as_view(), name='custom_login'),
    /api/boards/{board_id}/add_member/
    path( route: 'boards/<int:board_id>/add_member/', add_member_to_board)
]
```

Рисунок 3.14 – Реалізація URL-шляхів

3.3.2 Фронтенд-частина проєкту

Фронтенд-частина розроблена з використанням бібліотеки React, що дозволяє створювати динамічні інтерфейси користувача на основі компонентного підходу. Уся логіка клієнтської частини проєкту знаходиться в директорії frontend (рис. 3.15).

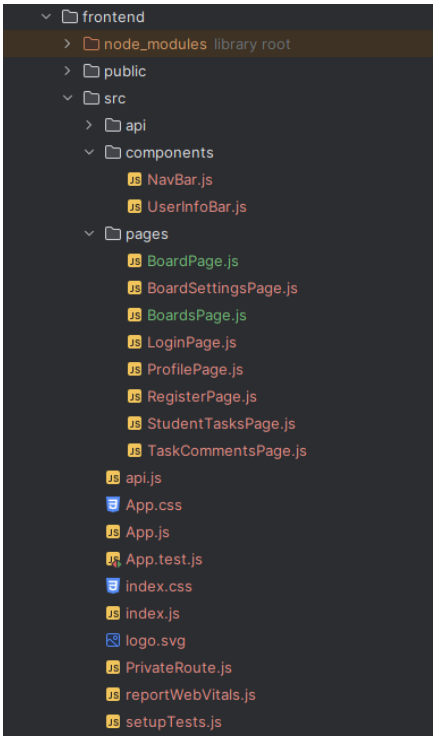

Рисунок 3.15 – Структура директорії frontend/src з компонентами, сторінками та API

Директорія зі статичними ресурсами – public/ – включає такі елементи:

– index.html – базовий HTML-шаблон для рендеру React-додатку;

– logo192.png, logo512.png – зображення, що використовуються для favicon або splash screen;

– manifest.json – конфігурація PWA;

– robots.txt – налаштування для SEO.

Основна логіка додатку міститься в src/:

а) api/ – містить файл index.js з методами для HTTP-запитів до бекенду через axios;

б) components/ – спільні візуальні компоненти:

– NavBar.js – панель навігації між сторінками;

– UserInfoBar.js – відображення поточного користувача, кнопки виходу та переходу.

в) pages/ – основні сторінки SPA-додатку:

– LoginPage.js – сторінка входу;

– RegisterPage.js – реєстрація користувача;

– BoardsPage.js – список усіх дошок користувача;

– BoardPage.js – перегляд однієї дошки;

– BoardSettingsPage.js – керування учасниками та колонками;

– ProfilePage.js – статистика користувача/учня;

– StudentTasksPage.js – перелік завдань здобувача освіти;

– TaskCommentsPage.js – перегляд і коментування завдання.

г) App.js – головний компонент додатку, що відповідає за маршрутизацію;

ґ) PrivateRoute.js – захист сторінок від неавторизованого доступу;

д) index.js – точка входу в додаток, де відбувається рендер у root;

е) App.css, index.css – глобальні стилі проєкту.

У додатку реалізовано авторизацію на основі токену, який генерується на бекенді після логіну через CustomLoginView (JWT не використовується – лише збереження токена вручну). Токен, а також роль (is_teacher) і user_id, зберігаються в localStorage браузера (рис. 3.16).

```
localStorage.setItem("token", res.data.token);
localStorage.setItem("username", res.data.username);
localStorage.setItem("is_teacher", res.data.is_teacher);
localStorage.setItem("user_id", res.data.user_id);
```

Рисунок 3.16 – Збереження сесійних даних у localStorage

На фронтенді перевірка ролей відбувається при завантаженні сторінки або під час рендеру компонентів. Наприклад, сторінка StudentTasksPage.js доступна лише здобувачам освіти (рис. 3.17).

```
if (isTeacher) return <p style={{ color: '#eee', padding: '2rem' }}>Only students can view this page.</p>;
```

Рисунок 3.17 – Обмеження доступу до сторінки лише для учнів

Аналогічно, у NavBar показуються різні кнопки в залежності від ролі користувача. Таким чином реалізовано обмеження інтерфейсу без потреби окремої логіки на сервері.

У застосунку реалізовано роботу з формами:

– форма реєстрації (RegisterPage.js);
– форма логіну (LoginPage.js);
– створення задач (через бекенд);
– створення предметів (тільки для ролі викладача).

Форми реалізовані з валідацією на фронтенді – наприклад, перевірка на порожні поля: if (!newComment.trim()) return.

А також на бекенді через Django REST Framework, де використовується ModelSerializer, який автоматично перевіряє наявність обов'язкових полів. У разі помилки форма може показувати повідомлення (через alert, або в майбутньому – UI повідомлення) (рис. 3.18).

```
} catch (err) {
  alert("Register failed");
}
```

Рисунок 3.18 – Обробка помилок реєстрації користувача

Також реалізовано фільтрацію задач за предметом та строком виконання (рис. 3.19).

```
const filtered = tasks.filter(task => {
  const due = task.due_date ? new Date(task.due_date) : null;

  if (subjectFilter && task.subject !== parseInt(subjectFilter)) return false;
});
```

Рисунок 3.19 – Фільтрація задач

3.3.3 Взаємодія між клієнтською та серверною частинами

У сучасних вебдодатках важливою складовою є ефективна взаємодія між клієнтською частиною (frontend) та серверною частиною (backend). У реалізованому проєкті було використано архітектуру типу «клієнт-сервер», де клієнтська частина, створена з використанням бібліотеки React, взаємодіє з серверною частиною, реалізованою на Django.

Frontend надсилає HTTP-запити до REST API, реалізованого на Django REST Framework. Усі запити реалізовано через модуль frontend/src/api/index.js, що містить функції для обробки:

- авторизації користувача;
- отримання та створення завдань;
- оновлення профілю;
- роботи з дошками, колонками, предметами тощо.

API використовує формат JSON як для надсилання, так і для отримання даних. Наприклад, для створення нового завдання клієнтська частина надсилає POST-запит на /tasks/, передаючи у тілі запиту об’єкт з полями title, description, subject, column тощо. У відповідь сервер повертає створене завдання з унікальним ідентифікатором та додатковими метаданими.

Обробка запитів у Django REST Framework здійснюється у модулі views.py, а перетворення моделей у зручний для передачі формат реалізовано у serializers.py. Це забезпечує чіткий розподіл відповідальностей між обробкою даних, логікою бізнес-процесів і представленням інформації.

Взаємодію клієнта і сервера відображено на схемі на рис. 3.20.

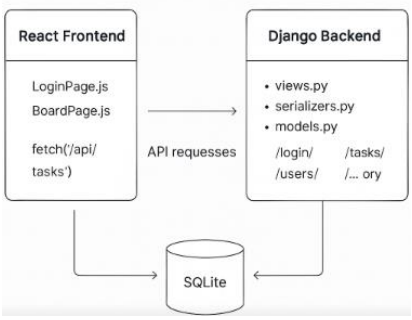

Рисунок 3.20 – Схема взаємодії клієнта (React) і сервера (Django REST API)

Крім того, у клієнтській частині використовується localStorage для збереження токена авторизації, що дозволяє підтримувати сесію користувача між перезавантаженнями сторінки. Заголовки з токеном автоматично додаються до кожного запиту API.

Таке розділення дозволило досягти незалежності між частинами системи: frontend може оновлюватися без змін backend, і навпаки. Це також спрощує масштабування проєкту та розробку окремих модулів різними командами.

3.4 Керівництво користувача

Доступ до системи надається у разі авторизації (рис. 3.21).

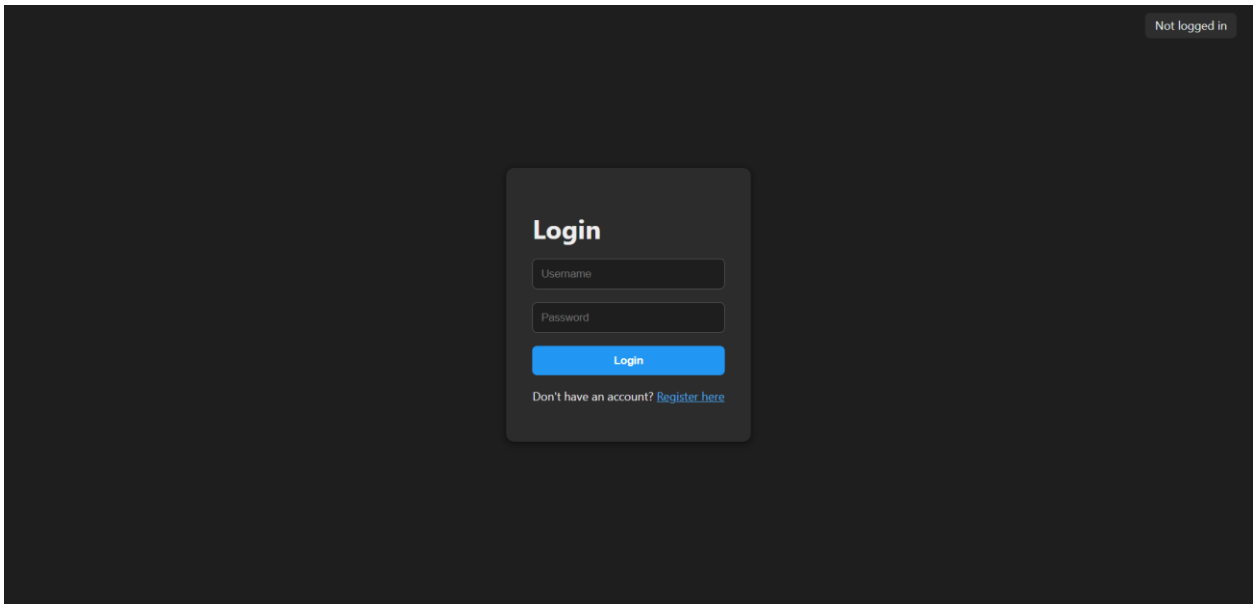

Рисунок 3.21 – Сторінка авторизації

У разі відсутності облікового запису користувач має можливість його створити. Система передбачає дві ролі: викладач та учень – тому під час реєстрації можна поставити прапорець, який позначає вчителя.

Сторінку реєстрації показано на рис. 3.22.

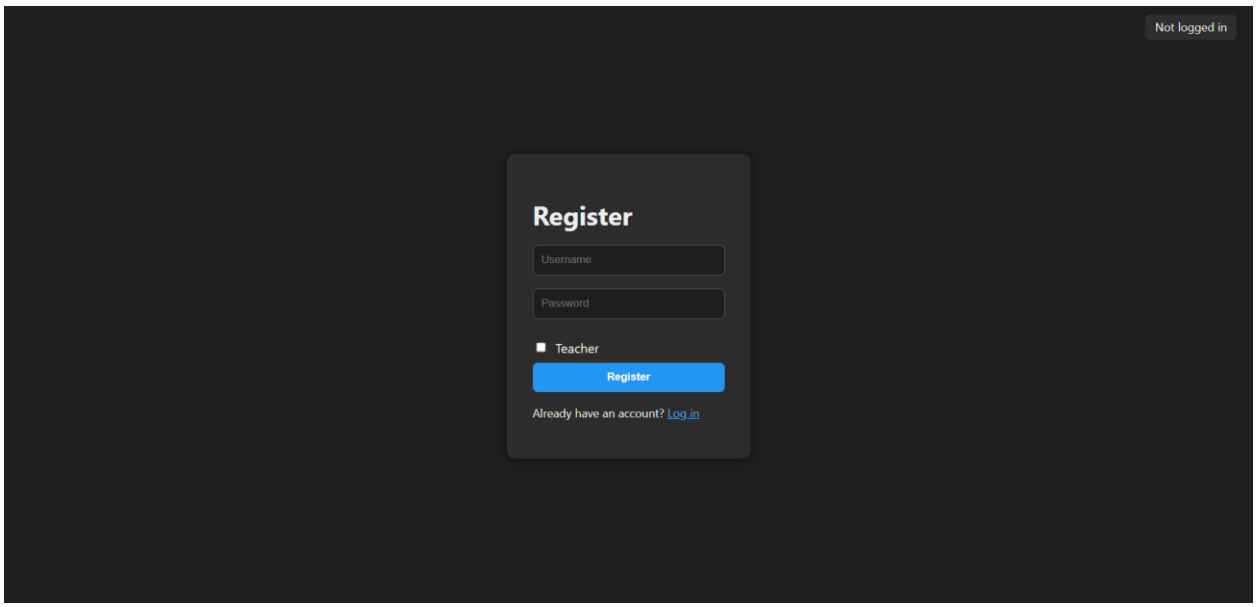

Рисунок 3.22 – Сторінка реєстрації

Після автентифікації студенти мають доступ до створення нової дошки, де вказують її назву та контролюють колонки (рис. 3.23). На цій сторінці також доступний перехід на навчальні та факультативні завдання.

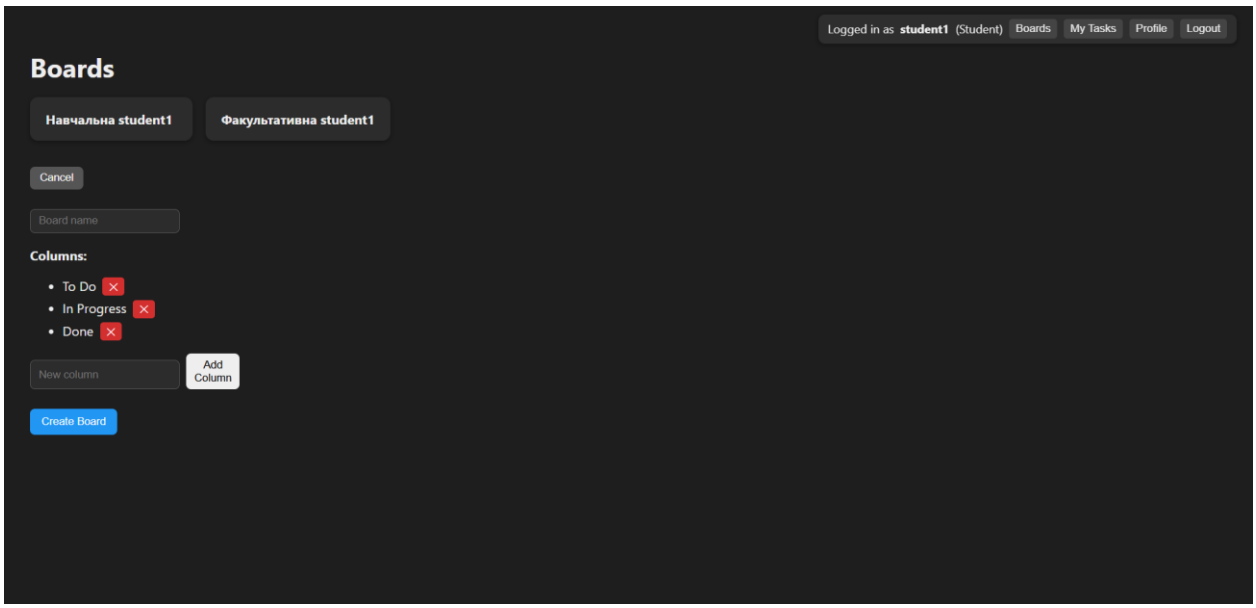

Рисунок 3.23 – Створення нової дошки

На сторінках факультативних та навчальних завдань відображаються дошки з наявною можливістю фільтрації, коментування та переміщення. Наприклад, на рис. 3.24 показано факультативні завдання студента з історії.

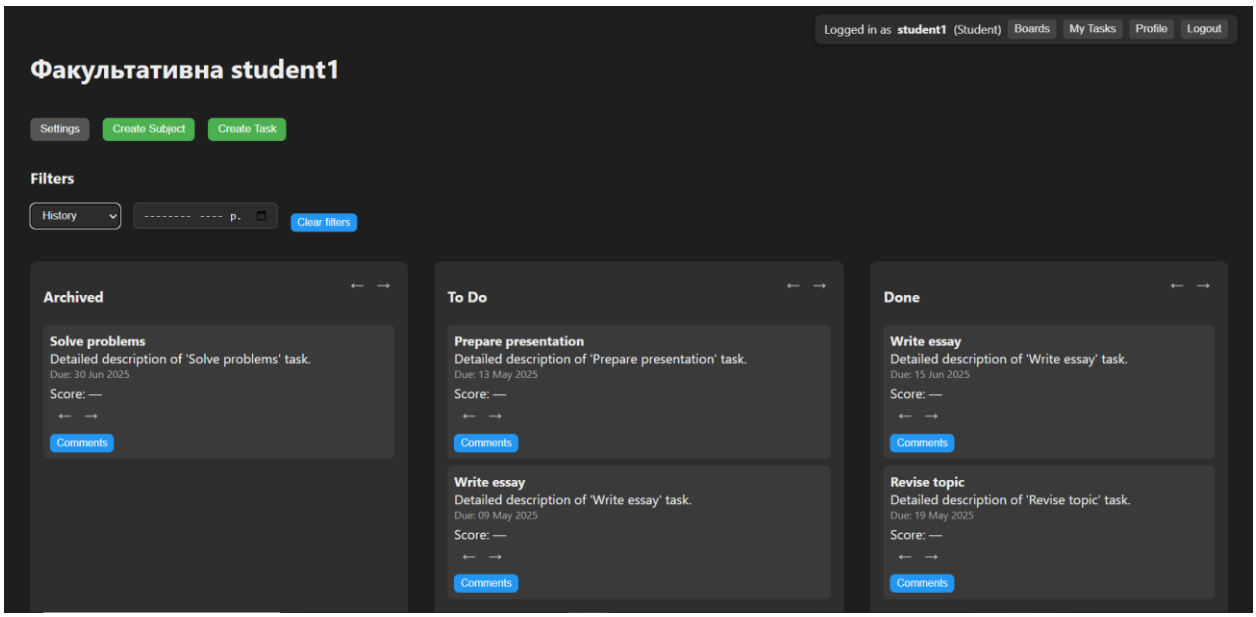

Рисунок 3.24 – Факультативні завдання студента

Завдання можна фільтрувати не лише за дисципліною, а й за датою, що показано на рис. 3.25.

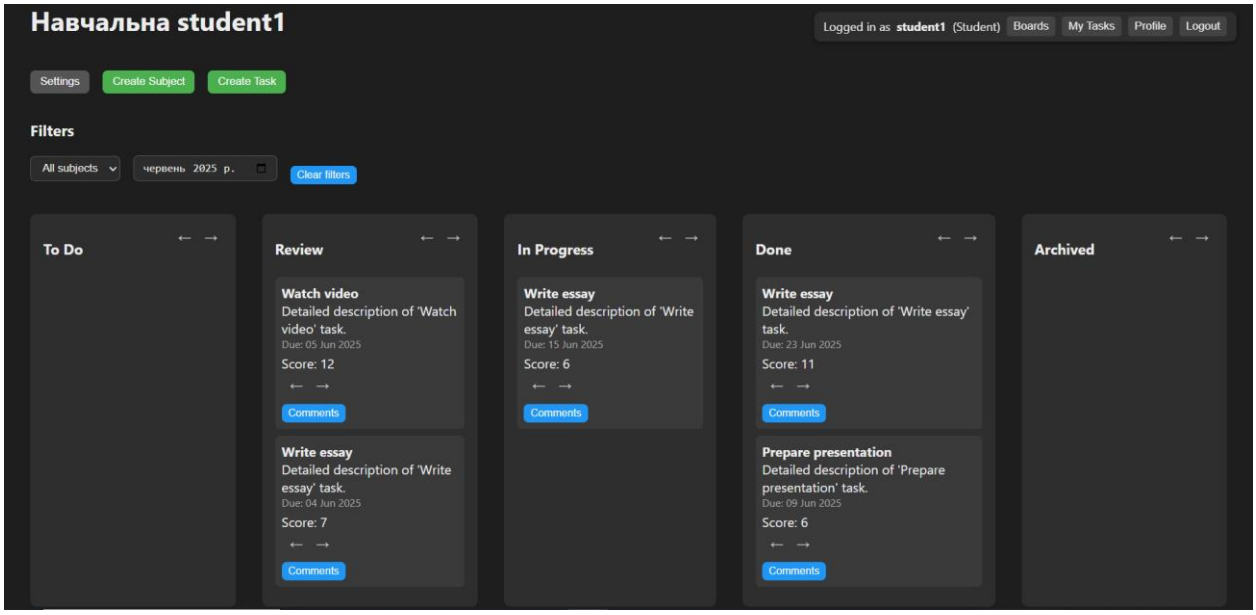

Рисунок 3.25 – Навчальні завдання студента

Окремо студент може переглянути список власних завдань, де також наявна можливість фільтрації за предметом та станом, який може свідчити про виконання, прострочення та наближення терміну здачі (рис. 3.26).

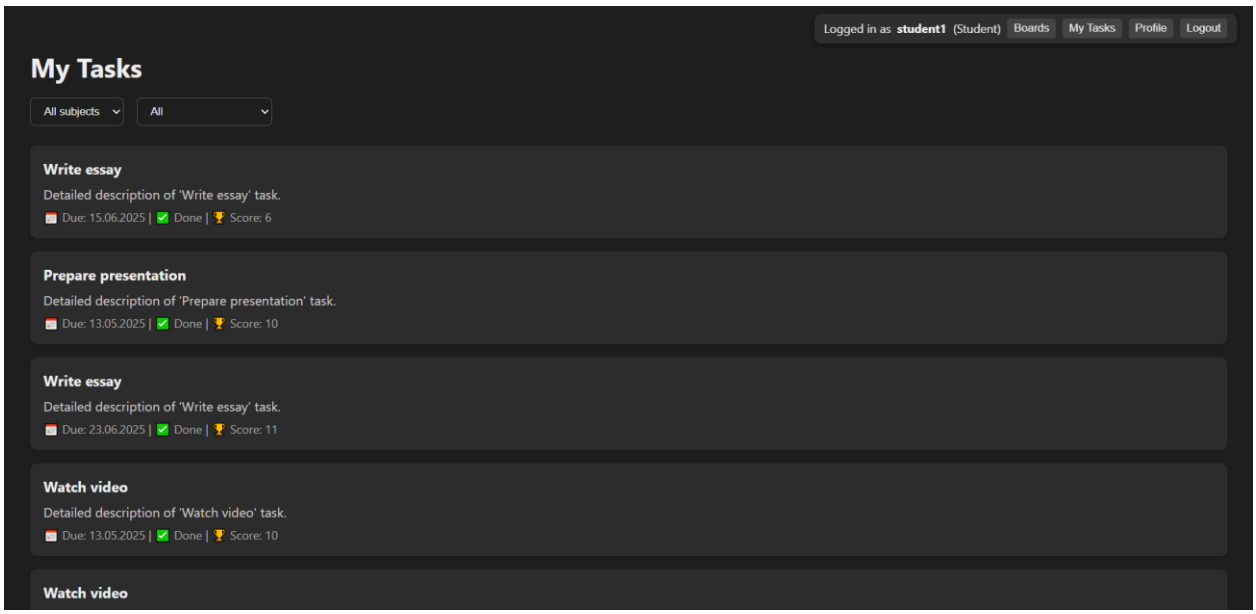

Рисунок 3.26 – Завдання студента

Якщо у фільтрах обрати завдання, термін яких спливає протягом 2 тижнів, відображається наступне (рис. 3.27):

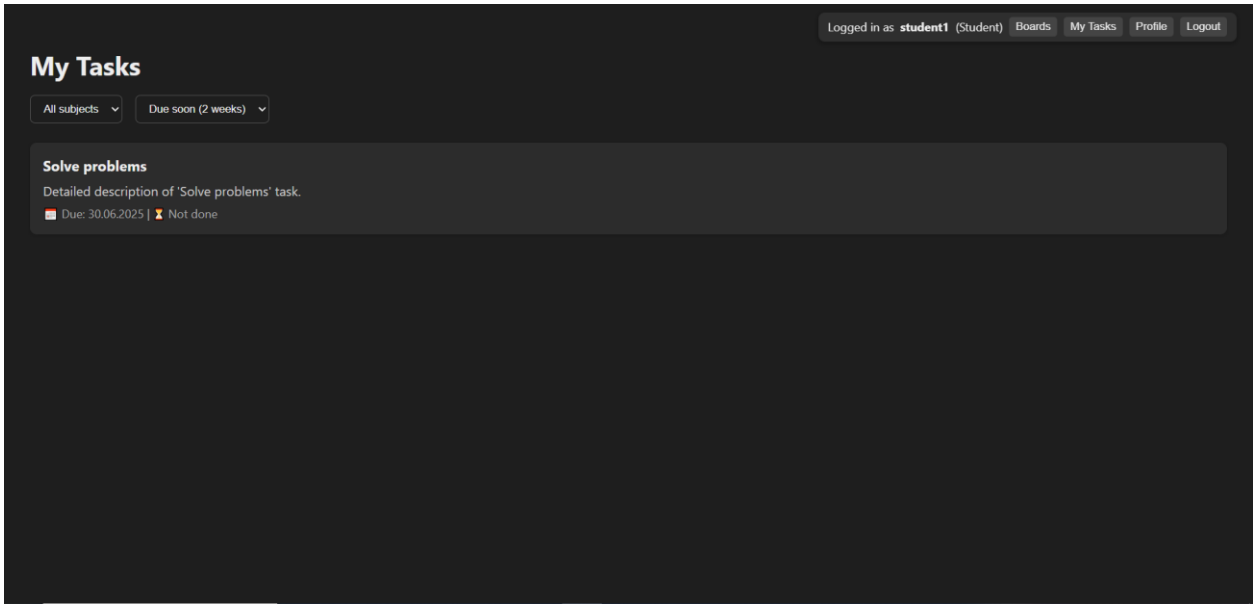

Рисунок 3.27 – Фільтрація завдань за спливаючим терміном виконання

На особистому профілі відображається успішність студента (рис. 3.28).

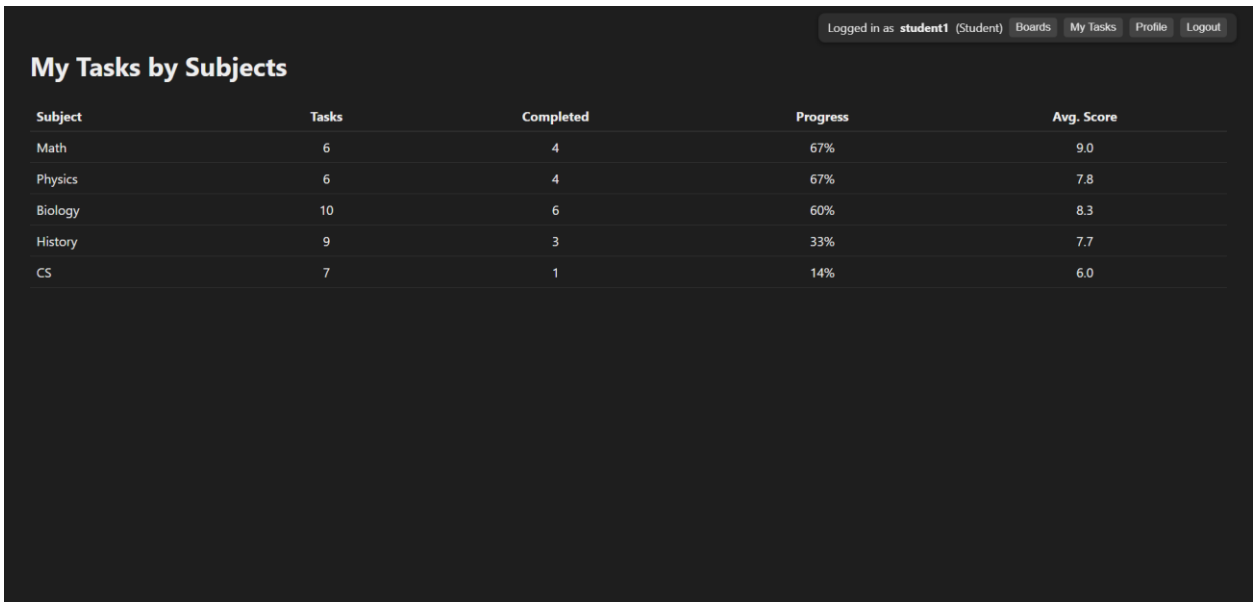

Рисунок 3.28 – Успішність студента

Викладач після автентифікації може бачити усі наявні дошки для різних студентів (рис. 3.29).

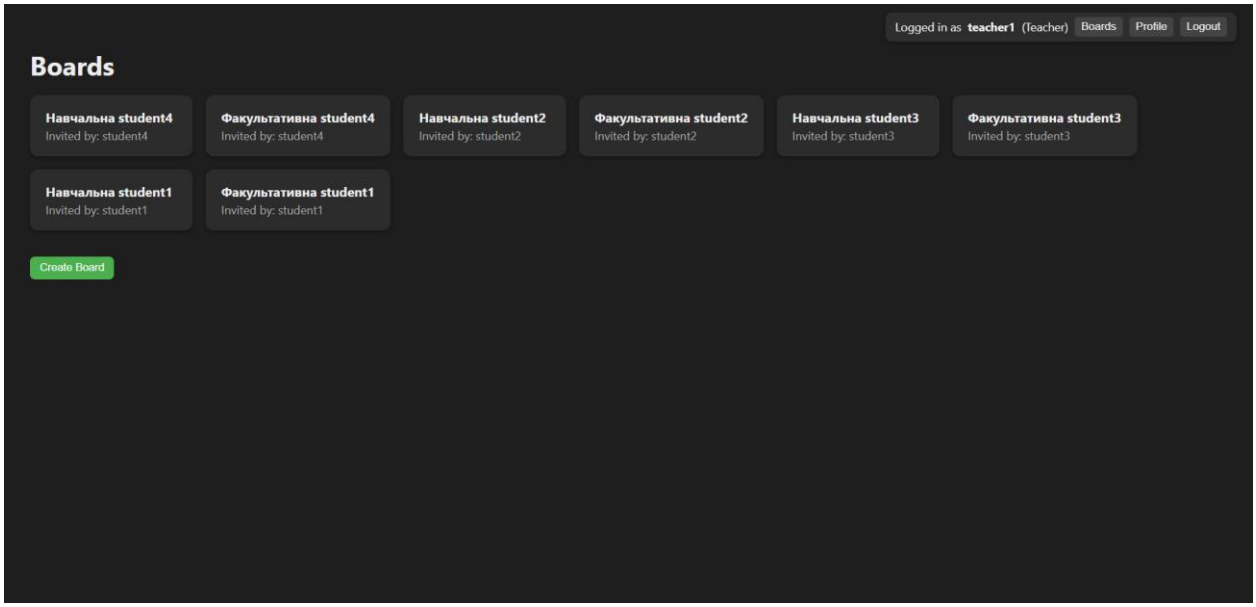

Рисунок 3.29 – Дошки викладача

В будь-якій дошці викладач має можливість залишати коментар та оцінку, що можна побачити на прикладі факультативних завдань з біології для студента 1 (рис. 3.30).

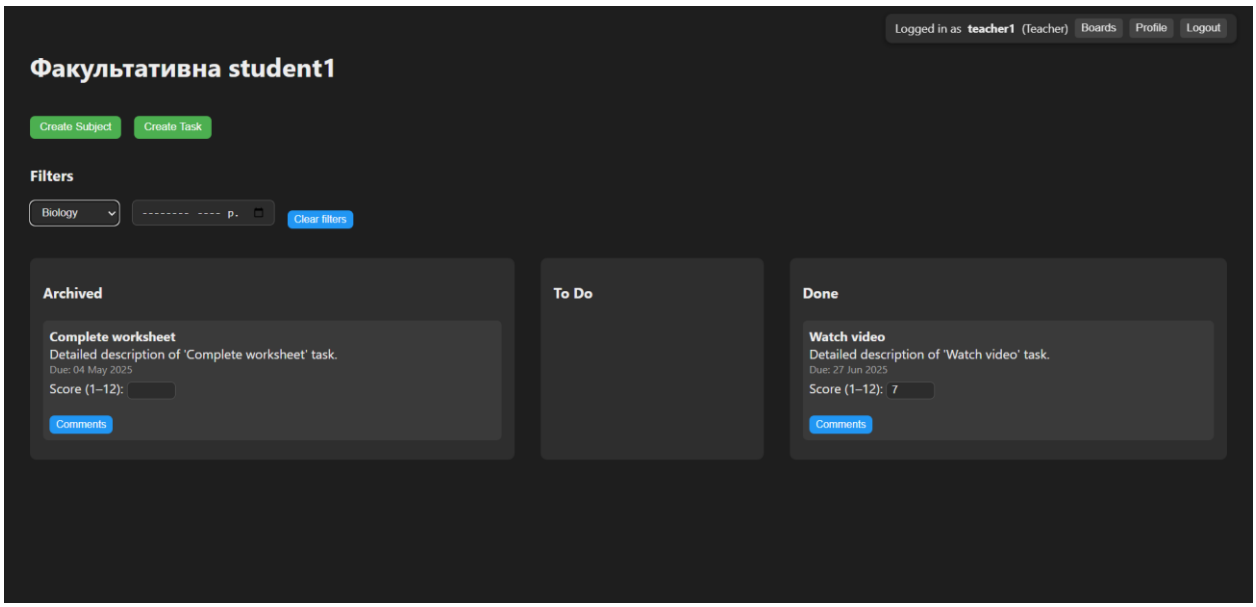

Рисунок 3.30 – Факультативні завдання студента з боку викладача

На власному профілі викладач може спостерігати за успішністю студентів (рис. 3.31).

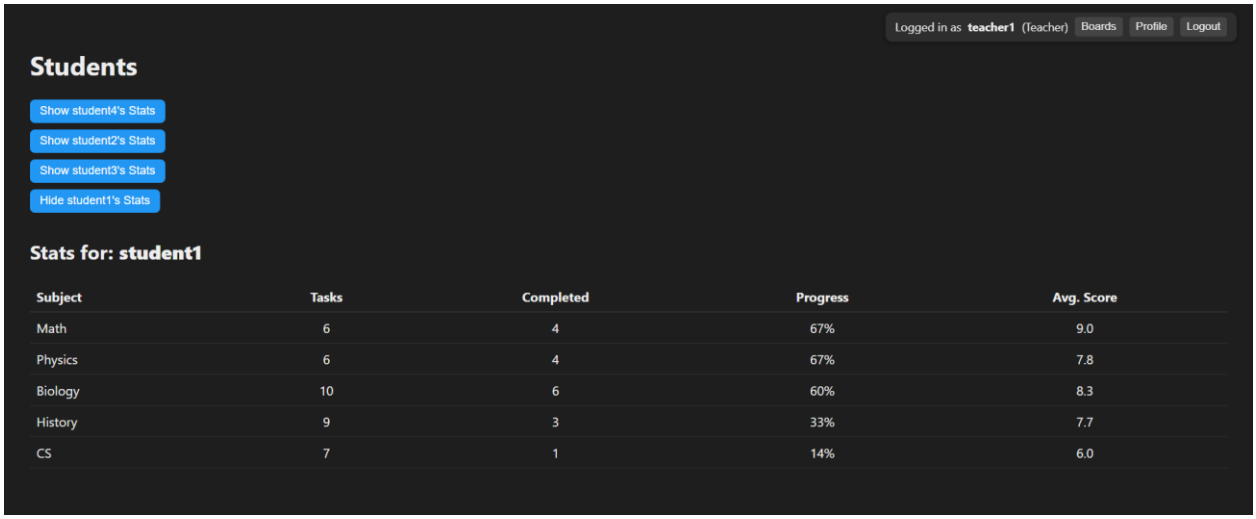

Рисунок 3.31 – Відображення успішності студентів у профілі викладача

Висновки до розділу

У розділі було детально розглянуто вибір і обґрунтування сучасного технологічного стеку для розробки вебзастосунку, що забезпечує ефективну організацію навчального процесу. Використання Python і фреймворку Django

для серверної частини дозволило реалізувати безпечний, масштабований і гнучкий бекенд з підтримкою REST API, а застосування JavaScript (ES6) і бібліотеки React для фронтенду забезпечило створення інтерактивного, зручного для користувача інтерфейсу.

Було описано структуру проєкту, основні компоненти бекенд- і фронтенд-частин, а також взаємодію між ними через HTTP-запити і збереження сесійних даних у браузері. Окремо наведено вимоги до технічного та програмного забезпечення, що гарантують коректну роботу системи, а також продемонстровано функціональні можливості, реалізовані в проєкті, зокрема систему аутентифікації, управління дошками, завданнями, коментарями та ролями користувачів.

Таким чином, обрана архітектура і технології забезпечують надійність, гнучкість і масштабованість системи, що є важливими умовами для подальшого розвитку і підтримки вебзастосунку у сфері освітніх технологій.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

4.1 Регулювання питань охорони праці на законодавчому рівні

Охорона праці є системою заходів, які спрямовані на збереження життя і здоров'я працівників під час їх трудової діяльності. Її мета – запобігання виробничим травмам, професійним захворюванням і створення належних умов праці.

Європейські стандарти організації праці встановлюють основні принципи гарантування безпеки й гігієни праці, вони визначають міжнародне трудове право. Таким чином, у співпраці з різними країнами, міжнародні стандарти сприяють підвищенню рівня охорони праці, забезпеченню прав працівників на належні умови у ході трудової діяльності та створенню єдиного підходу до управління ризиками.

Діючим прикладом гармонізації та уніфікації законодавства України у сфері охорони праці є застосування в якості ДСТУ стандарта ISO 45001:2019 [13, с. 162] – ДСТУ ISO 45001:2019 «Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування» [14]. Цей стандарт встановлює вимоги до системи управління охороною праці, згідно з якими відбувається своєчасне виявлення ризиків та контроль над протидією.

Законодавство України у сфері охорони праці створює правове підґрунтя для формування безпечного та здорового середовища на підприємствах, в організаціях та установах.

Закон України «Про охорону праці» [15] є основою для забезпечення охорони праці, оскільки визначає загальні принципи організації безпечних і здорових умов праці. Згідно зі статтею 13, роботодавець зобов'язаний надавати працівникам засоби індивідуального захисту, проводити навчання й інструктажі з охорони праці, а також здійснювати контроль за дотриманням

встановлених вимог безпеки. Відповідальність за належні умови праці, які не шкодитимуть життю та здоров'ю людей у процесі трудової діяльності, оцінку професійних ризиків та усування небезпек лежить саме на роботодавцеві.

Кодекс законів про працю України [16] є базовим нормативно-правовим актом, що регулює трудові відносини і встановлює гарантії безпечних умов праці. Він визначає права та обов'язки сторін трудового договору, механізми контролю за дотриманням трудового законодавства та гарантує захист працівників від несприятливих і небезпечних факторів на робочому місці.

Згідно зі статтею 54 Закону України «Про освіту» [17], педагогічні, науково-педагогічні та наукові працівники мають право на працю у безпечному та здоровому освітньому середовищі. Це положення закріплює відповідальність керівництва закладу освіти за створення відповідних умов для викладачів, у тому числі – під час організації дистанційного навчання.

4.2 Виявлення потенційних небезпек стосовно об'єкту проєктування

Для викладачів вищих навчальних закладів, незалежно від форми навчання, існує ряд потенційних небезпек, які можуть негативно впливати на їхнє здоров'я та працездатність у процесі професійної діяльності.

Фізичні небезпечні й шкідливі виробничі фактори можуть включати надмірне навантаження на зоровий апарат через тривалу роботу за комп'ютером, недостатню або надмірну освітленість робочої зони, підвищений рівень електромагнітного випромінювання від електронної техніки або шуму і вібрації в приміщенні. Крім того, негативний вплив можуть чинити й невідповідна ергономіки робочого місця або ж знижена чи підвищена температура повітря. Усі згадані шкідливі фактори є небезпечними для здоров'я викладачів, оскільки вони можуть спричиняти зорову втому, головний біль, порушення сну, зниження концентрації уваги, розвиток опорно-рухових розладів та загальне погіршення фізичного стану.

Хімічні небезпечні й шкідливі виробничі фактори можуть проявлятися у вигляді впливу шкідливих речовин, які застосовують для дезінфекції приміщень, прибирання чи ремонту обладнання. Вдихання токсичних, дратівних або сенсибілізуючих речовин може викликати подразнення дихальних шляхів, алергічні реакції та загострення хронічних захворювань.

Біологічними небезпечними й шкідливими виробничими факторами можуть бути такі патогенні мікроорганізми, як віруси, бактерії, гриби, які призводять до інфекційних захворювань та зниження імунітету, особливо якщо трудовий процес супроводжується скупченням людей у навчальних закладах.

До ряду психофізіологічних небезпечних та шкідливих виробничих факторів належать перенавантаження, пов’язані з високою розумовою активністю, емоційний стрес, викликаний професійними та соціальними чинниками, та монотонність роботи. Внаслідок тривалого перебування в сидячому положенні може розвиватися хронічна втома та погіршуватися самопочуття.

В умовах воєнного стану, з міркувань безпеки, дистанційна форма освіти може запроваджуватися як єдино можлива форма здобуття освіти на всій території України або в окремих місцевостях [18, с. 75].

Навчальний процес у дистанційному форматі повинен здійснюватися відповідно до вимог безпечного освітнього середовища, що передбачає організацію робочого місця викладача з дотриманням ергономічних норм, забезпечення належного рівня освітлення, вентиляції та мінімізації впливу шкідливих фізичних факторів.

Дистанційне навчання в умовах воєнного стану може супроводжуватися такими загрозами, як відсутність електропостачання, нестабільний інтернет-зв’язок, а також повітряні тривоги, спричинені атаками ворога, що створюють реальну небезпеку для життя викладачів і негативно впливають на їх психологічне здоров’я.

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проєктування та розробка заходів щодо їх попередження

Одним з ключових елементів ефективної системи управління охороною праці є процес оцінки ризиків на робочих місцях, який дозволяє ідентифікувати потенційні небезпеки та розробити заходи для їх усунення чи мінімізації [19, с. 242].

Оцінка ризику – це процедура визначення ступеня ймовірності виникнення небезпечних подій та їх можливих наслідків для здоров’я і життя працівників у процесі виконання трудових обов’язків. Вона включає збір і аналіз інформації про потенційні небезпеки, оцінку ймовірності їх настання та визначення рівня ризику для прийняття обґрунтованих рішень щодо заходів безпеки.

Основними задачами процедури оцінки ризику є:

- ідентифікація потенційних небезпек;
- визначення ймовірності виникнення ризиків;
- оцінка можливих наслідків для здоров’я і безпеки працівників;
- класифікація ризиків за рівнем небезпеки;
- створення заходів для зниження або повного усунення ризиків;
- контроль за ефективністю впроваджених заходів.

У результаті процедури оцінки ризику формується обґрунтована оцінка рівня небезпеки на робочому місці, що дозволяє своєчасно вжити необхідні заходи для знешкодження виявлених загроз задля запобігання травмам, професійним захворюванням та створення безпечних умов праці.

Однією з найпоширеніших небезпек під час трудової діяльності викладачів вищих навчальних закладів, особливо в умовах дистанційного навчання, є надмірне навантаження на зоровий апарат. Даний ризик спричиняється рядом факторів, які розглянуто на рисунку 4.1. Надмірне навантаження на зоровий апарат несе шкоду для зору, напружуючи очні м’язи, спричиняючи втому, подразнення, сльозотечу, а також синдром сухого ока. У

разі систематичного впливу без належного профілактичного втручання можуть спостерігатися погіршення гостроти зору, головні болі, зниження концентрації уваги та загальна втома, що впливає на ефективність професійної діяльності викладача та його самопочуття.

Рисунок 4.1 – Дерево відмов небезпеки надмірного навантаження на зоровий апарат

Для запобігання шкоди здоров'ю викладачів внаслідок надмірного навантаження на зоровий апарат важливо впровадити ряд заходів:

– регулярні перерви під час роботи за комп'ютером;

– забезпечення правильного освітлення в робочій зоні з урахуванням природного та штучного світла й уникнення відблисків на екрані;

– використання якісних моніторів, які не створюють додаткового навантаження на зір;

– розміщення монітора на правильні відстані від очей;

– проведення періодичних профілактичних оглядів у офтальмолога.

Серйозні наслідки для стану здоров'я та ефективності праці викладачів вищих навчальних закладів несе й професійне вигорання. Воно може бути пов'язане з багатьма факторами, які стосуються психоемоційного виснаження, недостатністю відпочинку та незадоволенням власною діяльністю (рис. 4.2).

Професійне вигорання може призводити до таких негативних наслідків, як постійна втома, зниження мотивації, тривожність, що безпосередньо впливає на якість викладання, взаємодію зі студентами та загальний психофізичний стан викладача.

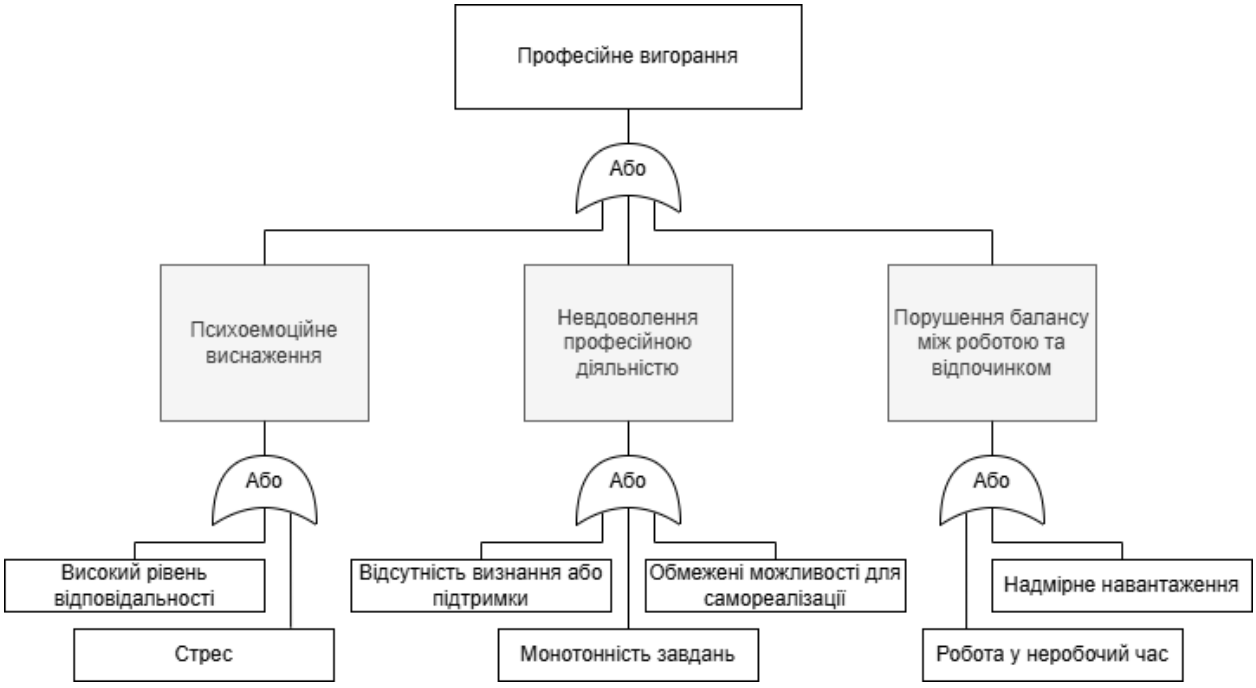

Рисунок 4.2 – Дерево відмов небезпеки професійного вигорання

Для протидії професійному вигоранню викладачів необхідно впровадити комплекс профілактичних та організаційних заходів, зокрема:

– планування робочого часу з врахуванням обмеження понаднормових навантажень;

– психологічна підтримка у вигляді консультацій з фахівцями або тренінгів зі збереження ментального здоров’я;

– створення середовища, в якому визнаються професійні досягнення;

– наявність можливості підвищення кваліфікації;

– застосування цифрових інструментів для автоматизації рутинних завдань з метою зниження навантаження та підвищення ефективності праці.

З метою попередження небезпек, які можуть виникати під час професійної діяльності викладачів, доцільно оптимізувати робочий графік з

урахуванням перерв для зменшення зорового та психоемоційного навантаження, надавати доступ до цифрових ресурсів та оновлювати програмне забезпечення, а також проводити зустрічі для формування навичок безпечної роботи за комп’ютером та управління стресом.

Висновки до розділу

У даному розділі було розглянуто, яку роль відіграє законодавство в охороні праці, зокрема і для викладачів вищих навчальних закладів. Було проаналізовано такі нормативно-правові акти, як Закон України «Про охорону праці», Кодекс законів про працю та Закон України «Про освіту». Крім того, було визначено важливість міжнародних стандартів та впровадження ДСТУ ISO 45001:2019.

У ході дослідження було виявлено низку потенційних небезпек, пов’язаних з професійною діяльністю викладачів, що включає фізичні, хімічні, біологічні та психофізіологічні чинники. Було приділено увагу забезпеченню належних охорони праці в умовах дистанційного навчання та воєнного стану.

На основі визначених ризиків було побудовано дерева відмов для двох ключових загроз – надмірного навантаження на зоровий апарат і професійного вигорання – та розроблено відповідні заходи для зменшення їх впливу. Рекомендовано впровадження перерв у роботі, оптимізацію освітлення, використання якісного обладнання, психологічну підтримку викладачів, цифрові інструменти для автоматизації рутинних завдань та організаційні рішення, спрямовані на збереження ментального і фізичного здоров’я.

ВИСНОВКИ

У ході роботи було проаналізовано предметне середовище, пов’язане зі створенням веб-системи для онлайн-навчання, що включало розгляд таких ролей, як викладач та студент, та побудову відповідних діаграм. Огляд наявних аналогів платформ для навчання дозволив виокремити ряд переваг та недоліків для врахування у розробці.

Було визначено вхідні та вихідні дані, проблематику організації онлайн-навчання та розглянуто проєктування бази даних для створеної веб-системи. Крім того, було спроєктовано об’єктно-орієнтовану модель, яка включала розгляд класів системи. Окремо було розглянуто роль математичного та алгоритмічного забезпечення у розробці платформи для дистанційного навчання.

Створення програмного та технічного забезпечення включало обґрунтування обраних засобів для реалізації веб-системи та розгляд вимог. Програмна реалізація включала створення бекенд- та фронтенд-частин за допомогою розглянутих засобів, а також реалізацію взаємодії між клієнтською та серверною частинами. У керівництві користувача було подано опис використання веб-системи як з боку вкиладача, так і з боку студента.

Результатом роботи стала веб-система, яка дозволяє організовувати повноцінний навчальний процес онлайн, забезпечуючи базовий функціонал для керування курсами, користувачами, завданнями, тестуванням, комунікацією між викладачами та студентами, а також відображенням прогресу навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головна сторінка платформи Moodle [Електронний ресурс] // Режим доступу: https://moodle.org

2. Головна сторінка платформи Google Classroom [Електронний ресурс] // Режим доступу: https://classroom.google.com/

3. Головна сторінка платформи Canvas LMS [Електронний ресурс] // Режим доступу: https://www.instructure.com/canvas

4. Головна сторінка платформи Udemy [Електронний ресурс] // Режим доступу: https://www.udemy.com/

5. Головна сторінка платформи Coursera [Електронний ресурс] // Режим доступу: https://www.coursera.org/

6. Головна сторінка Python [Електронний ресурс] // Режим доступу: https://www.python.org/

7. Головна сторінка Django [Електронний ресурс] // Режим доступу: https://www.djangoproject.com/

8. JavaScript ES6 [Електронний ресурс] // Режим доступу: https://262.ecma-international.org/6.0/

9. React – JavaScript Library for Building User Interfaces [Електронний ресурс] // Режим доступу: https://react.dev/

10. Django REST framework [Електронний ресурс] // Режим доступу: https://www.django-rest-framework.org/

11. SQLite Home Page [Електронний ресурс] // Режим доступу: https://www.sqlite.org/

12. PyCharm: the Python IDE for Professional Developers [Електронний ресурс] // Режим доступу: https://www.jetbrains.com/pycharm/

13. Малько О.Д., Бригада О.В., Цимбал Б.М. Адаптація нормативно-правового забезпечення охорони праці до європейських стандартів. Комунальне господарство міст. 2022. Т. 6, № 173. С. 160–169.

[Електронний ресурс] // Режим доступу: https://doi.org/10.33042/2522-1809-2022-6-173-10-160-169

14. ДСТУ ISO 45001:2019. Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування. На заміну ДСТУ OHSAS 18001:2010 ; чинний від 2021-01-01. Київ : ДП «УкрНДНЦ», 2019.

15. Про охорону праці : Закон України від 14.10.1992 № 2694-XII : станом на 4 квіт. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

16. Кодекс законів про працю України : Кодекс України від 10.12.1971 № 322-VIII : станом на 2 трав. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08#Text

17. Про освіту : Закон України від 05.09.2017 № 2145-VIII : станом на 1 черв. 2025 р. [Електронний ресурс] // Режим доступу: https://zakon.rada.gov.ua/laws/show/2145-19#Text

18. Бондарчук І. М., Горащенко І. І, Олійник І. М. Удосконалення навчання з безпеки життєдіяльності та охорони праці в закладах фахової передвищої освіти в умовах воєнного стану. Охорона праці: освіта і практика. Проблеми та перспективи розвитку охорони праці: Зб. наук. праць III Всеукраїнської науково–практичної конференції викладачів та фахівців–практиків та XIII Всеукраїнської науково-практичної конференції курсантів, студентів, аспірантів та ад'юнктів. – Львів: ЛДУ БЖД, 2023. С. 75–76. [Електронний ресурс] // Режим доступу: http://repositsc.nuczu.edu.ua/bitstream/123456789/18948/1/%D0%97%D0%B1%D1%96%D1%80%D0%BD%D0%B8%D0%BA%20%D0%9A%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D1%8F%20%D0%9E%D0%9F%202023_29.05.pdf#page=75

19. Власюк О.М., Березюк О.В. Застосування програмного забезпечення для оцінки ризиків на робочому місці. 2024. Матеріали V Міжнародної науково-практичної інтернет-конференції 12-13 листопада 2024 р. С. 242–244. [Електронний ресурс] // Режим

доступу: https://science.kname.edu.ua/images/dok/konferentsii/2024/Tezy_2024/Materiali%20konferencii%20HNUMG%202024.pdf#page=242