МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Онлайн платформа дистанційного навчання, з модулем рейтингу успішності студента, виконання практичних робіт та системою тестування і перевірки знань»

Виконав студент групи ІПЗ-20-3
ЦВІК Олександр Сергійович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Завідувач кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЦВІК Олександр Сергійович**
Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**
Тема роботи: **«Онлайн платформа дистанційного навчання, з модулем рейтингу успішності студента, виконання практичних робіт та системою тестування і перевірки знань»**,
затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74с**
Вихідні дані для роботи: **процес організації та проведення дистанційного навчання, що охоплює етапи взаємодії користувачів з веборієнтованою платформою, включаючи виконання практичних завдань, проходження тестування та перегляд успішності.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Толстой І. А.	27.02.2024р.	27.02.2024р.
2	Толстой І. А.	27.02.2024р.	27.02.2024р.
3	Толстой І. А.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з програмного комплексу вебдодатку онлайн платформи дистанційного навчання та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить: список використаних джерел з 40 найменувань; в роботі наведено 39 рисунків та 30 таблиць; загальний обсяг роботи складає 98 сторінок, з них 87 сторінок основного тексту.

Метою роботи є розробка веборієнтованої платформи дистанційного навчання, спрямованої на забезпечення ефективної організації навчального процесу в онлайн-форматі із використанням сучасних технологій та засобів розробки.

У роботі було визначено функціональні та нефункціональні вимоги до платформи, а також обґрунтовано вибір архітектури системи. Також було проведено аналіз існуючих аналогів систем дистанційного навчання та обрано відповідні інструменти для реалізації серверної та клієнтської частин. Розроблено структуру бази даних для ефективного зберігання та обробки навчальних матеріалів, завдань, тестів та результатів студентів.

КЛЮЧОВІ СЛОВА: ASP.NET CORE, EF CORE, ORM, SPA, БД, ВЕБДОДАТОК, ЧИСТА АРХІТЕКТУРА.

ABSTRACT

The bachelor's thesis consists of a software complex of a web application for an online distance learning platform and an explanatory note. The explanatory note to the qualification work contains: a list of 40 references; the work contains 39 figures and 30 tables; the total volume of the work is 98 pages, including 87 pages of the main text.

The aim of the work is to develop a web-based distance learning platform aimed at ensuring the effective organization of the educational process in an online format using modern technologies and development tools.

The paper identifies functional and non-functional requirements for the platform and justifies the choice of system architecture. Also, an analysis of existing analogues of distance learning systems was conducted and appropriate tools were selected to implement the server and client parts. A database structure was developed for the efficient storage and processing of training materials, assignments, tests, and student results.

KEYWORDS: ASP.NET CORE, EF CORE, ORM, SPA, DATABASE, WEB APPLICATION, CLEAN ARCHITECTURE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс (Application Programming Interface).

CRUD – 4 основні функції управління даними: створення (create), читання (read), оновлення (update) та видалення (delete).

ERD – модель «сутність-зв’язок» (Entity-Relationship Diagram).

GUID – глобальний унікальний ідентифікатор (Globally Unique Identifier).

JSON – запис об’єктів JavaScript (JavaScript Object Notation).

JWT – вебтокен JSON (JSON Web Tokens).

БД – база даних.

ЗК – зовнішній ключ.

ПЗ – програмне забезпечення.

ПК – первинний ключ.

СУБД – система управління базами даних.

ВСТУП

Актуальність теми. У сучасному світі, де технології швидко розвиваються, інтернет відіграє важливу роль у житті людей, він дає безліч можливостей та способів для розваг та розвитку людей, це місце де можна отримати якісну освіту не виходячи з дому. Дистанційна освіта стає все популярнішою через збільшення доступності та простоти у використанні, вона дозволяє економити час та гроші, адже не потрібно кудись збиратись та їхати.

Системи електронного навчання стають необхідним елементом для забезпечення доступу до освіти в будь-якому місці та у будь-який час. Впровадження даних систем дає можливість полегшити доступ до навчальних матеріалів, сприяє ефективній взаємодії студентів та викладачів, а також оптимізує процеси оцінювання.

Студенти мають можливість працювати з різними типами навчальних матеріалів, наприклад, презентаціями, тестами, лабораторними завданнями чи посиланнями на інші джерела, при цьому навчаючись у власному темпі та в зручний для них час. Викладачі можуть створювати різні курси та наповнювати їх різноманітними матеріалами, виставляти оцінки тощо. Така система не лише допомагає у засвоєнні матеріалу, а й стимулює до саморозвитку всіх учасників навчального процесу.

Метою роботи є розробка системи електронного навчання. Система має забезпечувати функціонал, який спрямований на оптимізацію процесів навчання та надасть студентам і викладачам можливість ефективно взаємодіяти між собою.

Поставлена мета обумовлює необхідність виконання наступних завдань:

– провести аналіз функціональних потреб системи;

– обрати оптимальну в сучасних умовах архітектуру функціонування програмного комплексу;

– визначити технології та інструменти для реалізації;

– здійснити проєктування структури БД й основних алгоритмів платформи;

– розробити програмне забезпечення та перевірити на відповідність вимогам.

Об’єктом дослідження є процес організації та проведення дистанційного навчання, що охоплює етапи взаємодії користувачів з веборієнтованою платформою, включаючи виконання практичних завдань, проходження тестування та перегляд успішності.

Предметом дослідження є методи та засоби проєктування і розробки онлайн платформи дистанційного навчання, з модулем рейтингу успішності студента, виконання практичних робіт та системою тестування і перевірки знань.

Розроблена онлайн платформа дистанційного навчання, яка включає модуль рейтингу успішності студентів, виконання практичних робіт та систему тестування і перевірки знань, може бути використана закладами вищої освіти та школами для організації дистанційного навчального процесу та забезпечення зручного доступу до навчальних матеріалів.

У роботі було використано наступні методи:

– метод порівняльного аналізу існуючих онлайн-платформ дистанційного навчання;

– метод моделювання для створення діаграм та проєктування архітектури системи;

– емпіричний метод для збору вимог;

– метод системного аналізу для деталізації та структурування необхідних компонентів та функціоналу майбутньої платформи.

РОЗДІЛ 1. АНАЛІЗ АНАЛОГІВ ТА ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПЛАТФОРМИ ДИСТАНЦІЙНОГО НАВЧАННЯ З ДОДАТКОВИМИ МОДУЛЯМИ ПЕРЕВІРКИ ЗНАНЬ

1.1. Постановка задачі

Платформи та сервіси дистанційного навчання представляють собою віртуальну освітню систему, яка надає викладачам та студентам розширені можливості використання сучасних технологій для навчання на відстані. Вони забезпечують ефективну організацію взаємодії між викладачами та студентами в інтерактивному форматі, а також спрощують керування процесом дистанційного навчання. Це середовище та інструмент для передачі знань [34].

Основною метою створення платформи дистанційного навчання є спрощення процесу обміну навчальними матеріалами, керування ними та розповсюдження за допомогою інтернету. На ній викладачі можуть завантажувати матеріали курсів, створювати тести та оцінювати роботи студентів. Студенти можуть переглядати матеріали курсів й власну успішність, виконувати практичні завдання, проходити тести та отримувати зворотній зв’язок.

Назва платформи, що розробляється – Elers.

Передбачається, що цільовою аудиторією платформи дистанційного навчання будуть студенти 17-30 років та працівники закладів вищої освіти (ЗВО) 25-60 років.

Для того, щоб успішно завершити створення даної платформи потрібно виконати наступні завдання:

– проаналізувати аналоги програмних продуктів для дистанційного навчання;

– визначити функціональні та нефункціональні вимоги до платформи;

– обрати оптимальну архітектуру для функціонування системи;

– обрати мови програмування та фреймворки для розробки;

– визначити акторів та варіанти використання системи;

– спроєктувати базу даних, яку буде легко масштабувати;

– розробити програмний комплекс з наступними функціональними можливостями:

а. створення та редагування курсів, практичних робіт та тестів;

б. завантаження та перегляд навчальних матеріалів;

в. перегляд та виставлення оцінок;

г. завантаження та оцінювання практичних робіт;

д. підтримка різних типів запитань у тестах.

– створити адаптивний, зрозумілий та простий дизайн;

– провести тестування роботи додатку на відповідність поставленим вимогам.

Важливим кроком є аналіз аналогів, тобто додатків, що мають подібний функціонал. На цьому етапі потрібно визначити переваги та недоліки інших систем, які варто врахувати при розробці власної, а також знайти корисні та додаткові можливості. Виконання даного кроку дозволить уникнути помилок, які наявні в інших системах й отримати якісний та унікальний продукт.

Наступним кроком потрібно визначити функціональні та нефункціональні вимоги до системи електронного навчання, зокрема можливості перегляду та завантаження навчальних матеріалів, проведення тестування, оцінювання студентів тощо.

Розробку програмного коду цієї платформи можна розділити на три ключові етапи: проєктування бази даних, створення користувацького інтерфейсу та розробка RESTful сервера. Останнє забезпечує можливість взаємодії з функціями додатку через визначені URL-адреси та HTTP-методи, такі як GET, POST, PUT, PATCH, DELETE та інші [31].

Результатом успішної реалізації поставленого завдання є онлайн платформа дистанційного навчання, яка забезпечує доступ до широкого спектру навчальних матеріалів, тестів, практичних завдань та успішності студентів.

1.2. Дослідження аналогів систем дистанційного навчання

Для проведення дослідження аналогів систем дистанційного навчання було використано метод порівняльного аналізу. За допомогою цього методу можна визначити як спільні, так і відмінні характеристики різних платформ, шляхом порівняння їхніх однотипних властивостей, таких як функціональність, дизайн інтерфейсу, зручність використання та інші. При цьому, системи, які порівнюються, повинні бути схожими за своїми цілями, цільовою аудиторією та функціональними можливостями. Застосування порівняльного аналізу дозволяє виявити переваги та недоліки кожної системи, а також знайти нові ідеї та рішення для покращення функціональності та дизайну власного проєкту [36].

Після детального аналізу і перегляду подібних систем стало зрозуміло, що світовий ринок має багато схожих продуктів. Існує безліч сервісів, які пропонують різні функціональні можливості, які іноді є подібними, а іноді представляють унікальні рішення.

Отже, розглянемо найбільш популярні платформи для дистанційного навчання з метою виокремлення ключових особливостей та визначення аспектів, які можна впровадити у власний продукт.

1.2.1. Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment, вимовляється «Мудл») – це модульне об’єктно-орієнтоване динамічне навчальне середовище, яке називають також системою управління навчанням, курсами, віртуальним навчальним середовищем або просто платформою для навчання, яка надає викладачам та студентам великий набір інструментів для дистанційного навчання. Цю платформу можна

використовувати для навчання здобувачів освіти, при підвищенні кваліфікації, бізнес-навчанні, як в комп'ютерних аудиторіях, так і для самостійної роботи вдома [19, 40].

Moodle надає багато різного функціоналу для створення онлайн-курсів та взаємодії зі студентами. Використовуючи дану платформу, викладачі мають змогу створювати власні дистанційні курси та публікувати навчальні матеріали (тексти лекцій, практичні завдання, самостійні роботи, тести для студентів тощо) та контролювати прогрес учнів. Вона широко використовується багатьма закладами вищої та середньої освіти, адже є відмінним помічником в організації дистанційного навчання. Ця система може бути використана не тільки для організації традиційних дистанційних курсів, але й для підтримки очного навчання [38].

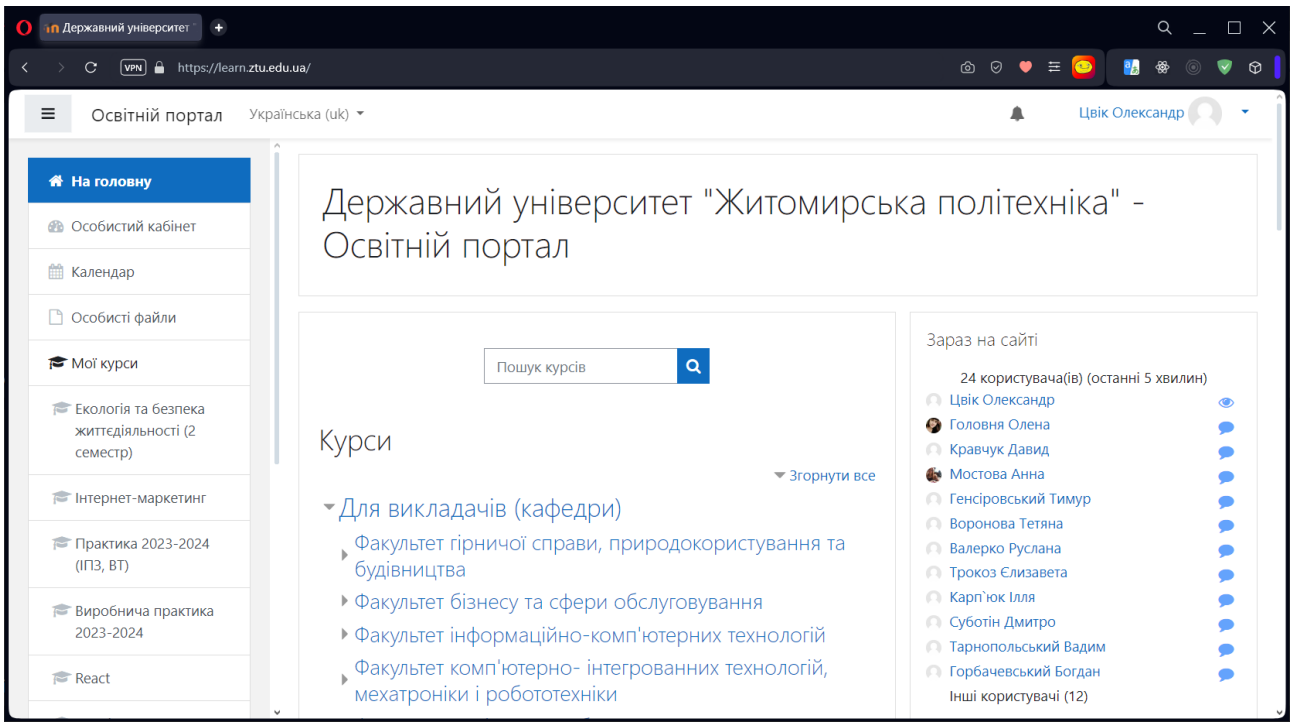

Рисунок 1.1 – Інтерфейс Moodle налаштований для Житомирської політехніки

Однак, на відміну від Google Classroom, ця платформа вимагає більш серйозного підходу і більш глибокого вивчення інструментів роботи.

Moodle повністю безкоштовна платформа, яку можна вільно завантажувати, встановлювати та змінювати. Вона відноситься до Open Source систем, тобто системам з відкритим вихідним кодом, що дозволяє

багатьом програмістам створювати додаткові, дуже корисні розширення або модулі.

Основні функціональні можливості Moodle:

– створення курсів та наповнення різноманітними матеріалами;

– можливість публікації навчального контенту різного формату – аудіо, відео, текст, різні формати файлів, каталоги тощо;

– велика кількість інструментів для управління курсами;

– містить багато функціоналу для створення різноманітних тестів;

– містить налаштування варіантів керування доступом користувачів до курсу та розміщених матеріалів;

– відстеження прогресу студентів;

– наявність особистого кабінету, який містить всі дані про користувача;

– користувачі мають можливість обмінюватися повідомленнями;

– перегляд активних користувачів на поточній сторінці.

Оскільки Moodle програма з відкритим вихідним кодом, платформа має велику кількість плагінів та доповнень до системи. Такі доповнення як правило безкоштовні, їх можна просто завантажити і встановити для своєї системи.

Прикладами таких плагінів є:

– модулі відеоконференції;

– аудіо та відео чати;

– масова розсилка повідомлень;

– засоби проєктної роботи;

– налаштування зовнішнього вигляду курсів;

– різні інтерактивні елементи ігрового підходу (кросворд, вікторина, мільйонер, судоку та інші);

– формування електронного портфоліо.

1.2.2. Google Classroom

Google Classroom є популярною платформою для електронного навчання, розробленою компанією Google. Цей інструмент значно спрощує

процес організації та створення навчальних завдань для вчителів та студентів, забезпечуючи ефективний зворотній зв’язок [11].

Цей продукт інтегрується з іншими Google-сервісами, такими як Google Диском, Документами, Календарем, Формами та Gmail. У сервісі викладач може коментувати та оцінювати отримані роботи, відсилати їх студенту на доопрацювання та повторно оцінювати після внесення правок. Крім того, викладач має можливість публікувати оголошення, які потім можуть коментувати учасники курсу [38].

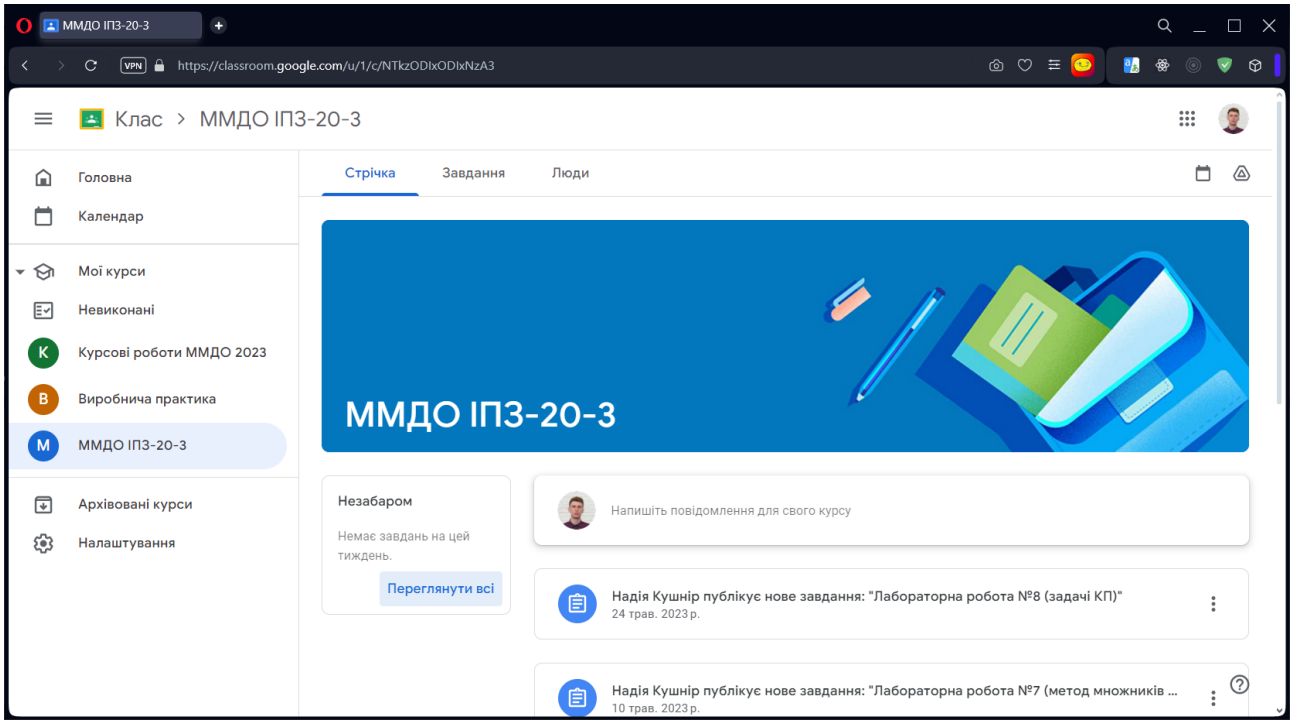

Рисунок 1.2 – Курс в Google Classroom

Google Classroom навряд чи можна назвати класичною системою дистанційного навчання, це скоріше середовище для спільної роботи з можливістю обмінюватися файлами, де Google об’єднали всі необхідні інструменти для ефективної освіти в одному місці. Але дана система відмінно підійде для знайомства з онлайн-навчанням. Для того щоб скористатися функціями даного сервісу, достатньо лише створити обліковий запис в Google. Інтерфейс платформи дуже зручний і простий, тому складнощів у знайомстві з функціями не виникне.

Основні функціональні можливості Google Classroom:

– викладачі можуть створювати курси, додавати студентів і ділитися різними навчальними матеріалами;

– над курсом можуть працювати декілька викладачів;

– можливість створення завдань та тестів;

– оцінювання виконаних студентами завдань і перегляд їх прогресу;

– викладач може залишати коментарі стосовно виконаного завдання;

– легка інтеграція з іншими інструментами Google для зручного обміну документами та іншими матеріалами.

1.2.3. ClassDojo

ClassDojo – це сервіс, який допомагає забезпечити комунікацію між вчителями, учнями та їхніми батьками. Дана платформа дає змогу максимально відтворити шкільне середовище вдома й намагається зацікавити дітей шкільного віку за допомогою ігрового підходу та заохочувальних бейджів. Кожен учень на платформі має свою власну анімовану аватарку, яка «радіє», коли отримано похвалу від учителя і «сумує», коли ставлять негативну оцінку [2].

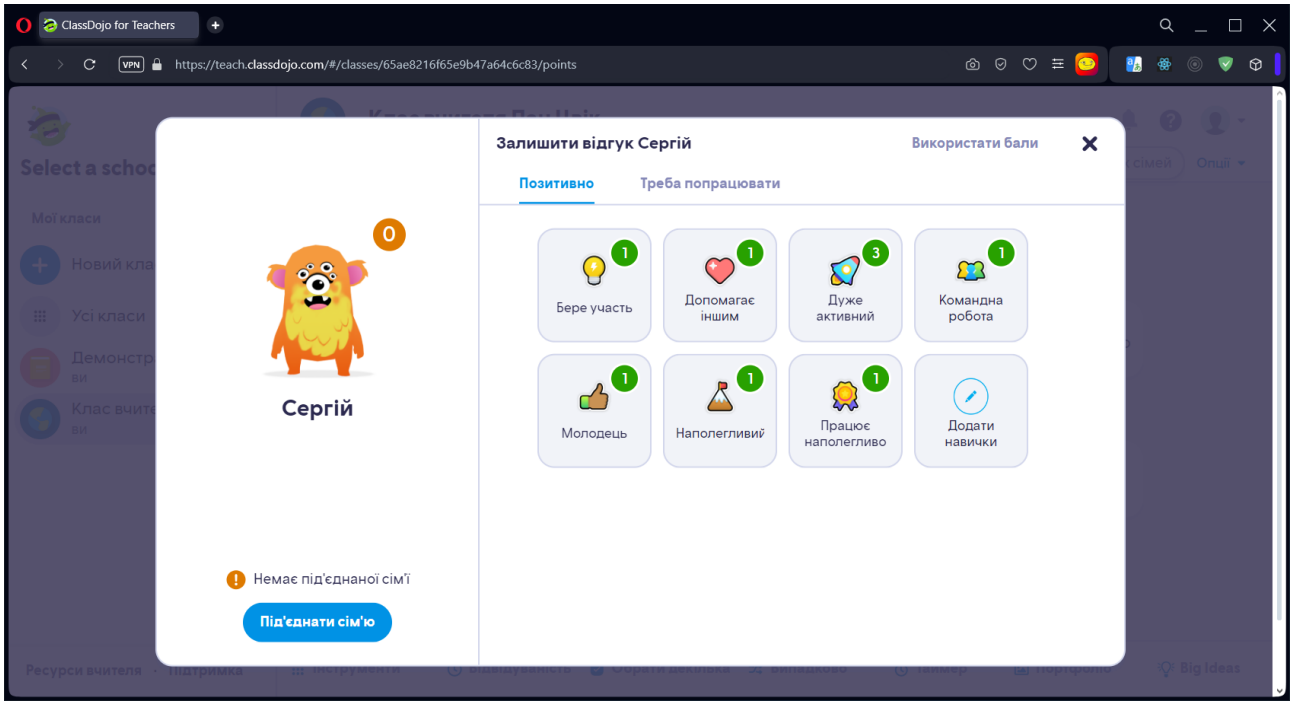

Рисунок 1.3 – Можливість залишити відгук учню в ClassDojo

Заохочувальні бейджи можуть давати як позитивні оцінки, так і негативні з закликом до праці (рис. 1.3). Також вчителі можуть створювати

нові бейджи та визначати скільки балів вони будуть додавати чи віднімати учню.

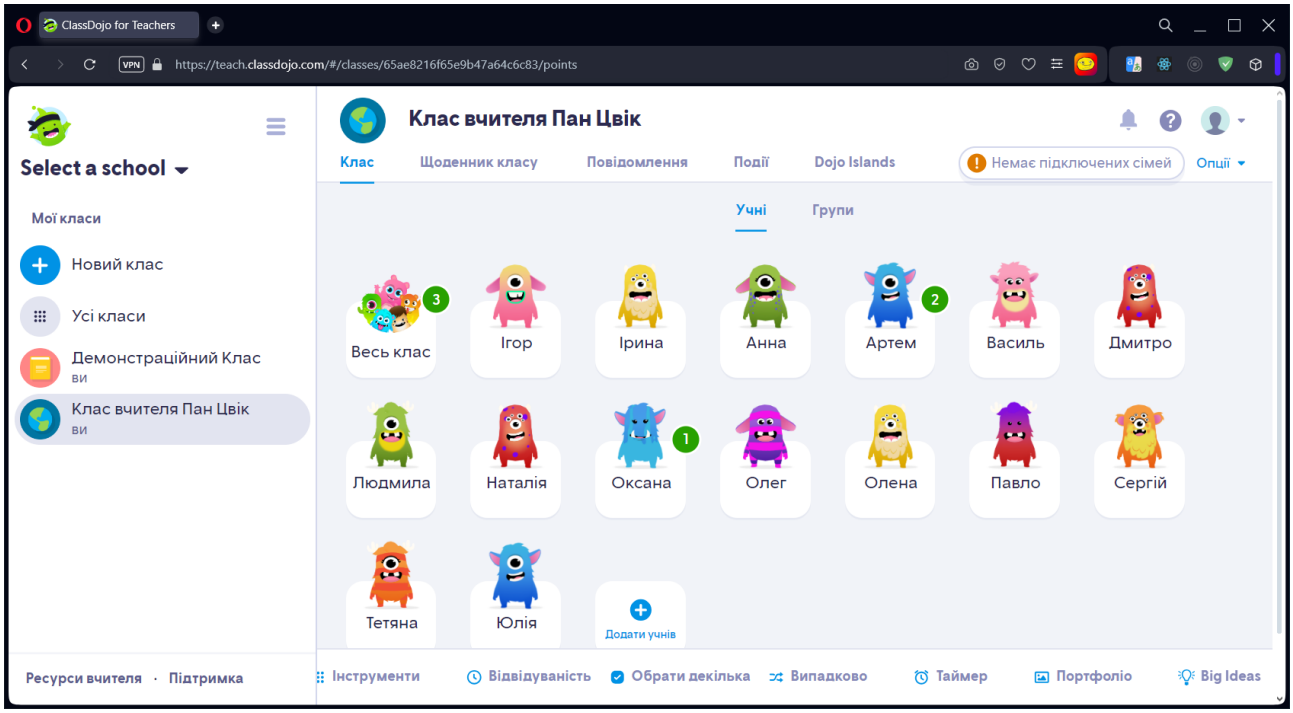

Рисунок 1.4 – Інтерфейс класу в ClassDojo

Основні функціональні можливості ClassDojo:

– легко запрошувати учнів і їх батьків, за допомогою згенерованого QR-коду, готового посилання для входу в акаунт або запрошень на електронну пошту;

– яскрава анімація аватарів, яка привертає увагу (рис. 1.4);

– стрічка подій для інформування класу;

– створення завдань з різними типами відповідей (текст, відео, фото, малювання, робочий лист);

– створення індивідуальних завдань для кожного учня окремо або спільне завдання для всього класу;

– бейджи з різними зображеннями для позитивних та негативних оцінок;

– генерація статистики прогресу кожного учня і всього класу для обраного періоду;

– батьки можуть переглядати успішність їхньої дитини.

Після проведеного аналізу аналогів систем дистанційного навчання, було виконано порівняння їх основних функціональних можливостей з платформою Elers (табл. 1.1).

Таблиця 1.1

Порівняння аналогів систем дистанційного навчання

Характеристика	Moodle	Google Classroom	ClassDojo	Elers
Створення курсів та наповнення матеріалами	+	+	-	+
Створення тестів	+	+	-	+
Створення та оцінювання завдань	+	+	+	+
Перегляд успішності студентів	+	+	+	+
Чат	+	-	+	-
Налаштування доступу до курсу	+	-	-	+
Керувати курсом можуть декілька викладачів	+	+	+	+
Календар подій	+	+	-	-
Батьки мають змогу переглядати успішність своїх дітей	+	+	+	-
Можливість групувати матеріали курсу та налаштовувати відображення кожної окремої вкладки	±	-	-	+

Платформа Moodle має широкий набір інструментів для управління курсами. Цей набір включає можливість публікації різних типів матеріалів курсу, створення тестів та практичних завдань, перегляду успішності студентів на курсі, налаштування доступу до курсу та формування календаря подій. Крім того, за допомогою додаткових налаштувань та плагінів, можна підключити модуль чату для обміну повідомленнями між учасниками платформи, додати отримання відзнак за навчання, а також створити акаунти батьків, які матимуть доступ до перегляду успішності своїх дітей. Є можливість додати плагін для відображення вкладок, але він має обмежені опції налаштувань і вимагає врахування сумісності версій як самого плагіну, так і всієї налаштованої системи.

Сервіс Google Classroom має спрощений функціонал для управління курсами. Завдяки інтеграції з Google Формами, викладачі можуть легко створювати тести, а з Google Документами – в реальному часі спостерігати за роботою студентів, переглядати документи, над якими ті працюють, а також

виправляти помилки. У сервісі викладачі можуть коментувати виконані завдання студентів та виставляти оцінки, а також відправляти роботу студентів на доопрацювання. Після внесення правок студентом, викладач може повторно оцінити роботу. Щоб додати батьків до курсу, викладачеві необхідно мати обліковий запис Google Workspace for Education, оскільки звичайні акаунти не надають такої можливості.

ClassDojo не пропонує функцію створення курсів та їх наповнення матеріалами. Натомість, він дозволяє користувачам створювати класи, публікувати нові пости та планувати події. Цей сервіс також не підтримує створення тестів, а доступ до класів обмежений лише учасниками, які можуть переглядати опубліковані матеріали. Календар подій також відсутній. Даний сервіс пропонує окремі акаунти для батьків, де вони можуть переглядати успішність своїх дітей та їх виконані роботи. Однією з переваг цього сервісу є ігровий підхід, який реалізований за допомогою мотиваційних бейджів.

За результатами порівняння аналогів проведемо формування вимог до платформи, яка розробляється. Метою визначення вимог є детальний опис функціональних та нефункціональних вимог, які учасники проєкту хочуть бачити в реалізованій та впровадженій системі [37, с. 53].

Функціональні вимоги:

– відсутність реєстрації, лише адміністратори можуть реєструвати нових користувачів з різними ролями;

– вхід для студентів і викладачів з різними рівнями доступу;

– створення курсів та можливість додавати різні матеріали (файли, посилання, текстове наповнення);

– можливість завантаження практичних завдань студентами та їх оцінювання викладачем;

– можливість групування матеріалів курсу викладачами за тематичними вкладками;

– створення тестів з різними типами питань (вибір однієї чи декількох правильних відповідей із запропонованих варіантів, відкрита відповідь, встановлення відповідності) та автоматизована перевірка результатів;

– відображення оцінок студентів.

Нефункціональні вимоги:

– платформа повинна бути сумісною з різними браузерами та пристроями користувачів;

– мінімалістичний та легкий у використанні інтерфейс;

– інтерфейс платформи повинен мати українську та англійську локалізацію;

– можливість вибрати темну чи світлу тему сайту;

– масштабованість для збільшення кількості користувачів та обсягу навчального матеріалу.

Підсумовуючи результати дослідження було виділено основні переваги та недоліки розглянутих систем (табл. 1.2).

Таблиця 1.2

Переваги та недоліки систем дистанційного навчання

Вебплатформа	Переваги	Недоліки
Moodle	– створення курсів, завдань та тестів; – підтримка різноманітних типів запитань для тестів; – наявні чати; – налаштування дозволів для користувацьких ролей; – календар подій; – наявність відзнак за навчання; – відкритий вихідний код та велика кількість додаткових плагінів.	– інтерфейс не є інтуїтивно зрозумілим; – додавання нового функціоналу потребує встановлення та налаштування нових плагінів; – вивчення та налаштування системи займає багато часу.
Google Classroom	– створення курсів, завдань та тестів; – календар подій; – легка інтеграція з іншими сервісами Google – наявність стрічки новин, в якій викладач може писати повідомлення з приводу начальних заходів.	– відсутні чати; – відсутні відзнаки за навчання; – відсутність налаштування дозволів для користувацьких ролей.

Продовження таблиці 1.2

ClassDojo	– створення класів та завдань; – публікація постів та подій; – наявні чати; – анімовані аватари; – мотиваційні бейджи; – швидка реєстрація для учнів та батьків.	– відсутність тестів; – відсутність налаштування дозволів для користувацьких ролей; – відсутній календар подій; – відсутня можливість групувати пости за тематикою.
Elers	– створення курсів, завдань та тестів; – налаштування загальних дозволів для користувачів платформи; – налаштування дозволів та ролей для учасників курсу; – групування матеріалів курсу по вкладках.	– відсутні чати; – відсутній календар подій; – відсутні акаунти батьків.

1.3. Планування та проєктування архітектури вебплатформи дистанційного навчання

Для реалізації вебзастосунків використовується клієнт-серверна архітектура. Модель такої системи полягає в тому, що клієнт, використовуючи браузер, відправляє запит на віддалений сервер. На стороні сервера отриманий запит обробляється, формується відповідь і далі готовий результат відправляється клієнтові [32].

Перевагами такого архітектурного стилю є:

– простота в обслуговуванні, якщо один сервер виходить з ладу через ремонт, оновлення або переміщення, це не впливає на роботу клієнта, за умови, що інші, резервні сервери будуть налаштовані та доступні для продовження роботи;

– забезпечує безпеку даних, адже вся інформація зберігається на віддаленому сервері, який має кращий захист, ніж клієнтські пристрої;

– централізований доступ до даних, оскільки дані зберігаються тільки на сервері.

Основний недолік даної архітектури є те, що несправність сервера зробить систему недоступною для користувачів, якщо відсутні резервні сервери. Також, якщо будь-який один модуль матиме більше навантаження

ніж інші, то при недостатній потужності сервера з’являються сильні затримки в роботі всієї системи [32].

Даний архітектурний підхід розділяє функціональні обов’язки між клієнтською та серверною частинами.

Функції, які реалізуються на сервері:

– зберігання, доступ, захист і резервне копіювання даних;

– обробка клієнтського запиту;

– відправлення результату (відповіді) клієнту у вигляді вебсторінки (html) або іншого файлу (json, зображення тощо).

Функції, які реалізуються на стороні клієнта:

– надання користувальницького інтерфейсу;

– формулювання запиту до сервера і його відправка;

– отримання та відображення результатів запиту.

Для розробки платформи дистанційного навчання буде використано шаблон проєктування Модель-Представлення-Контролер (англ. Model-View-Controller, MVC). Цей шаблон є ідеальним вибором для створення вебдодатків, оскільки дозволяє відокремити інтерфейс користувача, тобто HTML-розмітку, від бізнес-логіки системи [27].

Основна концепція шаблону MVC полягає в розподілі відповідальності, де кожна частина архітектури MVC є чітко визначеною та автономною. Цей шаблон розділяє систему на три взаємопов’язані складові (рис. 1.5): модель даних, представлення даних та контролер, який керує обміном даних між ними. Кожна складова виконує лише свої визначені функції [27]:

1. Модель відповідає за роботу з даними. Вона може бути простою моделлю представлення, яка лише передає дані між представленням і контролером, або моделлю предметної області, яка містить бізнес-дані, а також операції та правила для їх обробки.

2. Представлення відповідає за візуалізацію даних, зазвичай частини моделі, у формі користувацького інтерфейсу.

3. Контролер обробляє запити користувачів, виконує операції з моделлю і вибирає відповідне представлення для оновлення інтерфейсу користувача.

Кожен з цих компонентів має чітко визначені обов'язки та виконує лише певні функції, що забезпечує легкість у супроводі та гнучкість розробки.

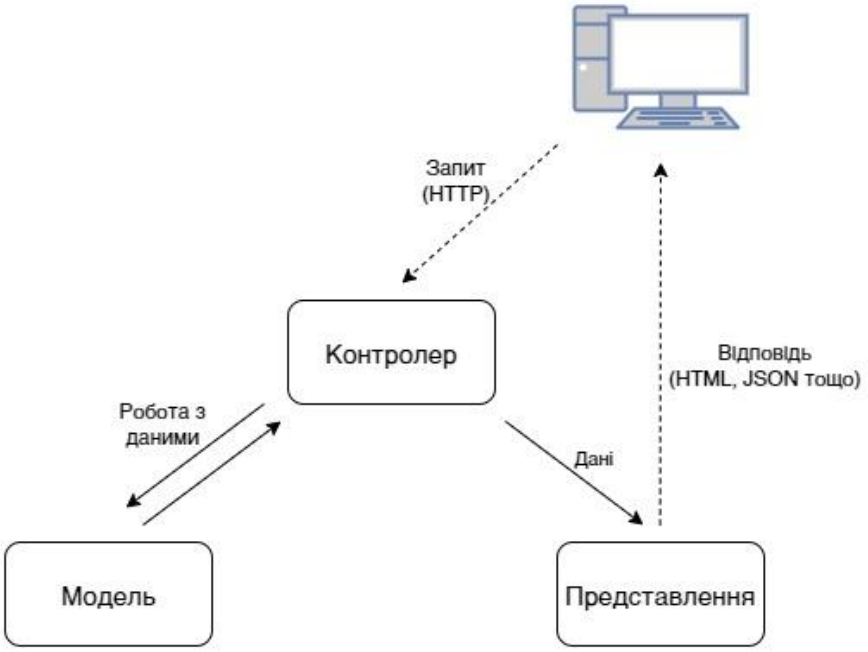

Рисунок 1.5 – Схема MVC шаблону

Щоб прискорити та спростити розробку системи буде використано монолітну архітектуру. Цей підхід зазвичай використовується у невеликих командах та проєктах з невеликим навантаженням. Монолітна архітектура має простий процес деплою на сервер, оскільки весь код розглядається як єдина програма. Однак, це є й головним недоліком, оскільки він впливає на можливість масштабування та може призвести до єдиної точки відмови, що може вивести всю програму з ладу [25].

Для проєктування серверної частини доцільно використати основні принципи чистої архітектури Роберта Мартіна для розділення програмного забезпечення на різні рівні. Головним правилом, що приводить цю архітектуру в дію, є правило залежностей (англ. Dependency Rule):

залежності у вихідному коді мають бути спрямовані лише всередину (рис. 1.6) [33, с. 215].

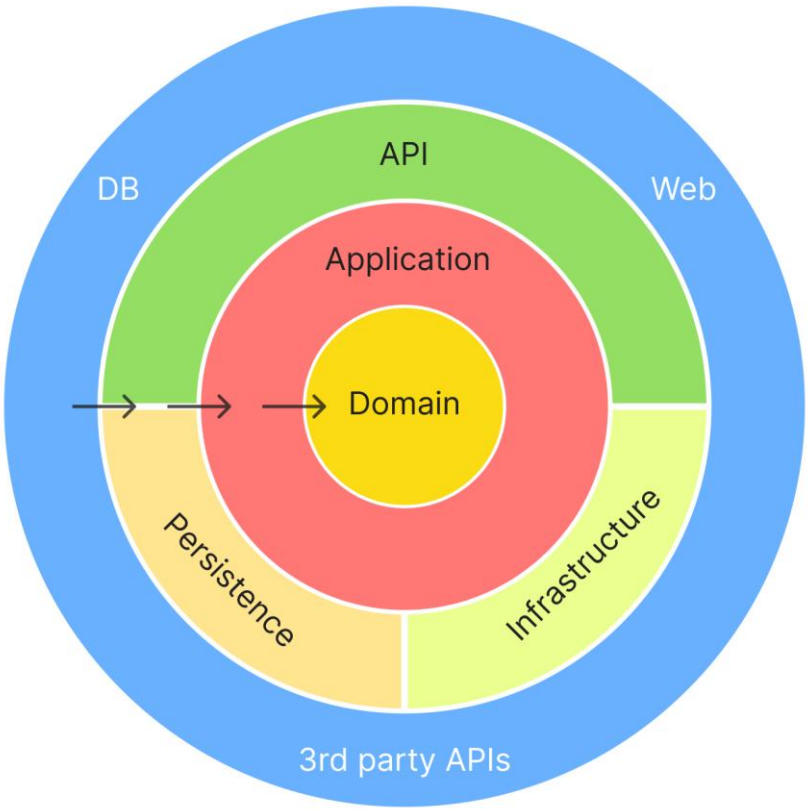

Рисунок 1.6 – Схема рівнів чистої архітектури

Усе, що перебуває у внутрішньому колі, взагалі не може нічого знати про зовнішні кола. Наприклад, імена, оголошені в зовнішніх колах, не повинні згадуватись в коді, що міститься у внутрішніх колах. Сюди належать функції, класи, змінні та будь-які інші іменовані елементи програми [33, с. 215-216].

Усі головні компоненти системи чітко розділені та інкапсульовані, утворюючи кілька шарів з односторонніми залежностями:

1. Domain. Компонент Domain – це фундамент системи, навколо нього будується весь проєкт. Цей рівень немає жодних залежностей від зовнішніх і містить лише сутності, перерахування, константні дані і, можливо, деякі спеціальні винятки на рівні Domain.

2. Application. Разом із доменним рівнем (Domain) прикладний рівень (Application) утворює ядро рішення, яке повинно мати можливість працювати та забезпечувати бізнес-логіку незалежно від зовнішніх рівнів і залежить

виключно від Domain. Він створює об’єкти сутностей, виконує бізнес-правила та повертає результати для API.

3. Infrastructure. Це рівень, у якому має бути реалізована вся логіка зв’язку із зовнішніми системами, як-от надсилання електронних листів, зв’язок із стороннім API тощо. Він залежить лише від Application та реалізує його інтерфейси.

4. Persistence. Порівняно з Infrastructure, рівень Persistence також містить логіку для зв’язку із зовнішніми системами, але його конкретна мета – зв’язуватися з базами даних. Цей рівень зазвичай залежить від Application та Domain.

5. API. Це рівень взаємодії із «зовнішнім світом», який дозволяє клієнтам отримувати видимі результати після запиту даних. Цей рівень може бути у будь-якій формі, наприклад: веб-API, консольна програма, графічний інтерфейс тощо. Він також залежить лише від Application, але також може використовувати елементи з Persistence. Наприклад, при першому запуску програми потрібно завантажити початкові дані (англ. seed data) в БД, які зберігаються в Persistence. Він не містить бізнес-логіки, його єдине завдання – перетворити запити користувача на зрозумілу для Application форму та повернути отримані результати у зручному форматі для компонента Web.

6. Web. Верхній рівень Web, відповідає за реалізацію клієнтської частини, яка може включати в себе HTML, CSS, JavaScript та будь-які інші фронт-енд технології для створення вебінтерфейсу. Користувачі можуть використовувати даний компонент для взаємодії з системою за допомогою HTTP запитів.

Застосування підходу чистої архітектури з її чітко визначеними компонентами у монолітному проєкті може спростити його майбутнє перенесення на мікросервісну архітектуру. Мікросервісна архітектура забезпечує більш зручну роботу для великої команди розробників, легше масштабується та дозволяє використовувати різні мови програмування для кожного окремого сервісу. Головними її недоліками є складність

проєктування та деплою на сервер, що вимагає більше часу та ресурсів порівняно з монолітною архітектурою (рис. 1.7) [25].

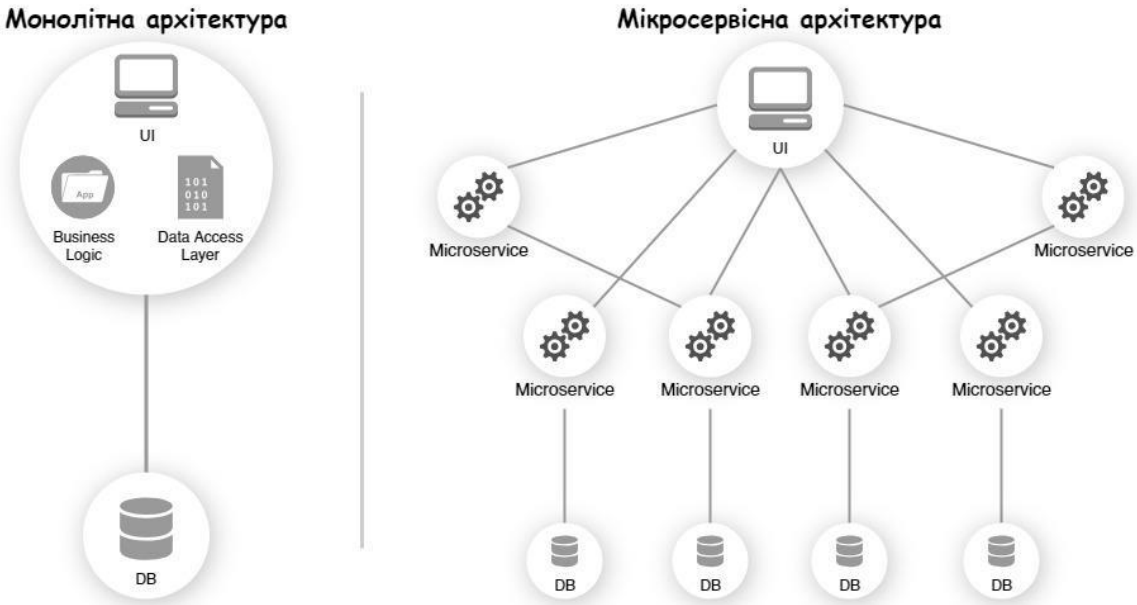

Рисунок 1.7 – Монолітна та мікросервісна архітектури

Клієнтську частину, яка відповідає за складний інтерфейс з багатьма інтерактивними елементами, такими як таблиця успішності або інструменти створення та проходження тестів, варто реалізувати у вигляді односторінкового додатку (англ. Single Page Application, SPA). Завдяки цьому підходу, користувач отримує швидке та плавне відображення змін інтерфейсу на його дії. Крім того, технології для розробки SPA мають більше можливостей ніж звичайний JavaScript, HTML та CSS.

SPA – це вебдодаток, який завантажує сторінку лише один раз та динамічно змінює її вміст без необхідності перезавантаження всієї сторінки. SPA використовує підхід AJAX (англ. Asynchronous JavaScript and XML) для обміну даними з сервером та оновлення лише необхідних частин сторінки. SPA забезпечує такі переваги, як [35]:

1. Постійна взаємодія з користувачем, за допомогою динамічної зміни контенту й без завантаження нової сторінки з сервера. Це дає змогу зменшити навантаження на сервер, адже серверу достатньо відправити дані у JSON форматі, а не генерувати всю HTML сторінку.

2. При завантаженні нових модулів (сторінок) контент на них оновлюється лише частково, тому що немає необхідності повторно завантажувати елементи, які не було змінено.

3. Розробникам доступні різні фреймворки та бібліотеки, які спрощують створення архітектури проєкту та дають чимало готових елементів для роботи.

Односторінкові додатки мають і деякі недоліки, які можна виправити шляхом використання сторонніх інструментів та проведення додаткової оптимізації [35]:

1. Проблеми з індексацією динамічного контенту, тобто складнощі з налаштуванням SEO.

2. Тривале завантаження під час першого відвідування, через велику кількість скриптів, які генерують контент на сторінці.

3. Важливо, щоб у користувачів було увімкнено підтримку JavaScript у браузері.

Після того, як було обрано архітектуру платформи, необхідно розробити її діаграму компонентів. Розуміння точної поведінки кожної частини програмного забезпечення є важливим для будь-якого програміста. Діаграми компонентів можуть допомогти іншим розробникам зрозуміти структуру конкретної системи. Крім того, ці діаграми можуть описувати програмні системи, реалізовані будь-якою мовою чи стилем програмування.

Діаграма компонентів допомагає перейти від загального розуміння проєкту до реалізації у вигляді коду. Деякі компоненти існують тільки під час компіляції коду, інші – під час його виконання. У багатьох середовищах розробки один компонент (або модуль) відповідає окремому файлу. Головними елементами діаграми компонентів є самі компоненти, інтерфейси та зв’язки між ними [29, с. 103].

Головна мета цих діаграм – показати взаємозв’язок між різними компонентами в системі. Тут «компоненти» можуть означати модулі класів, які представляють незалежні системи або підсистеми з можливістю взаємодії

з рештою системи, також у ролі компонента можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети тощо [30].

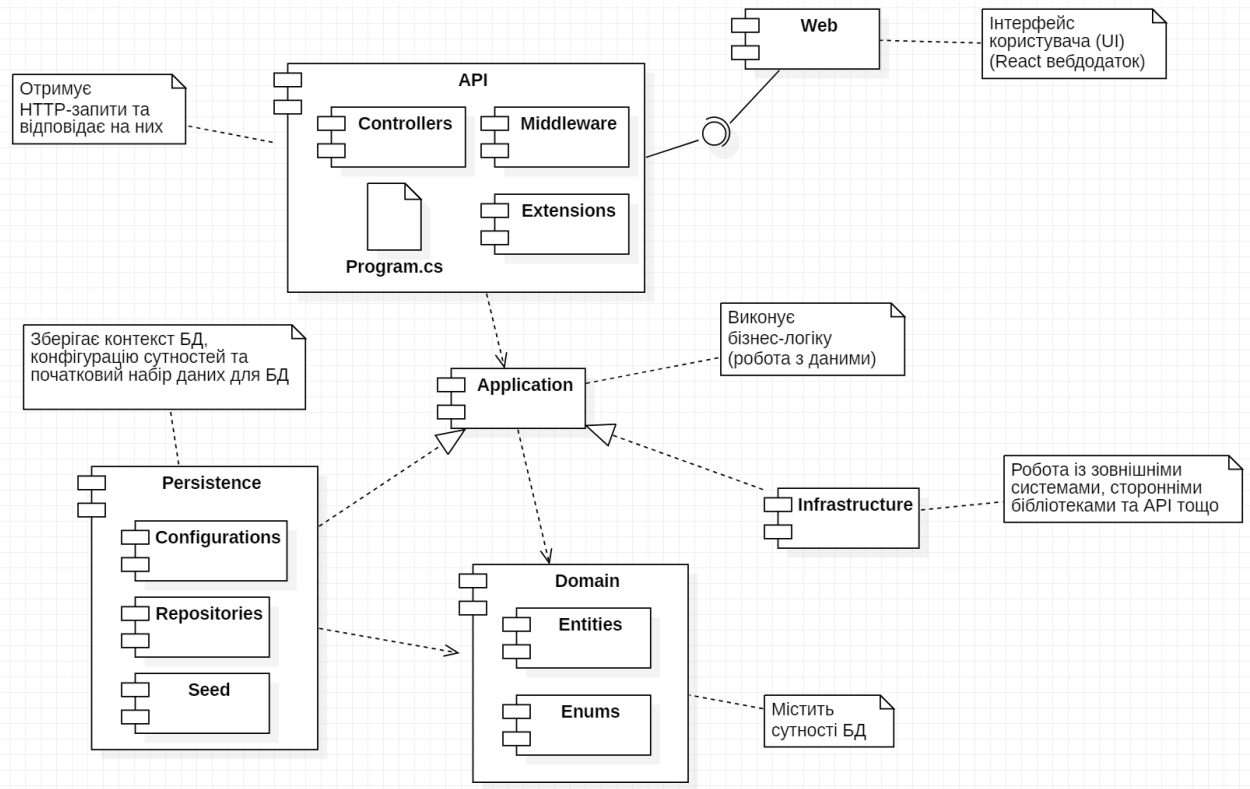

Рисунок 1.8 – Діаграма компонентів

Діаграма компонентів (рис. 1.8) показує загальну структуру платформи дистанційного навчання. Для зручності сприйняття діаграми компонентів, до кожного компонента було додано відповідні нотатки.

Компонент «Web» зв’язаний з компонентом «API» через інтерфейс тому, що взаємодія між ними відбувається через певний набір методів HTTP запитів. Ці методи визначають спосіб передачі даних та обмін інформацією між компонентами, що дозволяє взаємодіяти без прив’язки до конкретних реалізацій методів.

Компонент «API» використовує компонент «Application» для доступу до даних та основної функціональності системи. В свою чергу «Application» використовує «Domain», де зберігаються класи сутностей, що відповідають таблицям в базі даних. Компоненти «Persistence» та «Infrastructure» реалізують різні інтерфейси компонента «Application» для роботи з

контекстом БД та зовнішніми системами відповідно. Також «Persistence» використовує сутності з «Domain» для того, щоб провести конфігурацію БД.

Компонент «Web» відповідає за представлення (View) з шаблону MVC. У компоненті «API» зберігаються контролери (Controller), а компоненти «Application», «Domain», «Persistence» та «Infrastructure» містять правила для маніпулювання даними (Model).

У підсумку проєктування архітектури ми отримаємо наступну схему обробки запитів (рис. 1.8), яка візуально підкреслює послідовність кроків від користувача до БД або стороннього API та назад.

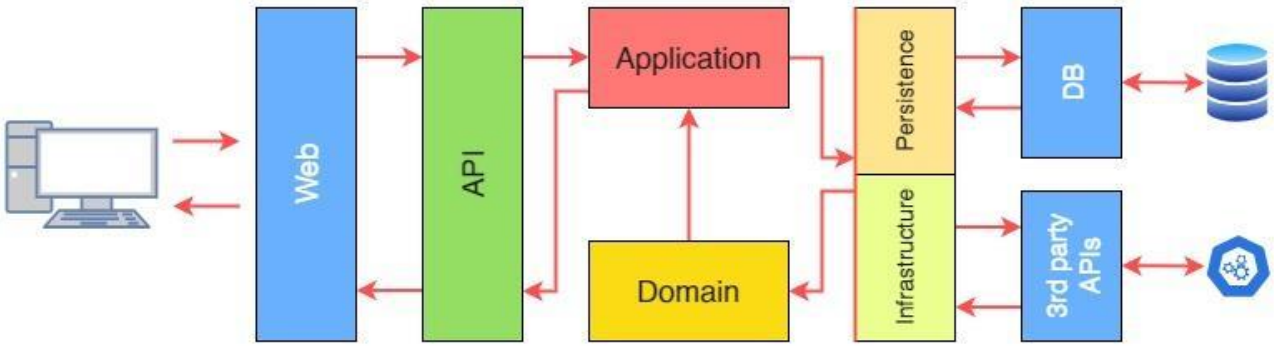

Рисунок 1.9 – Схема обробки запита [3]

1.4. Обґрунтування використання технологій та засобів

Для серверної частини обрано **ASP.NET Core** – це кросплатформенний фреймворк для створення вебзастосунків на платформі .NET з відкритим вихідним кодом, де в якості мови програмування можна використовувати C#. Має всі необхідні інструменти для роботи сучасного вебдодатку: маршрутизація, конфігурація, логування, можливість роботи з різними системами баз даних, налаштування авторизації, інтеграція зі сторонніми API та інші. Він пропонує ефективний підхід до розробки програми, спеціально орієнтований на роботу в стилі REST (англ. Representation State Transfer, передача стану уявлення) [17, с. 10-11].

Проєктування бази даних буде виконано за допомогою ORM (англ. object-relational mapping, реляційне відображення об’єктів), тобто відображення даних на реальних об’єктах, **Entity Framework Core** (EF Core)

та підходу Code First. Тобто спочатку потрібно написати код на мові програмування C#, а потім з цього коду буде автоматично згенеровано таблиці для бази даних. Для цього підходу дуже важливо визначити класи сутностей, що будуть відповідати таблицям, які будуть зберігатися в базі даних [4].

Конфігурацію таблиць зручно налаштовувати за допомогою **Fluent API**, адже він надає більший діапазон параметрів конфігурації, ніж атрибути анотації даних. Конфігурація Fluent API також сприяє чистому коду, оскільки конфігурацію можна зберігати в окремих файлах.

EF Core підвищує ефективність та швидкість розробки, оскільки тепер програмісти мають справу не з таблицями та полями, а з класами та об'єктам, що дозволяє не писати складних SQL-запитів. Даний фреймворк підтримує міграції бази даних, що дозволяє легко вносити зміни в схему бази даних та оновлювати дані в базі даних, не втрачаючи існуючої інформації. Це особливо корисно при розробці проєкту, коли схема бази даних може змінюватися. EF Core є одним з популярних ORM-фреймворків для .NET розробників, адже підтримує різні СУБД, включаючи PostgreSQL, MySQL, SQLite, Oracle та інші [8].

Бібліотека **MediatR** – спрощує реалізацію патернів CQRS та Mediator. Вона дозволяє розділити відповідальність між контролерами та обробниками команд і запитів, що робить код більш структурованим та гнучким [13].

FluentValidation – це бібліотека .NET для створення типізованих правил валідації даних. Допомагає уникнути помилок та виконання некоректних операцій з даними [1].

Обрано **PostgreSQL** як об'єктно-реляційну систему керування базами даних, яка має високу продуктивність та підтримує широкий спектр функцій, включаючи зберігання процедур, тригери, індекси, віртуальні таблиці (view) та налаштування обмежень (constraints). Гарантує цілісність даних завдяки ACID-транзакціям, що робить її безпечним вибором для зберігання важливої інформації про користувачів та курси [22].

MongoDB обрано як NoSQL базу даних, яка забезпечує високу швидкість та продуктивність при роботі з великими обсягами даних. Надає горизонтальне масштабування, що дозволяє практично необмежено збільшувати обсяги зберігання, що чудово підходить для створення платформи дистанційного навчання, адже майже щодня буде створюватися велика кількість нових записів: оцінки студентів, проходження тестів та завантаження виконаних практичних робіт [16, с. 4].

Для взаємодії між додатком та базою даних MongoDB буде використано **MongoDB C# Driver**. Цей драйвер надає зручний та ефективний інтерфейс для виконання CRUD-операцій з документами MongoDB [18].

Для реалізації клієнтського SPA додатку обрано бібліотеку **React**, яка дозволяє повторно використовувати раніше написані компоненти та спрощує процес створення динамічних вебсторінок. Наприклад, таблиці з успішністю студентів та інструменти створення та проходження тестів повинні мати багато динамічних елементів для зручного відображення та оновлення даних. Крім того, React краще використовувати разом з мовою програмування **TypeScript**, адже сувора типізація призводить до написання більш надійного коду та прискорює розробку.

Взято **Redux Toolkit** для керування станом додатку й **RTK Query** для відправки запитів до сервера [10]. Бібліотеку компонентів **Ant Design** для стилізації компонентів, а також додатково бібліотеку **dayjs** – для роботи з датами та часом, й відображення їх на сторінці у різних форматах.

Для навігації та маршрутизації в React додатках використовується бібліотека **React Router** (react-router-dom). Вона дозволяє легко визначати шляхи, обробляти параметри URL і ефективно керувати навігацією між сторінками. Має можливість гнучкого налаштування перенаправлення користувачів на інші сторінки, якщо вони намагаються перейти до неіснуючої сторінки або сторінки, яка вимагає авторизації [23].

Для локалізації клієнтського додатку було використано react-i18next, а для локалізації сервера – переносні об’єкти локалізації (англ. portable object localization).

Щоб покращити якість написання коду варто використати **Prettier** та **ESLint** – це інструменти, які допомагають дотримуватися загальних стандартів у процесі розробки програмного забезпечення. Prettier використовується для форматування коду, працює за допомогою правил, які можна налаштувати відповідно до потреб проєкту. ESLint допомагає виявити помилки у коді, підтримує широкий спектр правил, включаючи правила стилю, продуктивності та інші.

У якості середовища розробки обрано **Visual Studio Code** (VS Code). Цей редактор швидко аналізує структуру коду, показує підказки під час написання коду, підтримує автодоповнення і рефакторинг, що сприяє уникненню помилок та прискорює процес розробки. Має великий список доступних розширень, які дозволяють зручно працювати з великим переліком технологій, таких як C#, JavaScript, TypeScript, HTML, CSS тощо. Крім того, за допомогою VS Code можна легко та зручно відлагоджувати власний код.

Висновки до першого розділу

Проаналізовано вимоги до програмного продукту та предметної області. Визначено завдання, які необхідно вирішити для успішної реалізації проєкту.

Після дослідження аналогів Moodle, Google Classroom та ClassDojo, було визначено їхні ключові функціональні можливості та створено порівняльну таблицю у якій перераховано переваги та недоліки. На основі цього аналізу було сформовано функціональні та нефункціональні вимоги до майбутньої системи.

Для реалізації системи обрано монолітну архітектуру, в якій клієнтську частину буде реалізовано у вигляді SPA додатку, а сервер буде використовувати чисту архітектуру Роберта Мартіна.

Використання SPA дозволить забезпечити користувачам зручний, динамічний та швидкий інтерфейс, зменшуючи навантаження на сервер та забезпечуючи більш ефективну обробку запитів. Застосування чистої архітектури на сервері дозволяє розділити компоненти системи на незалежні шари, що полегшить тестування, розширення та підтримку коду.

Було обрано основні інструменти для розробки серверної частини, такі як ASP.NET Core, Entity Framework Core, MongoDB C# Driver, MediatR та FluentValidation. У якості СУБД обрано PostgreSQL та MongoDB, а VS Code як середовище розробки. Головні технології для реалізації клієнтського додатку: React, TypeScript, Redux Toolkit, RTK Query, React Router, Ant Design, Prettier та ESLint.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО НАВЧАННЯ

2.1. Визначення варіантів використання та структури платформи

Аналіз функціональних та нефункціональних вимог до системи електронного навчання дав підставу для визначення основних варіантів використання платформи. Ці варіанти описують ключові дії, які можуть виконувати користувачі системи, та відповідають їхнім потребам та ролям.

У системі розрізняють трьох ключових акторів: адміністратор, викладач та студент. Кожен з них має свій набір функціональних можливостей та список прав доступу (табл. 2.1).

Таблиця 2.1

Виявлення акторів

Актор	**Короткий опис**
Адміністратор	Реєструє нових викладачів та студентів або надає відповідні ролі уже зареєстрованим користувачам. Крім цього, він має право змінювати дані та видаляти будь-які зареєстровані акаунти на сайті. Має доступ до всіх частин сайту.
Викладач	Наповнює курси відповідним вмістом, створює завдання та тести, також перевіряє виконані завдання студентів та вводить оцінки в електронний журнал.
Студент	Має доступ до навчальних матеріалів, можливість проходити тестування, надсилати виконанні завдання та переглядати власну успішність.

Основні функціональні можливості онлайн-платформи дистанційного навчання можна згрупувати у вигляді варіантів використання для різних категорій користувачів, таких як студенти, викладачі та адміністратори.

Наступна таблиця (табл. 2.2) представляє перелік визначених варіантів використання, їх акторів, найменування та формулювання сутності даного варіанта. Ці варіанти охоплюють ключові аспекти платформи, включаючи авторизацію, перегляд навчальних матеріалів, завантаження та оцінювання робіт, проходження тестів, керування курсами та контентом, налаштування ролей і прав доступу, а також адміністрування користувачів та ролей на платформі.

Таблиця 2.2

Виявлення варіантів використання

Основний актор	**Найменування**	**Формулювання**
Студент / Викладач	Авторизація	Цей варіант дозволяє вводити свої облікові дані та отримати доступ до системи.
Студент	Перегляд навчальних матеріалів	Цей варіант дозволяє переглядати та завантажувати вміст курсів, а також проходити тестування.
	Завантаження виконаних завдань	Цей варіант дозволяє завантажувати виконані завдання, тобто текст та файли у певній кількості, для перевірки та оцінювання.
	Перегляд успішності навчання	Цей варіант дозволяє переглядати власні оцінки з усіх курсів.
	Проходження тестів	Цей варіант дозволяє проходити тести та миттєво отримувати за них оцінки.
Викладач	Перевірка завантажених студентських робіт	Цей варіант дозволяє перевіряти надіслані студентами роботи. Викладачі можуть завантажувати файли, надані студентами, оцінювати їхню роботу, виставляти оцінки або відправляти роботи на доопрацювання із зазначенням коментарів і рекомендацій для покращення.
	Налаштування ролей та прав доступу для учасників курсу	Цей варіант дозволяє створювати, налаштовувати та призначати нові ролі для учасників курсу.
	Перегляд та редагування оцінок	Цей варіант дозволяє переглядати всю таблицю оцінок з курсу, додавати нові колонки з оцінками, їх видаляти та виставляти оцінки у обраній комірці.
	Оцінки за пройдені тести	Цей варіант дозволяє лише переглядати результати проходження тестів студентами.
	Оцінки за виконані практичні завдання	Цей варіант дозволяє змінювати оцінки за уже перевірені виконанні завдання у загальній таблиці з оцінками.
Викладач / Адміністратор	Створення та редагування курсів і матеріалів	Цей варіант дозволяє створювати нові курси та наповнювати їх навчальним контентом, включаючи посилання, файли, тести та завдання. Крім того, передбачена можливість створювати вкладки для логічного групування матеріалів курсу, визначати, чи показувати на вкладках кількість матеріалів, змінювати колір вкладок, приховувати наявні вкладки та матеріали для поступового доступу студентів до контенту, а також змінювати порядок розміщення вкладок та матеріалів.
Адміністратор	Керування користувачами	Цей варіант дозволяє переглядати список користувачів системи та редагувати їх дані включаючи створення, редагування та видалення.

Продовження таблиці 2.2

	Редагування прав доступу користувача	Цей варіант дозволяє змінювати список наявних ролей у користувачів для надання або скасування прав доступу до певних функцій системи.
	Управління ролями платформи	Цей варіант дозволяє створювати нові ролі, редагувати або видаляти існуючі ролі, а також визначати набір прав для кожної ролі.

На основі таблиці виявлення варіантів використання була створена діаграма варіантів використання платформи електронного навчання (рис. 2.1). Ця діаграма наочно демонструє взаємодію між ключовими акторами системи та основними функціональними можливостями платформи.

Рисунок 2.1 – Діаграма варіантів використання платформи

Після визначення та відображення варіантів використання для різних категорій користувачів, було створено функціональну карту вебдодатка (рис. 2.2). Вона наочно демонструє структуру та взаємозв’язки основних функціональних компонентів платформи. Функціональна карта також візуалізує навігацію на сайті та вказує, які функції доступні на кожній сторінці.

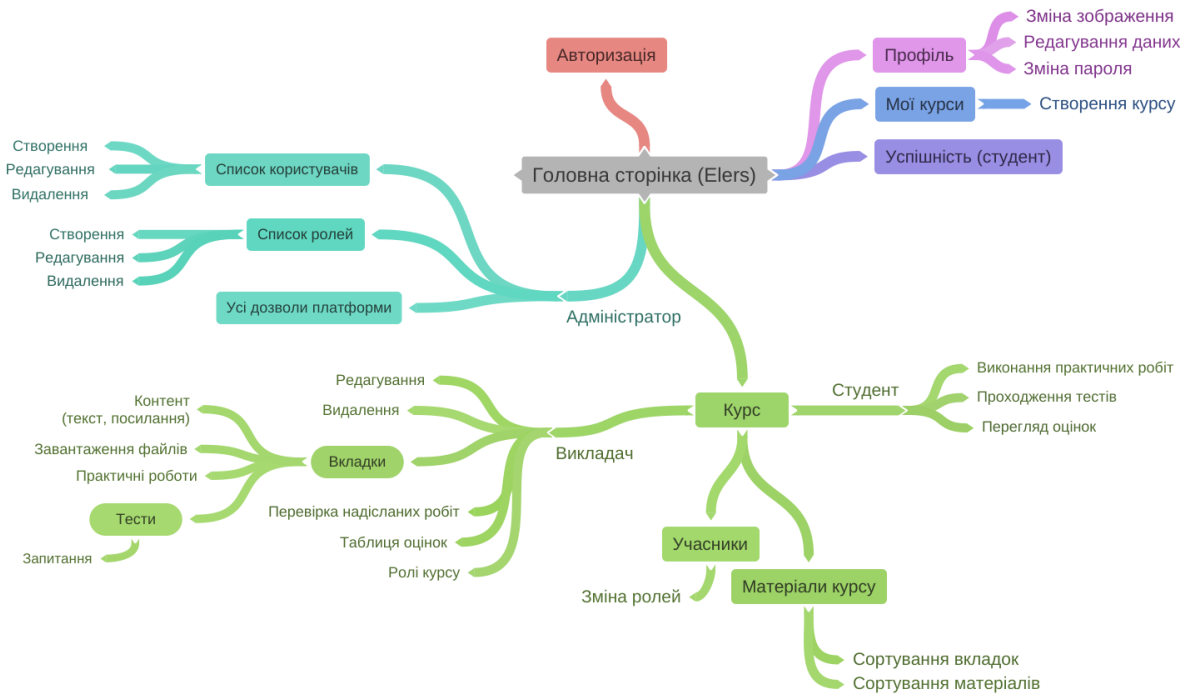

Рисунок 2.2 – Функціональна карта

Тепер, коли вже відомі дійові особи й варіанти використання, створено ілюстрацію поведінки системи і послідовність виконання операцій з описом, можна створити попередню об’єктно-орієнтовану модель системи.

Першим кроком буде моделювання діаграми класів для користувачів системи (рис. 2.3). Всі класи наслідуються від абстрактного класу Entity, який визначає базову структуру для всіх сутностей у системі. Це дозволяє створити загальний клас-репозиторій, що виконує основні CRUD-операції з відповідними сутностями [14].

Діаграма класів користувачів платформи містить наступні елементи:

1. User – зареєстрований користувач. Від цього класу наслідуються два дочірні класи: Teacher та Student, які наразі не мають додаткових властивостей, але можуть бути розширені у майбутньому.

2. UserType – перерахування, тип акаунта користувача.

3. RefreshToken – токен оновлення, який використовується для отримання нового JWT-токена доступу до платформи.

4. Role – роль користувача.

5. Permission – дозвіл, який може бути присвоєний певним ролям.

6. PermissionType – перерахування, дозволи для розмежування доступу користувачів до функцій платформи.

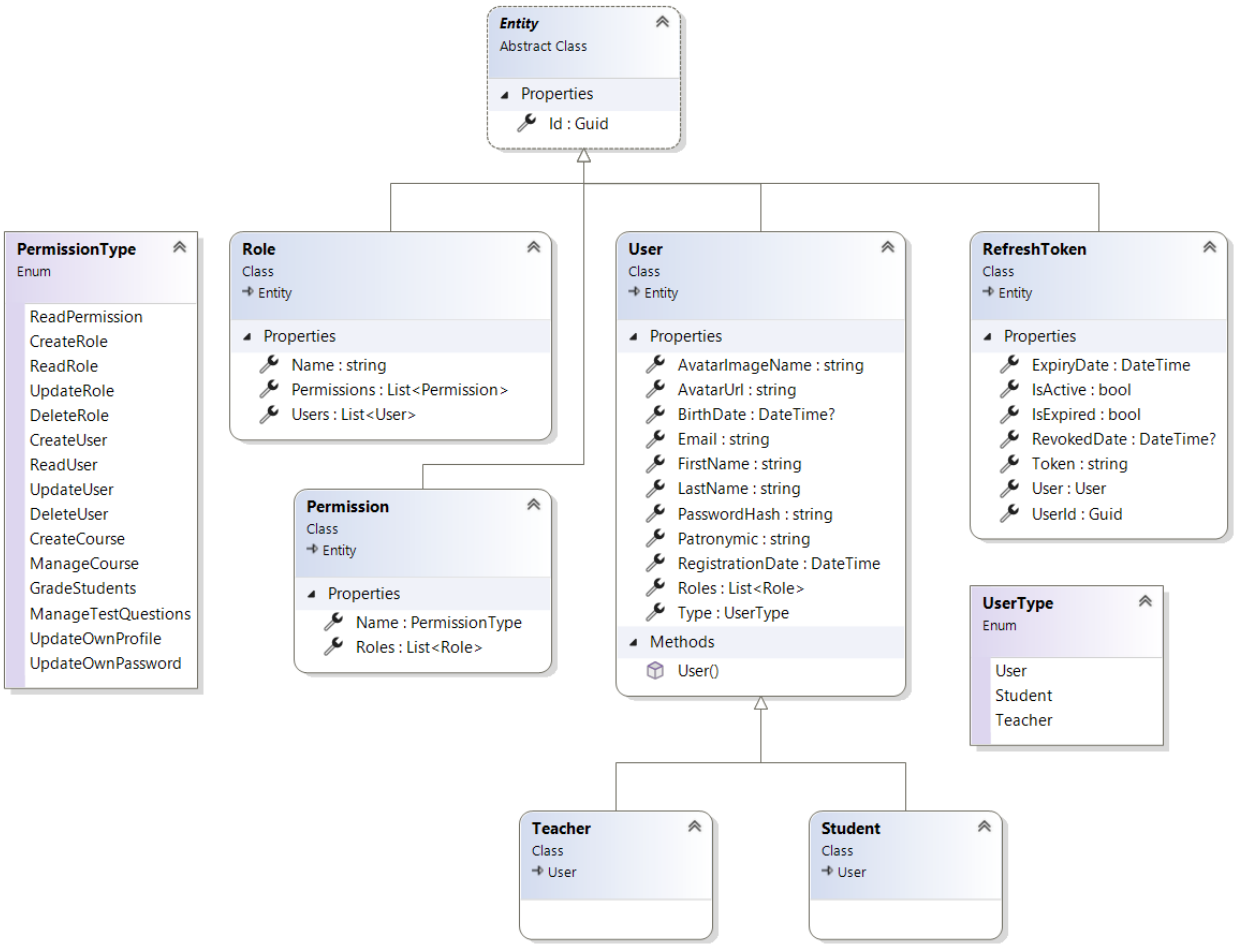

Рисунок 2.3 – Діаграма класів користувачів платформи

Діаграма класів для курсів (рис. 2.4) складається з:

1. Course – курс, містить основну інформацію про курс та тип для представлення вкладок курсу на сторінці.

2. CourseTabType – перерахування, тип представлення вкладок курсу.

3. CourseTab – вкладка курсу, групує матеріали курсу.

4. CourseMember – учасник курсу, також може мати додаткову роль на курсі.

5. CourseRole – роль курсу, містить список дозволів курсу.

6. CoursePermission – дозвіл курсу, визначає доступ до функціоналу для керування курсом.

7. CoursePermissionType – перерахування, дозволи для розмежування доступу учасників курсу до функцій керування курсом.

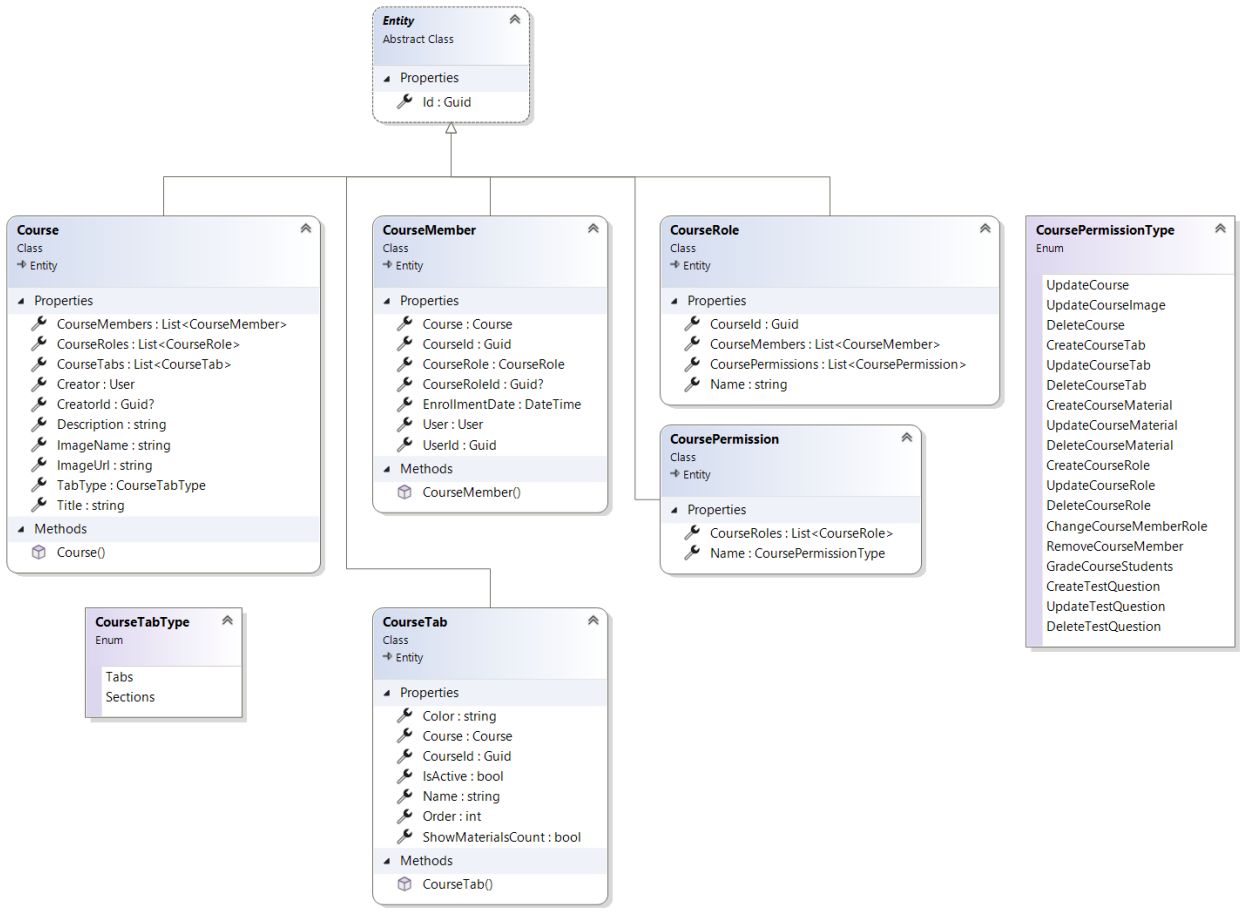

Рисунок 2.4 – Діаграма класів для курсів

До діаграми класів для матеріалів курсу (рис. 2.5), було включено всі підтипи матеріалів, а також перераховано відповідні типи матеріалів та методи оцінювання пройдених тестів студентами:

1. CourseMaterial – абстрактний батьківський клас для всіх типів матеріалів курсу, визначає основні поля.

2. CourseMaterialType – перерахування, тип навчального матеріалу.

3. CourseMaterialContent – контент.

4. CourseMaterialLink – посилання.

5. CourseMaterialFile – файл.

6. CourseMaterialAssignment – інформація про практичну роботу.

7. CourseMaterialTest – інформація про тест.

8. GradingMethod – перерахування, метод по якому буде визначено оцінку за проходження тесту за використаними спробами.

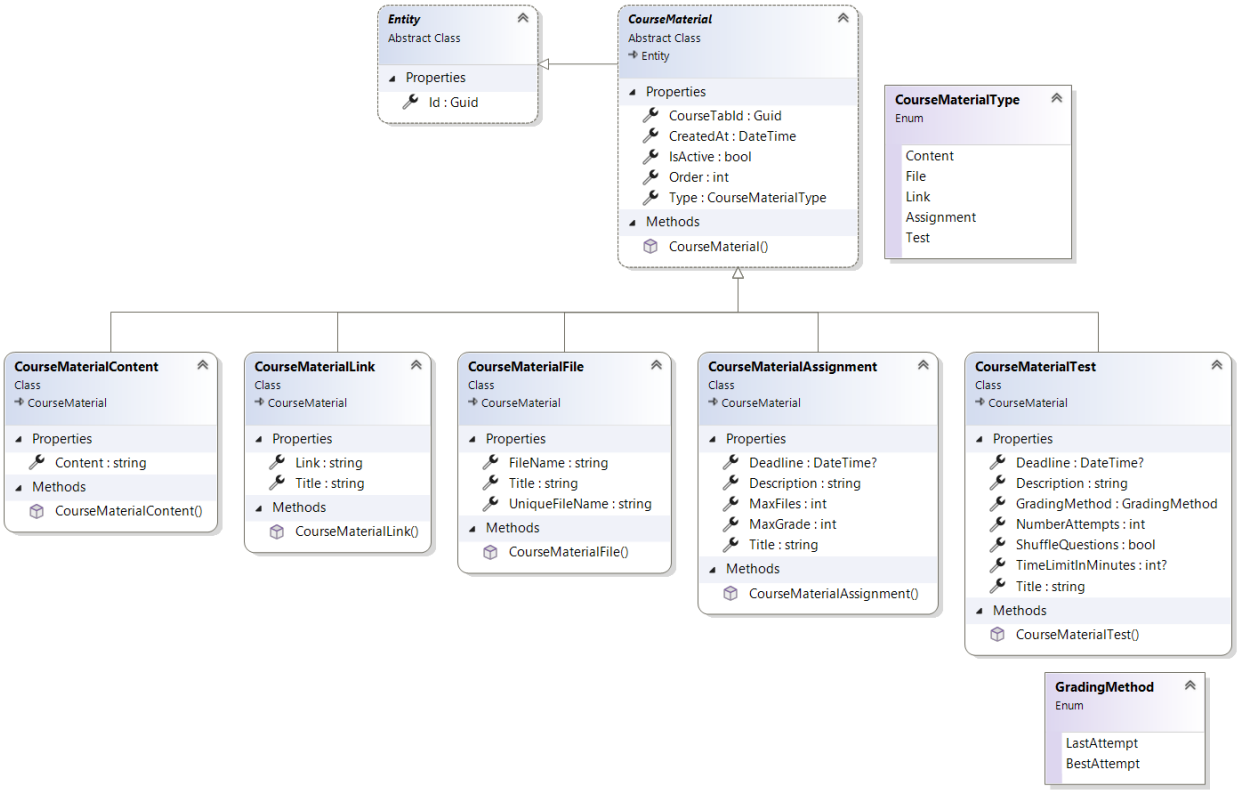

Рисунок 2.5 – Діаграма класів для матеріалів курсу

Діаграма класів для надісланих виконаних практичних робіт студентами (рис. 2.6) складається з:

1. SubmittedAssignment – надіслана робота.
2. SubmitAssignmentFile – файл прикріплений до виконаного завдання.
3. SubmittedAssignmentStatus – перерахування, статус перевірки виконаного завдання студентом.

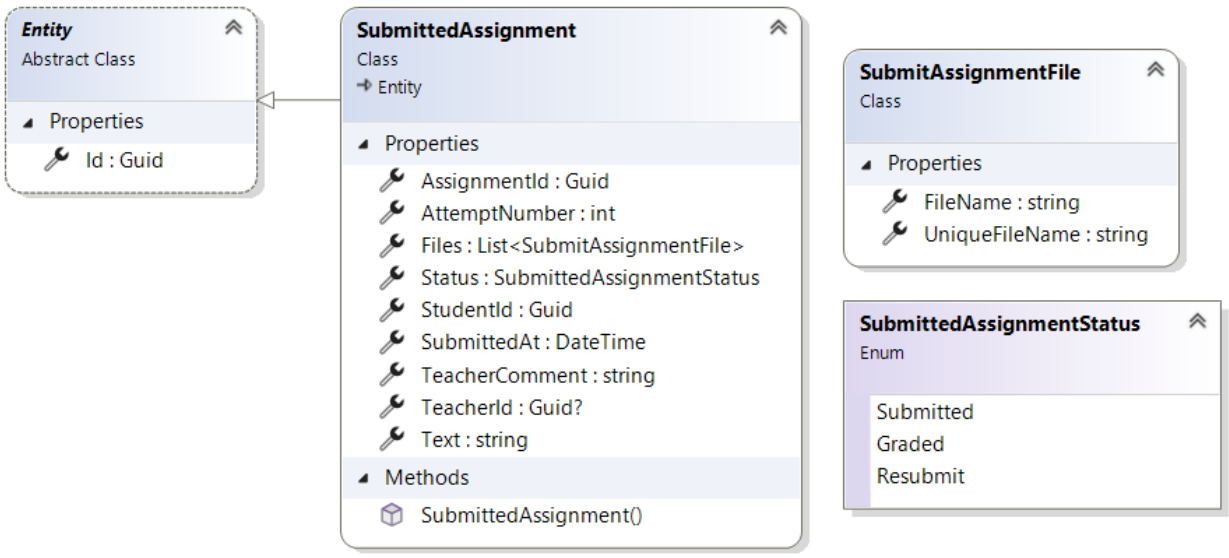

Рисунок 2.6 – Діаграма класів надісланих робіт

Діаграма класів для запитань, пов’язаних із тестами (рис. 2.7) включає:

1. TestQuestion – абстрактний батьківський клас для запитань тесту.

2. TestQuestionType – перерахування, тип запитання.

3. TestQuestionInput – запитання з відкритою відповіддю.

4. TestQuestionSingleChoice – запитання з вибором однієї правильної відповіді із запропонованих варіантів.

5. TestQuestionMultipleChoice – запитання з вибором декількох правильних відповідей із запропонованих варіантів.

6. TestQuestionChoiceOption – доступне значення для вибору для запитань з варіантами вибору.

7. TestQuestionMatching – запитання на встановлення відповідності.

8. TestQuestionMatchOption – доступне значення для встановлення відповідності.

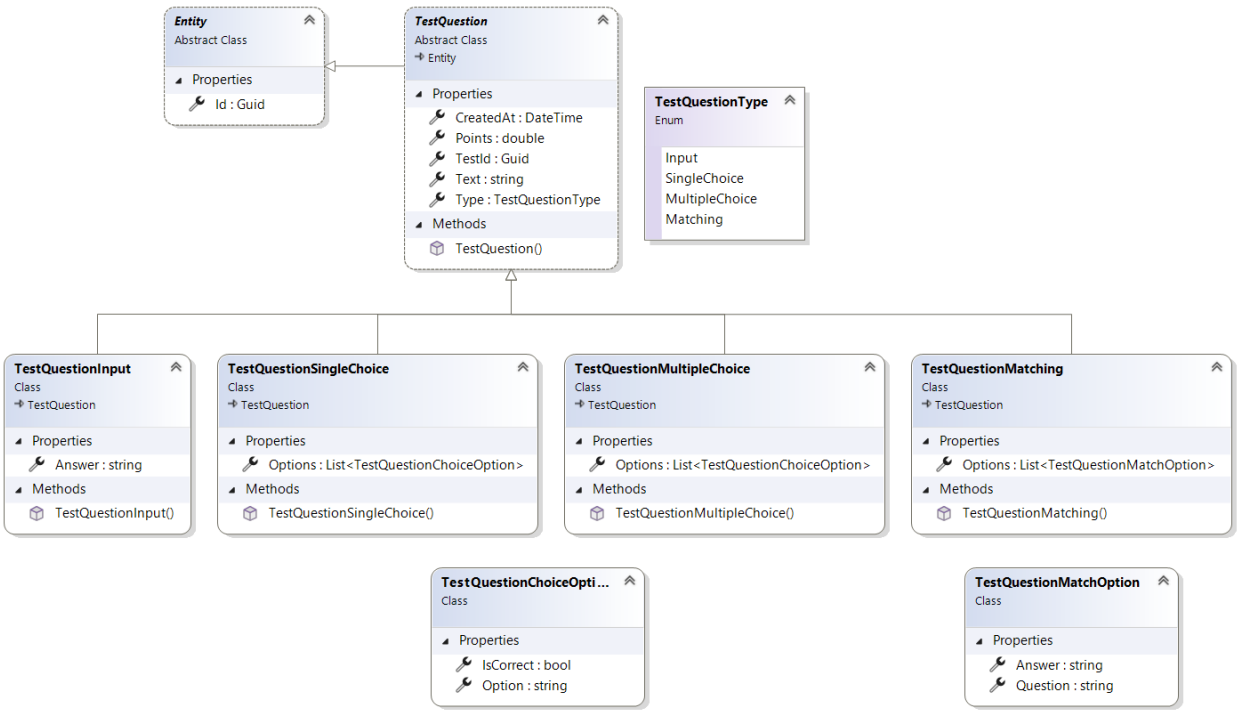

Рисунок 2.7 – Діаграма класів запитання тесту

Для зберігання відповідей студентів на тести використовується наступна діаграма класів (рис. 2.8):

1. TestSession – сесія тесту, містить інформацію про спробу проходження тесту студентом та список його відповідей.

2. TestSessionAnswer – абстрактний батьківський клас для всіх відповідей студента на різні типи запитань.

3. TestSessionAnswerInput – відповідь на запитання з відкритою відповіддю.

4. TestSessionAnswerSingleChoice – відповідь на запитання з вибором однієї правильної відповіді.

5. TestSessionAnswerMultipleChoice – відповідь запитання з вибором декількох правильних відповідей.

6. TestSessionAnswerMatching – відповідь запитання з встановлення відповідності.

7. AnswerMatchOption – обрані відповідності.

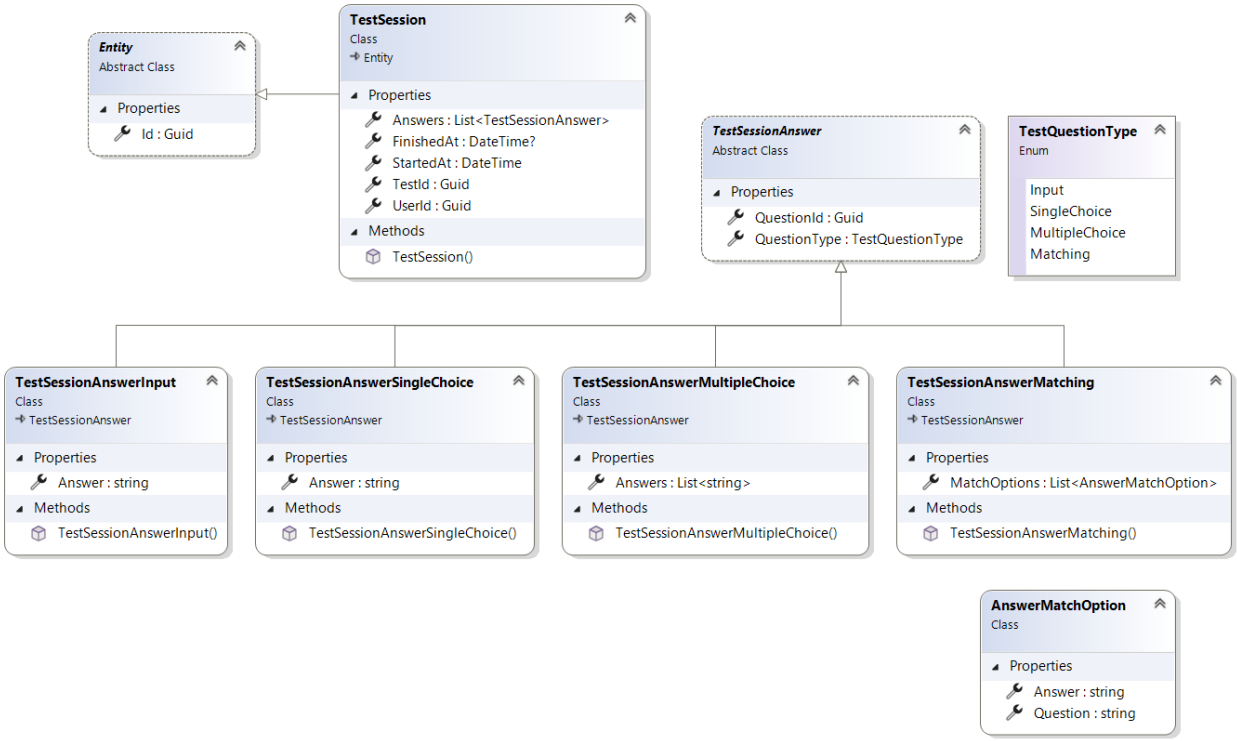

Рисунок 2.8 – Діаграма класів сесія тесту

Діаграма класів для оцінок студентів (рис. 2.9) складається з:

1. ManualGradesColumn – колонка з оцінками, які можна виставляти в ручному режимі, за будь-які активності наприклад відповіді на лекції чи бонусні бали за виконання завдань.

2. Grade – абстрактний батьківський клас для всіх оцінок студента.

3. GradeType – перерахування, тип оцінки.

4. GradeManual – оцінка виставлена в колонці.

5. GradeAssignment – оцінка виставлена за виконання практичної роботи.

6. GradeTest – оцінка за тест, містить список результатів всіх спроб проходження студентом конкретного тесту.

7. GradeTestItem – оцінка за використану спробу проходження тесту.

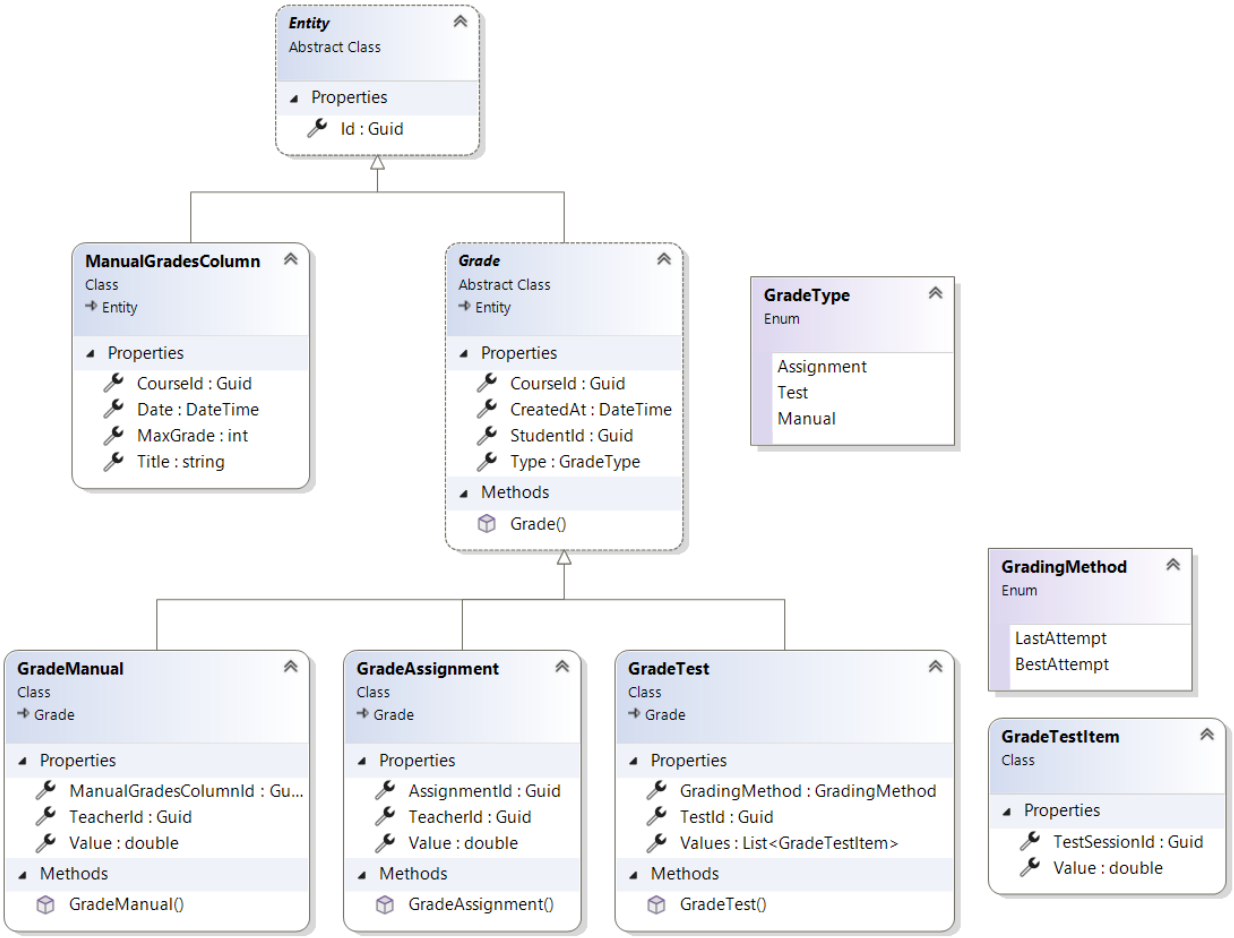

Рисунок 2.9 – Діаграма класів оцінок

2.2. Розробка структури бази даних вебсервісу

Для зберігання структурованих даних користувачів та курсів використовується база даних PostgreSQL. Оскільки ці дані мають чітко визначену та стабільну схему, використання реляційної бази даних є оптимальним рішенням, що забезпечує надійність, цілісність даних, ефективні операції читання та запису, а також можливість встановлення складних зв’язків між таблицями за допомогою ключів та обмежень.

Структура бази даних PostgreSQL складається з 12 таблиць, опис яких наведено у таблицях 2.3 – 2.14.

Таблиця «Users» (табл. 2.3) призначена для зберігання даних про всіх користувачів, що мають обліковий запис на сайті. Поле E-mail має бути унікальним для кожного користувача.

Далі за допомогою підходу TPH (англ. table-per-hierarchy), використовується одна таблиця для зберігання даних всіх типів користувачів, а для визначення типу кожного рядка у таблиці використовується дискримінаційний стовпець «Type», який було перевизначено за допомогою Fluent API [15].

Лістинг налаштування стовпця «Type»:

```
public class UserConfiguration : IEntityTypeConfiguration<User>
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder
            .HasDiscriminator(user => user.Type)
            .HasValue<User>(UserType.User)
            .HasValue<Student>(UserType.Student)
            .HasValue<Teacher>(UserType.Teacher);
    }
}
```

Таблиця 2.3

Опис полів таблиці «Users»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор користувача
Type	text	-	-	Тип користувача
Email	varchar(64), unique	-	-	Адреса електронної пошти
PasswordHash	varchar(512)	-	-	Хеш пароля
RegistrationDate	timestamp	-	-	Дата реєстрації на платформі
FirstName	varchar(64)	-	-	Ім’я
LastName	varchar(64)	-	-	Прізвище
Patronymic	varchar(64)	-	-	По батькові
AvatarUrl	varchar(2048), null	-	-	Посилання на аватар профілю
AvatarImageName	varchar(256), null	-	-	Назва зображення профілю, використовується для видалення файлу
BirthDate	timestamp, null	-	-	Дата народження

Таблиця «Roles» (табл. 2.4) призначена для збереження ролей, доступних у вебдодатку. Залежно від призначеної ролі, користувачі матимуть доступ до різних функцій та можливостей платформи.

Таблиця 2.4

Опис полів таблиці «Roles»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор ролі
Name	varchar(32), unique	-	-	Назва ролі

Таблиця «RoleUser» (табл. 2.5) потрібна, щоб встановити зв’язок багато до багатьох, між таблицями «Roles» та «Users». Один користувач може мати декілька ролей і одна роль може належати декільком користувачам.

Таблиця 2.5

Опис полів таблиці «RoleUser»

Назва	Тип даних	ПК	ЗК	Опис
RolesId	uuid	+	+	Ідентифікатор ролі
UsersId	uuid	+	+	Ідентифікатор користувача

Таблиця «Permissions» (табл. 2.6) зберігає дані про дозволи, які визначають доступ користувачів до певних функцій платформи.

Таблиця 2.6

Опис полів таблиці «Permissions»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор дозволу
Name	varchar(128), unique	-	-	Назва дозволу

Таблиця «PermissionRole» (табл. 2.7) потрібна, щоб встановити зв’язок багато до багатьох, між таблицями «Permissions» та «Roles».

Таблиця 2.7

Опис полів таблиці «PermissionRole»

Назва	Тип даних	ПК	ЗК	Опис
PermissionsId	uuid	+	+	Ідентифікатор дозволу
RolesId	uuid	+	+	Ідентифікатор ролі

Таблиця «RefreshTokens» (табл. 2.8) зберігає інформацію про токени оновлення, які використовуються для продовження сеансів користувачів у вебдодатку. Ці токени дозволяють користувачам повторно пройти

аутентифікацію без необхідності вводити пароль, коли термін дії їх токена доступу закінчується.

Таблиця 2.8

Опис полів таблиці «RefreshTokens»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор токена оновлення
UserId	uuid	-	+	Ідентифікатор користувача
Token	varchar(1024), unique	-	-	Токен, випадковий рядок
ExpiryDate	timestamp	-	-	Дата закінчення терміну дії
RevokedDate	timestamp, null	-	-	Дата відкликання токену

Таблиця «Courses» (табл. 2.9) призначена для зберігання інформації про курси, які створюються користувачами на платформі.

Таблиця 2.9

Опис полів таблиці «Courses»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор курсу
CreatorId	uuid	-	+	Ідентифікатор користувача, який є автором курсу
Title	varchar(64)	-	-	Заголовок
Description	varchar(512), null	-	-	Опис
ImageUrl	varchar(1024), null	-	-	Посилання на головне зображення курсу
ImageName	varchar(128), null	-	-	Назва головного зображення курсу, використовується для видалення файлу
TabType	varchar(16)	-	-	Тип представлення вкладок курсу

Таблиця «CourseTabs» (табл. 2.10) зберігає дані про вкладки курсів, що дозволяють логічно групувати навчальні матеріали. Вона визначає порядок, видимість, назву та інші параметри відображення вкладок на сторінці курсу.

Таблиця 2.10

Опис полів таблиці «CourseTabs»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор вкладки курсу
CourseId	uuid	-	+	Ідентифікатор курсу, до якого належить вкладка
Name	varchar(32)	-	-	Назва вкладки
IsActive	boolean	-	-	Значення, що означає, чи відображати вкладку на сторінці курсу

Продовження таблиці 2.10

Order	integer	-	-	Порядковий номер вкладки
Color	varchar(24), null	-	-	Колір назви вкладки
ShowMaterialsCount	boolean	-	-	Значення, що означає, чи відображати кількість матеріалів, розміщених у цій вкладці

Таблиця «CourseMembers» (табл. 2.11) забезпечує зв’язок між користувачами, курсами та їх ролями, зберігає інформацію про зарахування користувачів на курси та призначені їм ролі. Учасник курсу може мати лише одну роль на курсі, ця колонка є необов’язковою, тому учасник курсу може бути без ролі.

Таблиця 2.11

Опис полів таблиці «CourseMembers»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор учасника курсу
UserId	uuid	-	+	Ідентифікатор користувача
CourseId	uuid	-	+	Ідентифікатор курсу
CourseRoleId	uuid, null	-	+	Ідентифікатор ролі курсу
EnrollmentDate	timestamp	-	-	Дата зарахування на курс

Таблиця «CourseRoles» (табл. 2.12) визначає набір доступних ролей для курсів у вебдодатку. Вона містить інформацію про назву ролі та курс, до якого вона належить. Також, на дану таблицю посилається таблиця «CoursePermissions», це означає, що кожна роль курсу має набір дозволів, які визначають, які дії користувач з цією роллю може виконувати на курсі.

Таблиця 2.12

Опис полів таблиці «CourseRoles»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор ролі курсу
CourseId	uuid	-	+	Ідентифікатор курсу
Name	varchar(32)	-	-	Назва ролі курсу

Таблиця «CoursePermissionCourseRole» (табл. 2.13) потрібна, щоб встановити зв’язок багато до багатьох, між таблицями «CoursePermissions» та «CourseRoles». Одна роль курсу може мати декілька дозволів курсу і один дозвіл курсу може належати декільком ролям курсу.

Таблиця 2.13

Опис полів таблиці «CoursePermissionCourseRole»

Назва	Тип даних	ПК	ЗК	Опис
CoursePermissionsId	uuid	+	+	Ідентифікатор дозволу курсу
CourseRolesId	uuid	+	+	Ідентифікатор ролі курсу

Таблиця «CoursePermissions» (табл. 2.14) містить перелік дозволів, які можуть бути призначені ролям курсу. При запуску платформи виконується перевірка, чи містить таблиця всі наявні дозволи для курсів, визначені в системі. Якщо якісь дозволи відсутні, вони будуть автоматично додані в таблицю. Це забезпечує узгодженість та повноту списку дозволів для ролей курсу.

Таблиця 2.14

Опис полів таблиці «CoursePermissions»

Назва	Тип даних	ПК	ЗК	Опис
Id	uuid	+	-	Ідентифікатор дозволу курсу
Name	varchar(128), unique	-	-	Назва дозволу курсу

За допомогою інструменту ERD (англ. Entity-Relationship Diagram) у pgAdmin 4 було згенеровано діаграму бази даних для PostgreSQL (рис. 2.10). Завдяки інструменту ERD у pgAdmin 4 можливо легко створювати, редагувати та експортувати діаграми баз даних для PostgreSQL [9].

Ця діаграма, представлена у вигляді «сутність-зв’язок», детально відображає структуру БД, що використовується у системі. Вона включає в себе всі таблиці, які складають БД, а також їхні стовпці з вказаними типами даних. Для кожної таблиці позначено первинний ключ, який є унікальним ідентифікатором кожного запису в цій таблиці. Також показані зовнішні ключі, які зв’язують одну таблицю з іншою. Стрілки на діаграмі візуалізують зв’язки між таблицями. Тип зв’язку можна визначити за типом стрілки та позначками на ній [26, с. 52-56].

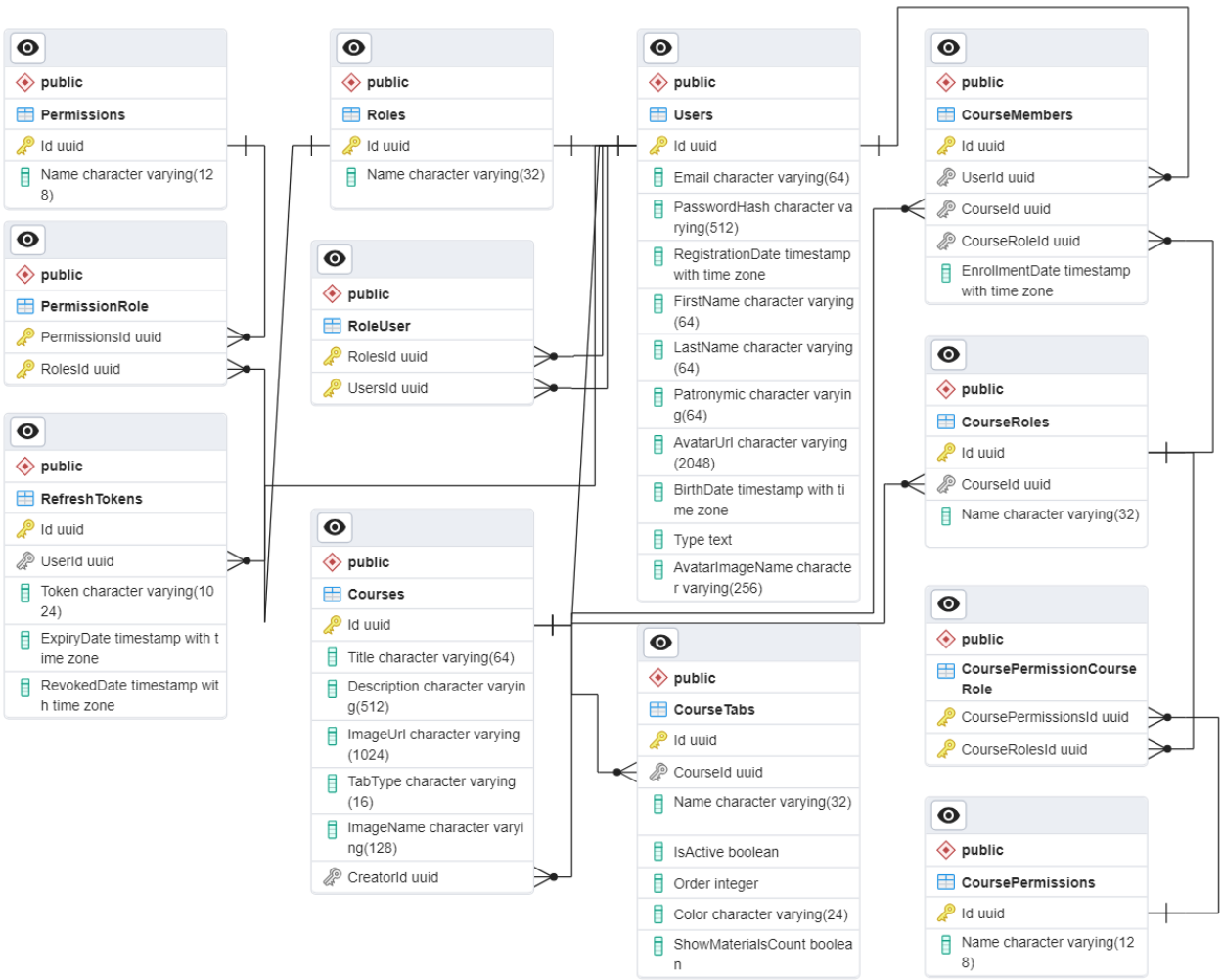

Рисунок 2.10 – Діаграма бази даних PostgreSQL

Для зберігання неструктурованих даних, таких як матеріали курсів, оцінки студентів та сесії проходження тестів, використовується база даних MongoDB. Ця NoSQL БД добре підходить для роботи з даними, які не мають чітко визначеної структури та можуть змінюватися або доповнюватися новими елементами в майбутньому. Вона дозволяє легко додавати нові типи даних та змінювати існуючі, не впливаючи на роботу всієї системи.

Для налаштування карт класів у MongoDB C# Driver було використано ручну конфігурація для поліморфних об’єктів за допомогою класів. Поліморфний об’єкт – це об’єкт, який може мати різні форми або типи, оскільки він успадковує властивості та методи від одного або кількох батьківських класів. Такий об’єкт вимагає спеціальної конфігурації для забезпечення коректної серіалізації та десеріалізації у документи MongoDB за допомогою .NET/C# драйвера [21].

Нижче наведено приклад лістингу конфігурації карти класів для різних типів матеріалів курсу, інші конфігурації мають подібний лістинг.

Лістинг конфігурації карти класів:

```
public static class CourseMaterialClassMap
{
    public static void RegisterClassMaps()
    {
        BsonClassMap.RegisterClassMap<CourseMaterial>(classMap =>
        {
            classMap.AutoMap();
            classMap.SetIsRootClass(true);

            classMap
                .MapMember(courseMaterial => courseMaterial.Type)
                .SetSerializer(new
EnumSerializer<CourseMaterialType>(BsonType.String));
        });

        BsonClassMap.RegisterClassMap<CourseMaterialContent>();
        BsonClassMap.RegisterClassMap<CourseMaterialLink>();
        BsonClassMap.RegisterClassMap<CourseMaterialFile>();
        BsonClassMap.RegisterClassMap<CourseMaterialAssignment>();

        BsonClassMap.RegisterClassMap<CourseMaterialTest>(classMap =>
        {
            classMap.AutoMap();

            classMap
                .MapMember(test => test.GradingMethod)
                .SetSerializer(new
EnumSerializer<GradingMethod>(BsonType.String));
        });
    }
}
```

У даному прикладі створюється статичний метод RegisterClassMaps(), який викликає BsonClassMap.RegisterClassMap для кожного типу, що залежить від CourseMaterial. Клас CourseMaterial визначається як кореневий (SetIsRootClass(true)). Також, додатково налаштований серіалізатор для перерахування «Type», для того щоб тип матеріалу зберігати у вигляді рядка, а не числа, це дозволяє додавати та змінювати порядок елементів перерахування без необхідності оновлення даних в БД [21].

Для реалізації поліморфізму використовується дискримінатор, який допомагає MongoDB відрізняти різні підкласи об’єкту. Драйвер зберігає

значення дискримінатора у полі документа з назвою «**_t**». Значення «**_t**» – це масив усіх класів в ієрархії успадкування типу документа. Дискримінатор автоматично додається до документа в БД і містить інформацію про конкретний тип об’єкта, що забезпечує коректну десеріалізацію при витягуванні об’єктів з MongoDB [21].

Схема MongoDB складається з 6 колекцій, опис яких наведено у таблицях 2.15 – 2.26.

Колекція «CourseMaterials» (табл. 2.15) містить наступні типи матеріалів курсу: текстове наповнення (контент), посилання, файл, практична робота та тест. За допомогою поля CourseTabId можна отримати всім матеріали вкладки.

Таблиця 2.15

Опис полів колекції «CourseMaterials»

Назва	**Тип даних**	**Дискримінатор**	**Опис**
_id	Binary	CourseMaterial	Ідентифікатор матеріалу курсу
CourseTabId	Binary	CourseMaterial	Ідентифікатор вкладки курсу
Type	String	CourseMaterial	Тип матеріалу курсу
IsActive	Boolean	CourseMaterial	Значення, що означає, чи відображати матеріал на сторінці курсу та чи доступна практична робота або тест
Order	Int32	CourseMaterial	Порядковий номер матеріалу у вкладці
CreatedAt	Date	CourseMaterial	Дата створення матеріалу
Content	String	CourseMaterialContent	Текстове наповнення матеріалу
Title	String	CourseMaterialLink	Заголовок посилання
Link	String	CourseMaterialLink	URL посилання
Title	String	CourseMaterialFile	Заголовок файлу
FileName	String	CourseMaterialFile	Ім’я файлу
UniqueFileName	String	CourseMaterialFile	Унікальне ім’я файлу, використовується для видалення
Title	String	CourseMaterialAssignment	Назва практичної роботи
Description	String	CourseMaterialAssignment	Опис практичної роботи
Deadline	Date, Null	CourseMaterialAssignment	Крайній термін здачі практичної роботи
MaxFiles	Int32	CourseMaterialAssignment	Максимальна кількість файлів, яку можуть завантажити студенти
MaxGrade	Int32	CourseMaterialAssignment	Максимальна оцінка за практичну оцінку

Продовження таблиці 2.15

Title	String	CourseMaterialTest	Назва тесту
Description	String, Null	CourseMaterialTest	Опис тесту
NumberAttempts	Int32	CourseMaterialTest	Кількість спроб на проходження тесту
TimeLimitInMinutes	Int32, Null	CourseMaterialTest	Обмеження в часі (в хвилинах)
Deadline	Date, Null	CourseMaterialTest	Дата завершення тесту
GradingMethod	String	CourseMaterialTest	Метод оцінювання, визначає з якої спроби буде взято оцінку за проходження тесту
ShuffleQuestions	Boolean	CourseMaterialTest	Значення, що означає, чи перемішувати питання у тесті

Колекція «SubmittedAssignments» (табл. 2.16) містить інформацію про виконані студентами практичні роботи, включаючи їх статус, надіслані файли (табл. 2.17), кількість спроб, коментарі викладача та дату відправлення.

Таблиця 2.16

Опис полів колекції «SubmittedAssignments»

Назва	Тип даних	Опис
_id	Binary	Ідентифікатор відправленого завдання
AssignmentId	Binary	Ідентифікатор завдання
StudentId	Binary	Ідентифікатор студента
TeacherId	Binary	Ідентифікатор викладача
Status	String	Статус відправленого завдання (відправлено на перевірку, повернуто на доопрацювання, оцінено)
AttemptNumber	Int32	Кількість спроб виконання завдання
Text	String, Null	Текстова відповідь студента
Files	Array< SubmitAssignmentFile>, Null	Масив файлів, відправлених із виконаним завданням
TeacherComment	String, Null	Коментар викладача
SubmittedAt	Date	Дата відправки завдання

Таблиця 2.17

Опис полів об’єкту «SubmitAssignmentFile»

Назва	Тип даних	Опис
FileName	String	Ім’я файлу
UniqueFileName	String	Унікальне ім’я файлу, використовується для видалення

Колекція «TestQuestions» (табл. 2.18) зберігає різні типи запитань, що використовуються у тестах. Містить інформацію про текст запитань, кількість балів за правильну відповідь, а також варіанти відповідей (табл. 2.19-2.20, залежно від типу запитання).

Таблиця 2.18

Опис полів колекції «TestQuestions»

Назва	Тип даних	Дискримінатор	Опис
_id	Binary	TestQuestion	Ідентифікатор запитання тесту
TestId	Binary	TestQuestion	Ідентифікатор тесту
Type	String	TestQuestion	Тип запитання
Text	String	TestQuestion	Текст запитання
Points	Double	TestQuestion	Кількість балів за правильну відповідь
CreatedAt	Date	TestQuestion	Дата створення запитання
Answer	String	TestQuestionInput	Правильна відповідь на запитання з відкритою відповіддю
Options	Array< TestQuestionChoiceOption>	TestQuestionSingleChoice, TestQuestionMultipleChoice	Масив варіантів відповіді на запитання з однією чи декількома правильними відповідями
Options	Array< TestQuestionMatchOption>	TestQuestionMatching	Масив пар «запитання-відповідь» для встановлення відповідності

Таблиця 2.19

Опис полів об’єкту «TestQuestionChoiceOption»

Назва	Тип даних	Опис
Option	String	Текст варіанту відповіді
IsCorrect	Boolean	Значення, що означає, чи даний варіант є правильною відповіддю

Таблиця 2.20

Опис полів об’єкту «TestQuestionMatchOption»

Назва	Тип даних	Опис
Question	String, Null	Текст запитання, може бути відсутнім, щоб кількість варіантів вибору правильних відповідей була більшою, за кількість запитань
Answer	String	Правильна відповідь на запитання

Колекція «TestSessions» (табл. 2.21) містить інформацію про сесії проходження тестів студентами, час початку та завершення, а також надані відповіді на запитання (табл. 2.22-2.23).

Таблиця 2.21

Опис полів колекції «TestSessions»

Назва	Тип даних	Опис
_id	Binary	Ідентифікатор сеансу проходження тесту
TestId	Binary	Ідентифікатор тесту
UserId	Binary	Ідентифікатор користувача, який проходить тест
StartedAt	Date	Дата та час початку проходження тесту
FinishedAt	Date, Null	Дата та час закінчення проходження тесту, може бути відсутнім, якщо тест не завершено вручну
Answers	Array<TestSessionAnswer>	Масив відповідей користувача на питання тесту

Таблиця 2.22

Опис полів об’єкту «TestSessionAnswer»

Назва	Тип даних	Дискримінатор	Опис
QuestionId	Binary	TestSessionAnswer	Ідентифікатор запитання тесту
QuestionType	Binary	TestSessionAnswer	Тип запитання
Answer	String, Null	TestSessionAnswerInput, TestSessionAnswerSingleChoice	Відповідь користувача на запитання з відкритою відповіддю або з вибором однієї правильної відповіді
Answers	Array<String>, Null	TestSessionAnswerMultipleChoice	Масив відповідей користувача на питання з декількома правильними відповідями
MatchOptions	Array<AnswerMatchOption>, Null	TestSessionAnswerMatching	Масив відповідей користувача на питання з встановлення відповідності

Таблиця 2.23

Опис полів об’єкту «AnswerMatchOption»

Назва	Тип даних	Опис
Question	String	Текст запитання, до якого було обрано відповідь
Answer	String, Null	Обрана відповідь користувача

Колекція «ManualGradesColumns» (табл. 2.24) містить інформацію про додаткові стовпці з оцінками, які викладачі можуть додавати вручну до курсу. Кожен стовпець містить заголовок, максимальну оцінку та дату додавання. При відображенні таблиці з оцінками студентів, стовпці сортуються за датою додавання. Оцінки студентів у цих стовпцях не можуть перевищувати вказаної максимальної оцінки для даної колонки.

Таблиця 2.24

Опис полів колекції «ManualGradesColumns»

Назва	Тип даних	Опис
_id	Binary	Ідентифікатор стовпця з оцінками, які можна додавати вручну за будь-яку активність
CourseId	Binary	Ідентифікатор курсу
Title	String	Заголовок стовпця
Date	Date	Дата додавання стовпця
MaxGrade	Int32	Максимальна оцінка, яку можна отримати в цій колонці

Колекція «Grades» (табл. 2.25) містить інформацію про оцінки студентів на курсах за різні види діяльності, включаючи завдання, тести та додаткові оцінки, що вводяться вручну. Зміна максимального балу на менше значення в практичних роботах чи колонках з додатковими оцінками, не впливає на наявні оцінки в цій колекції.

Для тестів зберігається масив оцінок (табл. 2.26) за всі спроби, але для підрахунку підсумкової оцінки використовується лише одна з них, обрана відповідно до налаштованого методу оцінювання.

Таблиця 2.25

Опис полів колекції «Grades»

Назва	Тип даних	Дискримінатор	Опис
_id	Binary	Grade	Ідентифікатор оцінки
CourseId	Binary	Grade	Ідентифікатор курсу
StudentId	Binary	Grade	Ідентифікатор студента
Type	String	Grade	Тип оцінки (завдання, тест, вручну)
CreatedAt	Date	Grade	Дата створення оцінки
TeacherId	Binary	GradeAssignment	Ідентифікатор викладача, який виставив оцінку
AssignmentId	Binary	GradeAssignment	Ідентифікатор завдання, за яке висталено оцінку
Value	Double	GradeAssignment	Оцінка за виконане завдання
TestId	Binary	GradeTest	Ідентифікатор тесту, за який виставлено оцінку
GradingMethod	String	GradeTest	Метод оцінювання тесту
Values	Array< GradeTestItem>	GradeTest	Масив результатів всіх спроб проходження тесту
ManualGradesColumnId	Binary	GradeManual	Ідентифікатор стовпця з оцінками, які можна додавати вручну
TeacherId	Binary	GradeManual	Ідентифікатор викладача, який виставив оцінку
Value	Double	GradeManual	Оцінка, виставлена викладачем вручну

Таблиця 2.26

Опис полів об’єкту «GradeTestItem»

Назва	Тип даних	Опис
TestSessionId	Binary	Ідентифікатор сеансу проходження тесту
Value	Double	Оцінка, отримана за використану спробу проходження тесту

За допомогою інструменту Moon Modeler було згенеровано діаграму колекцій бази даних для MongoDB (рис. 2.11). Дана програма може імпортувати дані з уже існуючої БД, а потім автоматично згенерувати діаграму, яка показує всі колекції, документи та зв’язки між ними [24].

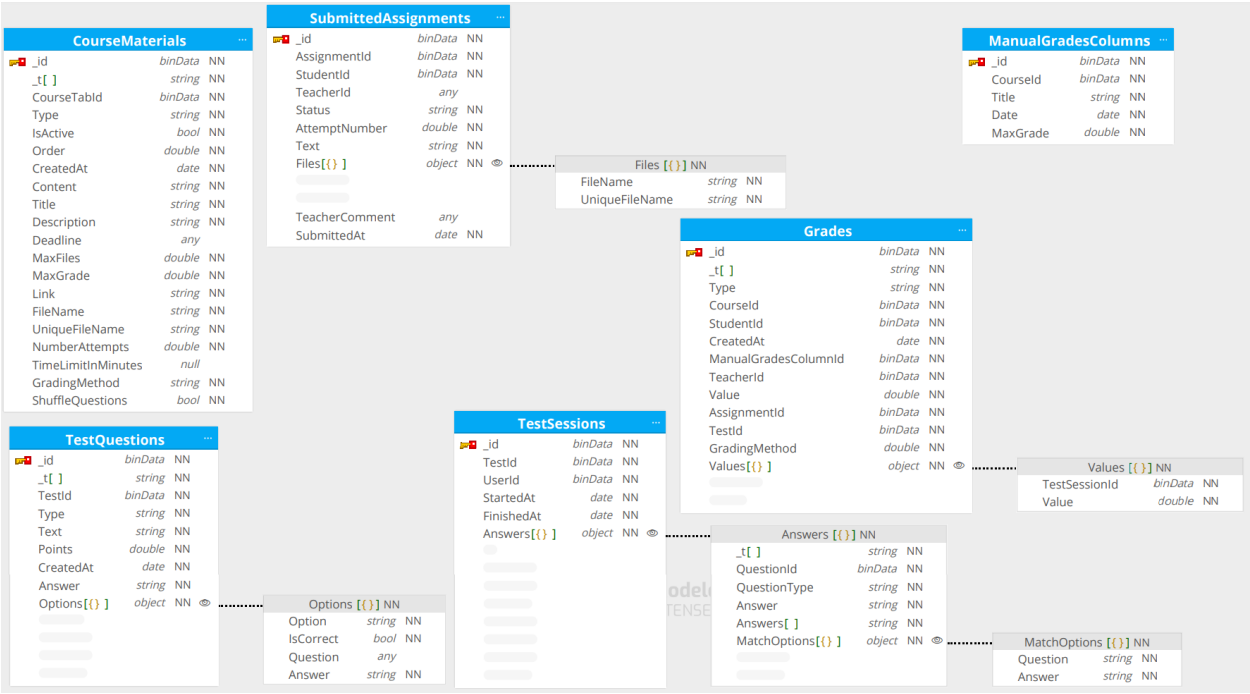

Рисунок 2.11 – Діаграма колекцій бази даних MongoDB

2.3. Проєктування та реалізація алгоритмів роботи системи

Усі запити до API обробляються на платформі ASP.NET Core за допомогою бібліотеки MediatR, яка спрощує реалізацію патернів CQRS та Mediator.

Патерн Mediator – це поведінковий патерн проєктування, що дає змогу зменшити зв’язаність великої кількості класів між собою, завдяки переміщенню цих зв’язків до одного класу-посередника. Цей шаблон обмежує пряме спілкування між об’єктами і змушує їх співпрацювати лише через об’єкт-посередник. Посередник використовується для зменшення

складності комунікації між різними класами або об’єктами. Цей патерн надає клас-посередник, який зазвичай відповідає за всю комунікацію між різними класами і сприяє легкому обслуговуванню коду через слабку залежність [39, с. 295-297].

Патерн CQRS, що означає «Command and Query Responsibility Segregation» або поділ відповідальності на команди та запити, тобто він розділяє операції над даними на дві категорії: команди, які вносять зміни до стану системи (створення, оновлення або видалення даних) та запити – операції отримання даних, без внесення змін до стану. Впровадження CQRS у додаток може поліпшити продуктивність та спростити масштабованість. Поділ операцій читання і запису дає змогу легко тестувати кожну з них окремо [5].

Застосування патернів CQRS та Mediator дозволяє перенести основну логіку виконання методів з контролерів до інших класів рівня Application (див. рис. 1.6). Методи для модифікації даних розміщуються у відповідних класах-командах, а методи для отримання даних – у класах-запитах. Такий підхід дозволяє зробити код більш модульним, зменшити залежності та полегшити обслуговування та розширення системи.

Кожна операція (Command або Query) зберігається у власному каталозі, де також можуть зберігатися інші класи, наприклад:

– Response – об’єкт, який буде повернуто після виконання команди чи запиту;

– Request – об’єкт, що використовується для передачі даних з контролера до команди чи запиту;

– Validator – клас, який наслідується від класу AbstractValidator бібліотеки FluentValidation, для визначення правил валідації даних, що передаються до команди чи запиту;

– Dto – об’єкт, призначений для передачі даних між компонентами системи.

Переваги використання патернів CQRS та Mediator:

– допомагають відокремити логіку обробки запитів від логіки контролера чи сервісу, що полегшує розширення та підтримку коду;

– дозволяють легко масштабувати додаток шляхом додавання нових запитів та обробників без змін у вже існуючому коді;

– сприяють уникненню прямих залежностей між компонентами системи, що робить код більш гнучким та легким для тестування [7, с. 277].

До недоліків використання патернів CQRS та Mediator можна віднести:

– написання подібного коду;

– створення більшої кількості класів.

Однією з ключових операцій в системі є завантаження виконаних завдань студентами. Цей процес можна проілюструвати за допомогою діаграми діяльності, яка містить дві доріжки (рис. 2.12). У діаграмах діяльності доріжки використовуються для групування дій за певною ознакою, які виконуються одним актором в одному потоці. Дана діаграма містить два потоки, які позначають роль студента та викладача.

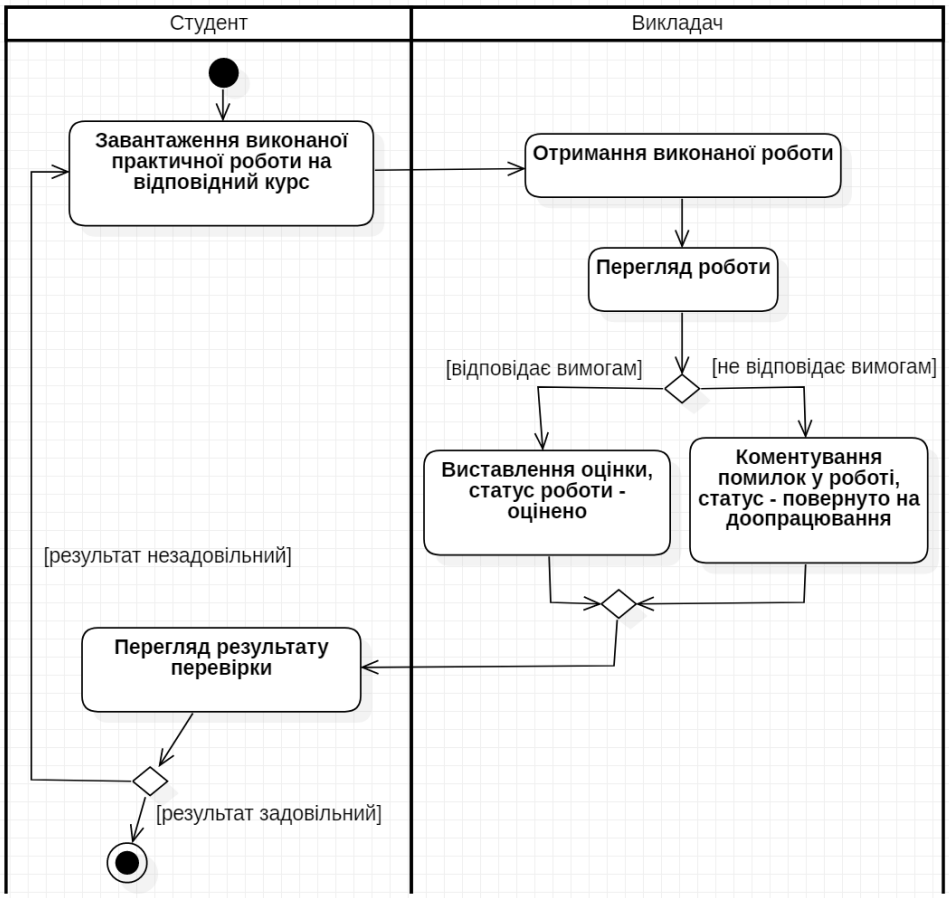

Рисунок 2.12 – Діаграма діяльності для завантаження виконаних завдань

Студент може виконати такі дії, як написати текстове повідомлення та завантажити дозволену кількість файлів з виконаним завданням до відповідного курсу й після перевірки переглянути результати. Якщо результат задовільний, то переходимо до кінцевого стану, інакше студент має можливість виправити свою роботу та знову її надіслати викладачу на повторну перевірку.

Після того як студент відправив виконанні завдання, викладач має можливість перевірити його роботу та завантажити надіслані файли. Далі викладач перевіряє роботу й якщо завдання було добре виконано він виставляє оцінку, інакше пише коментар, що потрібно виправити й позитивні або негативні результати перевірки відправляються студенту.

Наступна важлива частина системи дистанційного навчання – це можливість проходити тести. Студенти повинні мати можливість проходити різноманітні тести з різних курсів та на різні теми. Даний процес зображено на наступній діаграмі діяльності (рис. 2.13).

Першою дією є запуск тесту, але він може бути недоступним, якщо студент використав всі спроби, завершився термін тесту, у тесті відсутні запитання або його було зупинено.

Коли студент запускає тест, система перевіряє налаштування тесту та, за необхідності, виконує перемішування запитань у випадковому порядку. Далі система повертає список запитань відповідного тесту й студент може почати вводити відповіді. Введенні відповіді одразу зберігаються в системі та обчислюється поточний результат проходження тесту. Таким чином, викладачі можуть переглядати результати проходження тесту студентами в таблиці з оцінками курсу. При цьому студенти не мають можливості побачити свою оцінку під час проходження тесту.

Після того, як студент закінчить відповідати на запитання, він завершує тест. У системі фіксується дата та час завершення тесту, а отримана оцінка одразу стає доступною для перегляду студенту.

На основі отриманих результатів процес проходження тесту буде або завершено, або студент може знову його запустити, якщо отримав незадовільний результат і має можливість повторно пройти тест.

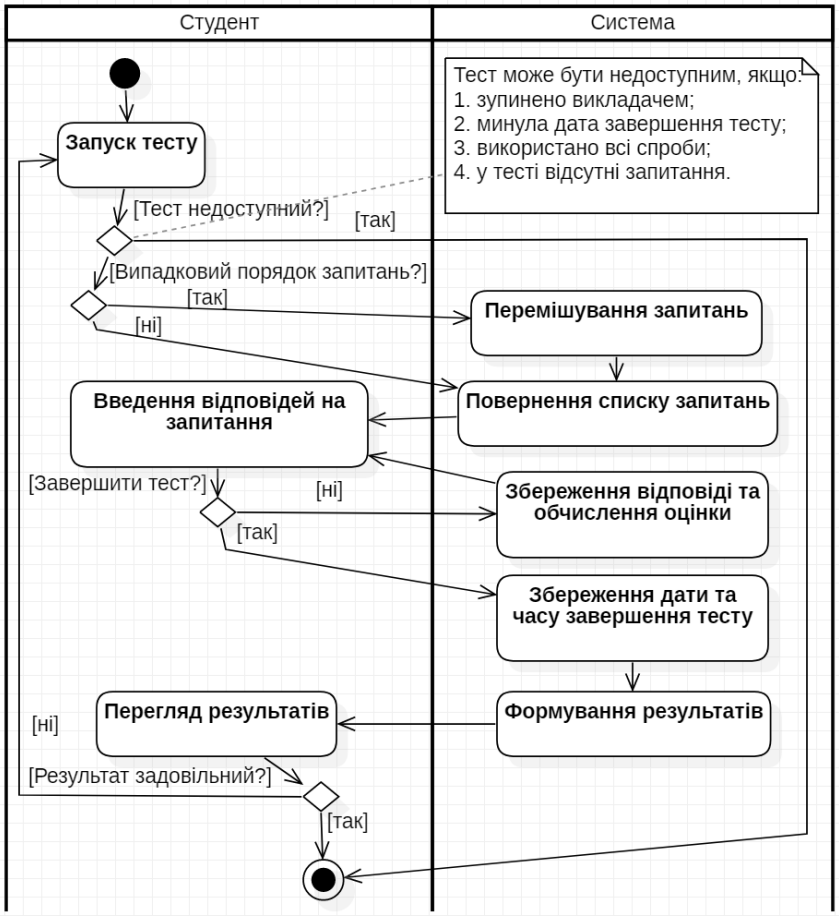

Рисунок 2.13 – Діаграма діяльності для проходження тестів

2.4. Реалізація функціональної частини вебдодатку

Розробка RESTful API на ASP.NET Core виконувалась наступним чином, було створено: сутності, репозиторії для роботи з БД, команди та запити для виконання основної логіки платформи та налаштовано авторизацію на основі прав доступу.

Спочатку було визначено загальні абстрактні класи репозиторіїв ApplicationDbRepository<TEntity> та MongoDbRepository<TDocument>. Лістинг коду наведено в додатку А. Вони використовуються для роботи з сутностями типу TEntity, де TEntity та TDocument мають успадковуватися від базового класу Entity. ApplicationDbRepository<TEntity> реалізує загальні CRUD-операції за допомогою Entity Framework Core, тоді як

MongoDbRepository<TDocument> використовує MongoDB C# Driver для виконання аналогічних операцій.

Класи є абстрактними, оскільки не призначені для прямого використання, а слугують базовими класами для інших конкретних репозиторіїв, які будуть відповідати за роботу з конкретними типами сутностей в системі. Цей підхід дозволяє уникнути дублювання коду та спростити реалізацію інших репозиторіїв додатку.

Для роботи з MongoDB за допомогою MongoDB C# Driver рекомендується налаштувати обробку GUID-значень. За замовчуванням драйвер використовує формат GuidRepresentation.CSharpLegacy, але щоб уникнути проблем із сумісністю при обміні даними між різними компонентами системи та забезпечити правильне збереження і читання, необхідно використати наступний код [12]:

```
BsonSerializer.RegisterSerializer(new
GuidSerializer(GuidRepresentation.Standard));
BsonDefaults.GuidRepresentation = GuidRepresentation.Standard;
```

Цей код налаштовує MongoDB C# Driver для використання стандартного формату GUID при серіалізації та десеріалізації. Спочатку реєструється новий серіалізатор для типу GUID, який використовує формат GuidRepresentation.Standard. Наступний крок встановлює формат GuidRepresentation.Standard як формат за замовчуванням для всіх GUID-значень, що використовуються в додатку [12].

Наступним кроком є налаштування генерації JWT токена доступу. Цей токен буде використовуватися для автентифікації користувачів та авторизації їх доступу до ресурсів та функцій системи. Для створення JWT токена створено метод GenerateAccessToken, який приймає параметром сутність користувача.

Лістинг методу GenerateAccessToken:

```
public TokenDto GenerateAccessToken(User user)
{
    var claims = new Claim[]
    {
        new(JwtRegisteredClaimNames.Sub, user.Id.ToString()),
```

```
        new(JwtRegisteredClaimNames.Email, user.Email)
    };

    var signingCredentials = new SigningCredentials(
        new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_jwtOptions.SecretKey)),
        SecurityAlgorithms.HmacSha256);

    DateTime expiresDate = DateTime.UtcNow
.AddMinutes(_jwtOptions.AccessTokenExpirationInMinutes);

    var securityToken = new JwtSecurityToken(
        issuer: _jwtOptions.Issuer,
        audience: _jwtOptions.Audience,
        claims: claims,
        expires: expiresDate,
        signingCredentials: signingCredentials);

    string token = new JwtSecurityTokenHandler()
.WriteToken(securityToken);

    return new TokenDto { Token = token, ExpiresDate = expiresDate };
}
```

Спочатку створюється масив claims (дані, які будуть закодовані в токені), який містить ідентифікатор користувача (sub) та його електронну адресу (email).

Далі створюються об’єкти SigningCredentials та SymmetricSecurityKey, які використовуються для підписування токена секретним ключем за допомогою алгоритму HMAC SHA-256. Потім обчислюється термін дії токена, який визначений в конфігурації платформи.

Після цього створюється об’єкт JwtSecurityToken, який містить інформацію про видавця (issuer), цільову аудиторію (audience), закодовані дані (claims), дату закінчення терміну дії (expires) та підпис токена (signingCredentials).

Нарешті, токен кодується в рядок за допомогою JwtSecurityTokenHandler і повертається разом із датою закінчення терміну дії в об’єкті TokenDto.

Після налаштування генерації JWT токена доступу, необхідно реалізувати генерацію токена оновлення, який дозволяє користувачам отримувати новий токен доступу без необхідності повторної аутентифікації,

коли у поточного токену доступу закінчиться термін дії. Для цього використовується метод GenerateRefreshToken, який створює випадковий токен оновлення.

Лістинг методу GenerateAccessToken:

```
public TokenDto GenerateRefreshToken()
{
    byte[] randomNumber = new byte[32];
    using var rng = RandomNumberGenerator.Create();

    rng.GetBytes(randomNumber);

    string token = Convert.ToBase64String(randomNumber);
    DateTime expiresDate =
DateTime.UtcNow.AddDays(_jwtOptions.RefreshTokenExpirationInDays);

    return new TokenDto { Token = token, ExpiresDate = expiresDate };
}
```

Даний метод створює випадковий рядок довжиною 32 байти за допомогою класу RandomNumberGenerator. Потім згенерований масив байтів конвертується в рядок і повертається як токен оновлення. Також встановлюється термін дії для токена оновлення.

Для забезпечення авторизації на основі прав доступу для користувачів і учасників курсу використовуються атрибути HasPermissionAttribute та HasCourseMemberPermissionAttribute. Вони створюють нові політики, що починається з визначених префіксів й далі перелік необхідних дозволів, розділяється комами. Ці атрибути дозволяють визначати, використовуючи перерахування PermissionType та CoursePermissionType, які дозволи мають бути наявні у користувача чи учасника курсу для доступу до певного методу в конторлері. Лістинг коду атрибутів та постачальника політик авторизації наведено в додатку Б.

Клас PermissionAuthorizationPolicyProvider використовується для налаштування політик авторизації. Він перевизначає метод GetPolicyAsync базового класу DefaultAuthorizationPolicyProvider, щоб створювати динамічно політики авторизації, а не реєструвати їх вручну. Також він дістає масив

дозволів з назви політики та передає їх до класів з вимогами PermissionRequirement та CourseMemberPermissionRequirement [6, 20].

Для перевірки доступу до методів контролера використовуються обробники авторизації PermissionAuthorizationHandler та CourseMemberPermissionAuthorizationHandler. Перший обробник відповідає за перевірку прав доступу користувачів платформи, а другий – учасників курсу. Ці обробники авторизації перевіряють, чи має користувач необхідні дозволи для виконання певної дії, і в разі успішної перевірки надають доступ до методу контролера.

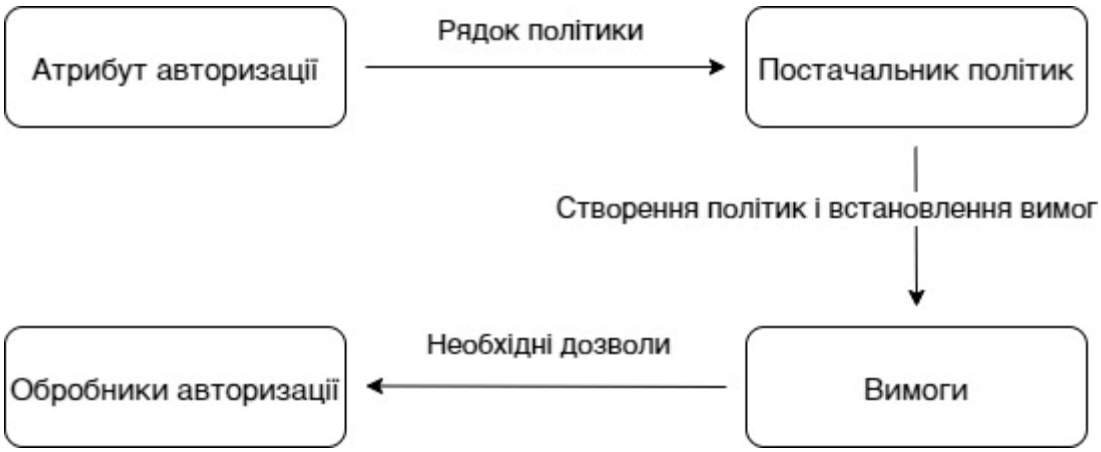

Рисунок 2.14 – Процес авторизації

Нижче наведено приклад використання атрибуту HasPermission над методом CreateCourse для створення курсу. Лише користувачі, які мають дозвіл на створення нових курсів, можуть виконати цей метод. Якщо користувач не має цього дозволу, система авторизації заборонить виконання методу та поверне помилку, що доступ до запрошеного ресурсу заборонено.

```
[HasPermission(PermissionType.UpdateOwnProfile)]
[HttpPut]
public async Task<IActionResult> UpdateCurrentProfile(
    [FromBody] UpdateCurrentProfileRequest request,
    CancellationToken cancellationToken)
{
    var command = new UpdateCurrentProfileCommand(
        request.Email,
        request.FirstName,
        request.LastName,
        request.Patronymic,
        request.BirthDate);
```

```
    return HandleResult(await Sender.Send(command,
cancellationToken));
}
```

Далі розміщено лістинг використання атрибуту HasCourseMemberPermission над методом CreateCourseMaterialAssignment для створення практичної роботи. Лише учасники курсу, які мають дозвіл на додавання нових матеріалів, можуть виконати цей метод. Перевірка наявності доступу до курсу виконується за ідентифікатором вкладки курсу (tabId), який береться з параметрів маршруту. Якщо учасник не має дозволу на створення матеріалу, але як користувач має дозвіл на керування курсами, то це також дозволяє йому змінювати дані курсу, навіть якщо він не є його учасником.

```
[HasCourseMemberPermission(
    [CoursePermissionType.CreateCourseMaterial],
    [PermissionType.ManageCourse])]
[HttpPost("assignment/{tabId:guid}")]
public async Task<IActionResult> CreateCourseMaterialAssignment(
    Guid tabId,
    [FromBody] CreateCourseMaterialAssignmentRequest request,
    CancellationToken cancellationToken)
{
    var command = new CreateCourseMaterialAssignmentCommand(
        tabId,
        request.Title,
        request.Description,
        request.Deadline,
        request.MaxFiles,
        request.MaxGrade);

    return HandleResult(await Sender.Send(command,
cancellationToken));
}
```

Розглянемо метод відправки виконаного завдання студентом. Лістинг методу наведено в додатку В. Лише студенти можуть завантажувати виконані практичні роботи в систему. Якщо завдання було видалено або на момент відправки не є активним, користувач не зможе завантажити завдання та отримає відповідне повідомлення про помилку. Також не можна повторно відправити завдання, якщо його вже було оцінено.

Якщо завдання відправлялося раніше, то крайній термін здачі не враховується, і студент може виправити помилки та відправити роботу повторно. Однак, якщо студент намагається відправити роботу вперше після закінчення терміну здачі, система не дозволить здати завдання. Далі виконується перевірка, чи студент не намагається завантажити більшу кількість файлів, ніж зазначено в умовах практичної роботи. Якщо кількість файлів не перевищує максимально допустиму, вони зберігаються.

Останній крок – збереження запису про відправку завдання в БД. Якщо це перша спроба, то буде створено новий запис, інакше – оновлено існуючий, але пере цим буде видалено файли, які були завантажені раніше, тобто відбудеться перезапис відповіді. Також при повторній відправці завдання статус роботи буде змінено на «Відправлено на перевірку».

Перейдемо до алгоритму обчислення оцінки під час проходження тесту (див. рис. 2.13). Коли студент надсилає відповідь на запитання тесту, система зберігає відповідь та оновлює оцінку за тест. При оновленні оцінки враховуються два випадки:

1. Якщо попередня відповідь була відсутня або неправильною, а поточна відповідь є правильною, до оцінки додаються бали за правильну відповідь на запитання.

2. Якщо попередня відповідь була правильною, а нова відповідь стала неправильною, від оцінки віднімаються бали за це запитання.

Нижче наведено лістинг методів для перевірки правильності відповідей на запитання різних типів:

1. Метод IsAnswerInputCorrect перевіряє правильність відповіді, на запитання з відкритою відповіддю, шляхом порівняння відповіді користувача з очікуваною відповіддю, зазначеною в сутності запитання:

```
private static bool IsAnswerInputCorrect(TestQuestionInput question,
string? answer) =>
    question.Answer == answer;
```

2. Перевірку правильності відповіді на запитання з одним варіантом відповіді виконує метод IsAnswerSingleChoiceCorrect. Він перевіряє, чи

співпадає відповідь студента з варіантом, позначеним як правильний у списку запропонованих варіантів на вибір.

```
private static bool
IsAnswerSingleChoiceCorrect(TestQuestionSingleChoice question, string?
answer) =>
    question.Options.FirstOrDefault(x => x.IsCorrect)?.Option ==
answer;
```

3. IsAnswerMultipleChoiceCorrect: перевіряє, чи всі правильні варіанти вибрані студентом і чи немає зайвих варіантів у відповіді.

```
private static bool IsAnswerMultipleChoiceCorrect(
    TestQuestionMultipleChoice question,
    List<string>? answers) =>
    answers is not null &&
    question.Options.Where(x => x.IsCorrect).Select(x => x.Option)
        .All(x => answers.Contains(x));
```

4. IsAnswerMatchingCorrect: перевіряє, чи співпадають всі відповіді студента з правильними відповідностями запитання.

```
private static bool IsAnswerMatchingCorrect(
    TestQuestionMatching question,
    List<AnswerMatchOption>? matchOptions) =>
    matchOptions is not null &&
    question.Options.Where(x => !string.IsNullOrEmpty(x.Question))
        .All(x => matchOptions
            .Any(answer => answer.Question == x.Question &&
answer.Answer == x.Answer));
```

Клієнтський додаток було розроблено на React з використанням Redux для керування станом додатку та RTK Query для роботи з API, також для побудови інтерфейсу використано готові стилізовані компоненти з бібліотеки Ant Design.

Використання Redux спрощує процес керування станом, оскільки він зберігає його в єдиному об’єкті, який можна легко оновлювати через виклик функції dispatch. Це особливо зручно, коли один і той самий стан потрібно використовувати в різних компонентах або коли необхідно забезпечити взаємодію між кількома компонентами.

Стан додатка зберігає інформацію про автентифікованого користувача, щоб забезпечити доступ тільки до тих частин сайту, до яких цей користувач має права. При першому завантаженні сторінки надсилається запит до

сервера для отримання даних про поточного користувача, і поки запит виконується, користувачу відображається індикатор завантаження.

Лістинг компонента AuthMiddleware, який відповідає за завантаження автентифікованого користувача в стан додатка:

```
interface Props {
  children: React.ReactNode;
}

export function AuthMiddleware({ children }: Props) {
  const { t } = useTranslation();
  const appDispatch = useAppDispatch();
  const { displayError } = useDisplayError();

  const { isAuth } = useAuth();
  const [refresh] = useRefreshMutation();

  const [isLoading, setIsLoading] = useState(true);

  useLayoutEffect(() => {
    if (isAuth) {
      setIsLoading(false);
      return;
    }

    refresh()
      .unwrap()
      .then((response) => appDispatch(setCredentials(response)))
      .catch((error) => displayError(error, { display: false }))
      .finally(() => setIsLoading(false));

    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, []);

  if (isLoading) {
    return <Spin fullscreen tip={t('loading.app')} />;
  }

  return children;
}
```

Висновки до другого розділу

В процесі проєктування платформи електронного навчання було визначено основні варіанти використання залежно від ролі користувача. На основі отриманого результату було створено функціональну карту вебдодатка та об’єктно-орієнтовану модель системи.

Також було спроєктовано базу даних, яка забезпечує ефективне зберігання та обробку даних про курси, їх матеріали, практичні роботи, тести та оцінки.

Було описано алгоритм завантаження виконаних завдань та проходження тестів. Під час проєктування та реалізації алгоритмів роботи системи було детально описано процес авторизації та перевірки дозволів, що дозволяє забезпечити надійний контроль доступу до різних функцій платформи залежно від ролі користувача.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБОРІЄНТОВАНОЮ СИСТЕМОЮ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1. Порядок встановлення та налаштування параметрів системи

Онлайн платформа дистанційного навчання складається з шести основних вузлів (рис. 3.1). Для відображення компонентів системи була створена діаграма розгортання, на якій зазначені технології та їх версії, що повинні бути встановлені та налаштовані на робочій машині.

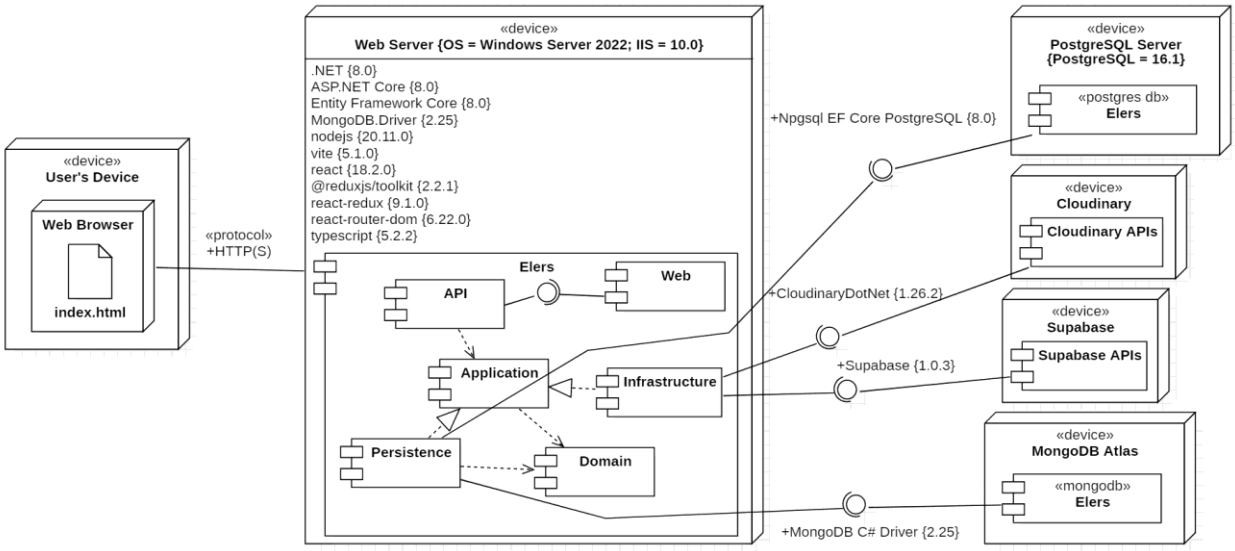

Рисунок 3.1 – Діаграма розгортання

Для розгортання та функціонування онлайн платформи дистанційного навчання використовуються такі вузли:

1. User’s Device – будь-який пристрій з якого користувач взаємодіє з системою (наприклад, комп’ютер, смартфон тощо).

2. Web Server – сервер, на якому розміщується розроблений програмний код системи.

3. PostgreSQL Server – сервер, на якому розміщена БД PostgreSQL.

4. MongoDB Atlas – хмарний сервіс MongoDB для розміщення та управління базами даних MongoDB.

5. Cloudinary – хмарне сховище для зберігання зображень.

6. Supabase – хмарне сховище для зберігання файлів різних типів.

Перед тим як запускати серверний додаток необхідно підготувати конфігураційний файл appsettings.json (табл. 3.1).

Таблиця 3.1

Опис властивостей конфігураційного файлу appsettings.json

Властивість			**Опис**
DatabaseSettings	ApplicationDb	ConnectionString	Рядок з'єднання з БД PostgreSQL
	MongoDb	ConnectionString	Рядок з'єднання з MongoDB
		DatabaseName	Назва бази даних MongoDB
Jwt	Issuer		Ідентифікатор видавця JWT-токенів
	Audience		Аудиторія JWT-токенів
	AccessTokenExpirationInMinutes		Термін дії токенів доступу в хвилинах
	RefreshTokenExpirationInDays		Термін дії токенів оновлення в дня
	SecretKey		Секретний ключ для підпису та перевірки JWT-токенів
Supabase	Url		URL-адреса Supabase
	Key		Ключ для доступу до Supabase
	BucketName		Назва Supabase Storage-бакету
Cloudinary	CloudName		Назва хмарного сховища Cloudinary
	ApiKey		Ключ API Cloudinary
	ApiSecret		Секретний ключ Cloudinary
FileSettings	SizeLimit		Максимальний розмір файлу в байтах
	ImageSizeLimit		Максимальний розмір зображення в байтах
AppVariables	MaxManualGrade		Максимальне значення оцінки, яку можуть додати вручну до таблиці
	MaxAssignmentGrade		Максимальна оцінка за виконання практичної роботи
	MaxFilesStudentUploadAssignment		Максимальна кількість файлів для завантаження студентом до практичної роботи

Для налаштування клієнтського додатку у файлі .env достатньо вказати URL-адресу серверного API у змінній середовища VITE_REACT_APP_API_URL.

3.2. Структура інтерфейсу вебдодатку та порядок роботи

Користувач, відкривши вебдодаток, потрапляє на головну сторінку, де йому одразу ж відображається список доступних курсів на платформі (рис. 3.2). Він може перейти на сторінку конкретного курсу або скористатися пошуком курсів.

Після заголовку та опису курсу відображається кількість: учасників курсу, доступних матеріалів, практичних робіт та тестів.

Крім того, на сайті передбачена можливість вибору темної або світлої теми оформлення, при цьому за замовчуванням обирається та тема, яка встановлена в системі користувача. Також передбачена підтримка української та англійської мов інтерфейсу, за замовчуванням обирається українська мова.

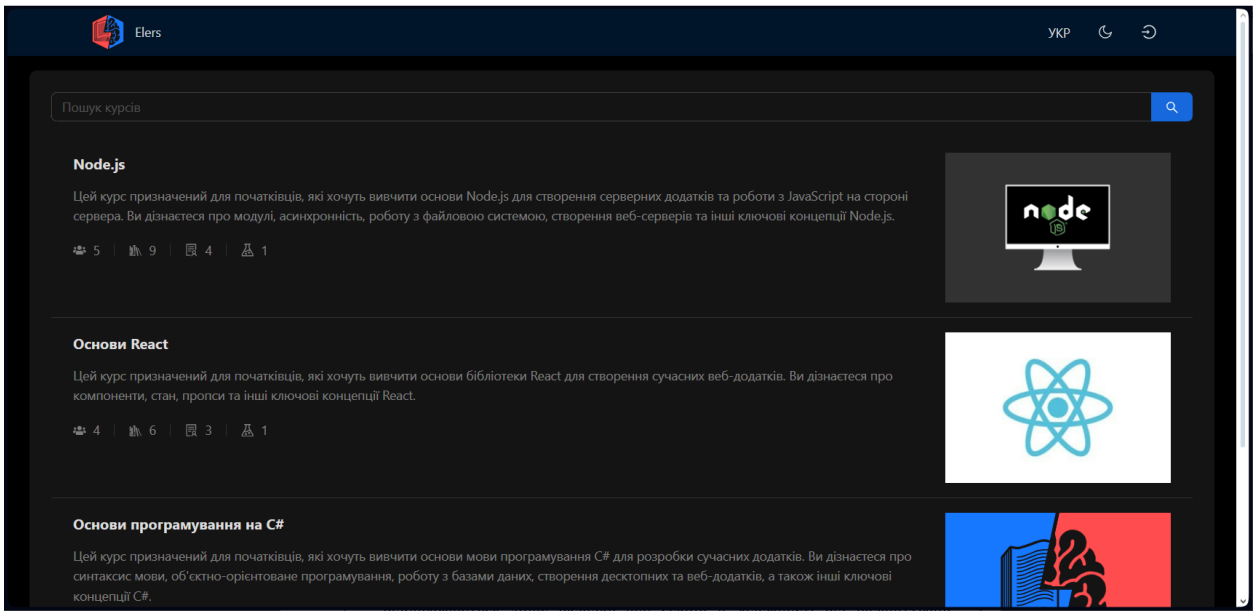

Рисунок 3.2 – Головна сторінка

Для неавторизованого користувача на сторінці курсу відображаються лише вкладки (рис. 3.3) або секції, в залежності від налаштувань курсу, та активні матеріали курсу (рис. 3.4). Також, неавторизовані користувачі можуть переглядати сторінки практичних завдань та тестів.

Кожен тип матеріалу курсу має свій унікальний значок для полегшення навігації. Крім того, над вкладками курсу може відображатися кількість матеріалів, розміщених на цій вкладці, але це налаштування можуть змінювати лише користувачі з відповідними дозволами.

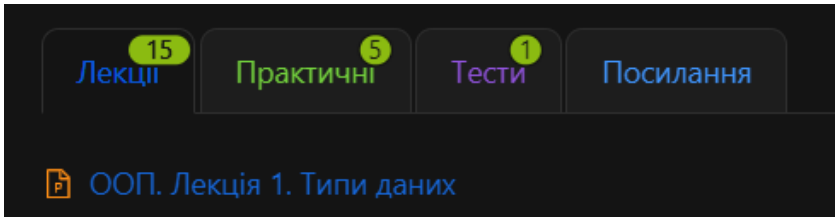

Рисунок 3.3 – Представлення курсу вкладками

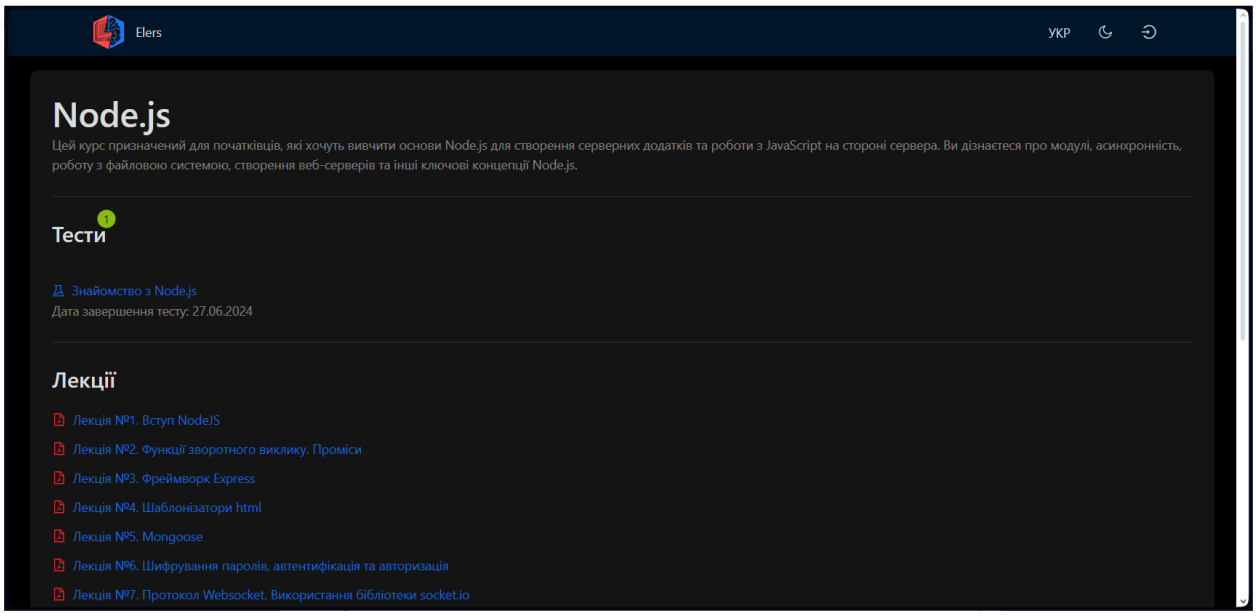

Рисунок 3.4 – Сторінка курсу для неавторизованого користувача, представлення – секції

Натиснувши на кнопку входу, у верхньому правому кутку панелі навігації, користувач перейде на сторінку для авторизації, де він може ввести електронну пошту та пароль для входу в систему.

Можливість зареєструватися на сайті відсутня, але нових користувачів можуть створювати інші користувачі, які мають відповідний дозвіл. Після авторизації як адміністратор, зліва з'являється бічна панель, яка відкриває доступ до інших сторінок платформи (рис. 3.5).

На сторінці з таблицею користувачів можна виконати такі дії як: додати нового користувача, редагувати дані наявного користувача або видалити користувача. При редагуванні або створенні нового користувача відкривається нова сторінка з формою для заповнення даних. При видаленні користувача з'являється модальне вікно (рис. 3.6) для підтвердження виконання дії.

Таблиця користувачів має розширені можливості для фільтрації та сортування даних. У кожній колонці таблиці присутні елементи керування, що дозволяють застосовувати фільтри для відображення лише певних записів за заданими критеріями. Крім того, користувач може сортувати записи за

зростанням або спаданням значень у будь-якій колонці, крім «Роль» та «Тип», клікаючи на заголовок відповідної колонки.

Сторінка «Ролі» має подібний інтерфейс та можливості.

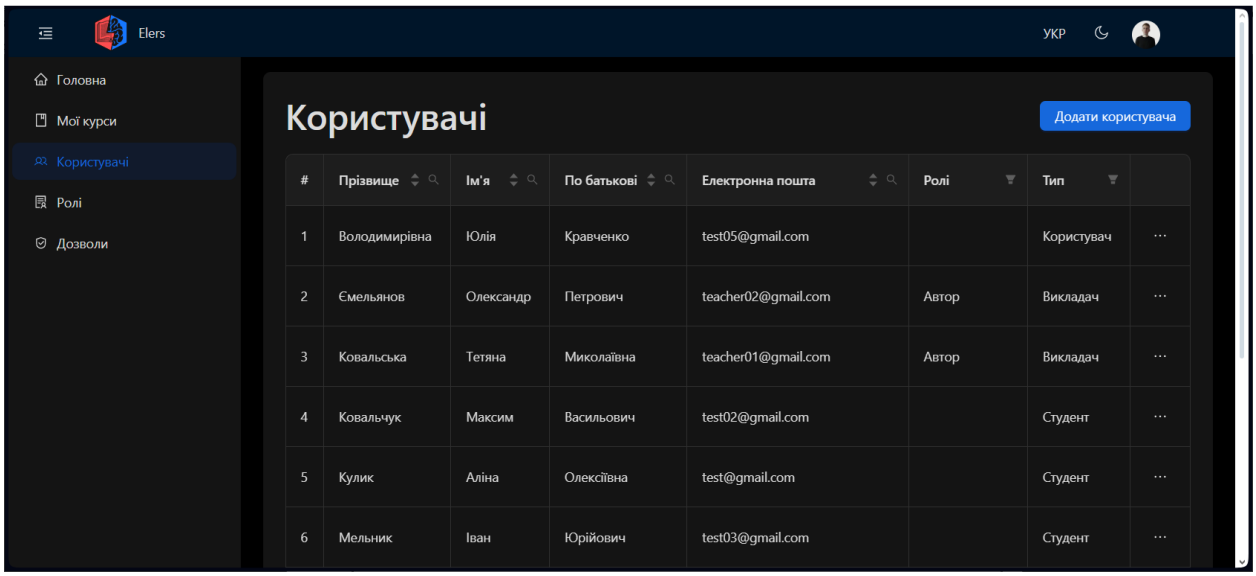

Рисунок 3.5 – Сторінка з користувачами

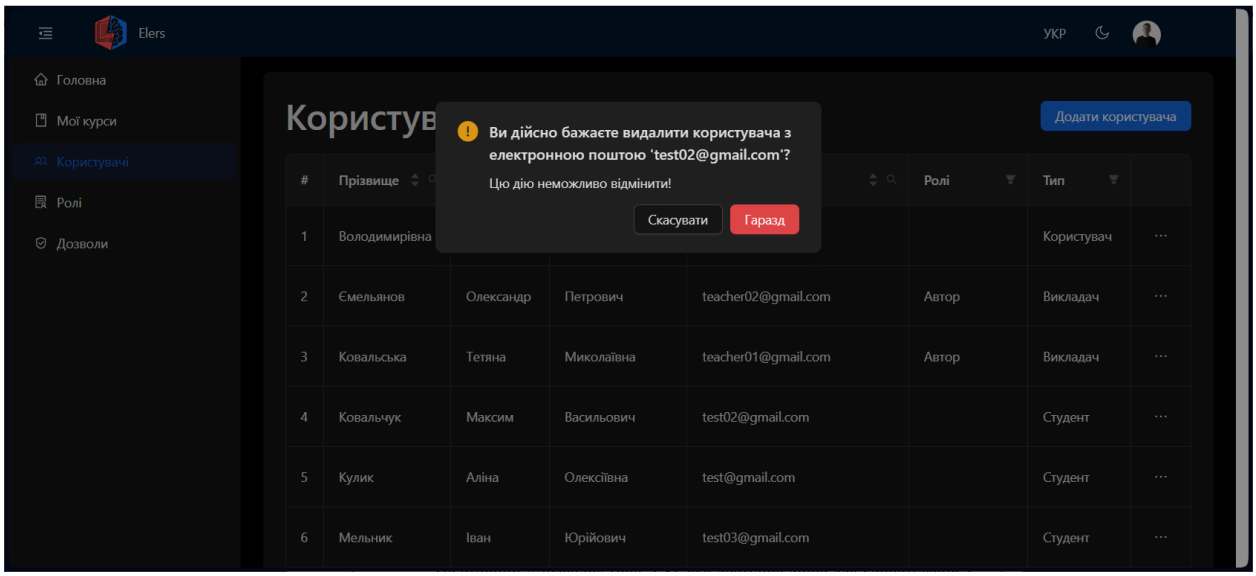

Рисунок 3.6 – Модальне вікно для підтвердження видалення користувача

На сторінці «Дозволи» (рис. 3.7), яка доступна лише для користувачів з відповідними правами доступу, відображається список усіх наявних у системі дозволів. Дозволи поділяються на загальні для всіх користувачів та спеціальні для учасників курсу.

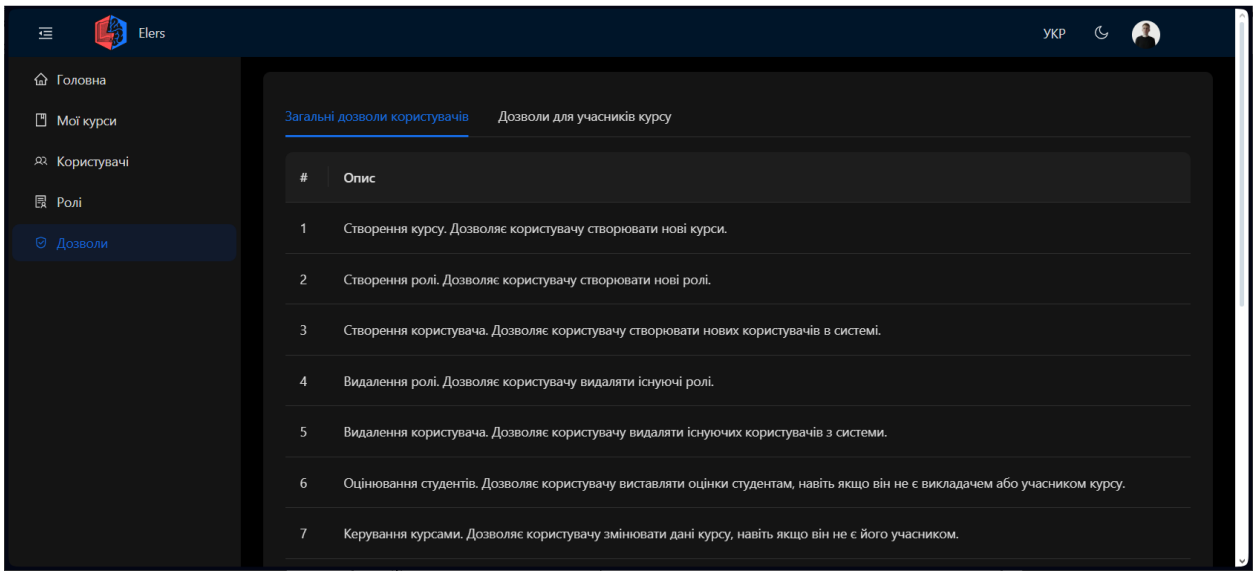

Рисунок 3.7 – Дозволи

Створення нових курсів відбувається на сторінці «Мої курси» (рис. 3.8). На цій сторінці розташована кнопка «Створити курс», яка доступна лише для користувачів, які мають відповідний дозвіл на створення курсів. Також на цій сторінці відображається список усіх курсів, на які записаний користувач або які він створив.

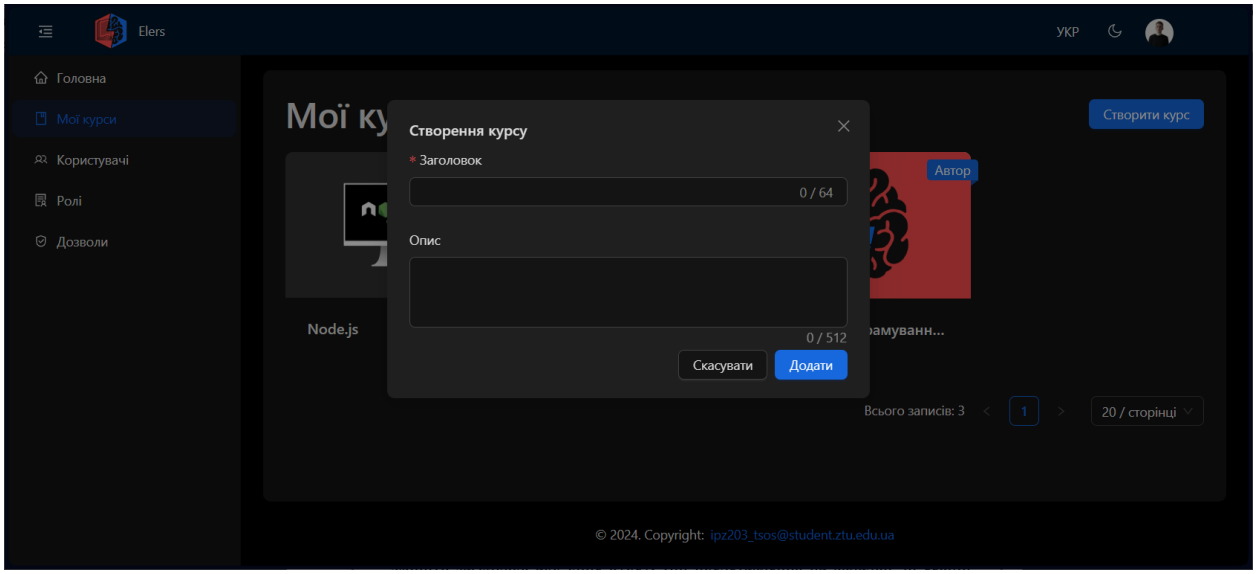

Рисунок 3.8 – Модальне вікно для створення нового курсу

У режимі редагування курсу (рис. 3.9) користувач має можливість змінити заголовок або опис курсу, тип представлення на вкладки чи секції. Також можливо змінити порядок вкладок або матеріалів курсу, додати новий матеріал, такий як контент, посилання, файл, завдання чи тест.

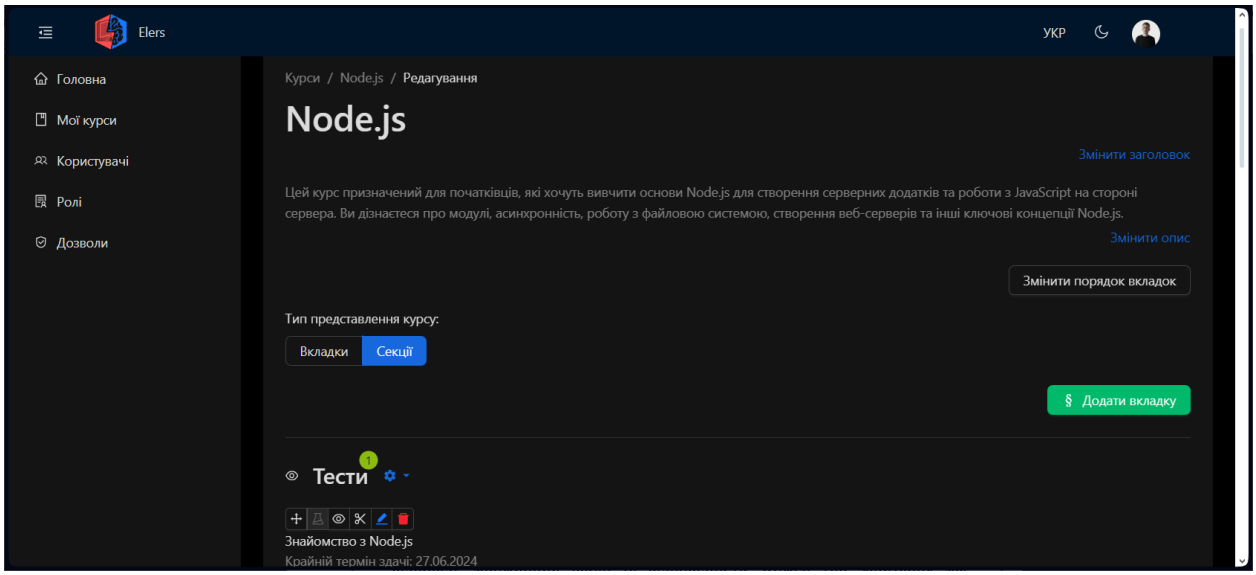

Рисунок 3.9 – Сторінка курсу для адміністратора

Запитання до тесту можна додати лише після створення тесту. В системі підтримується такі типи запитань, як коротка відповідь, вибір однієї відповіді, множинний вибір та відповідність. Кожен тип запитання має подібну форму з текстовим редактором CKEditor, який дозволяє форматувати текст, додавати зображення та таблиці (рис. 3.10).

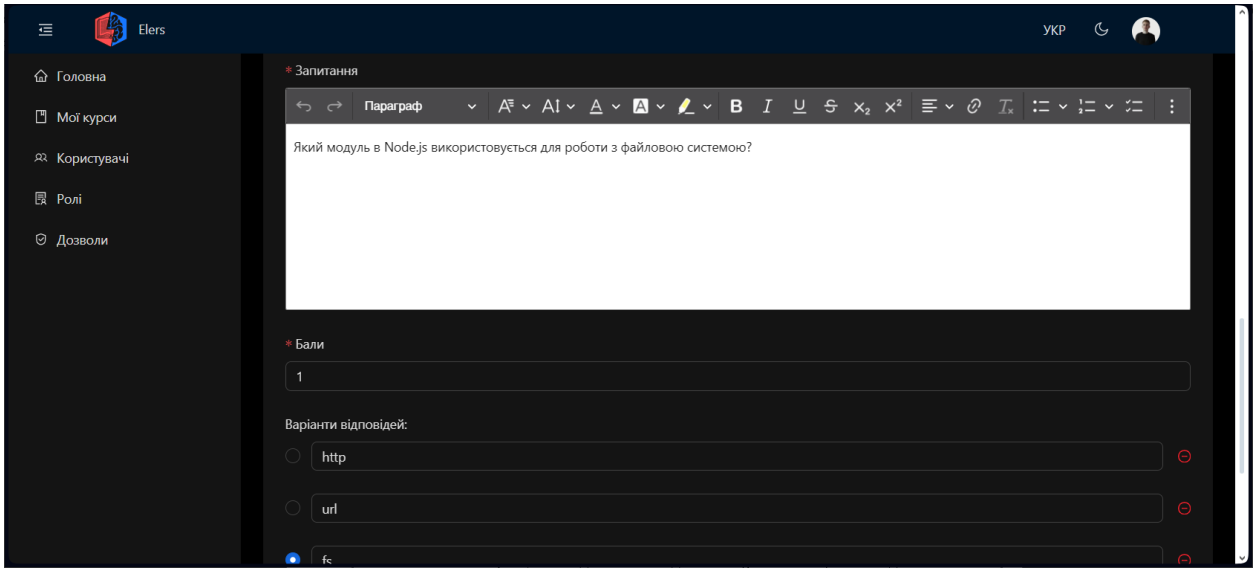

Рисунок 3.10 – Форма створення запитання до тесту

На сторінці «Надіслані завдання» (рис. 3.11) викладач може переглянути всі надіслані завдання студентами, які мають статус «Неперевірені», «Доопрацювання» або «Оцінені». Додатково, викладач може вибрати конкретне завдання та конкретного студента для детальнішого перегляду.

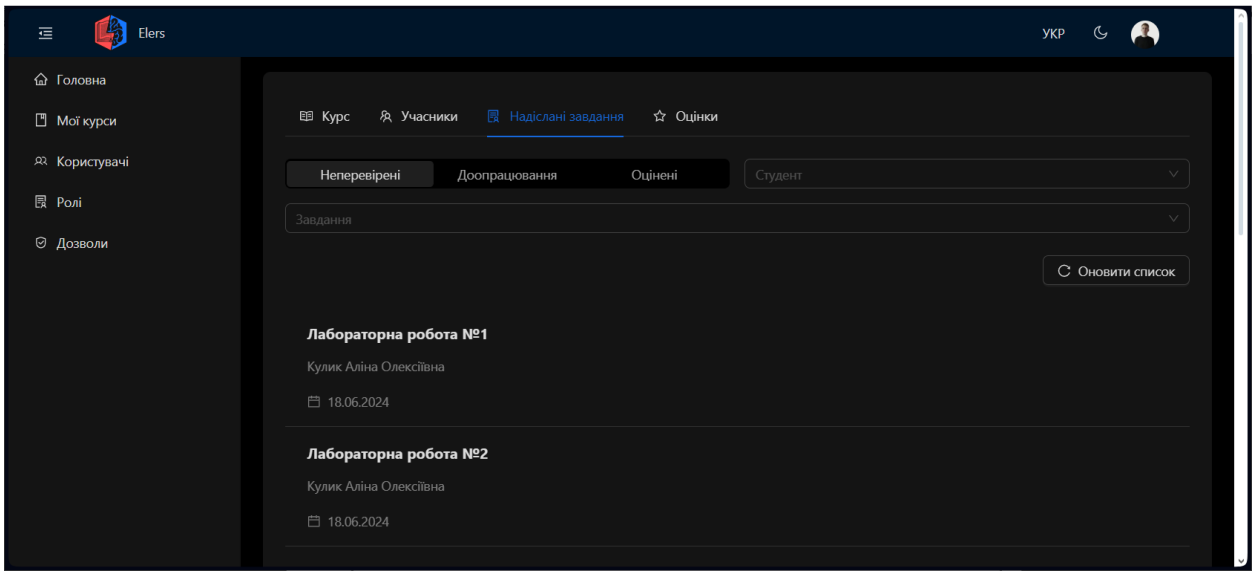

Рисунок 3.11 – Надіслані завдання

Таблиця з оцінками (рис. 3.12) містить усіх студентів курсу, відсортованих за прізвищем, та їх оцінки за практичні завдання, тести та додаткові оцінки, що вводяться вручну в додаткові колонки. Кожен тип колонки має свій унікальний значок. Комірки, які можна редагувати, клікнувши по ним, підсвічуються синім кольором.

Під час наведення курсору на заголовок колонки з’являється спливаюче вікно, в якому повна назва колонки відображається горизонтально, забезпечуючи кращу читабельність та зручність перегляду.

Якщо клікнути на заголовок колонки, який підсвічений синім кольором, відкриється модальне вікно з формою для зміни заголовка, максимальної оцінки, яку можна отримати в цій колонці та дату стовпця, за якою сортуються колонки. Також у цьому модальному вікні буде кнопка для видалення колонки разом з оцінками, які в ній розміщуються.

Оцінки можна змінювати в доданих колонках, а також є можливість змінити оцінку за раніше оцінену практичну роботу. Якщо робота раніше не перевірялася та не оцінювалась, то в таблиці неможливо виставити за неї оцінку. Також неможливо змінити чи виставити оцінку за тест, адже її виставляє система.

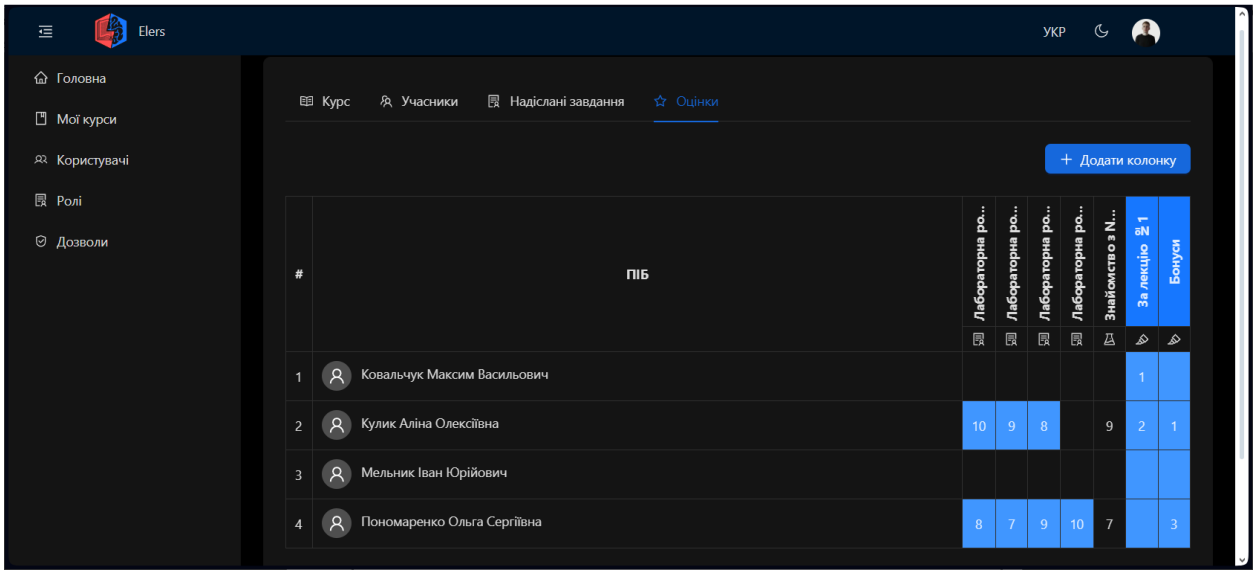

Рисунок 3.12 – Оцінки студентів курсу

На курсі додатково можна змінити зображення та додати нові ролі для учасників, які потім можна призначити їм на сторінці «Учасники».

Студенти курсу мають можливість проходити тести та переглядати свої результати за кожною спробою (рис. 3.13). Також вони можуть продовжити почату спробу, якщо вона не була завершена.

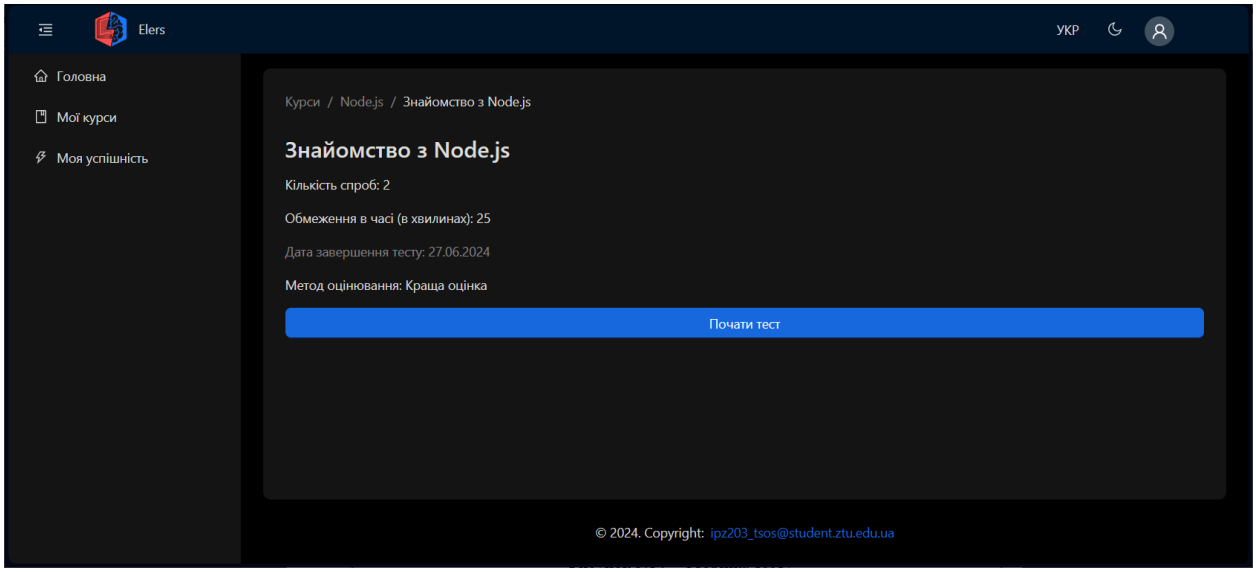

Рисунок 3.13 – Сторінка тесту

Сторінка проходження тесту (рис. 3.14) містить список номерів запитань для навігації, а також таймер у верхньому правому куті, якщо у тесті налаштовано обмеження часу на проходження. Після завершення часу тест автоматично закриється, і студент буде перенаправлений на сторінку

результатів. Після натискання на кнопку «Завершити тест» з’явиться модальне вікно для підтвердження дії.

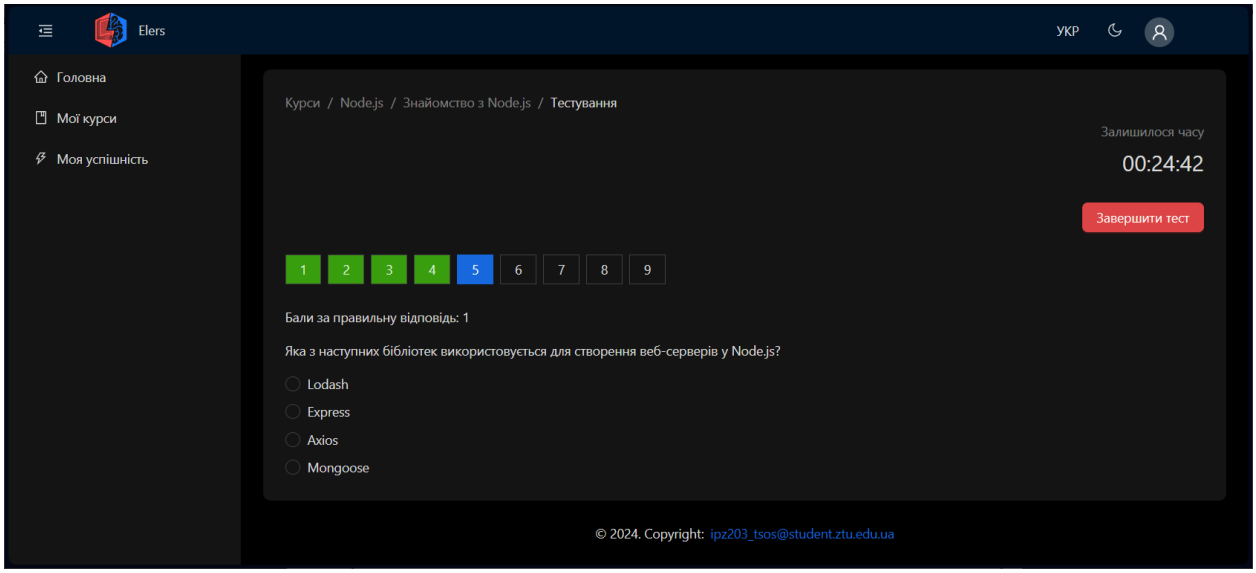

Рисунок 3.14 – Проходження тесту

На сторінці з завданням студенти мають можливість ввести текст та завантажити файли з виконаною роботою (рис. 15). Поки робота не була оцінена чи відправлена на доопрацювання, студент може змінювати свою відповідь без зарахування це як нової спроби.

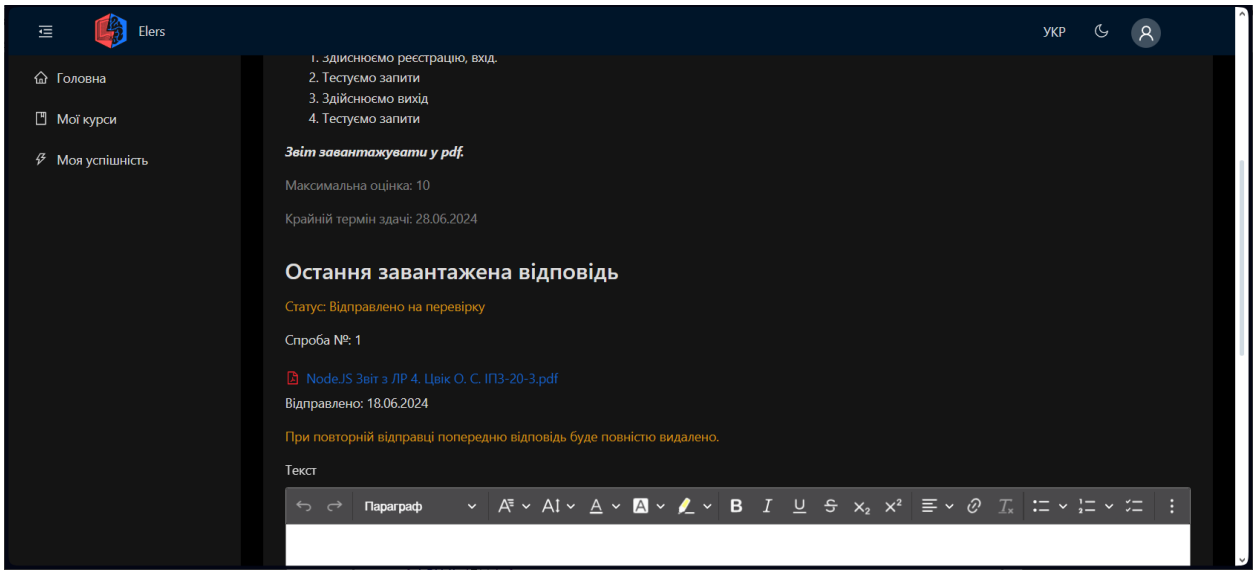

Рисунок 3.15 – Завантаження виконаної практичної роботи

Студенти мають можливість переглянути свою успішність по кожному курсу, на який вони записані (рис. 3.16).

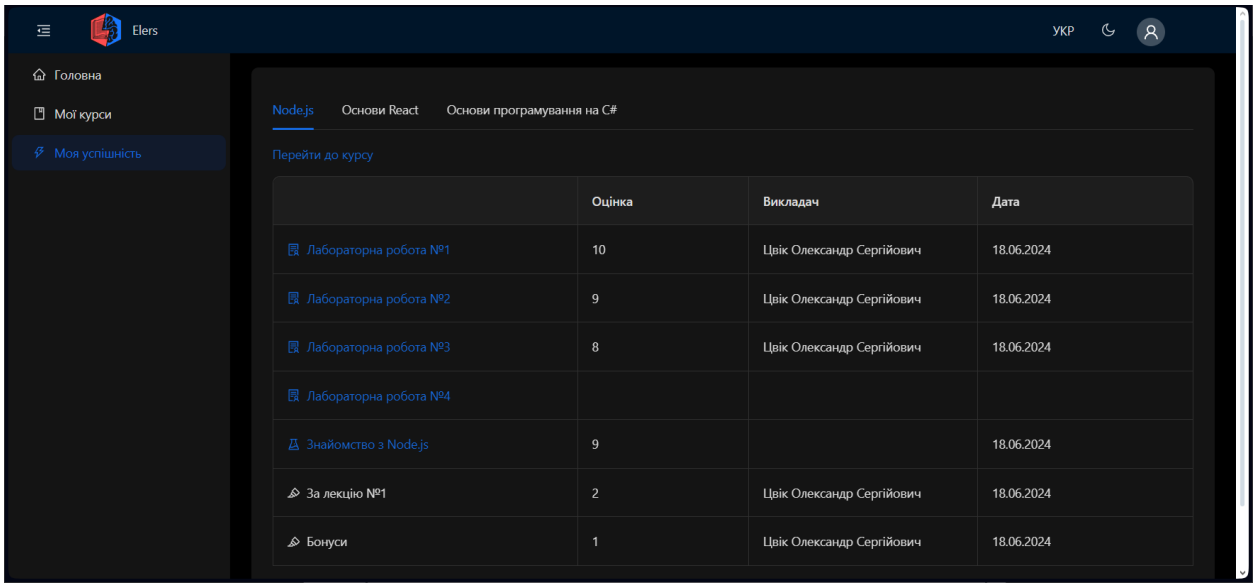

Рисунок 3.16 – Успішність студента

3.3. Тестування роботи та використання вебсервісу

Перевірка коректності роботи програмного забезпечення є важливим етапом у процесі створення будь-якого проєкту. Ця процедура дозволяє виявити наявні недоліки, помилки та дефекти, що сприяє підвищенню загальної якості кінцевого продукту. Лише після ретельного тестування і усунення всіх виявлених проблем розробники можуть бути впевнені, що їхній продукт готовий до випуску та впровадження на ринку. Основними цілями тестування є забезпечення стабільної, безвідмовної роботи додатків, перевірка всього функціоналу на відповідність специфікаціям.

Було проведено ручне тестування з використанням димового тестування, яке охоплює основну функціональність системи. Це тестування здійснюється з метою переконатися, що базові функції програми працюють правильно, без заглиблення в деталі. Воно спрямоване на те, щоб у найкоротші терміни охопити тестами якомога більше функціоналу [28, с. 25].

Під час тестування було застосовано техніку «причина-наслідок». Ця методика передбачає визначення вхідних умов (причин) та очікуваних результатів (наслідків) для перевірки коректної роботи програми відповідно до введених даних. Дану техніку було використано для тестування функціоналу авторизації користувачів, роботи з базою даних (створення,

редагування та видалення записів), а також для перевірки логіки роботи з файлами, зокрема завантаження та видалення файлів.

Крім того, під час тестування використовувалася техніка «аналіз граничних значень» для перевірки даних, які вводяться користувачами. Цей метод застосовувався для тестування функцій валідації вхідних даних, таких як перевірка довжини пароля, формату електронної пошти та діапазонів числових значень у полях введення. Наприклад, при виставленні оцінки неможливо вказати значення, яке перевищує максимальну оцінку, зазначену в налаштуваннях.

Також було створено чекліст для тестування онлайн платформи дистанційного навчання (табл. 3.2).

Таблиця 3.2

Чекліст для тестування платформи дистанційного навчання

№	Функціонал	Результат
	Користувачі та ролі	
1	Додавання користувачів (студенти, викладачі, адміністратори).	Пройдено
2	Налаштування ролей користувачів адміністратором.	Пройдено
3	Контроль доступу до функціоналу платформи залежно від наявних дозволів у користувача.	Пройдено
	Курси	
4	Створення нового курсу викладачем.	Пройдено
5	Редагування інформації про курс викладачем.	Пройдено
6	Видалення курсу викладачем.	Пройдено
7	Додавання навчальних матеріалів викладачем до курсу.	Пройдено
8	Редагування навчальних матеріалів викладачем.	Пройдено
9	Видалення навчальних матеріалів викладачем.	Пройдено
	Практичні роботи	
10	Створення практичної роботи викладачем.	Пройдено
11	Редагування практичної роботи викладачем.	Пройдено
12	Видалення практичної роботи викладачем.	Пройдено
13	Завантаження виконаної практичної роботи студентом.	Пройдено
14	Перегляд та оцінка виконаної практичної роботи викладачем.	Пройдено
	Тестування	
15	Створення тесту викладачем.	Пройдено
16	Редагування тесту викладачем та керування запитаннями.	Пройдено
17	Видалення тесту викладачем.	Пройдено
18	Проходження тесту студентом.	Пройдено
19	Перегляд результатів тесту студентом.	Пройдено
	Рейтинг успішності студента	
20	Перегляд рейтингу успішності студентом.	Пройдено

Висновки до третього розділу

Під час налаштування параметрів системи дистанційного навчання важливо дотримуватися вимог до встановлення визначених технологій та їх версій для забезпечення стабільної роботи програми. Необхідно встановити та налаштувати серверне та клієнтське програмне забезпечення, а також налагодити з’єднання з базами даних та сервісами Supabase і Cloudinary.

Розглянуто роботу з інтерфейсом вебдодатку, включаючи його основні елементи та функціонал. Система включає головну сторінку з переліком курсів, сторінки курсів з вкладками або секціями та матеріалами, сторінки тестів та завдань, а також сторінки з користувачами, ролями, дозволами та оцінками.

Протягом тестування роботи вебсервісу дистанційного навчання було використано різні методи, зокрема димове тестування, техніки «причина-наслідок» та «аналіз граничних значень». Це дозволило виявити і виправити помилки та дефекти, що сприяло поліпшенню якості продукту.

ВИСНОВКИ

У ході виконання даної роботи було розроблено веборієнтовану систему дистанційного навчання з додатковими модулями для перевірки знань студентів.

На початковому етапі було ретельно проаналізовано вимоги до програмного продукту та предметну область, а також досліджено існуючі аналоги систем дистанційного навчання. Це дозволило визначити ключові функціональні можливості майбутньої системи та сформувати відповідні функціональні й нефункціональні вимоги. Для реалізації системи було обрано монолітну архітектуру з використанням SPA для клієнтської частини та чистої архітектури на сервері, що забезпечує високу продуктивність, гнучкість та можливість подальшого розширення.

Під час проєктування було визначено основні варіанти використання системи залежно від ролі користувача, розроблено функціональну карту вебдодатку та об’єктно-орієнтовану модель системи. Також було спроєктовано базу даних для ефективного зберігання та обробки даних про курси, матеріали, практичні роботи, тести та оцінки. Детально описано алгоритми роботи системи, включаючи процеси авторизації, перевірки дозволів, завантаження завдань та проходження тестів.

Під час налаштування системи особливу увагу було приділено встановленню та налаштуванню необхідного програмного забезпечення, а також інтеграції з базами даних та сервісами Supabase і Cloudinary. Розглянуто інтерфейс вебдодатку, його основні елементи та функціонал.

Таким чином, розроблена веборієнтована система дистанційного навчання відповідає усім поставленим вимогам та забезпечує зручний і функціональний інструмент для організації процесу електронного навчання з можливостями перевірки знань студентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Core — FluentValidation documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.fluentvalidation.net/en/latest/aspnet.html.

2. Classdojo [Електронний ресурс] – Режим доступу до ресурсу: https://www.classdojo.com/uk-ua/.

3. Clean Architecture in Android – Crafting Testable, Adaptable, and Robust Apps [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://copperchips.com/clean-architecture-in-android-crafting-testable-adaptable-and-robust-apps/.

4. Code First to a New Database [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/ef/ef6/modeling/code-first/workflows/new-database.

5. CQRS pattern [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/azure/architecture/patterns/cqrs.

6. Custom Authorization Policy Providers using IAuthorizationPolicyProvider in ASP.NET Core [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/aspnet/core/security/authorization/iauthorizationpolicyprovider?view=aspnetcore-8.0.

7. Design Patterns: Elements of Reusable Object-Oriented Software / G.Erich, H. Richard, J. Ralph, V. John., 1994. – 416 с.

8. Entity Framework Core [Електронний ресурс] // 2021 – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/ef/core/.

9. ERD Tool [Електронний ресурс] – Режим доступу до ресурсу: https://www.pgadmin.org/docs/pgadmin4/8.6/erd_tool.html.

10. Getting Started with Redux Toolkit [Електронний ресурс] – Режим доступу до ресурсу: https://redux-toolkit.js.org/introduction/getting-started.

11. Google Classroom [Електронний ресурс] – Режим доступу до ресурсу: https://classroom.google.com/.

12. GUIDs [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/drivers/csharp/current/fundamentals/serialization/guid-serialization/.

13. Implement the microservice application layer using the Web API [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/microservice-application-layer-implementation-web-api.

14. Implementing the Repository and Unit of Work Patterns [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application.

15. Inheritance [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/ef/core/modeling/inheritance.

16. Kyle B. MongoDB in Action / Banker Kyle., 2011. – 312 с.

17. Lock A. ASP.NET Core in Action, Second Edition / Andrew Lock., 2021. – 832 с.

18. MongoDB C# Driver [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/drivers/csharp/v2.25/.

19. Moodle [Електронний ресурс] – Режим доступу до ресурсу: https://moodle.org/.

20. Policy-based authorization in ASP.NET Core [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/aspnet/core/security/authorization/policies?view=aspnetcore-8.0.

21. Polymorphic Objects [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/drivers/csharp/upcoming/fundamentals/serialization/polymorphic-objects/.

22. PostgreSQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/.

23. React Router [Електронний ресурс] – Режим доступу до ресурсу: https://reactrouter.com/en/main/start/overview.

24. Visualization of existing MongoDB databases [Електронний ресурс] – Режим доступу до ресурсу: https://www.datensen.com/blog/docs/mongodb-connection-reverse-engineering/.

25. Боков О. Г. Аналіз архітектури сучасних ВЕБ-додатків [Електронний ресурс] / О. Г. Боков – Режим доступу до ресурсу: https://openarchive.nure.ua/handle/document/21107.

26. Гайна Г. А. Основи проектування баз даних / Георгій Анатолійович Гайна. – Київ: Кондор, 2018. – 204 с.

27. Глабець І. Використання архітектурного шаблону MVC при проектуванні програмного забезпечення [Електронний ресурс] / І. Глабець. – 2013. – Режим доступу до ресурсу: https://api.core.ac.uk/oai/oai:elartu.tntu.edu.ua:123456789/9470.

28. Дідковська М. В. Тестування: Основні визначення, аксіоми та принципи. Текст лекцій. Частина І / М. В. Дідковська, Ю. О. Тимошенко. – МОН України: ННК НТУУ «КПІ». Кафедра математичних методів системного аналізу, 2010. – 62 с.

29. Дудзяний І. М. Об’єктно-орієнтоване моделювання програмних систем / Ігор Михайлович Дудзяний. – Львів: ЛНУ ім. Івана Франка, 2007. – 107 с.

30. Єфремов М. Ф. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ UML [Електронний ресурс] / М. Ф. Єфремов, Ю. М. Єфремов, В. М. Єфремов. – 2016. – Режим доступу до ресурсу: http://eztuir.ztu.edu.ua/123456789/3296.

31. Жаврук Н. В. ВИКОРИСТАННЯ ТЕХНОЛОГІЇ REST ДЛЯ ПОБУДОВИ ВЕБСЕРВІСІВ [Електронний ресурс] / Н. В. Жаврук – Режим доступу до ресурсу: http://eprints.zu.edu.ua/21125/1/ZhavrukAPSI2016.pdf.

32. КОВТУН Л. О. ВИБІР АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ НАВЧАЛЬНОЇ СИСТЕМИ

[Електронний ресурс] / Л. О. КОВТУН, Р. ФРАНЧУК, В. М. ТКАЧУК // Вісник Хмельницького національного університету. – 2019. – Режим доступу до ресурсу: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/01/42-6.pdf.

33. Мартін Р. Чиста архітектура: Мистецтво створення програмного забезпечення / Роберт Мартін. – Харків: Фабула, 2019. – 368 с. – (Видання друге).

34. Платформи та сервіси дистанційного навчання [Електронний ресурс] – Режим доступу до ресурсу: https://www.krok.edu.ua/ua/pro-krok/pidrozdili/navchalni/tsentr-distantsijnogo-navchannya/platformi-ta-servisi-distantsijnogo-navchannya.

35. Плоха О. Б. ОСНОВНІ ПЕРЕВАГИ ТА НЕДОЛІКИ ОДНОСТОРІНКОВИХ ТА БАГАТОСТОРІНКОВИХ ВЕБ-ДОДАТКІВ. Тези доповідей, 8. [Електронний ресурс] / О. Б. Плоха, А. Ю. Верещака. – 2020. – Режим доступу до ресурсу: https://it.hneu.edu.ua/wp-content/uploads/2021/10/tezy-dopovidej-mizhnarodnoyi-naukovo-praktychnoyi-konferencziyi-informaczijni-tehnologiyi-ta-systemy-2020.pdf#page=8.

36. Савельєв Ю. Б. Аналіз порівняльний [Електронний ресурс] / Ю. Б. Савельєв, В. В. Чепак // Велика українська енциклопедія. – 2019. – Режим доступу до ресурсу: https://vue.gov.ua/Аналіз_порівняльний.

37. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т. І. Каплієнко, О. А. Петрова. – Запоріжжя: Дике Поле, 2016. – 250 с.

38. Ткаченко Л. В. ОСОБЛИВОСТІ ВПРОВАДЖЕННЯ ДИСТАНЦІЙНОГО НАВЧАННЯ В ОСВІТНІЙ ПРОЦЕС ЗАКЛАДУ ВИЩОЇ ОСВІТИ [Електронний ресурс] / Л. В. Ткаченко, О. С. Хмельницька. – 2021. – Режим доступу до ресурсу: http://pedagogy-journal.kpu.zp.ua/archive/2021/75/part_3/20.pdf.

39. Швець О. Занурення в ПАТЕРНИ ПРОЕКТУВАННЯ / Олександр Швець., 2022. – 396 с.

40. Що таке Moodle [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://moodle.org/mod/page/view.php?id=8174.

ДОДАТКИ

Додаток А

Лістинг репозиторію для PostgreSQL:

```
using Domain.Primitives;
using Microsoft.EntityFrameworkCore;

namespace Persistence.Repositories;

internal abstract class ApplicationDbRepository<TEntity>
    where TEntity : Entity
{
    protected readonly ApplicationDbContext DbContext;
    protected ApplicationDbRepository(ApplicationDbContext dbContext)
    {
        DbContext = dbContext;
    }

    public virtual Task<TEntity?> GetByIdAsync(Guid id,
CancellationToken cancellationToken = default)
    {
        return DbContext.Set<TEntity>().FirstOrDefaultAsync(x => x.Id
== id, cancellationToken);
    }

    public void Add(TEntity entity)
    {
        DbContext.Set<TEntity>().Add(entity);
    }

    public void Update(TEntity entity)
    {
        DbContext.Set<TEntity>().Update(entity);
    }

    public void Remove(TEntity entity)
    {
        DbContext.Set<TEntity>().Remove(entity);
    }

    public void RemoveRange(IEnumerable<TEntity> entities)
    {
        DbContext.Set<TEntity>().RemoveRange(entities);
    }

    public Task<bool> ExistsByIdAsync(Guid id, CancellationToken
cancellationToken = default)
    {
        return DbContext.Set<TEntity>().AnyAsync(x => x.Id == id,
cancellationToken);
    }
}
```

Лістинг репозиторію для MongoDB:

```
using Domain.Primitives;
using MongoDB.Driver;
using MongoDB.Driver.Linq;

namespace Persistence.Repositories;

internal abstract class MongoDbRepository<TDocument>
    where TDocument : Entity
{
    protected readonly IMongoCollection<TDocument> Collection;

    protected MongoDbRepository(IMongoDatabase mongoDatabase, string
collectionName)
    {
        Collection =
mongoDatabase.GetCollection<TDocument>(collectionName);
    }

    public virtual async Task<TDocument?> GetByIdAsync(Guid id,
CancellationToken cancellationToken = default)
    {
        return await Collection.Find(x => x.Id ==
id).FirstOrDefaultAsync(cancellationToken);
    }

    public async Task<TEntity?> GetByIdAsync<TEntity>(
        Guid id,
        CancellationToken cancellationToken = default)
        where TEntity : TDocument
    {
        return await Collection
            .OfType<TEntity>()
            .Find(x => x.Id == id)
            .FirstOrDefaultAsync(cancellationToken);
    }

    public async Task AddAsync(TDocument document, CancellationToken
cancellationToken = default)
    {
        await Collection.InsertOneAsync(document, null,
cancellationToken);
    }

    public async Task RemoveAsync(Guid id, CancellationToken
cancellationToken = default)
    {
        await Collection.DeleteOneAsync(x => x.Id == id,
cancellationToken);
    }
```

```
    public async Task RemoveAsync(TDocument document,
CancellationToken cancellationToken = default)
    {
        await Collection.DeleteOneAsync(x => x.Id == document.Id,
cancellationToken);
    }

    public async Task RemoveRangeAsync(
        IEnumerable<TDocument> documents,
        CancellationToken cancellationToken = default)
    {
        await Collection.DeleteManyAsync(
            x => documents.Select(document =>
document.Id).Contains(x.Id),
            cancellationToken);
    }

    public Task<bool> ExistsByIdAsync(Guid id, CancellationToken
cancellationToken = default)
    {
        return Collection.Find(x => x.Id ==
id).AnyAsync(cancellationToken);
    }
}
```

Лістинг атрибуту авторизації для користувачів:

```
using Domain.Enums;
using Microsoft.AspNetCore.Authorization;

namespace Infrastructure.Authentication;

public class HasPermissionAttribute : AuthorizeAttribute
{
    public const string PolicyPrefix = "PERMISSION:";

    public HasPermissionAttribute(params PermissionType[] permissions)
    {
        Policy = $"{PolicyPrefix}{string.Join(",", permissions)}";
    }
}
```

Лістинг атрибуту авторизації для учасників курсу:

```
using Domain.Enums;
using Microsoft.AspNetCore.Authorization;

namespace Infrastructure.CourseMemberPermissions;

/// <summary>
/// Attribute for checking the course member's permissions to access
the controller's methods.
/// </summary>
/// <remarks>
/// This attribute should be used on controller methods that have one
of the following parameters
/// in the reference: <c>courseId</c>, <c>tabId</c>,
<c>materialId</c>, <c>roleId</c>,
/// <c>memberId</c>, <c>testQuestionId</c>, <c>gradeId</c> or
<c>columnGradesId</c>.
/// </remarks>
public class HasCourseMemberPermissionAttribute : AuthorizeAttribute
{
    public const string CourseMemberPolicyPrefix =
"COURSE_MEMBER_PERMISSION:";
    public const string UserPolicyPrefix = "USER_PERMISSION:";
    public const string Separator = "---";

    public HasCourseMemberPermissionAttribute(
         CoursePermissionType[] courseMemberPermissions,
         PermissionType[]? userPermissions = null)
    {
        string courseMemberPolicy =
$"{CourseMemberPolicyPrefix}{string.Join(",",
courseMemberPermissions)}";
```

```
        string userPolicy = $"{UserPolicyPrefix}{string.Join(",",
userPermissions ?? [])}";

        Policy = $"{courseMemberPolicy}{Separator}{userPolicy}";
    }
}
```

Лістинг постачальника політик авторизації:

```
using Infrastructure.CourseMemberPermissions;
using Microsoft.AspNetCore.Authorization;
using Microsoft.Extensions.Options;

namespace Infrastructure.Authentication;

public class PermissionAuthorizationPolicyProvider :
DefaultAuthorizationPolicyProvider
{
    public
PermissionAuthorizationPolicyProvider(IOptions<AuthorizationOptions>
options)
        : base(options)
    {
    }

    public override async Task<AuthorizationPolicy?>
GetPolicyAsync(string policyName)
    {
        AuthorizationPolicy? policy = await
base.GetPolicyAsync(policyName);

        bool isPermission =
policyName.StartsWith(HasPermissionAttribute.PolicyPrefix);

        bool isCourseMemberPermission = policyName
            .StartsWith(HasCourseMemberPermissionAttribute.CourseMembe
rPolicyPrefix);

        if (!isPermission && !isCourseMemberPermission)
        {
            return policy;
        }

        if (policy is not null)
        {
            return policy;
        }

        if (isPermission)
        {
            return BuildPermissionPolicy(policyName);
        }
```

```
        else if (isCourseMemberPermission)
        {
            return BuildCourseMemberPermissionPolicy(policyName);
        }

        return null;
    }

    private static AuthorizationPolicy BuildPermissionPolicy(string
policyName)
    {
        string[] permissions = policyName
            .Substring(HasPermissionAttribute.PolicyPrefix.Length)
            .Split(",");

        return new AuthorizationPolicyBuilder()
            .AddRequirements(new PermissionRequirement(permissions))
            .Build();
    }

    private static AuthorizationPolicy
BuildCourseMemberPermissionPolicy(string policyName)
    {
        string[] permissions =
policyName.Split(HasCourseMemberPermissionAttribute.Separator);

        string[] courseMemberPermissions = permissions[0]
            .Substring(HasCourseMemberPermissionAttribute.CourseMember
PolicyPrefix.Length)
            .Split(",");

        string[] userPermissions = permissions[1]
            .Substring(HasCourseMemberPermissionAttribute.UserPolicyPr
efix.Length)
            .Split(",");

        var requirements = new
CourseMemberPermissionRequirement(courseMemberPermissions,
userPermissions);

        return new AuthorizationPolicyBuilder()
            .AddRequirements(requirements)
            .Build();
    }
}
```

Додаток В

Лістинг методу відправки виконаного завдання студентом:

```
using Application.Common.Interfaces;
using Application.Common.Messaging;
using Application.Common.Models;
using Application.Common.Services;
using Domain.Entities;
using Domain.Enums;
using Domain.Errors;
using Domain.Repositories;
using Domain.Shared;

namespace Application.Assignments.SubmitAssignment;

public class SubmitAssignmentCommandHandler :
ICommandHandler<SubmitAssignmentCommand>
{
    private readonly ICourseMaterialRepository
_courseMaterialRepository;
    private readonly IStudentRepository _studentRepository;
    private readonly ISubmittedAssignmentRepository
_submittedAssignmentRepository;
    private readonly ICourseMemberService _courseMemberService;
    private readonly IFileService _fileService;
    private readonly IUserContext _userContext;

    public SubmitAssignmentCommandHandler(
        IStudentRepository studentRepository,
        ISubmittedAssignmentRepository submittedAssignmentRepository,
        ICourseMaterialRepository courseMaterialRepository,
        ICourseMemberService courseMemberService,
        IFileService fileService,
        IUserContext userContext)
    {
        _studentRepository = studentRepository;
        _courseMaterialRepository = courseMaterialRepository;
        _submittedAssignmentRepository =
submittedAssignmentRepository;
        _courseMemberService = courseMemberService;
        _fileService = fileService;
        _userContext = userContext;
    }

    public async Task<Result> Handle(SubmitAssignmentCommand request,
CancellationToken cancellationToken)
    {
        if (string.IsNullOrEmpty(request.Text) && (request.Files is
null || request.Files.Length == 0))
        {
            return AssignmentErrors.EmptyFields();
```

```
        }

        Student? student = await
_studentRepository.GetByIdAsync(_userContext.UserId,
cancellationToken);

        if (student is null)
        {
            return AssignmentErrors.StudentsOnly();
        }

        CourseMaterialAssignment? assignment = await
_courseMaterialRepository
            .GetByIdAsync<CourseMaterialAssignment>(request.Assignment
Id, cancellationToken);

        if (assignment is null)
        {
            return AssignmentErrors.NotFound(request.AssignmentId);
        }

        if (!assignment.IsActive)
        {
            return AssignmentErrors.NotActive();
        }

        SubmittedAssignment? submittedAssignment = await
_submittedAssignmentRepository
            .GetByAssignmentIdAndStudentIdAsync(assignment.Id,
student.Id, cancellationToken);

        if (submittedAssignment?.Status ==
SubmittedAssignmentStatus.Graded)
        {
            return AssignmentErrors.AlreadyGraded();
        }

        if (submittedAssignment?.Status !=
SubmittedAssignmentStatus.Submitted &&
            submittedAssignment?.Status !=
SubmittedAssignmentStatus.Resubmit &&
            assignment.Deadline.HasValue &&
            assignment.Deadline.Value.AddDays(1).Date <
DateTime.UtcNow.Date)
        {
            return AssignmentErrors.DeadlinePassed();
        }

        if (!await
_courseMemberService.IsCourseMemberByCourseTabIdAsync(
            student.Id, assignment.CourseTabId, cancellationToken))
        {
```

```
            return AssignmentErrors.StudentsOnly();
        }

        var files = new List<SubmitAssignmentFile>();

        if (request.Files is not null)
        {
            if (request.Files.Length > assignment.MaxFiles)
            {
                return
AssignmentErrors.ManyFiles(assignment.MaxFiles);
            }

            foreach (IFile file in request.Files)
            {
                Result<FileUploadResult> addFileResult = await
_fileService.AddAsync(file, cancellationToken);

                if (addFileResult.IsFailure || addFileResult.Value is
null)
                {
                    return addFileResult.Error;
                }

                files.Add(new SubmitAssignmentFile
                {
                    FileName = addFileResult.Value.FileName,
                    UniqueFileName =
addFileResult.Value.UniqueFileName
                });
            }
        }

        if (submittedAssignment is null)
        {
            var newSubmittedAssignment = new SubmittedAssignment
            {
                AssignmentId = assignment.Id,
                StudentId = student.Id,
                Status = SubmittedAssignmentStatus.Submitted,
                Text = request.Text,
                Files = files,
                SubmittedAt = DateTime.UtcNow
            };

            await
_submittedAssignmentRepository.AddAsync(newSubmittedAssignment,
cancellationToken);
        }
        else
        {
            if (submittedAssignment.Files.Count != 0)
```

```
            {
                await _fileService.RemoveRangeAsync(
                    submittedAssignment.Files.Select(x =>
x.UniqueFileName), cancellationToken);
            }

            submittedAssignment.Status =
SubmittedAssignmentStatus.Submitted;
            submittedAssignment.Text = request.Text;
            submittedAssignment.Files = files;
            submittedAssignment.SubmittedAt = DateTime.UtcNow;

            await
_submittedAssignmentRepository.UpdateAsync(submittedAssignment,
cancellationToken);
        }

        return Result.Success();
    }
}
```