Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – магістр

за освітньо-професійною програмою

«Комп'ютерні науки»

зі спеціальності

122 – Комп'ютерні науки

тема роботи:

«Розробка web-платформи для онлайн-навчання та спільної роботи»

Виконав студент гр. КН-23м _____ Шрамко І. С.

Керівник _____ Харламенко В. Ю.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Магістр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *05* » *липня* 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи КН-23м *Шрамко Ігорю Сергійовичу*

1. Тема кваліфікаційної роботи: *«Розробка web-платформи для онлайн-навчання та спільної роботи»*

затверджено наказом по університету № *594с* від *04.07.2024 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 80 с., додатки, презентація у Microsoft PowerPoint (12 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 ст. викл. Харламенко В. Ю.

_____

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.07.24*
2	*Розділ 1*	*15.07.24*
3	*Розділ 2*	*15.08.24*
4	*Висновки*	*15.09.24*
5	*Оформлення кваліфікаційної роботи*	*20.11.24*
6	*Підготовка презентації та графічного матеріалу*	*28.11.24*
7	*Підготовка доповіді до захисту*	*01.12.24*

6. Дата видачі завдання: *28.06.2024р.*

Керівник _____ / Хармаленко В. Ю./

7. Запевнення: *Я, Шрамко Ігор Сергійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Шрамко І. С./

АНОТАЦІЯ

Шрамко І. С. Розробка web-платформи для онлайн-навчання та спільної роботи.

Кваліфікаційна робота на здобуття ступеня вищої освіти – магістр, за спеціальністю 122 Комп'ютерні науки. Криворізький національний університет, Кривий Ріг, 2024.

Кваліфікаційна робота на здобуття ступеня магістр, за спеціальністю 122 Комп'ютерні науки. Національний університет, 2024.

Робота складається зі вступу, трьох розділів, висновків та переліку використаної літератури з 26 позицій. Загальний обсяг роботи становить 82 сторінок, з яких основний зміст викладено на 76 сторінках. Включено 1 таблиця та 34 рисунків.

У роботі розглянуто теоретичні та практичні аспекти розробки web-платформи для онлайн-навчання, що підтримує інструменти для спільної роботи та взаємодії користувачів. Досліджено існуючі технології та фреймворки для створення таких платформ, зокрема, використання React, Firebase, TypeScript, та інших інструментів для реалізації функціоналу. Описано етапи проектування структури бази даних, функціональність основних компонентів системи, а також технічні особливості розробки веб-додатків для навчання в онлайн-форматі.

Практична частина роботи включає розробку та тестування основних функціональних елементів платформи, таких як реєстрації користувачів, інтерфейс для створення та перегляду курсів, а також інструменти для спільної роботи.

Ключові слова:

WEB-ПЛАТФОРМА, ОНЛАЙН-НАВЧАННЯ, FIREBASE, DATABASE, CSS, REACT, SCSS,TYPESCRIPT,JSON.

ANNOTATION

Shramko I. S. Development of a web platform for online learning and collaboration.

Qualification work for the degree of higher education - Master, specialty 122 Computer Science. Kryvyi Rih National University, Kryvyi Rih, 2024.

Qualification work for the degree of Master, specialty 122 Computer Science. National University, 2024.

The work consists of an introduction, three sections, conclusions and a list of used literature from 26 items. The total volume of the work is 82 pages, of which the main content is presented on 76 pages. 1 table and 34 figures are included.

The work considers theoretical and practical aspects of developing a web platform for online learning that supports tools for collaboration and user interaction. Existing technologies and frameworks for creating such platforms are studied, in particular, the use of React, Firebase, TypeScript, and other tools for implementing functionality. The stages of designing the database structure, the functionality of the main components of the system, as well as the technical features of developing web applications for online learning are described.

The practical part of the work includes the development and testing of the main functional elements of the platform, such as user registration, an interface for creating and viewing courses, as well as tools for collaboration.

Keywords:

WEB PLATFORM, ONLINE LEARNING, FIREBASE, DATABASE, CSS, REACT, SCSS, TYPESCRIPT, JSON.

ЗМІСТ

ВСТУП

Навчання сьогодні відкриває нові горизонти для освітнього процесу. Хоча традиційні методи залишаються основою систем освіти, вони поступово поступаються новітнім цифровим підходам, що роблять процес навчання доступнішим, більш індивідуалізованим та інтерактивним. Одним із таких підходів є створення онлайн-платформ для навчання, які дозволяють користувачам здобувати знання та розвивати навички в зручний для них час і в будь-якому місці. Швидкий розвиток технологій та необхідність безперервного навчання для фахівців у різних галузях створюють значний попит на ефективні та функціональні web-платформи для дистанційного навчання та спільної роботи.

Впровадження таких платформ дозволяє забезпечити доступ до освітніх ресурсів, підвищити рівень інтерактивності навчального процесу та полегшити співпрацю між учасниками. Вони пропонують користувачам різноманітні інструменти для створення, обміну та обговорення навчальних матеріалів, а також для спільної роботи над проєктами в реальному часі.

Кваліфікаційна робота з розробки web-платформи для онлайн-навчання та спільної роботи має на меті дослідити підходи до реалізації таких систем, обрати оптимальні архітектурні та технологічні рішення, а також розробити функціональні можливості, які максимально відповідають потребам користувачів. Дослідження та впровадження ефективної платформи є актуальним завданням, оскільки такі системи підтримують процес навчання і співпраці у багатьох сферах діяльності, зокрема в освіті, бізнесі та науковій роботі.

Таким чином, ця робота спрямована на створення комплексного рішення, яке поєднає зручність використання, високий рівень інтерактивності та можливість адаптації під конкретні потреби користувачів, забезпечуючи їм ефективний інструмент для навчання. Організація спільної роботи та забезпечення ефективної комунікації між учасниками навчального процесу є

важливими компонентами сучасних освітніх платформ. Вони об'єднують інструменти для обміну знаннями та досвідом, що створює сприятливе середовище для навчання. Це дозволяє здобувачам освіти обговорювати навчальний матеріал, ділитися своїми ідеями та отримувати зворотний зв'язок від викладачів.

Розробка онлайн-платформи для навчання – це складний багатоступеневий процес, який потребує знань із різних напрямків, зокрема програмістів, дизайнерів, освітніх методистів та аналітиків. Важливо ретельно опрацювати архітектуру платформи, її функціональні можливості та зручність використання. До того ж, платформа повинна легко інтегруватися з іншими навчальними інструментами та системами, стабільно працювати на різних пристроях і відповідати високим вимогам безпеки даних.

Один із ключових аспектів – вибір оптимальних методик навчання. Оскільки онлайн-освіта має свої особливості порівняно з традиційними підходами, навчальні програми мають бути адаптовані для максимальної ефективності в цифровому середовищі. Це включає розробку коротких, структурованих уроків, інтерактивних завдань, а також системи оцінювання, яка дозволяє відстежувати успіхи студентів і за потреби коригувати навчальний процес.

Предмет дослідження – можливості та засоби JavaScript для розробки функціоналу web-платформи, призначених для онлайн-навчання та спільної роботи.

Об’єкт дослідження – web-платформа, призначена для організації онлайн-навчання, спільної роботи користувачів, управління навчальним процесом і забезпечення інтерактивної взаємодії між учасниками.

Отже, розробка web-платформи для онлайн-навчання та спільної роботи має значний потенціал для поліпшення організаційних процесів, підвищення ефективності навчання, а також створення інтерактивного середовища для обміну знаннями. Така платформа забезпечить зручний доступ до навчальних матеріалів, спростить комунікацію між учасниками та сприятиме їх залученості.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ МОЖЛИВИХ ПІДХОДІВ ДО ВИРІШЕННЯ ЗАВДАННЯ РОЗРОБКИ WEB-ПЛАТФОРМИ ДЛЯ ОНЛАЙН-НАВЧАННЯ

1.1 Актуальність предметної області

З розвитком суспільства зростає потреба в якісних освітніх послугах, які дозволяють підвищувати кваліфікацію без відриву від основної діяльності. Сьогодні освіта, здобута в молоді роки, вже не є достатньою – постійне вдосконалення знань і навичок стає необхідним протягом усього активного життя. Сучасні комп'ютери та мережі відкривають можливості для створення ефективних навчальних засобів, які базуються на інформаційних базах даних. Основу таких систем складають електронні навчальні матеріали.

Розробка електронних навчальних засобів – це складний і відповідальний процес. Для їхньої ефективності необхідно грамотно поєднувати всі можливості сучасних технологій: гіпертекст, графіку, відео, звук та анімацію.

Більшість існуючих електронних навчальних засобів складається з матеріалів для студентів і школярів, навчальних курсів з використання комп'ютерних програм, а також енциклопедій та довідників. Такі засоби можна умовно поділити на кілька категорій:

- електронні копії друкованих підручників;
- оцифровані відеозаписи лекцій;
- електронні версії довідників, словників та енциклопедій;
- мультимедійні електронні навчальні посібники.

Електронні навчальні засоби – це програмні педагогічні інструменти, створені для надання нової інформації, яка доповнює друковані видання, а також для індивідуального вивчення й обмеженої перевірки знань і вмінь, здобутих студентами.

Електронні навчальні засоби мають важливу перевагу: досвідчений розробник, володіючи знаннями синтаксису та технологій, що

використовуються для їх створення, може у будь-який момент оновити або змінити навчальний матеріал. Це особливо актуально, оскільки стандарти й вимоги до освітніх ресурсів постійно змінюються. Наприклад, може бути необхідність скоротити або розширити обсяг лекційного матеріалу з певних тем.

Метою магістерської роботи є створення навчально-методичного ресурсу для студентів. Це платформа управління навчанням, яка надає доступ до різноманітних матеріалів, домашніх і практичних робіт, спрямованих на підвищення рівня знань студентів.

Електронне навчання можна умовно поділити на два основних сегменти:

1. Компанії-розробники програмного забезпечення.

2. Компанії, що створюють електронні курси та навчальний контент.

Основну частину ринку послуг електронного навчання займають компанії-розробники програмного забезпечення. Їх, у свою чергу, можна класифікувати за напрямками:

– розробка систем управління навчанням (LMS – Learning Management System) та управління навчальним контентом (LCMS – Learning Content Management System);
– інструменти для організації вебінарів і віртуальних класів;
– засоби для створення електронних курсів та навчального контенту (системи розробки курсів).

Цей поділ є умовним, оскільки багато компаній працюють у всіх сегментах e-learning, тоді як інші спеціалізуються лише в одному або декількох напрямах.

LMS являє собою платформу для розгортання електронного навчання. Платформа виконує функції організації навчального процесу, забезпечуючи інтерфейс для взаємодії між викладачами та студентами. Вона надає доступ до навчального порталу, який слугує центральною точкою для отримання навчального контенту, а також містить інструменти для створення навчальних програм, контролю їх виконання та формування звітів про результати навчання.

Дистанційне навчання має низку переваг, серед яких доступність освіти з

будь-якого місця, гнучкість у плануванні часу, економічність і використання інноваційних методів навчання. Воно дозволяє персоналізувати процес навчання, даючи можливість студентам проходити матеріал у власному темпі. Проте існують і недоліки, такі як потреба в самодисципліні, відсутність живого спілкування, технічні труднощі, обмеження практичних занять та мотиваційні бар'єри. Крім того, дистанційне оцінювання знань може стикатися з проблемами чесності й точності. Таким чином, для ефективного впровадження платформ дистанційного навчання необхідно враховувати ці аспекти, мінімізуючи недоліки і підсилюючи переваги.

Ефективне дистанційне навчання вимагає добре продуманої платформи, яка б забезпечувала якісний контент, зручний інтерфейс і інтерактивні елементи для підтримки зацікавленості студентів. Важливим аспектом є організація взаємодії між студентами та викладачами через форуми, відеоконференції чи інші засоби комунікації, що дозволить частково компенсувати відсутність живого спілкування. Також слід передбачити технічну підтримку користувачів і впровадження технологій для боротьби з академічною недоброчесністю під час онлайн-оцінювання. Урахування цих факторів дозволить створити платформу, яка стане ефективним інструментом для онлайн-навчання та спільної роботи, задовольняючи потреби сучасних студентів та викладачів.

Крім того, слід зосередити увагу на можливостях для спільної роботи, які є важливим елементом сучасного навчання. Інтеграція таких інструментів, як спільні дошки, чати, системи управління проєктами та сховища файлів, дозволить студентам працювати над командними завданнями, обмінюватися ідеями та розвивати навички комунікації. Значну роль відіграє використання адаптивних технологій, які можуть підлаштовувати навчальний контент під потреби кожного користувача. Це не лише покращить ефективність навчання, але й зробить платформу привабливішою для широкого кола користувачів. Таким чином, розробка web-платформи з фокусом на онлайн-навчанні та спільній роботі відкриває перспективи створення інноваційного інструменту, що сприятиме розвитку освіти в умовах цифрової трансформації.

Сучасний світ зазнав значних змін у підходах до освіти та організації робочих процесів, що зумовлено швидким розвитком цифрових технологій і глобальним переходом до дистанційного формату навчання та співпраці. Зростання популярності онлайн-освіти є особливо помітним у зв'язку з потребою забезпечення доступу до знань і ресурсів незалежно від географічного розташування учасників, що робить освітній процес гнучкішим та доступнішим. Усе це створює об'єктивні передумови для розробки web-платформи, яка могла б об'єднати функції дистанційного навчання і спільної роботи, забезпечуючи користувачів усіма необхідними інструментами для ефективного засвоєння знань і продуктивної взаємодії. Актуальність розробки такої платформи зумовлена низкою чинників:

1. Потреба в доступності та гнучкості навчання. Сьогодні значна кількість людей обирає дистанційне навчання як зручну альтернативу традиційній освіті. Онлайн-платформа забезпечує можливість доступу до навчальних матеріалів у будь-який час і з будь-якого місця, що особливо важливо для людей, які не можуть відвідувати освітні заклади через географічну віддаленість, робочий графік чи особисті обставини. Така гнучкість дозволяє користувачам отримувати освіту за зручним для них розкладом, що сприяє підвищенню мотивації та якості засвоєння матеріалу.

2. Необхідність індивідуалізації навчального процесу. Сучасна система освіти все більше орієнтується на індивідуальний підхід, який враховує особливості кожного студента. Онлайн-платформа для навчання може пропонувати персоналізовані маршрути навчання, адаптивні тести та інтерактивні завдання, які відповідають рівню підготовки і потребам кожного користувача. Такі інструменти дозволяють підвищити ефективність навчального процесу та досягати кращих результатів за рахунок індивідуального підходу.

3. Забезпечення ефективної комунікації та співпраці. Навчання та робота в умовах командної взаємодії потребують платформи, яка здатна підтримувати багатосторонню комунікацію та надавати можливості для спільної роботи. Web-

платформа з функціями чату, відеоконференцій, обміну документами та інтеграцією з іншими онлайн-інструментами створює середовище для активної комунікації між студентами та викладачами, а також для ефективної командної роботи. Це особливо важливо для виконання групових проєктів, де від якості співпраці залежить успіх виконання завдань.

4. Потреба в інтерактивності та залученості користувачів. Однією з основних переваг онлайн-платформ є можливість інтерактивної взаємодії, яка робить навчання динамічним і цікавим. Модулі для проведення інтерактивних занять, тестування в реальному часі та вбудовані інструменти для зворотного зв'язку дають змогу підтримувати високий рівень залученості студентів, що покращує сприйняття матеріалу та підвищує рівень мотивації до навчання.

5. Безпека даних та конфіденційність. Під час онлайн-навчання та роботи в мережі питання захисту даних і конфіденційності стають особливо актуальними. Розробка web-платформи, яка дотримується сучасних стандартів безпеки, забезпечує збереження особистої інформації користувачів, конфіденційність навчальних матеріалів та захищеність інформації під час комунікації між учасниками. Це особливо важливо для збереження довіри користувачів до платформи та підтримання її репутації.

6. Інтеграція з сучасними технологіями та освітніми інструментами. Актуальність розробки такої платформи також пов'язана з можливістю інтеграції з інноваційними освітніми технологіями, такими як штучний інтелект, автоматичне оцінювання та аналіз прогресу. Це дозволяє створювати більш ефективне середовище для навчання, де кожен учасник може отримувати підтримку на основі своїх успіхів і недоліків.

Таким чином, розробка web-платформи для онлайн-навчання та спільної роботи є не тільки актуальним завданням, але й необхідною умовою для ефективного розвитку сучасної освіти та вдосконалення процесів спільної діяльності.

На рис. 1.1 представлена схема, що відображає актуальність предметної області розробки веб-платформи для онлайн-навчання.

Рисунок 1.1 – Актуальність предметної області

Центральний вузол «Онлайн-навчання» представляє основну ідею – створення інтерактивного та доступного освітнього середовища. Від нього виходять п'ять ключових напрямків, які визначають важливість цієї області:

1. Глобальний доступ до освіти. Веб-платформи забезпечують можливість навчання незалежно від місця проживання користувача. Вони зменшують освітню нерівність і відкривають нові можливості для людей з різних куточків світу.
2. Гнучкі моделі навчання. Платформи дозволяють адаптувати освітній процес під потреби кожного студента, підтримуючи самостійне навчання, онлайн-курси та змішаний формат навчання.
3. Покращена інтерактивність. Впровадження інструментів для інтерактивного навчання, таких як відеоконференції, інтерактивні завдання та чат-боти, покращує взаємодію між учнями та викладачами.
4. Технологічні досягнення. Використання новітніх технологій, таких як штучний інтелект, аналіз великих даних та адаптивне навчання, робить процес більш ефективним і персоналізованим.
5. Глобальні події, такі як COVID-19 та війна, значно збільшили потребу в онлайн-навчанні, роблячи його незамінним інструментом для забезпечення освітнього процесу навіть у кризових умовах.

Розробка веб-платформи для онлайн-навчання є стратегічно важливою, оскільки вона сприяє підвищенню доступності, якості та гнучкості освіти у світі, що стрімко змінюється.

1.2 Аналіз існуючих учбових веб-ресурсів

Аналіз існуючих учбових веб-ресурсів є важливим етапом у дослідженні теми кваліфікаційної роботи, оскільки він дозволяє зрозуміти, які функції та інструменти є найбільш затребуваними серед користувачів. Сучасні платформи, такі як Google Classroom, Coursera, Udemy, edX, Khan Academy, Moodle та інші, пропонують різноманітний функціонал, який можна класифікувати за кількома ключовими аспектами: якісний навчальний контент, інструменти для командної взаємодії, технічну надійність та зручний користувацький досвід [1]. Ретельне вивчення цих елементів допомагає врахувати сучасні тенденції, усунути недоліки існуючих рішень і створити конкурентоспроможну платформу.

На сучасному етапі розвитку технологій з'являється все більше веб-ресурсів для онлайн-навчання та організації спільної роботи. Одним із найбільш популярних інструментів є Google Classroom, який створений для полегшення комунікації між викладачами та студентами, а також для організації навчального процесу в дистанційному форматі. У цьому розділі буде проаналізовано функціональні можливості, переваги та недоліки Google Classroom у контексті розробки власної веб-платформи.

Google Classroom – це безкоштовна платформа, створена компанією Google у 2014 році, для підтримки процесу навчання. Вона є частиною пакета Google Workspace for Education і орієнтована на полегшення взаємодії між викладачами та учнями. Основні можливості включають:

- створення та управління курсами;
- розміщення навчальних матеріалів;
- проведення тестів та опитувань;
- управління завданнями і дедлайнами;
- інтеграція з іншими інструментами Google (Google Drive, Google Meet, Gmail тощо).

Google Classroom має низку переваг, які роблять його ефективним інструментом для організації навчального процесу. По-перше, платформа

забезпечує зручне управління завданнями, дозволяючи викладачам створювати, роздавати та оцінювати роботи в одному місці. По-друге, Google Classroom інтегрується з іншими сервісами Google, такими як Docs, Sheets, Drive і Meet, що полегшує спільну роботу та обмін матеріалами. Також платформа підтримує миттєві повідомлення та комунікацію між учасниками, сприяючи інтерактивності навчання. Крім того, Classroom працює на будь-якому пристрої з доступом до Інтернету, що робить його доступним для широкого кола користувачів. Простота інтерфейсу та відсутність плати за використання додають йому популярності серед закладів освіти.

Окрім того, Google Classroom дозволяє викладачам легко відстежувати прогрес кожного студента через вбудовані інструменти оцінювання та аналізу результатів. Це сприяє персоналізації навчання та забезпеченню індивідуальної підтримки. Платформа також має функцію автоматичного збереження завдань і документів у Google Drive, що мінімізує ризик втрати даних. Завдяки інтеграції з різними додатками, такими як Quizizz або Kahoot, викладачі можуть урізноманітнити навчальний процес і зробити його більш цікавим. Нарешті, Google Classroom підтримує багатомовний інтерфейс і постійно оновлюється, щоб відповідати сучасним освітнім потребам.

Попри численні переваги, Google Classroom має й деякі недоліки. По-перше, платформа має обмежені функції налаштування, що може бути незручним для викладачів, які хочуть адаптувати інтерфейс під свої потреби. По-друге, Google Classroom вимагає постійного доступу до Інтернету, що створює труднощі для користувачів із нестабільним підключенням. Також інструменти для оцінювання на платформі можуть здаватися недостатньо гнучкими, особливо для складних форм завдань, таких як проєкти або творчі роботи.

Окрім цього, відсутність інтегрованих інструментів для роботи в офлайн-режимі обмежує можливості навчання в регіонах із поганою інфраструктурою. Google Classroom має відносно базовий дизайн і не пропонує розширених аналітичних функцій, які доступні в деяких альтернативних платформах для

навчання. Нарешті, у платформи є залежність від Google-акаунтів, що може викликати проблеми з конфіденційністю та доступом у випадках, коли установа чи студенти не використовують екосистему Google.

На рис. 1.2 представлена головна сторінка платформи Google Classroom, яка демонструє сучасний мінімалістичний дизайн інтерфейсу, орієнтований на зручність користувача. Вона забезпечує швидкий доступ до основних функцій, таких як створення та приєднання до курсів, перегляд списку завдань, керування матеріалами курсу, спілкування між учасниками та отримання сповіщень про оновлення.

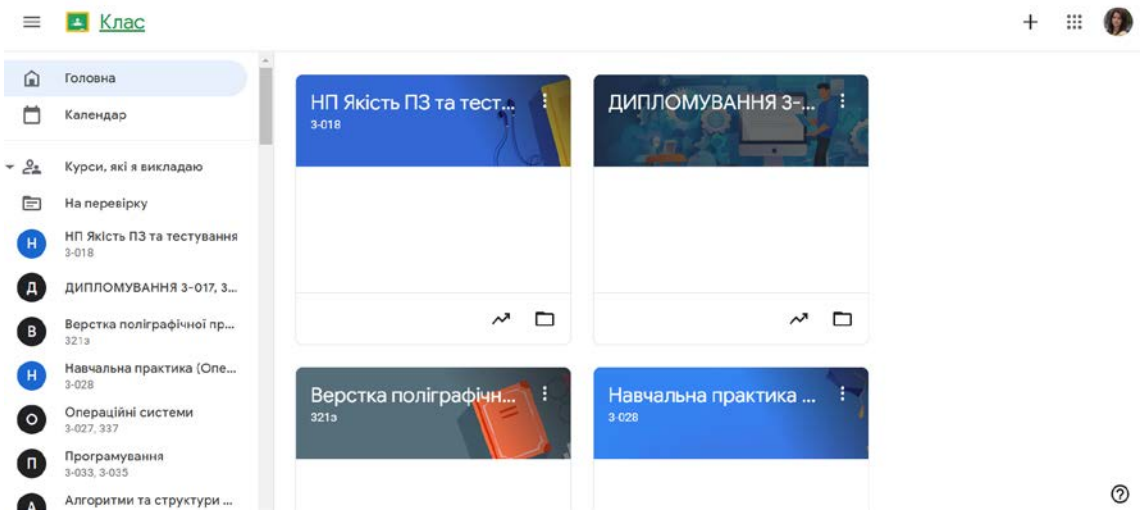

Рисунок 1.2 – Інтерфейс платформи Google Classroom

Google Classroom активно використовується у школах, університетах та корпоративному секторі. Основними причинами його популярності є легкість інтеграції та підтримка дистанційного навчання. Однак у процесі його використання були виявлені проблеми, зокрема обмеження функціоналу для складних курсів та великих проєктів.

На основі аналізу Google Classroom для створення власної веб-платформи варто врахувати кілька ключових аспектів. По-перше, важливо забезпечити інтеграцію зі сторонніми сервісами, щоб спростити роботу користувачів і розширити функціонал. По-друге, платформа повинна бути гнучкою, дозволяючи налаштовувати інтерфейс і функції відповідно до потреб викладачів і студентів. Не менш важливим є включення інструментів для

детального аналізу результатів навчання, що сприятиме оцінюванню прогресу. Окрім цього, необхідно розробити ефективні механізми для підтримки спільної роботи в реальному часі, які забезпечать інтерактивність і командну взаємодію.

Coursera є однією з найбільших платформ для масових відкритих онлайн-курсів (MOOC). Вона пропонує курси від провідних університетів та компаній, таких як Stanford, Google та IBM. Coursera використовує інтерактивні відеолекції, тести та проєкти для перевірки знань. Її основними перевагами є гнучкість у навчанні, можливість отримання сертифікатів та доступ до професійних програм. Технологічно платформа побудована на основі сучасних веб-технологій, що забезпечує масштабованість і стабільність роботи.

На рис. 1.3 представлена головна сторінка платформи Coursera, яка демонструє сучасний дизайн інтерфейсу, орієнтований на зручність користувача, з доступом до основних функцій, таких як пошук курсів, перегляд категорій, рекомендації для користувачів [1].

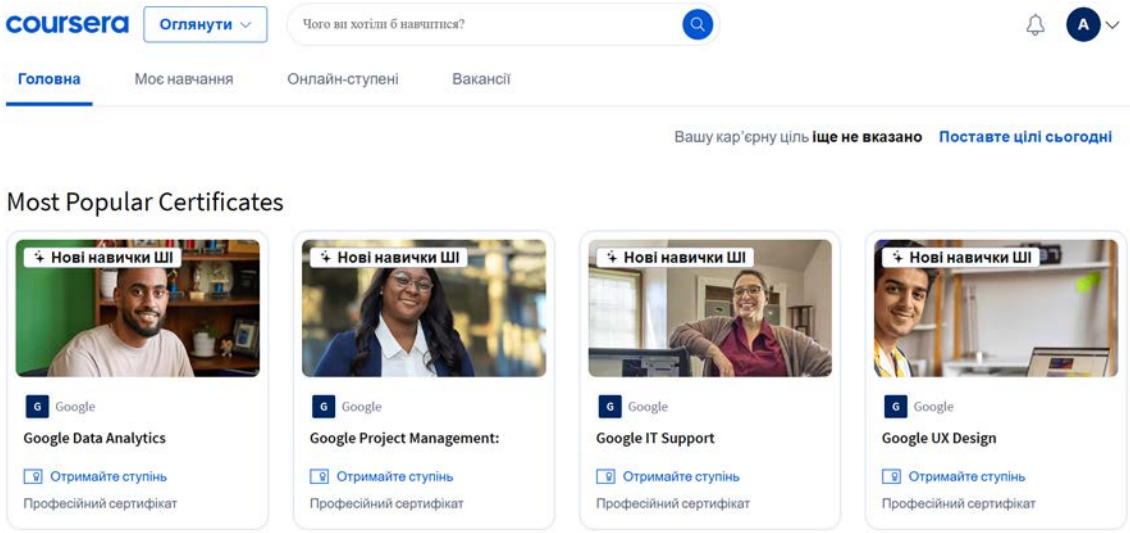

Рисунок 1.3 – Головна сторінка платформи Coursera

Udemy – це онлайн-ринок, де інструктори можуть створювати свої курси на різні теми, починаючи від програмування і закінчуючи особистісним розвитком. Платформа підтримує різні формати контенту, включаючи відео, документи, інтерактивні тести та завдання. Відкрита модель Udemy дозволяє швидко додавати нові курси, що робить її популярною серед професіоналів та ентузіастів. Технології, що використовуються, включають Node.js для бекенду

та React для фронтенду, що забезпечує динамічність та високу продуктивність.

На рис. 1.4 представлена головна сторінка платформи Udemy, яка відображає інтуїтивно зрозумілий інтерфейс із центральним акцентом на пошуку курсів, популярних категоріях, акційних пропозиціях, а також персоналізованих рекомендаціях для користувачів, що сприяє швидкому доступу до навчальних матеріалів [2].

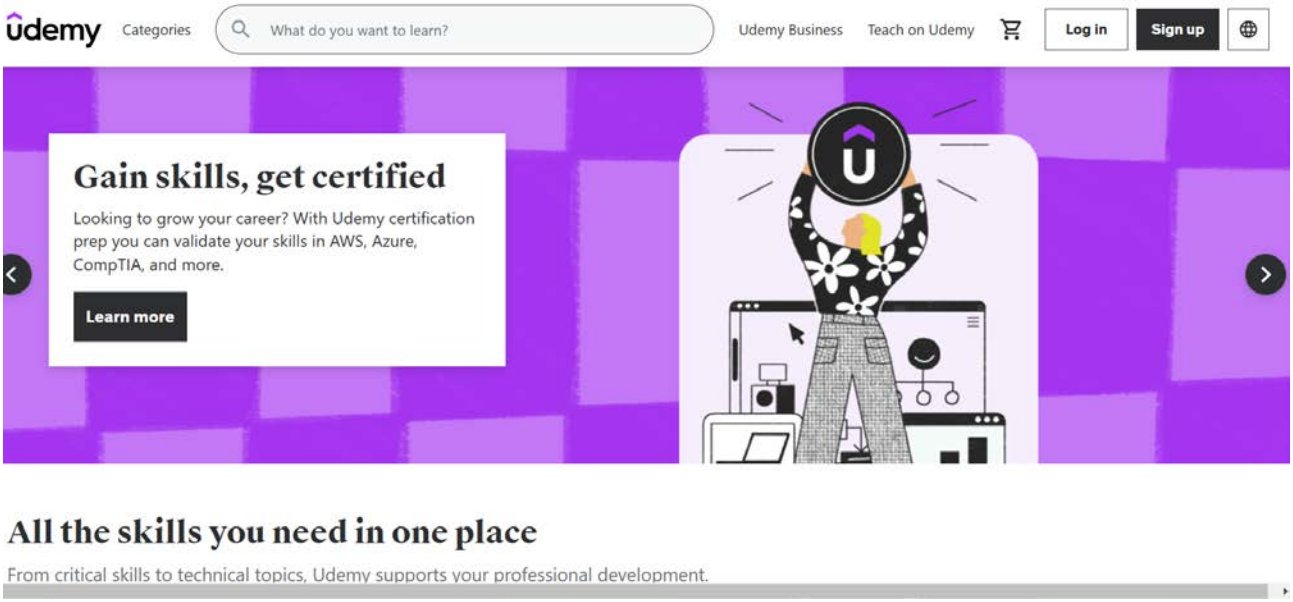

Рисунок 1.4 – Інтерфейс платформи Udemy

edX, заснована Гарвардом та MIT, пропонує безкоштовні курси з можливістю отримання сертифікатів за додаткову плату. Платформа використовує технологію Open edX, яка є відкритим програмним забезпеченням для створення навчальних курсів. edX пропонує не лише відеолекції, а й інтерактивні вправи, лабораторії та форуми для обговорень. Вона орієнтована на наукові та технічні курси і використовує технології Python (Django), що забезпечує гнучкість та масштабованість.

На рис. 1.5 представлена головна сторінка платформи edX, яка демонструє сучасний дизайн із фокусом на академічній освіті, де користувачам пропонуються курси від провідних університетів світу, інструменти для пошуку програм, а також детальна інформація про професійні сертифікати та ступені.

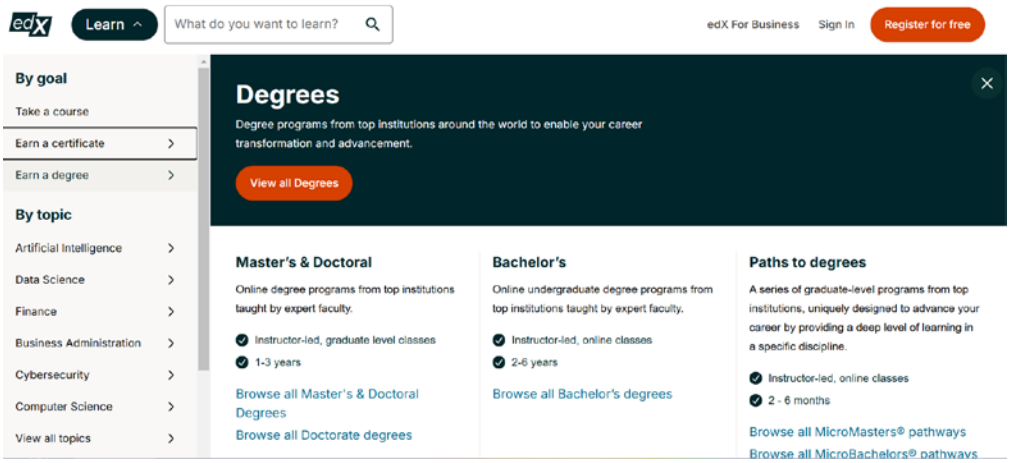

Рисунок 1.5 – Головна сторінка платформи edX

Khan Academy надає безкоштовний доступ до навчальних матеріалів для школярів і студентів у форматі відеолекцій, інтерактивних завдань та тестів. Основна ціль платформи – забезпечити якісне безкоштовне навчання для всіх. Використовує Node.js і React, що дозволяє інтерактивно взаємодіяти з учнями в реальному часі. Крім того, платформа має систему гейміфікації для підвищення мотивації користувачів.

На рис. 1.6 представлена головна сторінка платформи Khan Academy, яка відображає простий і зрозумілий інтерфейс, орієнтований на навчання для всіх вікових категорій, із доступом до інтерактивних уроків, відеоматеріалів, вправ для перевірки знань, а також персоналізованих рекомендацій для самостійного навчання.

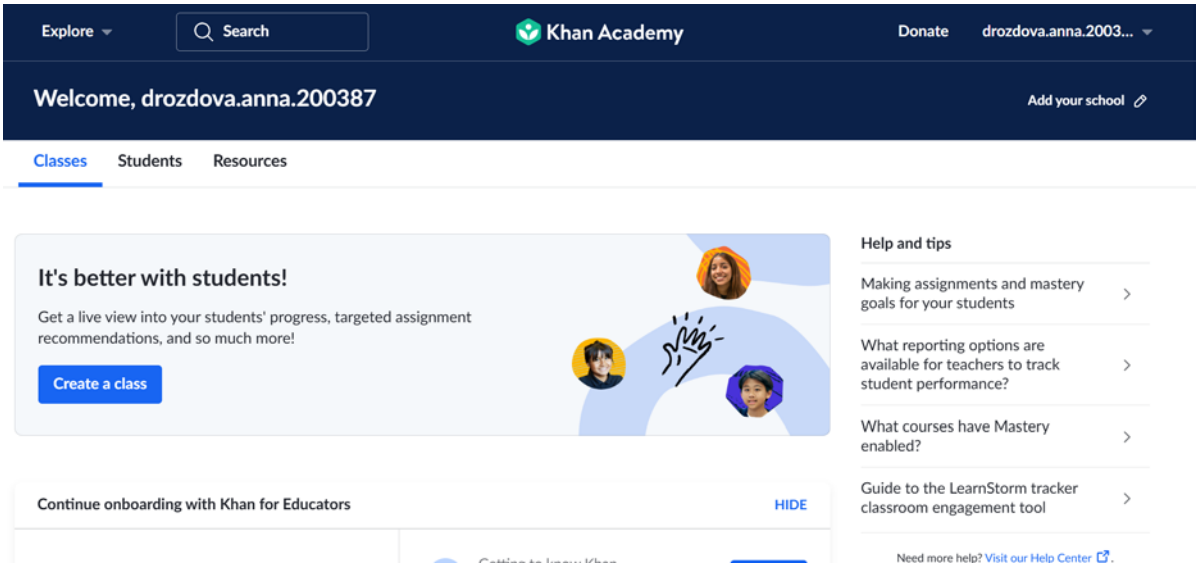

Рисунок 1.6 – Інтерфейс платформи Khan Academy

Udacity фокусується на курсах у сфері технологій та IT, пропонуючи так звані «нанодипломи» у співпраці з компаніями-лідерами індустрії (наприклад, Google, Amazon). Платформа пропонує проєктно-орієнтоване навчання, де студенти виконують практичні завдання і отримують відгуки від менторів. Udacity використовує сучасні веб-технології, включаючи Python і React, що забезпечує інтерактивність і масштабованість.

Moodle – це багатофункціональна платформа для управління навчанням, яка пропонує широкий набір інструментів для організації освітнього процесу. Вона дозволяє створювати та налаштовувати курси, завантажувати навчальні матеріали, проводити тестування, організовувати опитування та оцінювати результати студентів. Платформа підтримує інтеграцію зі сторонніми сервісами, такими як відеоконференції та зовнішні інструменти, для розширення функціональності.

Moodle також включає функції управління користувачами, спільної роботи через форуми та чати, а також інструменти для відстеження прогресу студентів і проведення аналітики. Завдяки можливостям персоналізації та підтримці численних плагінів, Moodle є потужним і гнучким рішенням для навчальних закладів і корпоративного навчання.

На рис. 1.7 представлена головна сторінка платформи Moodle, яка демонструє гнучкий дизайн інтерфейсу, що підлаштовується під потреби користувача. Вона забезпечує доступ до основних функцій, таких як створення курсів, управління матеріалами, перегляд прогресу студентів, організація тестувань і обговорень. Moodle також підтримує інтеграцію з зовнішніми сервісами, налаштування інтерфейсу та розширені інструменти аналітики, що робить платформу універсальною для закладів освіти різного рівня.

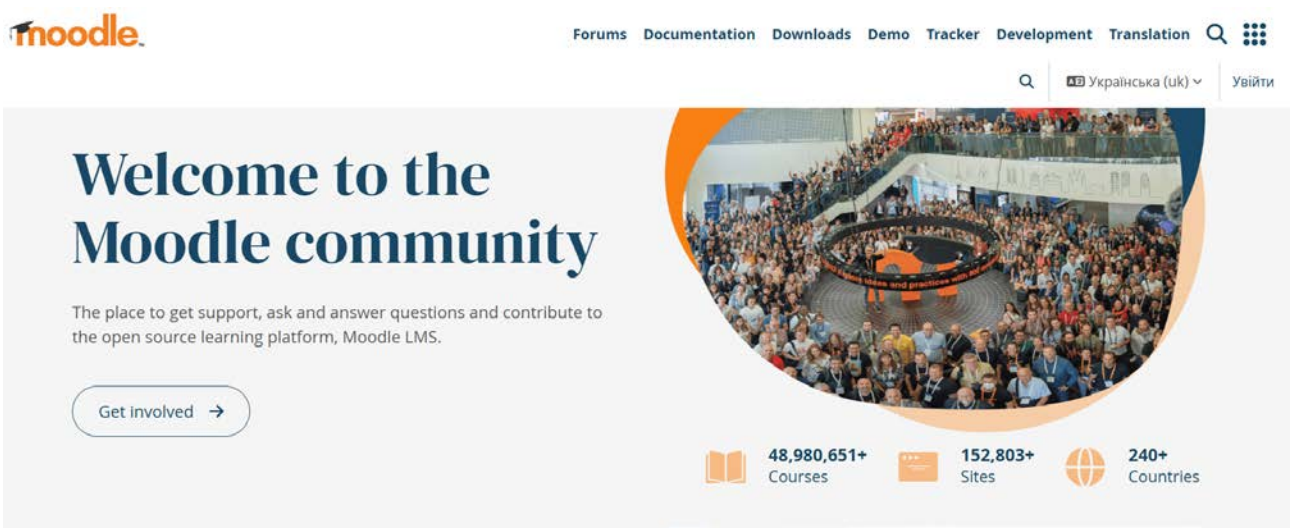

Рисунок 1.7 – Інтерфейс платформи Moodle

Платформа також підтримує адаптивний дизайн, що забезпечує зручність використання на різних пристроях, таких як комп'ютери, планшети та смартфони. Moodle дозволяє викладачам встановлювати дедлайни для завдань, налаштовувати доступ до матеріалів за прогресом студента та автоматизувати процес перевірки тестів. Для забезпечення інтерактивності навчання передбачені інструменти для групових проєктів, спільного редагування документів та обговорення в реальному часі. Окрім цього, Moodle надає можливість використання аналітики для моніторингу ефективності навчання, визначення слабких місць та надання персоналізованих рекомендацій студентам. Завдяки відкритому коду, платформу можна модифікувати відповідно до специфічних потреб закладу або організації, що робить її універсальним інструментом для дистанційного навчання.

На рис. 1.8 представлена сторінка платформи Moodle Documentation, яка надає користувачам доступ до детальних інструкцій, порад і рекомендацій щодо використання різноманітних функцій Moodle для ефективної організації навчального процесу.

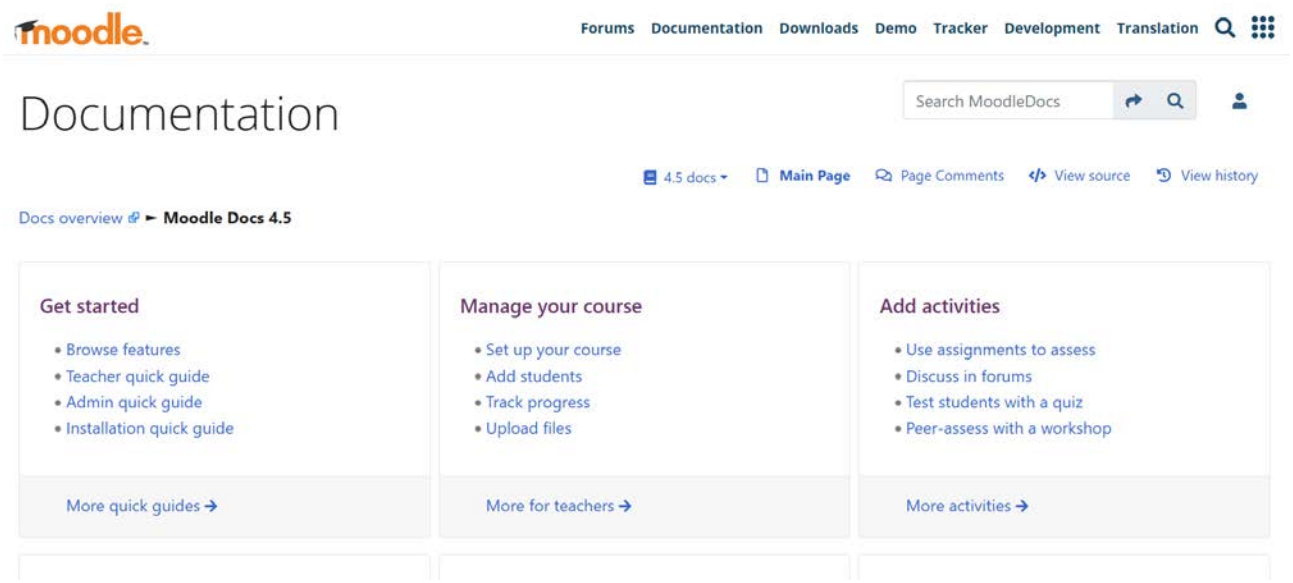

Рисунок 1.8 – Сторінка платформи Moodle Documentation

Moodle має численні переваги, які роблять його популярним вибором для організації онлайн-навчання. По-перше, це платформа з відкритим кодом, що дозволяє безкоштовно використовувати її базові функції та адаптувати під специфічні потреби закладу чи організації. По-друге, Moodle пропонує широку гнучкість у налаштуванні інтерфейсу, курсів та ролей користувачів, що дозволяє викладачам і адміністраторам повністю контролювати навчальний процес. Завдяки модульній структурі платформи можна легко додавати плагіни, розширюючи функціонал.

Moodle також підтримує багатомовність, що робить його доступним для користувачів у всьому світі. Платформа забезпечує інтеграцію з різними сторонніми сервісами, такими як системи відеоконференцій, електронні бібліотеки та інструменти для спільної роботи, що сприяє інтерактивності навчального процесу. Важливою перевагою є наявність потужних інструментів для оцінювання, моніторингу прогресу студентів та створення аналітичних звітів. Окрім цього, Moodle пропонує адаптивний дизайн, який забезпечує комфортну роботу на будь-яких пристроях, від смартфонів до настільних комп’ютерів.

Багато платформ впроваджують індивідуальні траєкторії навчання на основі аналізу даних про користувачів. Це дозволяє підвищити ефективність

навчання за рахунок персоналізованого підходу. Аналіз існуючих платформ допомагає визначити найкращі практики та інструменти, які можна використовувати для створення нових або вдосконалення наявних веб-ресурсів для дистанційного навчання.

Таким чином, результати аналізу формують базу для визначення вимог до проєкту та розробки web-платформи, яка задовольнятиме потреби як індивідуальних користувачів, так і команд, що працюють над спільними завданнями.

1.3 Аналіз технологій для розробки сучасних web-платформ для дистанційного навчання

Сучасні web-платформи для дистанційного навчання базуються на поєднанні передових технологій, що забезпечують гнучкість, масштабованість та інтерактивність. Розглянемо основні технології та їх значення.

1. Фронтенд-технології забезпечують створення зручного та інтерактивного інтерфейсу веб-платформ для дистанційного навчання. Основу складають HTML для структури, CSS для стилізації та JavaScript для динамічності. Сучасні фреймворки, такі як React, Angular і Vue.js, спрощують розробку масштабованих інтерфейсів, забезпечуючи високу продуктивність та адаптивність. Інструменти, такі як WebRTC для відеозв’язку, та бібліотеки візуалізації, як D3.js, додають інтерактивності. Використання TypeScript і CSS-фреймворків, таких як Tailwind CSS, підвищує продуктивність розробки, а оптимізатори коду, як Webpack, забезпечують швидке завантаження платформи. Це дозволяє створювати сучасні, адаптивні та ефективні рішення для онлайн-освіти [5].

2. Сучасні JavaScript-фреймворки та бібліотеки, такі як React, Angular і Vue.js, є основними інструментами для розробки динамічних і масштабованих веб-платформ. React, створений Facebook, забезпечує високопродуктивні інтерфейси завдяки компоненто-орієнтованому підходу. Angular від Google

надає повний набір інструментів для створення односторінкових додатків (SPA) з підтримкою двосторонньої прив'язки даних. Vue.js вирізняється простотою використання і гнучкістю, поєднуючи переваги React і Angular. Ці інструменти дозволяють ефективно розробляти інтерактивні застосунки, що відповідають сучасним вимогам веб-розробки.

3. Інструменти для розширення функціональності веб-платформ спрощують розробку, підвищують продуктивність і забезпечують гнучкість проектів. TypeScript додає статичну типізацію до JavaScript, що полегшує роботу з великими кодовими базами. Webpack і Vite оптимізують збірку коду, зменшують розмір файлів і прискорюють завантаження сторінок. CSS-фреймворки, такі як Bootstrap і Tailwind CSS, забезпечують адаптивний дизайн за допомогою готових стилів і компонентів, що значно скорочує час розробки. Ці інструменти дозволяють створювати більш масштабовані, ефективні та зручні для користувачів веб-додатки.

4. Інструменти для інтерактивності дозволяють створювати веб-платформи з багатим функціоналом, що підвищує залученість користувачів. WebRTC забезпечує відео- та аудіозв'язок у реальному часі без використання сторонніх плагінів, що ідеально підходить для інтерактивних занять і конференцій. Бібліотеки, такі як Chart.js і D3.js, дозволяють візуалізувати дані у вигляді діаграм, графіків і таблиць, роблячи процес навчання більш наочним. Використання цих інструментів допомагає створювати сучасні веб-платформи, які відповідають потребам користувачів і забезпечують зручний освітній досвід.

Фронтенд-технології є ключовим елементом у розробці веб-платформ, орієнтованих на якісний користувацький досвід. Їх вибір залежить від цілей проекту, масштабу платформи та очікуваного рівня інтерактивності.

Бекенд-технології є невидимою частиною веб-платформи, що відповідає за логіку роботи системи, обробку даних, зберігання інформації та взаємодію з клієнтською частиною (фронтендом). Саме бекенд забезпечує функціонування ключових елементів, таких як реєстрація користувачів, збереження прогресу навчання, робота з базами даних, інтеграція з іншими сервісами тощо. Давайте

детальніше розглянемо основні бекенд-технології, що використовуються при створенні сучасних платформ для дистанційного навчання.

Для розробки бекенду сучасних платформ для дистанційного навчання використовуються різні мови програмування, кожна з яких має свої переваги. Python із фреймворками Django та Flask є популярним вибором завдяки простоті синтаксису, швидкій розробці та потужним можливостям для масштабування, що особливо важливо для платформ із великою кількістю користувачів. JavaScript на основі Node.js дозволяє створювати високопродуктивні асинхронні застосунки, що ідеально підходить для інтерактивних елементів реального часу, як-от чати або відеоконференції. PHP із фреймворком Laravel широко використовується для створення веб-сайтів завдяки своїй простоті та великій спільноті розробників. Java з фреймворком Spring Boot забезпечує надійність і стабільність для масштабованих корпоративних проєктів, що робить її хорошим вибором для великих освітніх платформ із підвищеними вимогами до безпеки та продуктивності.

Для зберігання та обробки даних у платформах для дистанційного навчання використовуються різні типи баз даних залежно від потреб проєкту. Реляційні бази даних (MySQL, PostgreSQL) забезпечують надійне зберігання структурованих даних, таких як профілі користувачів, курси та результати тестів, дозволяючи виконувати складні запити для аналізу та звітів. Водночас NoSQL бази даних (MongoDB, Firebase) підходять для зберігання неструктурованих даних та забезпечують гнучкість при роботі зі змінними обсягами інформації, такими як мультимедійні файли або логи активності користувачів. Firebase також має функції реального часу, що є корисним для миттєвого оновлення даних на платформі, наприклад, у чатах чи інтерактивних сесіях. Вибір між SQL і NoSQL базами залежить від складності даних та вимог до масштабування платформи [6].

API (Application Programming Interface) є ключовим компонентом бекенд-технологій для забезпечення взаємодії між різними частинами системи та зовнішніми сервісами. Веб-платформи для дистанційного навчання часто

використовують REST API для обміну даними між сервером і клієнтом за допомогою HTTP-запитів, що дозволяє ефективно організувати роботу з користувачами, курсами та контентом. GraphQL є сучасною альтернативою, яка дає змогу запитувати саме ті дані, які потрібні клієнту, скорочуючи обсяг переданих даних і підвищуючи швидкість роботи. API дозволяють інтегрувати освітні платформи з іншими сервісами, такими як платіжні системи для прийому оплат, сервіси для відеоконференцій (Zoom, Jitsi), соціальні мережі для авторизації користувачів, а також забезпечують інтеграцію зі сторонніми системами для розширення функціональності.

На рис. 1.9 представлена схема, що демонструє, як різні технології взаємодіють для створення сучасної веб-платформи для дистанційного навчання.

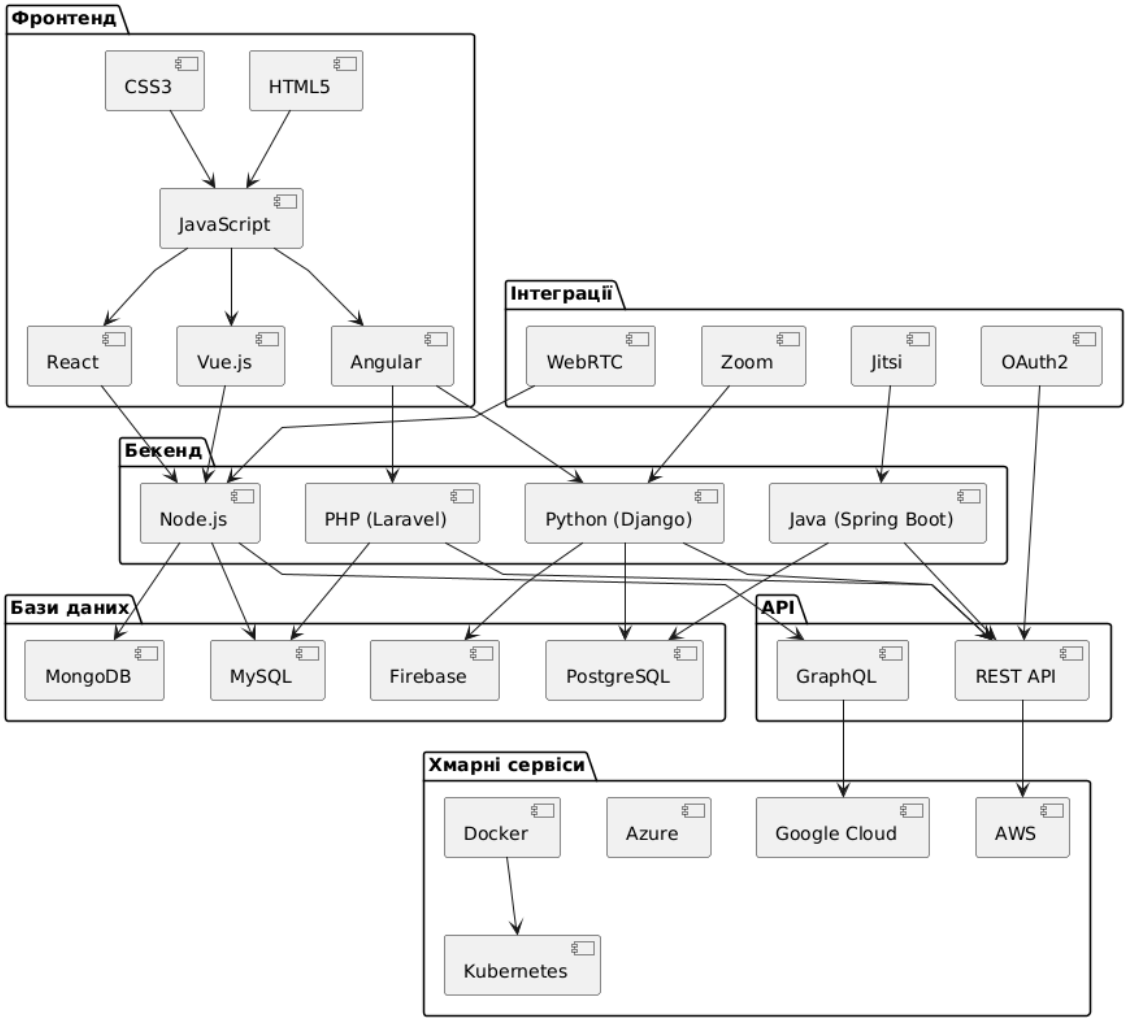

Рисунок 1.9 – Технології для створення сучасної веб-платформи для дистанційного навчання

Технології для розробки сучасних веб-платформ для дистанційного навчання повинні забезпечувати масштабованість, інтерактивність, доступність та безпеку. Їх правильний вибір є основою для створення якісного навчального середовища, здатного задовольнити потреби сучасних користувачів.

1.4 Постановка задачі досліджень

Основними варіанти використання для web-платформи для онлайн-навчання та спільної роботи можуть бути:

1. Реєстрація та аутентифікація: спільний для всіх акторів процес входу в систему. Ця функція дозволяє користувачам (студентам, викладачам і менеджерам) реєструватися на платформі та входити у свої облікові записи. Система перевіряє роль користувача та надає доступ до відповідного функціоналу.

2. Створення курсу: викладачі можуть створювати нові курси, визначаючи назву, опис та структуру курсу. Ця функція дозволяє додати базові навчальні модулі, які пізніше наповнюються матеріалами та завданнями.

3. Перегляд навчальних матеріалів: студенти отримують доступ до лекцій, презентацій, відео та інших навчальних ресурсів, які викладачі завантажують до своїх курсів.

4. Завантаження навчальних матеріалів: ця функція надає можливість викладачам додавати навчальні матеріали до курсів. Матеріали можуть включати документи, відео, зображення чи посилання на зовнішні ресурси.

5. Створення завдань: викладачі створюють завдання для студентів, задають терміни виконання, додають опис і, за потреби, прикріплюють інструкції або шаблони.

6. Виконання завдань: студенти виконують завдання, слідуючи інструкціям викладачів, і завантажують свої роботи через платформу.

7. Завантаження виконаних робіт: студенти передають готові виконані завдання через платформу для перевірки викладачем.

8. Участь в обговореннях: ця функція забезпечує взаємодію між студентами та викладачами в межах курсів. Вони можуть створювати текстові повідомлення для уточнення завдань через електрону пошту, обговорення матеріалів або отримання додаткових пояснень.

9. Перегляд зворотного зв’язку: студенти переглядають коментарі від викладачів щодо виконаних завдань. Це допомагає оцінити якість виконаної роботи та внести покращення.

10. Керування курсами: менеджери забезпечують загальне адміністрування курсів: додають або видаляють курси, архівують старі курси, керують їхньою видимістю для викладачів і студентів.

Розробка веб-платформи для онлайн-навчання та спільної роботи вимагає детального аналізу структури, функціональності та інтерфейсу системи. Одним із ключових етапів є побудова діаграм, які відображають архітектуру, взаємодію компонентів, та логіку роботи платформи.

Для створення UML-діаграми використання було використано синтаксис PlantUML [10]:

```
@startuml
left to right direction
actor Менеджер as Manager
actor Студент as Student
actor Викладач as Teacher
rectangle "Веб-платформа для онлайн-навчання" as System {

    usecase "Реєстрація та аутентифікація" as Auth
    usecase "Створення курсу" as CreateCourse
    usecase "Перегляд навчальних матеріалів" as ViewMaterials
    usecase "Завантаження навчальних матеріалів" as UploadMaterials
    usecase "Створення завдань" as CreateAssignments
    usecase "Виконання завдань" as CompleteAssignments
    usecase "Завантаження виконаних робіт" as SubmitWork
    usecase "Участь в обговореннях" as ParticipateDiscussions
    usecase "Перегляд зворотного зв’язку" as ViewFeedback
    usecase "Керування курсами" as ManageCourses


    Student --> Auth
    Teacher --> Auth
    Manager --> Auth
```

```
    Student --> ViewMaterials
    Student --> CompleteAssignments
    Student --> SubmitWork
    Student --> ParticipateDiscussions
    Student --> ViewFeedback
    Teacher --> ParticipateDiscussions
    Teacher --> CreateCourse
    Teacher --> UploadMaterials
    Teacher --> CreateAssignments
    Manager --> ManageCourses
}
@enduml
```

У середовищі розробки Visual Studio Code побудовано діаграму та експортовано файл із зображенням діаграми. Цей файл буде використовуватися для подальшого аналізу, обговорення та документування проєкту. Побудова діаграми є важливим етапом у проєктуванні та розробці веб-платформи, оскільки вона дає змогу візуалізувати складні процеси та структури (рис. 1.10).

Ця діаграма демонструє основні функції веб-платформи для онлайн-навчання та взаємодію між трьома типами користувачів: Менеджер, Викладач і Студент.

Всі функції згруповані в межах системи платформи, що забезпечує ефективну взаємодію між її користувачами.

1. Менеджер – адміністратор платформи, який відповідає за загальне управління курсами. Його обов’язки включають: налаштування структури платформи; додавання, редагування та видалення курсів; управління обліковими записами викладачів і студентів.

2. Викладач – користувач, який створює та наповнює курси. Основні функції викладача: розробка навчальних програм і курсів; завантаження та редагування навчальних матеріалів (текстових документів, відео, презентацій); створення тестів, завдань і контрольних робіт; взаємодія зі студентами через форуми, чати та систему обговорень.

3. Студент – користувач, який має доступ до навчальних матеріалів і курсів. Функції студента включають: вивчення матеріалів курсу; виконання завдань і тестів; завантаження готових робіт для перевірки викладачем; участь в обговореннях і спільних проектах.

Рисунок 1.10 – Діаграма використання для інформаційної системи аналізу результатів веб-платформи для онлайн-навчання

Ці три ролі визначають основну взаємодію в межах платформи, що забезпечує її функціональність і підтримує освітній процес. Зв'язки в діаграмі визначають взаємодію між акторами та варіантами використання (функціями) платформи.

Student --> Auth: студент реєструється та входить у систему, щоб отримати доступ до курсів, навчальних матеріалів, завдань та обговорень.

Teacher --> Auth: викладач проходить процес реєстрації та аутентифікації для створення курсів, додавання матеріалів та взаємодії зі студентами.

Manager --> Auth: менеджер аутентифікується для отримання доступу до функцій керування курсами.

Student --> ViewMaterials: студент переглядає навчальні матеріали, які були завантажені викладачем у межах відповідного курсу.

Student --> CompleteAssignments: студент виконує завдання, які створив викладач, у межах курсу.

Student --> SubmitWork: студент завантажує готові роботи у відповідь на завдання викладача.

Student --> ParticipateDiscussions: студент бере участь в обговореннях, ставлячи запитання або обговорюючи завдання та навчальні матеріали.

Student --> ViewFeedback: студент переглядає коментарі, рекомендації або відгуки, залишені викладачем стосовно виконаних завдань.

Teacher --> ParticipateDiscussions: викладач бере участь в обговореннях для взаємодії зі студентами, надаючи відповіді на запитання чи роз'яснення щодо матеріалів і завдань.

Teacher --> CreateCourse: викладач створює курси, додаючи назву, опис і структуру.

Teacher --> UploadMaterials: викладач завантажує навчальні матеріали, які стають доступними студентам у межах курсу.

Teacher --> CreateAssignments: викладач створює завдання для студентів, визначаючи дедлайни, інструкції та прикріплюючи додаткові матеріали.

Manager --> ManageCourses: менеджер управляє курсами, змінює їхній статус (активний/архівований), налаштовує видимість курсів для викладачів і студентів, здійснює моніторинг активності.

Типи зв’язків у системі платформи визначають взаємодію між її компонентами та акторами, забезпечуючи цілісність і коректну роботу. Основними типами зв’язків є асоціації, залежності, агрегації та композиції.

Асоціації (-->): прямі стрілки між акторами та варіантами використання

вказують на те, що актор виконує або ініціює певну функцію. Наприклад, Student --> ViewMaterials означає, що студент може переглядати навчальні матеріали.

Залежності (include/extend): у цій діаграмі залежності не використовуються, але їх можна додати для уточнення, якщо певна функція є залежною від іншої.

Усі три актори (Менеджер, Викладач, Студент) підключені до функції «Реєстрація та аутентифікація», оскільки це обов'язковий початковий етап для доступу до системи. Зв'язки між студентами та функціями зосереджені на перегляді матеріалів, виконанні завдань і отриманні зворотного зв'язку. Зв'язки викладача включають функції створення та адміністрування курсів і матеріалів. Зв'язок менеджера обмежується керуванням курсами, що відображає його адміністративну роль у платформі. Ця структура зв'язків забезпечує чіткий розподіл функцій між акторами, відповідно до їхньої ролі на платформі.

Висновки до розділу:

У першому розділі виконано аналіз існуючих веб-платформ та програмних рішень, що дозволяють організувати онлайн-навчання та спільну роботу. Зокрема, були досліджені можливості, переваги й недоліки таких популярних платформ, як Google Classroom, Coursera, Udemy, edX, Khan Academy. Ці продукти забезпечують широкий функціонал, який включає доступ до якісного навчального контенту, інтерактивні інструменти для користувачів, а також можливості для організації командної роботи у реальному часі. Вони дозволяють не лише підвищувати рівень знань користувачів, але й створювати ефективне середовище для колективного виконання завдань. Проте основними недоліками таких рішень є залежність від стабільного інтернет-з'єднання, відсутність персоналізації для вузькоспеціалізованих навчальних потреб та висока вартість преміум-функцій у більшості платформ.

Також розглянуто інший можливий варіант реалізації веб-платформи для онлайн-навчання та спільної роботи з використанням інтерфейсів прикладного

програмування (API), що надаються провідними освітніми платформами та сервісами. Зокрема, виконано аналіз можливостей API, що підтримують інтеграцію освітнього контенту й інструментів для спільної роботи. На основі виконаних досліджень прийнято рішення розробляти власну платформу для забезпечення спільного редагування документів і персоналізації навчального процесу.

Розроблено вимоги до функціоналу інформаційної системи для онлайн-навчання, які візуалізовано за допомогою діаграми варіантів використання, побудованої з використанням синтаксису PlantUML.

Діаграма варіантів використання відображає ключові функціональні компоненти платформи, зокрема, реєстрацію користувачів, перегляд курсів, організацію спільної роботи, інтерактивні інструменти для виконання завдань. Застосування цієї діаграми дозволяє чітко визначити основні сценарії використання системи і наочно представити структуру її функціональності, що забезпечує ефективне проектування та подальшу розробку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ РЕЗУЛЬТАТІВ WEB-ПЛАТФОРМИ ДЛЯ ОНЛАЙН-НАВЧАННЯ

2.1 Вибір та обґрунтування технологій реалізації web-платформи для онлайн-навчання та спільної роботи

З урахуванням вимог до функціональності інформаційної системи для онлайн-навчання та спільної роботи, реалізацію системи було вирішено виконувати на фреймворку React з використанням мови програмування TypeScript. Такий вибір зумовлений здатністю React забезпечувати інтерактивність і динамічність інтерфейсу, а також перевагами TypeScript, зокрема статичною типізацією, масштабованістю та інтеграцією з іншими сучасними інструментами.

Даний підхід дозволяє розробляти компонентно-орієнтовану архітектуру, яка спрощує підтримку і розширення системи. React забезпечує швидке оновлення інтерфейсу за допомогою віртуального DOM, що є критично важливим для додатків з високою взаємодією користувачів, наприклад, для функцій спільної роботи чи чату. Використання TypeScript додає строгість у визначенні типів даних, що зменшує кількість помилок і підвищує надійність програмного забезпечення.

Обраний стек технологій також підтримує інтеграцію з SCSS для організації стилів, Firebase для зберігання даних і обробки реального часу, а також формат JSON для обміну даними між клієнтською та серверною частинами системи. Основними перевагами даного рішення є:

1. Підвищена безпека коду: Статична типізація допомагає виявляти помилки ще на етапі розробки, що сприяє більшій надійності системи.

2. Зручність масштабування: Об’єктно-орієнтований підхід і модульна структура полегшують розвиток проєкту та впровадження нових функцій.

3. Сумісність із сучасними технологіями: TypeScript добре інтегрується з

React, SCSS, і такими сервісами, як Firebase, що дозволяє реалізувати складний функціонал, зокрема реєстрацію, чат та інтерактивний інтерфейс.

4. Простота підтримки: Використання інструментів Visual Studio Code, таких як IntelliSense, ESLint і Prettier, покращує якість коду та полегшує співпрацю в команді розробників.

Швидка інтеграція з API: Завдяки підтримці JSON TypeScript забезпечує ефективну передачу даних між клієнтською та серверною частинами платформи.

Таким чином, поєднання TypeScript та Visual Studio Code забезпечує гнучкість розробки, надійність роботи системи та відповідність сучасним вимогам до веб-додатків.

Для розробки web-платформи для онлайн-навчання та спільної роботи важливо використовувати сучасні технології, які забезпечують продуктивність, гнучкість і масштабованість. У цій роботі було вирішено застосувати CSS, React, SCSS, TypeScript та JSON як основні технології для побудови функціональної, інтерактивної та легкої у підтримці системи.

CSS – мова стилізації, що використовується для створення візуального оформлення веб-сторінок. У контексті розробки платформи для онлайн-навчання CSS дозволяє забезпечити естетичність і зручність інтерфейсу. Роль у проєкті:

React – це бібліотека JavaScript, яка дозволяє створювати динамічні та багатофункціональні інтерфейси користувача. Для веб-платформи React забезпечує компонентно-орієнтований підхід до розробки, що значно спрощує підтримку та масштабування системи. Роль у проєкті:

1. Реалізація інтерактивних компонентів, таких як список курсів, тести, чат для спільної роботи.

2. Забезпечення швидкого оновлення інтерфейсу завдяки використанню віртуального DOM.

3. Реалізація односторінкової структури веб-додатка (SPA), що дозволяє користувачам швидко взаємодіяти із системою без перезавантаження сторінок.

SCSS є розширенням CSS, яке додає додаткові функції для спрощення

роботи з великою кількістю стилів. У масштабних проєктах, таких як платформа онлайн-навчання, SCSS дозволяє підтримувати порядок і структуру в коді стилів. Вкладені структури стилів SCSS використовуються для створення унікального вигляду модулів платформи, таких як панель навігації, вікно чату чи форми зворотного зв’язку.

TypeScript – це мова програмування, яка є надбудовою над JavaScript і додає статичну типізацію. Використання TypeScript у розробці забезпечує високу якість коду і дозволяє уникати багатьох помилок на етапі написання. Забезпечення безпеки даних завдяки статичній типізації, що зменшує ймовірність логічних помилок. Роль технологій у розробці web-платформи для онлайн-навчання та спільної роботи:

– підтримка об’єктно-орієнтованого підходу для організації складного функціоналу платформи (наприклад, управління курсами та користувачами);
– спрощення роботи з великим кодом завдяки IntelliSense і підказкам у середовищі розробки (Visual Studio Code);
– TypeScript використовується для реалізації логіки платформи, наприклад, валідації форм реєстрації, управління станом додатка та обробки подій.

JSON – це формат даних, що використовується для обміну інформацією між клієнтом (веб-додатком) і сервером. Він легкий у використанні та сумісний із більшістю мов програмування. Основними перевагами цієї технології є:

1. Збереження і передача даних про курси, завдання, профілі користувачів.
2. Формат для зберігання конфігурацій платформи (ролей користувачів).
3. Використовується для інтеграції з Firebase (бази даних і аутентифікація).

Дані про результати студентів зберігаються у форматі JSON у хмарній базі даних і передаються у веб-додаток для відображення.

Застосування цих технологій дозволяє створити ефективну, інтерактивну та масштабовану платформу для онлайн-навчання та спільної роботи. Вони забезпечують як якісний інтерфейс для користувачів, так і зручність для

розробників при реалізації складного функціоналу.

Також для розробки системи було вирішено використовувати наступну бібліотеку MUI, що дозволяє ефективно вирішувати задачі інтерактивності, візуального оформлення та зручності використання платформи.

MUI (Material-UI) – це популярна бібліотека компонентів для React, заснована на дизайні Material Design, створеному Google. Бібліотека надає готові стилізовані компоненти, які допомагають швидко створювати сучасні, естетичні та функціональні інтерфейси користувача. У рамках розробки web-платформи для онлайн-навчання та спільної роботи MUI дозволяє значно скоротити час розробки UI та забезпечити зручний і зрозумілий для користувачів дизайн.

Роль MUI у проєкті полягає в наданні готових адаптивних та стилізованих компонентів, які дозволяють швидко створювати сучасний, функціональний і зручний інтерфейс користувача для веб-платформи онлайн-навчання та спільної роботи, дотримуючись принципів Material Design.

Швидка розробка інтерфейсу MUI надає набір попередньо стилізованих компонентів (кнопки, форми, таблиці, панелі навігації, діалогові вікна тощо), які можна швидко інтегрувати у проєкт. Компоненти адаптовані для React, що дозволяє легко їх використовувати в інтерактивних сценаріях платформи. Використання компонентів, таких як AppBar для створення панелі навігації, або Button для кнопок керування завданнями студентів. До основних переваг бібліотеки відноситься:

1. Адаптивний дизайн. MUI має вбудовану підтримку адаптивності через Grid system і Breakpoints, що забезпечує коректне відображення платформи на різних пристроях. Це особливо важливо для онлайн-навчання, адже користувачі можуть працювати як з комп'ютера, так і з мобільного телефону. Адаптивні розклади для сторінок з курсами та матеріалами.

2. Тема та кастомізація. MUI підтримує налаштування теми, що дозволяє адаптувати кольорову схему платформи до корпоративного стилю або побажань клієнта. Завдяки ThemeProvider розробники можуть змінювати кольори,

шрифти та стилі компонентів у глобальному масштабі.

MUI надає компоненти для складних функцій, таких як DataGrid для роботи з таблицями, Snackbar для показу сповіщень, Dialog для модальних вікон.

Це значно полегшує створення функцій, таких як управління користувачами, перегляд результатів тестів, сповіщення студентів. Використання Snackbar для виведення повідомлень про успішне завантаження файлу чи завершення тесту.

3. Модульність і легкість інтеграції. Компоненти MUI легко інтегруються з TypeScript, що забезпечує строгість типізації і спрощує підтримку коду. MUI добре поєднується зі SCSS, дозволяючи розширювати стилі компонентів за потреби. Інтеграція кастомних стилів для унікального оформлення платформи без порушення логіки роботи компонентів.

Роль у розробці ключових функцій платформи охоплює реалізацію важливих складових системи, які забезпечують зручність використання, інтерактивність та ефективність роботи користувачів. Для аутентифікації та реєстрації застосовуються компоненти TextField, Button, FormControl, що дозволяють створювати форми з полями вводу та валідаторами, спрощуючи процес реєстрації й входу в систему. Навігація реалізується за допомогою адаптивної панелі, створеної з компонентів AppBar, Toolbar і Drawer, що забезпечує швидкий доступ до основних розділів платформи. Для роботи з курсами та матеріалами використовуються таблиці (DataGrid) для відображення списків курсів, завдань і файлів, а також карточки (Card) для візуально привабливого представлення матеріалів. Функція комунікації, як чат або сповіщення, реалізується за допомогою Snackbar у поєднанні з інтеграцією Firebase Realtime Database, що забезпечує обмін повідомленнями в реальному часі. Ці рішення сприяють створенню сучасної, функціональної та зручної платформи для онлайн-навчання та спільної роботи.

Використання MUI у розробці web-платформи для онлайн-навчання та спільної роботи забезпечує швидку розробку сучасного інтерфейсу, який

відповідає стандартам зручності й адаптивності. Завдяки гнучкості у налаштуванні та широкому набору компонентів, бібліотека MUI дозволяє реалізувати всі вимоги до дизайну та функціональності системи.

2.2 Проєктування структури бази даних інформаційної системи web-платформи для онлайн-навчання та спільної роботи

Функціонал web-платформи для онлайн-навчання та спільної роботи (рис. 1.11), відповідно до визначених вимог, передбачає реалізацію механізмів аутентифікації та авторизації користувачів. З огляду на те, що розробка планується на базі технології Firebase, доцільно використати її вбудовані сервіси Firebase Authentication для забезпечення безпечного та зручного керування доступом користувачів.

Firebase надає набір інструментів для зберігання інформації про користувачів системи, їхні ролі, зареєстровані облікові записи та методи аутентифікації (наприклад, за допомогою пароля або сторонніх OAuth-сервісів). Також платформа дозволяє управляти відомими твердженнями (claims) про користувачів для налаштування доступу та розмежування прав.

Firebase Database – це хмарний сервіс баз даних від Google, призначений для зберігання та синхронізації даних у реальному часі. Firebase пропонує два основних типи баз даних:

1. Firebase Realtime Database (RTDB)
   - це документно-орієнтована база даних JSON, яка забезпечує синхронізацію даних у режимі реального часу між клієнтами;
   - підходить для додатків, які потребують швидкого оновлення даних, таких як чати, віджети оновлення статусів або багатокористувацькі системи.
2. Cloud Firestore

- гнучка, масштабована та вдосконалена документно-орієнтована база даних NoSQL, що підтримує складні запити та інтеграцію з іншими сервісами Google Cloud;
- більш сучасна та гнучка альтернатива Realtime Database.

Firebase Authentication – це інструмент, який входить до складу платформи Firebase і забезпечує просте та безпечне управління процесом аутентифікації користувачів у веб- та мобільних додатках. Він підтримує різні методи аутентифікації, дозволяючи інтегрувати як базові механізми, так і популярні сторонні сервіси.

Cloud Firestore використовує більш складну і організовану структуру даних, засновану на колекціях і документах. Це дозволяє забезпечити гнучкість при виконанні складних запитів, фільтрації, сортуванні даних та роботи з великими обсягами. Firestore також має перевагу в масштабованості, підтримуючи глобальне розподілене середовище для високонавантажених додатків. Крім того, Firestore дозволяє більш точно налаштувати правила безпеки на рівні документів та колекцій, що дає більшу гнучкість при контролю доступу до даних.

У Firestore також реалізовані більш потужні механізми транзакцій і підтримка батч-записів, що робить його кращим вибором для більш складних і великих проектів. Проте, з огляду на структуру та можливості, налаштування Cloud Firestore може бути більш складним порівняно з Firebase Realtime Database

В таблиці 2.1 представлено порівняння між Firebase Realtime Database та Cloud Firestore. Firebase Realtime Database є більш простим у налаштуванні та використанні, оскільки використовує структуру даних у вигляді дерева JSON, що дозволяє легко додавати та змінювати дані. Однак, це може призвести до обмежень при роботі з великими або складними даними, оскільки структура дерева не завжди оптимальна для складних запитів. Крім того, масштабованість Realtime Database є обмеженою при високих навантаженнях.

Таблиця 2.1 – Перелік характеристик між Firebase Realtime Database та Cloud Firestore

Характеристика	Realtime Database	Cloud Firestore
Структура даних	Дерево JSON	Колекції та документи
Масштабованість	Обмежена при роботі з великими наборами даних	Краще масштабування для великих додатків
Синхронізація	Дані синхронізуються в реальному часі	Синхронізація в реальному часі, але з додатковими можливостями, такими як офлайн-доступ
Запити	Обмежена підтримка складних запитів	Підтримка складних запитів із фільтрацією, сортуванням, обмеженням та об'єднанням
Безпека	Правила безпеки JSON на вузловій основі	Більш потужні правила доступу на рівні документів та колекцій
Транзакції	Простий підхід до транзакцій	Підтримка складних транзакцій на кількох документах
Вартість	Дешевше для невеликих додатків	Ефективніше для масштабних додатків через модель ціноутворення на основі операцій

Таким чином, вибір між Firebase Realtime Database і Cloud Firestore залежить від потреб проекту, складності структури даних, вимог до масштабованості та необхідності виконання.

Приклад роботи з Firebase Realtime Database на мові JavaScript: запис даних здійснюється за допомогою методу set(), який дозволяє зберігати інформацію в заданому місці бази даних.

```
import { getDatabase, ref, set } from "firebase/database";

const db = getDatabase();
set(ref(db, 'users/' + userId), {
  username: "example_user",
  email: "user@example.com",
  role: "student"
});
```

Зчитування даних з Firebase Realtime Database на мові JavaScript здійснюється за допомогою методу onValue(), який дозволяє підписатися на зміну даних у реальному часі та отримувати їх при кожній зміні.

```
import { getFirestore, collection, getDocs } from "firebase/firestore";

const db = getFirestore();
const querySnapshot = await getDocs(collection(db, "users"));
querySnapshot.forEach((doc) => {
  console.log(`${doc.id} =>`, doc.data());
});
```

На рис. 2.1 наведено UML діаграму класів [8], що показує основні взаємозв'язки між сутностями предметної області, що приймають участь у процесах онлайн-навчання та спільної роботи для інформаційної системи веб-платформи, яка використовує функціональність управління курсами, завданнями, повідомленнями та оцінками.

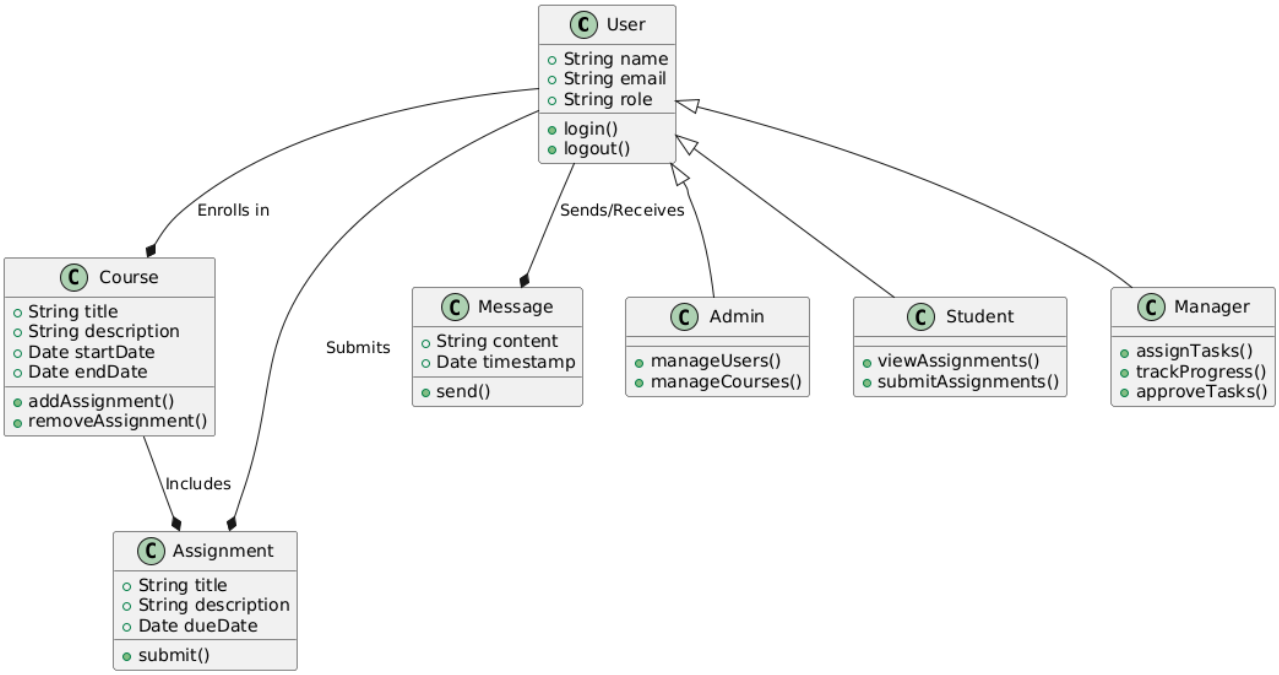

Рисунок 2.1 – Діаграма класів UML основних сутностей інформаційної системи web-платформи для онлайн-навчання та спільної роботи

Клас User представляє основну сутність системи – користувача веб-платформи для онлайн-навчання та спільної роботи. Він містить ключові властивості для зберігання інформації про користувача, такі як name (ім’я), email (електронна пошта), password (пароль для входу в систему) та role (роль користувача: адміністратор, студент або менеджер). Крім того, клас включає методи для взаємодії з системою: login() для входу, logout() для виходу, а також updateProfile() для оновлення персональних даних користувача. Завдяки таким властивостям і методам, цей клас забезпечує базову функціональність управління користувачами в системі.

Клас Course представляє сутність курсу в системі онлайн-навчання та спільної роботи. Він містить основні властивості для опису курсу: title (назва курсу), description (опис, що надає деталі про зміст курсу). Такі властивості забезпечують зберігання структурованої інформації про кожен курс. Крім того, клас включає методи addAssignment() та removeAssignment(), які дозволяють додавати та видаляти завдання, пов’язані з курсом. Завдяки цим методам клас Course надає функціональність для управління навчальним процесом і його структурою.

Клас Assignment представляє сутність завдання, яке призначається користувачам у межах курсу. Він має такі властивості: title (назва завдання), description (опис із деталями про завдання). Метод submit() дозволяє користувачам подавати виконані завдання через платформу. Цей клас забезпечує структуру для управління завданнями та їх виконанням.

Клас Message реалізує функціональність обміну повідомленнями між користувачами системи. Він має властивості content (текст повідомлення) та timestamp (дата і час відправлення). Метод send() відповідає за надсилання повідомлень між користувачами. Клас забезпечує базову комунікацію в межах платформи, сприяючи взаємодії між учасниками навчального процесу.

У системі онлайн-навчання класи мають такі зв’язки:

User і Course: Зв’язок типу User --* Course з позначенням «Enrolls in» вказує, що користувачі можуть записуватися на курси. Це забезпечує взаємодію між користувачами та курсами.

User і Message: Зв’язок типу User --* Message з позначенням «Sends/Receives» демонструє, що користувачі можуть відправляти та отримувати повідомлення через платформу.

Course і Assignment: Зв’язок типу Course --* Assignment з позначенням «Includes» означає, що кожен курс може містити одне або кілька завдань.

User і Assignment: Зв’язок типу User --* Assignment з позначенням «Submits» вказує на те, що користувачі виконують і подають завдання, які входять до курсу.

Клас Admin успадковує основний функціонал класу User і доповнюється специфічними методами:

- manageUsers(): метод для управління користувачами системи.
- manageCourses(): метод для створення, редагування та видалення курсів.

Такі зв’язки та спадкування дозволяють чітко структурувати систему та забезпечити її гнучкість для виконання різних ролей у платформі.

Опис додаткових класів та методів:

1. Grade: Клас для оцінок, що дає можливість присвоювати оцінки за виконані завдання та отримувати деталі оцінки. Це забезпечує збереження інформації про успішність користувачів у процесі виконання завдань та тестів.

2. Profile: Клас для профілю користувача, який містить біографічну інформацію, фотографії та інші дані про користувача. Цей клас дозволяє зберігати персоналізовану інформацію для кожного користувача системи.

3. Методи: Кожен з класів має певні методи для взаємодії з іншими об'єктами. Методи для класу Grade – assignGrade() для присвоєння оцінки, для класу Message – send() для відправки повідомлень, а для класу Assignment – submit() для подачі завдання.

4. Ролі: В системі передбачено три основні ролі користувачів: Admin, Student та Manager. Кожна з цих ролей має специфічні методи для виконання своїх функцій. Admin може управляти користувачами, курсами та завданнями, Student – переглядати курси, виконувати завдання та проходити тести, а Manager – призначати завдання, відслідковувати прогрес.

2.3 Розробка основного функціоналу веб-додатку

Розробка основного функціоналу веб-додатку складається зі сторінки HomeworkPage, яка забезпечує користувачам можливість переглядати перелік призначених завдань, отримувати детальну інформацію про кожне завдання, завантажувати необхідні файли для виконання та подавати виконані роботи на перевірку.

Компонент React реалізує сторінку для перегляду, оцінки та управління домашніми завданнями студентів у контексті навчальної платформи. Код використовує інтеграцію з Firebase Firestore для роботи з даними та включає управління станом через Redux.

Основні елементи коду web-платформи для онлайн-навчання та спільної роботи включають такі компоненти:

1. Firebase Firestore – для отримання документів з бази даних (doc, getDoc).

2. React – для створення компонентів і керування станом через хуки (useState, useEffect).

3. React Router – для отримання параметрів маршруту (useParams).

4. Redux – для використання сховища стану програми (useAppDispatch, useAppSelector) та дій (editHomework, modalsActions).

5. Матеріали інтерфейсу (@mui/material) – для формування зручного інтерфейсу (TextField, Button).

6. Іконки (@mui/icons-material) – для роботи з інтерфейсом (іконка завантаження файлів).

Функціонал HomeworkPage включає отримання даних про домашнє завдання з бази даних, їх відображення у вигляді списку із заголовками, описом та можливістю завантаження необхідних матеріалів.

1. Отримання даних про домашнє завдання. При завантаженні сторінки useEffect ініціалізує запит до Firebase Firestore, щоб отримати дані про конкретне домашнє завдання, використовуючи параметри маршруту (organizationId, classroomId, lessonId, studentId). Отримані дані обробляються функцією formatDoc для форматування документа в зручний для роботи об'єкт.

2. Управління оцінкою. Поле для введення оцінки (newMark) дозволяє вчителю змінити оцінку студента. Після натискання кнопки Save викликається дія editHomework, яка оновлює дані у Firestore через Redux, а нова оцінка зберігається локально у стані компонента.

3. Функція відхилення домашнього завдання. Кнопка Decline встановлює статус завдання як «не виконано» (isDone: false) і «відхилено» (isDeclined: true). Це також оновлюється через Redux і Firestore.

4. Робота з прикріпленими файлами. Перегляд файлів, робимо клік по файлу, відкриває модальне вікно для його перегляду (openFile). Завантаження файлів відбувається при натискання на іконку, завантаження створює HTML-елемент <a> для завантаження файлу за посиланням (downloadFile).

5. Відображення даних. Тема уроку (lesson.topic), опис уроку (lesson.description), кількість викладачів і прикріплені файли відображаються в головному і бічному контейнерах.

6. Стан завантаження. Компонент обробляє стан завантаження даних через isLoading. Поки дані завантажуються, виводиться текст «Loading...». Якщо домашнє завдання відсутнє, з’являється повідомлення «This student hasn't completed homework yet».

Особливості HomeworkPage полягають у зручності використання для студентів та викладачів, включаючи інтуїтивний інтерфейс, динамічне оновлення даних у режимі реального часу, можливість завантаження матеріалів для виконання завдань, подачу виконаних робіт через систему, а також інтеграцію з іншими сторінками платформи, такими як сторінка оцінок чи профілю користувача.

Функціонал сторінки HomeworkPage побудований на інтеграції з Firebase, що забезпечує отримання даних із Firestore через методи, такі як getDoc, та їх оновлення за допомогою Redux для централізованого управління станом. У додатку використовуються React-хуки, зокрема, useState для локального управління станом (наприклад, для відображення домашніх завдань або обробки стану завантаження) та useEffect для виконання побічних ефектів, таких як запити до бази даних. Redux забезпечує централізоване управління станом за допомогою дій, таких як editHomework чи modalsActions, що дозволяють змінювати дані у відповідь на дії користувача. Інтерфейс сторінки є інтерактивним та інтуїтивним завдяки використанню бібліотеки Material-UI, яка забезпечує підтримку таких функцій, як перегляд та завантаження файлів, покращуючи користувацький досвід (рис. 2.2).

```
import { doc, getDoc } from "firebase/firestore";
import React from "react";
import { useParams } from "react-router-dom";
import { useAppDispatch, useAppSelector } from "store";
import { Homework } from "types/Lessons";
import { firestoreDB } from "utils/firebase";
import formatDoc from "utils/helpers/formatDoc";
import { TextField, Button } from "@mui/material";
import "./styles.scss";
import { editHomework } from "store/slices/homework";
import Divider from "components/Divider";
import { modalsActions } from "store/slices/modals";
import DownloadIcon from "@mui/icons-material/Download";

const cssName = "homework-page";
const getNameFromUrl = (url = "") => {
  const s = url.split("?")[0].split("%2F");
  return s[s.length - 1];
};
```

Рисунок 2.2 – Фрагмент коду, який демонструє підключення функцій перегляду та завантаження файлів

Потенційні поліпшення для сторінки HomeworkPage включають впровадження обробки помилок, що дозволить коректно реагувати на збої при отриманні даних із Firestore або оновленні документів (рис. 2.3). Для покращення продуктивності можна оптимізувати запити до бази даних і зменшити кількість повторних рендерів компонентів. Також варто розширити функціонал, додавши можливості сортування завдань, фільтрування за їх статусом та реалізацію пошуку для полегшення навігації користувачів.

Цей компонент демонструє, як сучасний стек технологій (React, Redux, Firebase) можна використовувати для створення ефективного та функціонального інтерфейсу для управління домашніми завданнями.

```
function HomeworkPage() {
  const dispatch = useAppDispatch();
  const { organizationId, classroomId, lessonId, studentId } = useParams();
  const lessons = useAppSelector((state) => state.lessons.lessons);
  const lesson = lessons?.find((org) => org.id === lessonId);
  const classrooms = useAppSelector((state) => state.classrooms.classrooms);
  const classroom = classrooms?.find((x) => x.id === classroomId);
  const [homework, setHomework] = React.useState<Homework | null>(null);
  const [isLoading, setIsLoading] = React.useState(true);
  const [newMark, setNewMark] = React.useState("");
  React.useEffect(() => {
    setIsLoading(true);
    getDoc(
      doc(
        firestoreDB,
        `/organizations/${organizationId}/classrooms/${classroomId}
                        /lessons/${lessonId}/homeworks/${studentId}`
      )
    )
      .then((res) => {
        const hom = formatDoc(res) as Homework;
        setHomework(hom);
        setNewMark(hom.mark);
      })
      .finally(() => setIsLoading(false));

    return () => {
      setHomework(null);
    };
  }, [lessonId]);
```

Рисунок 2.3 – Фрагмент коду сторінки HomeworkPage, який демонструє реалізацію основного функціоналу

Опис коду ClassroomPage. Цей компонент React відображає список уроків у межах класної кімнати (Classroom). Код інтегрується з Redux для отримання стану уроків та їх відображення у вигляді прев'ю. Компонент реалізує просту обробку помилок, стан завантаження та динамічне рендерення даних.

Функціональність компонента забезпечує обробку даних, отриманих із бази даних, їх відображення у зручному форматі для користувача, а також підтримує інтерактивні дії, такі як завантаження необхідних матеріалів, подача виконаних робіт і відстеження статусу завдання. Основними з них є:

1. Імпорти: розділювач для інтерфейсу, імпортується з локальних компонентів. LessonPreview: компонент для відображення окремого уроку у списку. React: використовується для створення компонента. useAppSelector: хук для вибірки стану уроків із Redux Store. Підключення CSS-файлу (styles.scss) для оформлення інтерфейсу.

2. Стан уроків. За допомогою useAppSelector отримується частина стану Redux: lessons – список уроків, які потрібно відобразити; isLessonsLoading – стан завантаження уроків (булеве значення); lessonsError – помилка, якщо не вдалося отримати дані про уроки.

3. Логіка компонента. Обробка помилок: Якщо у Redux Store присутня помилка lessonsError, компонент відображає повідомлення «Error». Стан завантаження: Поки isLessonsLoading має значення true, виводиться текст «Loading...». Відображення уроків: Якщо уроки успішно завантажені, компонент використовує метод map, щоб для кожного уроку створити компонент LessonPreview.

Рендеринг інтерфейсу включає відображення заголовка «Lessons», який вказує на вміст сторінки, та використання елемента Divider для створення візуального поділу між заголовком і списком уроків. У контейнері classrooms-page__lessons рендеряться різні елементи залежно від стану завантаження: повідомлення «Loading...» у разі завантаження уроків, повідомлення «Error» при виникненні помилки та компоненти LessonPreview, якщо дані були успішно отримані.

Особливості компонента включають обробку помилок, що забезпечує базову стабільність програми, адже у разі виникнення проблем виводиться відповідне повідомлення. Динамічне завантаження реалізоване через стан isLessonsLoading, що гарантує, що користувач побачить процес завантаження даних. Модульний підхід передбачає, що компонент LessonPreview виділений окремо, що спрощує його масштабування та повторне використання в інших частинах додатку.

CSS-класи, що використовуються на сторінці, включають classrooms-page, який є основним контейнером для всієї сторінки, та classrooms-page__lessons, який містить список уроків.

Потенційні покращення:

– Розширена обробка помилок: додати більше деталей про причину помилки або функцію для повторного завантаження уроків.
– Місцева обробка стану: використовувати локальний стан для тимчасового зберігання завантажених уроків, щоб зменшити навантаження на Redux Store.
– Анімація завантаження: замість тексту «Loading...» реалізувати анімацію для покращення користувацького досвіду.

Компонент ClassroomPage є зручним інструментом для відображення списку уроків. Він реалізує базові функції динамічного рендерингу даних з урахуванням стану завантаження та можливих помилок, забезпечуючи стабільність і гнучкість інтерфейсу (рис. 2.4).

```
import Divider from "components/Divider";
import LessonPreview from "components/LessonPreview";
import React from "react";
import { useAppSelector } from "store";
import "./styles.scss";

function ClassroomPage() {
  const { lessons, isLessonsLoading, lessonsError } =
                useAppSelector((state) => state.lessons);
  if (lessonsError) return <div>Error</div>;
  return (
    <div className="classrooms-page">
      <h2>Lessons</h2>
      <Divider />
      <div className="classrooms-page__lessons">
        {isLessonsLoading
          ? "Loading..."
          : (lessons || []).map((lesson) => <LessonPreview lesson={lesson}
      </div>
    </div>
  );
}

export default ClassroomPage;
```

Рисунок 2.4 – Фрагмент коду сторінки ClassroomPage, який демонструє реалізацію основного функціоналу

Компонент ClassroomSettingsPage реалізує сторінку налаштувань для класної кімнати з використанням вкладок (tabs) для організації даних. Компонент дозволяє перемикатися між загальною інформацією, списками викладачів і студентів, забезпечуючи багаторівневу навігацію.

Основні функціональні елементи:

1. Імпорти
   - EditClassroomForm: Форма для редагування загальної інформації про класну кімнату.
   - Tabs, Tab (Material-UI): Компоненти вкладок для побудови зручної та інтуїтивної навігації.
   - TeachersTab і StudentsTab: Компоненти для відображення списків викладачів та студентів.
2. Компонент TabPanel
   - Використовується для відображення вмісту, пов’язаного з конкретною вкладкою.
   - Параметри: children – вміст вкладки; value – поточний вибір вкладки; index – індекс вкладки, яка має відображатися.
   - Якщо вкладка неактивна (value !== index), компонент приховується за допомогою властивості hidden.
3. Функції для налаштування вкладок
   - a11yProps: Додає стилі вкладкам для покращення їх вигляду.
   - a11yInnerProps: Можна розширити для додавання атрибутів вкладкам внутрішнього рівня.
4. Стан компонента
   - value: Визначає активну головну вкладку («General Information» або «Users»).
   - innerValue: Визначає активну внутрішню вкладку («Teachers» або «Students»).
5. Обробники подій

- handleChange: Оновлює стан value при виборі нової головної вкладки.
- handleInnerChange: Оновлює стан innerValue при перемиканні між внутрішніми вкладками.

Рендеринг сторінки включає головні вкладки, серед яких «General Information» відображає компонент EditClassroomForm, що дозволяє редагувати загальну інформацію про класну кімнату, а вкладка «Users» містить другий набір вкладок для управління списками користувачів. Внутрішні вкладки для «Users» включають «Teachers», де відображається компонент TeachersTab, що представляє список викладачів, та «Students», що містить компонент StudentsTab для відображення списку студентів.

Особливості дизайну сторінки включають багаторівневу організацію вкладок, що дозволяє ефективно групувати пов’язані дані на двох рівнях. Завдяки використанню Material-UI вкладки адаптуються до різних розмірів екрану, забезпечуючи хорошу інтерактивність та зручність для користувачів. Можливі покращення включають додавання обробки помилок, що дозволить відображати повідомлення у разі проблем із завантаженням даних, логічне кешування для збереження стану вибраних вкладок під час перемикання між іншими сторінками, а також додавання анімації переходів для покращення користувацького досвіду при перемиканні між вкладками (рис. 2.5).

```
import EditClassroomForm from "components/forms/EditClassroomForm";
import React from "react";
import Tabs from "@mui/material/Tabs";
import Tab from "@mui/material/Tab";
import TeachersTab from "./components/TeachersTab";
import StudentsTab from "./components/StudentsTab";

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

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

  return (
    <div role="tabpanel" hidden={value !== index} {...other}>
      {value === index && children}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    style: {
      width: "50vw",
      maxWidth: "unset",
    },
  };
}
function a11yInnerProps(index: number) {
  return {};
}

function ClassroomSettingsPage() {
  const [value, setValue] = React.useState(0);
  const [innerValue, setInnerValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) =>
    setValue(newValue);
  };
```

Рисунок 2.5 – Фрагмент коду сторінки ClassroomSettingsPage, який демонструє реалізацію управління налаштуваннями класної кімнати

Компонент ClassroomSettingsPage надає зручний і багаторівневий інтерфейс для управління налаштуваннями класної кімнати. Завдяки

використанню вкладок користувачі можуть легко перемикатися між різними розділами, такими як загальна інформація, викладачі та студенти, що робить взаємодію з інтерфейсом інтуїтивною та ефективною.

Розглянемо компонент React, який реалізує сторінку для перегляду, редагування та управління домашніми завданнями студентів у контексті класу. Він використовує таблицю даних для відображення студентів та інформації про виконання домашніх завдань, дозволяє змінювати оцінки, переглядати завдання та виконувати інші дії через контекстне меню.

Основні елементи компонента включають імпорти з Material-UI, такі як Avatar, Box, Menu, MenuItem, Typography, MoreVertIcon для побудови інтерфейсу, а також DataGrid з @mui/x-data-grid для відображення даних у вигляді інтерактивної таблиці. Для роботи з маршрутизацією використовуються useNavigate та useParams з React Router. Redux забезпечує роботу зі станом додатка через useAppSelector та useAppDispatch, а також включає дії, як-от editHomework та declineHomework для редагування та відхилення домашніх завдань. Для отримання списку студентів використовується утиліта getUsers.

Функціональність компонента

1. Колонки таблиці (DataGrid), columns визначає структуру таблиці.

Основні стовпці:

- Avatar: відображає аватар студента.
- Name: ім’я студента.
- Email: електронна пошта студента.
- Mark: оцінка домашнього завдання, доступна для редагування.
- Status: стан виконання завдання.

Меню дій: кнопка з іконкою MoreVertIcon, яка відкриває контекстне меню для кожного студента.

2. Контекстне меню

- Реалізоване через Material-UI Menu.
- Доступні дії: «Open homework»: переходить до сторінки конкретного домашнього завдання студента.

- «Decline homework»: відхиляє виконане завдання, якщо його ще не було оцінено.

3. Отримання даних
   - Список студентів: отримується через функцію getUsers, яка використовує ідентифікатори студентів, збережені в Redux Store.
   - Домашні завдання: беруться зі стану Redux (homeworks) та об'єднуються зі списком студентів для створення об’єднаних рядків таблиці.
4. Редагування оцінок
   - onEditDone дозволяє змінювати оцінки безпосередньо у таблиці.
   - Перевіряється, чи завдання виконано (isDone), перш ніж дозволити зміну оцінки.
   - Викликається дія Redux editHomework для оновлення даних у Store.
5. Рендеринг
   - Таблиця даних: виводить об'єднаний список студентів та їхніх домашніх завдань через DataGrid.
   - Підтримує редагування комірок (оцінок) та інтерактивність через меню.
   - Box використовується для задання розмірів таблиці.

Особливості компонента:

- Динамічне злиття даних: список студентів і домашніх завдань об'єднується у єдину структуру перед відображенням у таблиці.
- Контекстне меню: меню дозволяє виконувати специфічні дії, такі як перегляд або відхилення домашнього завдання.
- Редагування оцінок: поле Mark у таблиці дозволяє змінювати оцінки студентів без переходу на інші сторінки.

```
function HomeworksPage() {
  const { organizationId, classroomId, lessonId } = useParams();
  const [students, setStudents] = React.useState([]);
  const dispatch = useAppDispatch();
  const homeworks = useAppSelector((state) => state.homework.homeworks);
  const classrooms = useAppSelector((state) => state.classrooms.classrooms);
  const classroom = classrooms?.find((org) => org.id === classroomId);
  const studentsIds = classroom?.students;

  React.useEffect(() => {
    getUsers(studentsIds).then(setStudents);
  }, [studentsIds]);

  const onEditDone = ({ field, id, value }) => {
    const homework = homeworks?.find((x) => x.id === id);
    if (!homework?.isDone) {
      alert("You can't mark a not finished homework");
      return;
    }
    dispatch(
      editHomework({
        organizationId,
        classroomId,
        lessonId,
        userId: id,
        homework: { ...homework, [field]: value },
      })
    );
  };
```

Рисунок 2.6 – Фрагмент коду сторінки HomeworksPage

Компонент HomeworksPage створює функціональну сторінку для управління домашніми завданнями студентів. Завдяки використанню таблиці даних, інтеграції з Redux та контекстного меню, компонент забезпечує зручність і ефективність у виконанні основних завдань, таких як редагування оцінок, перегляд і відхилення домашніх завдань.

Компонент OrganizationClassroomsPage відображає сторінку зі списком класів, що належать до конкретної організації. Цей компонент інтегрується з Redux для отримання даних про класи та організації, а також використовує елементи інтерфейсу для покращення візуалізації.

Основні елементи компонента включають ClassroomCard, який використовується для відображення окремого класу у вигляді картки, а також Divider, що служить візуальним розділювачем для структуризації інтерфейсу. Хоча SearchBar в коді не використовується, він може бути доданий для пошуку класів. Для отримання параметрів URL, зокрема organizationId, використовується хук useParams з React Router, а для доступу до стану Redux Store застосовується хук useAppSelector, через який отримуються дані про класи та організації. Компонент використовує стан Redux, зокрема classrooms – список класів організації, isClassroomsLoading – стан завантаження класів, classroomsError – помилки, що виникають при отриманні класів, та organizations – список організацій, що дозволяє знайти поточну організацію за organizationId.

ункціональність компонента полягає в наступному. Спочатку useParams витягує organizationId з URL, а метод organizations.find шукає відповідну організацію в стані Redux, щоб визначити, яка організація повинна відображатися. Якщо у Redux Store є помилка (наприклад, classroomsError), компонент повертає елемент із повідомленням «Error». Під час завантаження даних (коли isClassroomsLoading дорівнює true), компонент відображає текст «Loading...». Після завершення завантаження, метод map генерує список компонентів ClassroomCard, кожен з яких представляє окремий клас, передаючи дані про клас через пропс classroom.

Відображення інтерфейсу компонента включає кілька ключових елементів. Спочатку відображається заголовок сторінки, який містить назву організації (організація отримується через organization?.name) і слово «classrooms». Нижче заголовка знаходиться розділювач Divider, який візуально розділяє заголовок та список класів. Якщо класи доступні, вони відображаються у вигляді карток за допомогою компонента ClassroomCard, кожна з яких представляє окремий клас.

CSS-класи:

- org-classrooms-page: Основний контейнер для сторінки;
- org-classrooms-page__header: Контейнер для заголовка сторінки.

Можливі покращення:

1. Пошук і фільтрація: Реалізувати пошук і сортування класів за допомогою SearchBar.
2. Додатковий стан помилки: Розширити обробку помилок, наприклад, додати кнопку "Спробувати знову".
3. Скелетон-екран: Додати анімацію завантаження (skeleton screen) замість тексту "Loading...".
4. Пагінація: Реалізувати підтримку великих списків класів із розбивкою на сторінки.

```
function OrganizationClassroomsPage() {
  const { classrooms, isClassroomsLoading, classroomsError } = useAppSelector(
    (state) => state.classrooms
  );
  const organizations = useAppSelector((state) => state.organizations.organizations);
  const { organizationId } = useParams();

  const organization = organizations?.find((org) => org.id === organizationId);

  if (classroomsError) return <div>Error</div>;

  return (
    <div className="org-classrooms-page">
      <div className="org-classrooms-page__header">
        <h2>{organization?.name} classrooms</h2>
      </div>
      <Divider />
      {isClassroomsLoading
        ? "Loading..."
        : (classrooms || []).map((classroom) => <ClassroomCard classroom={classroom} />)}
    </div>
  );
}
```

Рисунок 2.7 – Фрагмент коду сторінки OrganizationClassroomsPage

Компонент OrganizationClassroomsPage є простим і функціональним інструментом для відображення списку класів організації. Він ефективно використовує дані з Redux Store і дозволяє користувачеві зручно переглядати класи, пов’язані з вибраною організацією.

Діаграма класів представляє структуру ключових об’єктів системи навчальної платформи (рис. 2.8). Вона показує класи, їхні атрибути, методи, а також взаємозв’язки між ними [10].

```
@startuml
class ClassroomPage {
  + lessons: Lesson[]
  + isLessonsLoading: boolean
  + lessonsError: string
  + render(): void
  + handleLoading(): void
  + handleError(): void
}

class ClassroomSettingsPage {
  + value: number
  + innerValue: number
  + render(): void
  + handleChange(event): void
  + handleInnerChange(event): void
  + editClassroomDetails(): void
}

class HomeworkPage {
  + homework: Homework
  + students: Student[]
  + render(): void
  + onMarkSave(): void
  + onDecline(): void
  + downloadFile(url): void
  + openHomeworkDetails(studentId: string): void
}

class LessonPage {
  + lesson: Lesson
  + isLoading: boolean
  + render(): void
  + fetchLessonDetails(): void
  + updateLesson(): void
  + handleLoading(): void
}

class OrganizationClassroomsPage {
  + classrooms: Classroom[]
  + organizationId: string
  + render(): void
  + fetchClassrooms(): void
  + addClassroom(): void
}

class OrganizationsPage {
  + organizations: Organization[]
  + isLoading: boolean
  + render(): void
  + fetchOrganizations(): void
```

```
  + addOrganization(): void
  + handleLoading(): void
}

class LoginPage {
  + username: string
  + password: string
  + render(): void
  + handleLogin(): void
  + handleInputChange(): void
  + validateCredentials(): boolean
}

ClassroomPage --> LessonPage
ClassroomSettingsPage --> ClassroomPage
HomeworkPage --> LessonPage
OrganizationClassroomsPage --> ClassroomPage
OrganizationsPage --> OrganizationClassroomsPage
LoginPage --> OrganizationsPage

@enduml
```

Клас ClassroomPage відповідає за відображення сторінки класної кімнати та включає ключові атрибути й методи для забезпечення функціональності. Основні атрибути: lessons – список уроків, isLessonsLoading – стан завантаження уроків, та lessonsError – помилки, пов'язані з уроками. Функціонал класу забезпечують методи: render() для виведення інтерфейсу сторінки, handleLoading() для обробки стану завантаження, та handleError() для обробки можливих помилок, що забезпечує стабільну роботу компонента.

Клас ClassroomSettingsPage використовується для управління налаштуваннями класної кімнати, надаючи користувачеві можливість змінювати параметри через систему вкладок. Серед атрибутів класу є value, який визначає активну вкладку, і innerValue, що відповідає за вкладки внутрішнього рівня. Основні методи включають render() для відображення сторінки налаштувань, handleChange(event) для обробки змін активної вкладки, handleInnerChange(event) для управління внутрішніми вкладками, та editClassroomDetails(), який забезпечує функціонал редагування деталей класної кімнати, спрощуючи процес налаштування для користувача.

Клас HomeworkPage забезпечує управління домашніми завданнями, надаючи функціонал для роботи з поточними завданнями та студентами, які їх виконують. Основними атрибутами є homework, що зберігає інформацію про поточне завдання, і students, який містить список студентів, прив'язаних до завдання. Методи класу включають: render() для відображення інтерфейсу сторінки домашніх завдань, onMarkSave() для збереження оцінки, onDecline() для відхилення завдання, downloadFile(url) для завантаження прикріпленого файлу, та openHomeworkDetails(studentId) для відкриття деталей завдання, пов'язаного з конкретним студентом, що забезпечує ефективну взаємодію викладача із завданнями.

Клас LessonPage забезпечує управління сторінкою уроку, дозволяючи працювати з даними уроку та обробляти стан завантаження. Основними атрибутами є lesson, який містить інформацію про поточний урок, та isLoading, що відображає стан завантаження сторінки. Методи включають: render() для відображення інтерфейсу сторінки уроку, fetchLessonDetails() для отримання деталей уроку, updateLesson() для оновлення даних та handleLoading() для управління станом завантаження.

Клас OrganizationClassroomsPage дозволяє керувати сторінкою класів, пов'язаних із певною організацією. Основними атрибутами є classrooms, який містить список класів, та organizationId, що визначає ідентифікатор організації. Функціональність забезпечується методами: render() для виведення інтерфейсу сторінки, fetchClassrooms() для отримання списку класів, та addClassroom() для додавання нових класів, що спрощує адміністрування.

Клас OrganizationsPage забезпечує управління списком організацій, дозволяючи переглядати, отримувати або додавати нові організації. Основними атрибутами є organizations, що містить список усіх доступних організацій, та isLoading, який відображає стан завантаження сторінки. Для реалізації функціональності використовуються методи: render() для відображення інтерфейсу сторінки організацій, fetchOrganizations() для отримання списку організацій, addOrganization() для створення нових організацій та

handleLoading() для управління станом завантаження, що покращує стабільність програми.

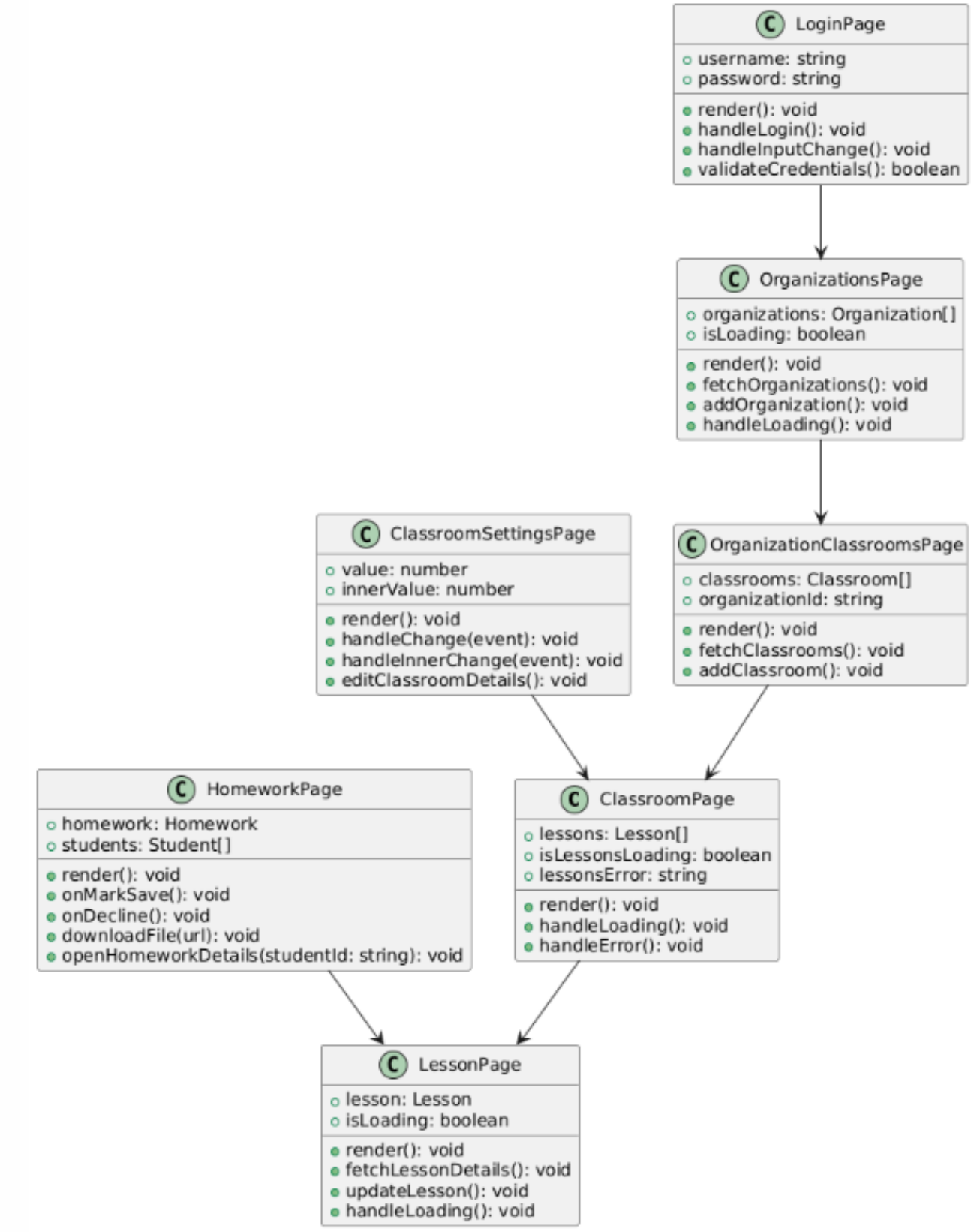

Рисунок 2.8 – UML-діаграма класів

Клас LoginPage призначений для автентифікації користувачів, надаючи інтерфейс для входу в систему. До атрибутів належать username, який зберігає ім'я користувача, та password, що відповідає за введений пароль. Основні методи включають: render() для відображення сторінки входу, handleLogin() для перевірки введених даних та виконання автентифікації, handleInputChange() для оновлення стану полів вводу та validateCredentials() для перевірки коректності введених даних, забезпечуючи безпечний доступ до системи.

Зв’язки між класами відображають взаємодію компонентів системи. ClassroomPage взаємодіє з LessonPage, оскільки використовує уроки для відображення їхнього списку. ClassroomSettingsPage залежить від ClassroomPage, дозволяючи змінювати налаштування відповідної сторінки класу. HomeworkPage прив’язана до LessonPage, оскільки домашні завдання пов’язані з конкретними уроками. OrganizationClassroomsPage оперує списком класів через ClassroomPage, тоді як OrganizationsPage працює зі списком організацій, кожна з яких має свої класи через OrganizationClassroomsPage. Нарешті, LoginPage після автентифікації спрямовує користувача на OrganizationsPage, забезпечуючи доступ до відповідних організацій.

Загальні особливості системи включають модульність, що дозволяє кожному класу відповідати за окрему частину функціоналу платформи, забезпечуючи зрозумілу структуру та легкість підтримки. Взаємозалежності класів організовані у багаторівневу систему, де класи взаємопов’язані через логіку сторінок, наприклад, Classroom → Lesson → Homework, що сприяє зручності навігації та організації даних. Завдяки розширюваності структури, нові функції можна додавати до кожного класу без суттєвого впливу на інші класи, забезпечуючи адаптивність та масштабованість платформи.

На рис. 2.9 зображено діаграму послідовності, яка демонструє взаємодію між користувачем та основними компонентами системи веб-платформи для онлайн-навчання і спільної роботи з використанням синтаксису PlantUML [19].

Вона відображає потік дій, які виконує користувач, переходячи між сторінками, і як ці сторінки взаємодіють між собою, дозволяє зрозуміти архітектуру системи та зв’язки між її ключовими компонентами.

```
@startuml

actor Користувач as User
participant "LoginPage" as LoginPage
participant "OrganizationsPage" as OrganizationsPage
participant "OrganizationClassroomsPage" as ClassroomsPage
participant "ClassroomPage" as ClassroomPage
participant "ClassroomSettingsPage" as ClassroomSettingsPage
participant "LessonPage" as LessonPage
participant "HomeworkPage" as HomeworkPage

User -> LoginPage: Вхід у систему
LoginPage -> User: Підтвердження успішного входу

User -> OrganizationsPage: Перехід на сторінку організацій
OrganizationsPage -> User: Відображення списку організацій

User -> ClassroomsPage: Вибір організації
ClassroomsPage -> User: Відображення списку класів

User -> ClassroomPage: Вибір конкретного класу
ClassroomPage -> User: Інформація про клас
User -> ClassroomSettingsPage: Перехід до налаштувань класу
ClassroomSettingsPage -> User: Інтерфейс для налаштувань

User -> LessonPage: Вибір уроку в класі
LessonPage -> User: Деталі уроку

User -> HomeworkPage: Перегляд домашніх завдань
HomeworkPage -> User: Список завдань для студентів

User -> HomeworkPage: Створення нового домашнього завдання
HomeworkPage -> HomeworkPage: Збереження даних про завдання
HomeworkPage -> User: Підтвердження створення завдання

User -> HomeworkPage: Редагування домашнього завдання
HomeworkPage -> HomeworkPage: Оновлення інформації про завдання
HomeworkPage -> User: Підтвердження редагування

User -> HomeworkPage: Відхилення завдання
HomeworkPage -> HomeworkPage: Оновлення статусу завдання
HomeworkPage -> User: Підтвердження відхилення

User -> ClassroomPage: Повернення до списку класів
ClassroomPage -> User: Оновлений список уроків
```

```
User -> OrganizationsPage: Перехід до іншої організації
OrganizationsPage -> User: Оновлений список класів

@enduml
```

Учасники:

1. Користувач (User) – головний актор, який виконує дії в системі, такі як вхід, вибір класів, робота з уроками та домашніми завданнями.
2. LoginPage – сторінка для автентифікації користувача.
3. OrganizationsPage – сторінка для перегляду списку організацій.
4. OrganizationClassroomsPage – сторінка, яка показує класи вибраної організації.
5. ClassroomPage – сторінка з інформацією про конкретний клас.
6. ClassroomSettingsPage – сторінка для редагування налаштувань класу.
7. LessonPage – сторінка для перегляду деталей уроку.
8. HomeworkPage – сторінка для перегляду і редагування домашніх завдань.

Основний сценарій роботи користувача починається зі входу до системи через LoginPage, де здійснюється автентифікація, після чого у разі успіху користувач перенаправляється на OrganizationsPage, щоб переглянути доступні організації. Далі, вибравши конкретну організацію, він переходить до OrganizationClassroomsPage, де отримує список класів цієї організації. Вибір класу веде користувача на ClassroomPage, де він може переглянути загальну інформацію про клас, або на ClassroomSettingsPage для зміни його налаштувань. Переходячи до уроків через LessonPage, користувач має змогу переглядати їхні деталі. Крім того, на HomeworkPage доступні функції для роботи з домашніми завданнями: перегляд списку, створення нових завдань, редагування існуючих або зміна їхнього статусу.

Характеристики діаграми демонструють основні аспекти роботи системи. Взаємодія між компонентами відображає, як кожна дія користувача активує відповідний метод, що забезпечує отримання даних або виконання певних функцій. Лінійна послідовність показує логічний порядок дій, починаючи з

автентифікації і закінчуючи управлінням домашніми завданнями. Особлива увага приділяється користувацькому досвіду: дії, як-от вибір організації, класу, уроку чи завдання, забезпечують інтуїтивність та зручність для користувача.

Діаграма розроблена з використанням синтаксису PlantUML [10] та відображає ключові етапи роботи користувача із системою. Зокрема, послідовність починається з автентифікації користувача, після чого демонструється отримання списку організацій та вибір конкретного класу. Далі показано взаємодію з компонентами, відповідальними за управління курсами та домашніми завданнями. Такий підхід дозволяє чітко відобразити логіку роботи платформи та забезпечує розуміння взаємодії між її модулями.

Діаграма дозволяє візуалізувати основні етапи роботи системи, підтримує аналіз ідеального сценарію користування та стає основою для покращень або розширення функціоналу.

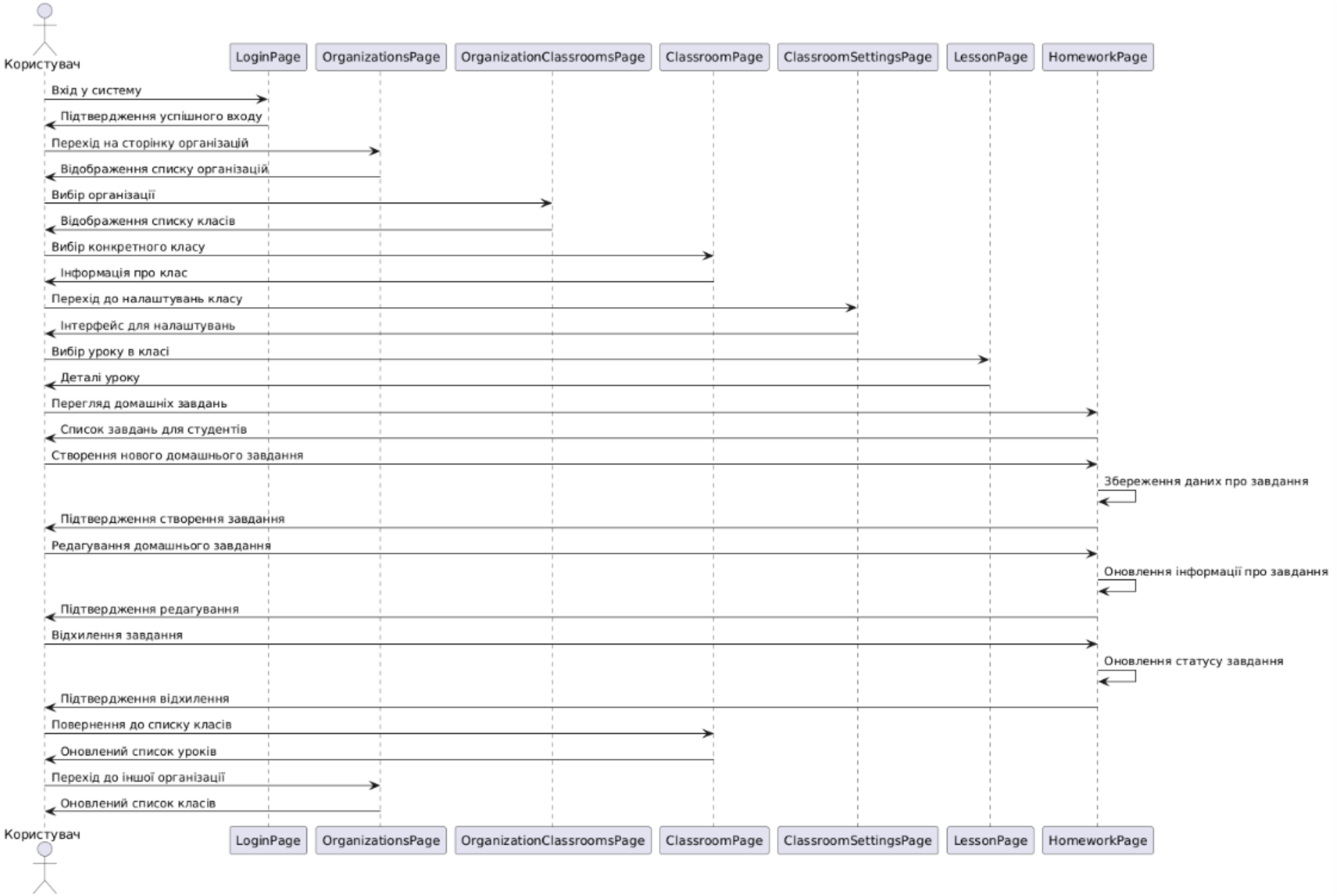

Рисунок 2.9 – UML- діаграма послідовності

Висновки до розділу:

У цьому розділі було розглянуто ключові аспекти розробки web-платформи для онлайн-навчання та спільної роботи.

Перш за все, було здійснено вибір та обґрунтування технологій, необхідних для реалізації проєкту. Обрані технології забезпечують високу продуктивність, гнучкість, інтерактивність і зручність використання веб-додатку. Використання сучасних фреймворків та інструментів дозволило створити ефективну систему, яка відповідає вимогам користувачів.

Далі проведено проєктування структури бази даних інформаційної системи. Це включало визначення основних сутностей, їхніх атрибутів та зв’язків між ними. Спроєктована база даних забезпечує швидкий доступ до даних, цілісність інформації та підтримує масштабованість для розширення функціоналу в майбутньому.

Окрім того, у межах розробки було реалізовано основний функціонал веб-додатку. Створено зручний інтерфейс користувача, який дозволяє виконувати ключові операції, такі як управління курсами, уроками, завданнями та користувачами. Функціонал було протестовано та оптимізовано для досягнення максимальної зручності використання.

Таким чином, розглянуті пункти демонструють завершеність етапів вибору технологій, проєктування бази даних та реалізації основного функціоналу, що є необхідними для успішної реалізації web-платформи для онлайн-навчання та спільної роботи. Отримані результати створюють надійну основу для подальшого вдосконалення платформи та інтеграції нових можливостей.

РОЗДІЛ 3
ПРАКТИЧНА АПРОБАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АНАЛІЗУ РЕЗУЛЬТАТІВ РОБОТИ WEB-ПЛАТФОРМИ ДЛЯ ОНЛАЙН-НАВЧАННЯ ТА СПІЛЬНОЇ РОБОТИ

3.1 Практична апробація та опис методики використання розробленої панелі реєстрації для аналізу результатів web-платформи для онлайн-навчання

З метою продемонструвати функціональність розробленої web-платформи для онлайн-навчання та спільної роботи, розглянемо послідовність операцій та завдань, які можна автоматизувати за допомогою цього проєкту.

По-перше, адміністратор платформи повинен пройти процес реєстрації в системі. Для цього він здійснює реєстрацію та вхід через особистий кабінет користувача на веб-сайті. Щоб увійти в систему, на головній сторінці платформи натисніть кнопку «Locaut with Google account» (рис. 3.1).

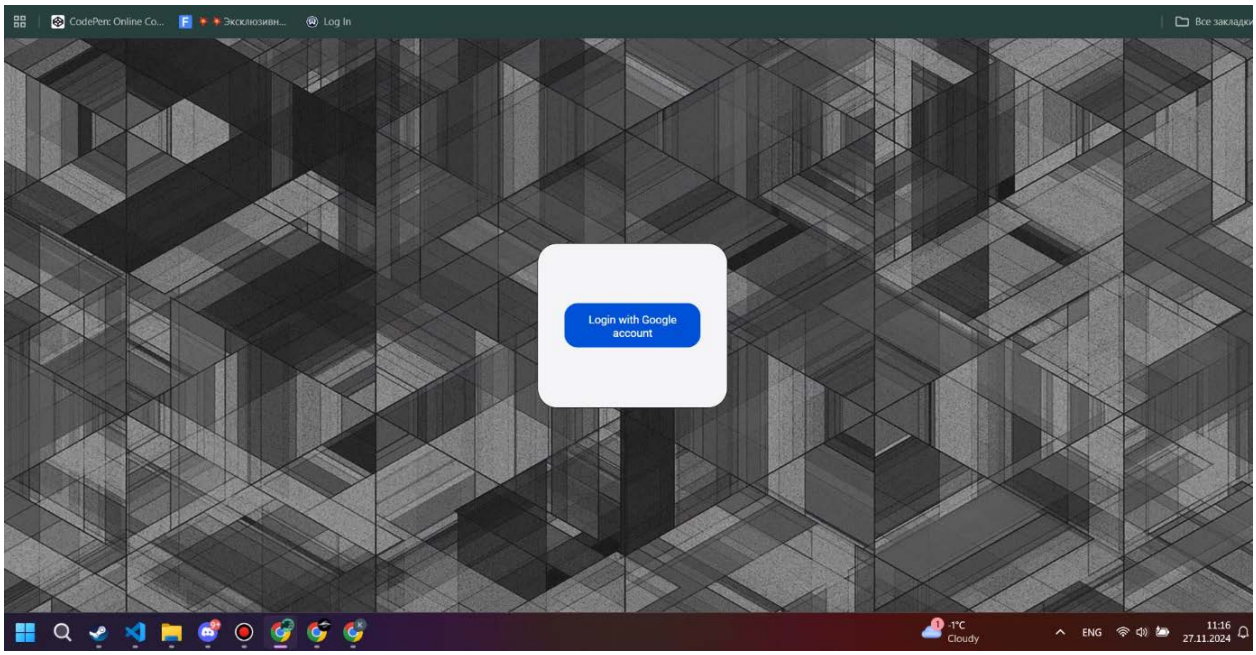

Рисунок 3.1 – Вхід у систему особистого кабінету адміністратора

Платформа підтримує зручний вхід за допомогою облікового запису Google. Система перенаправить вас на сторінку автентифікації Google, де потрібно обрати свій обліковий запис або ввести облікові дані. Після успішної авторизації ви будете автоматично зареєстровані або ввійдете в систему (якщо вже зареєстровані). Цей спосіб входу забезпечує швидкість і безпеку доступу до платформи (рис. 3.2).

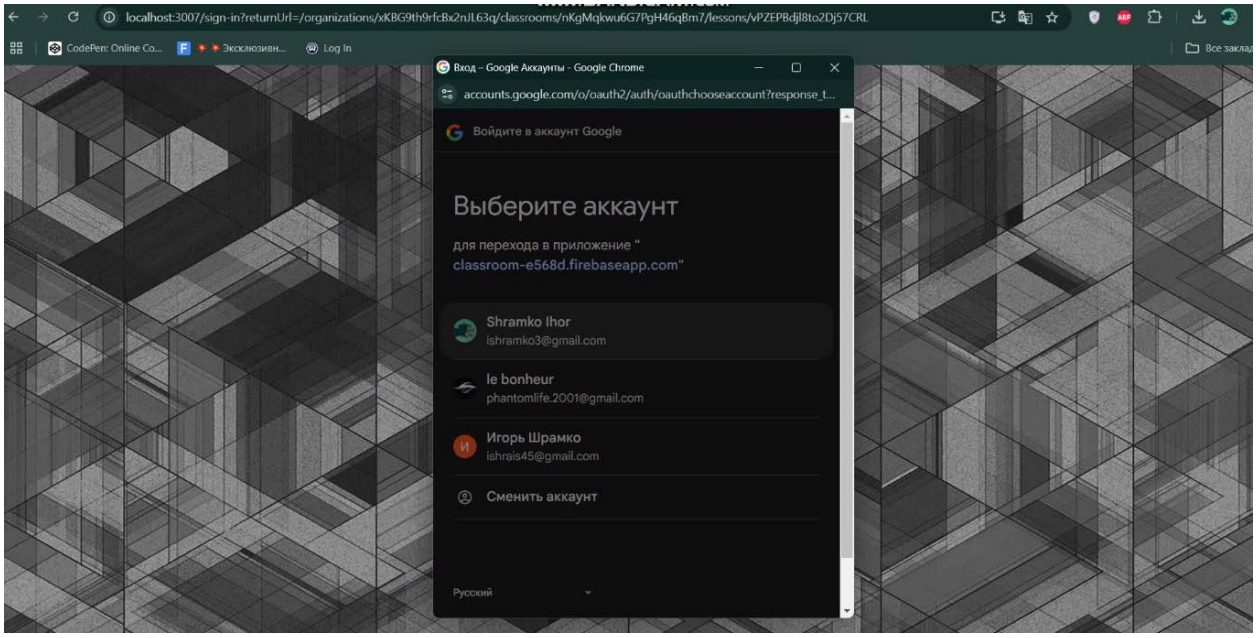

Рисунок 3.2 – Вікно реєстрації в системі сторінку через автентифікацію Google

Після входу у систему користувач потрапляє на головну сторінку профілю. У верхньому меню або на бічній панелі можна швидко переходити між доступними функціями. Кожен розділ забезпечує зручний доступ до пов’язаних даних і функцій, таких як перегляд уроків, управління курсами чи робота із завданнями. Інтуїтивний інтерфейс дозволяє легко орієнтуватися навіть новачкам (рис. 3.3).

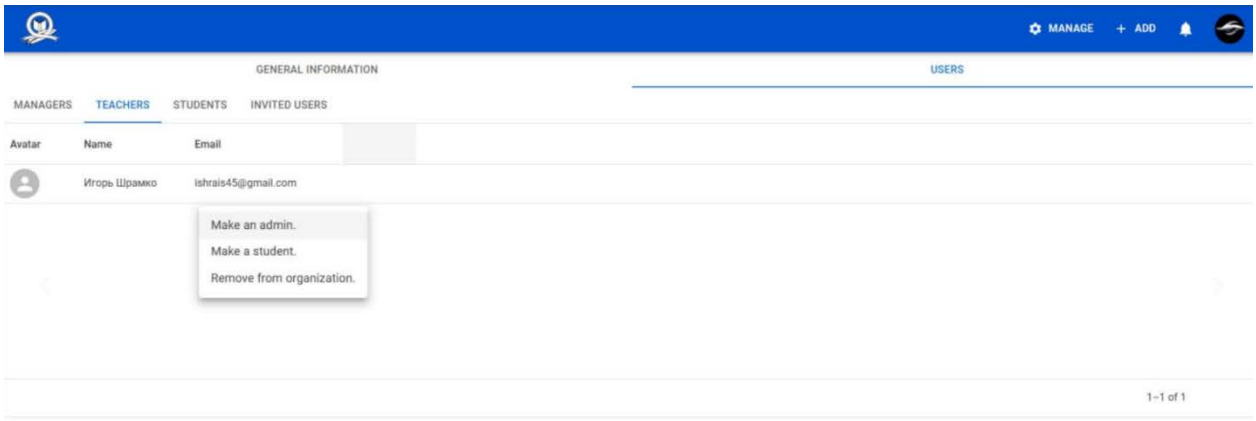

Рисунок 3.3 – Панель профілю адміністратора

Викладачі на платформі мають можливість створювати класи для організації навчального процесу. Для цього потрібно перейти до розділу та заповнити форму, зазначивши назву, опис і категорію класу. За бажанням можна прикріпити зображення або логотип для візуального оформлення. Після збереження створений клас додається до списку, і викладач може додавати до нього уроки, завдання та запрошувати студентів через унікальний код доступу. Це дозволяє легко структурувати навчальний процес і забезпечує зручність у взаємодії зі студентами.

Рисунок 3.4 – Створення класу для організації навчального процесу

Викладач може легко додавати навчальні матеріали, щоб забезпечити студентів актуальним контентом. Необхідно перейти на сторінку курсу, до

якого хочете додати матеріали, і оберіть урок, де буде розміщено контент. Для завантаження файлів натисніть кнопку «Attach files», оберіть необхідні документи (наприклад, PDF, презентації або відео) з вашого комп’ютера і підтвердіть вибір (рис. 3.5). Після збереження матеріал автоматично з’явиться у списку доступних для перегляду студентами. Вони зможуть переглядати матеріали або завантажувати файли залежно від налаштувань доступу. Ця функція дозволяє викладачам забезпечувати студентів актуальними навчальними ресурсами, організувати навчання ефективніше та швидко оновлювати матеріали в разі потреби (рис. 3.6).

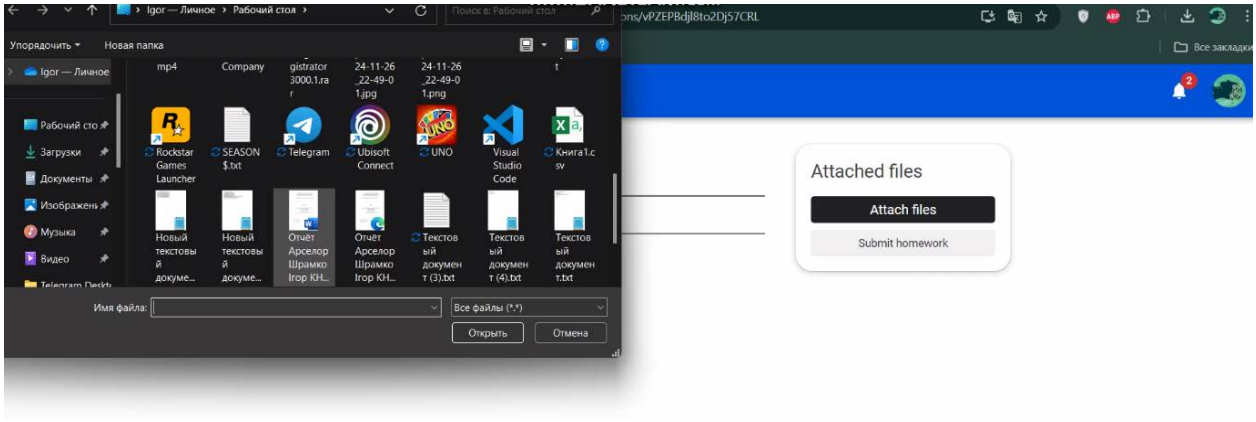

Рисунок 3.5 – Завантаження навчального матеріалу

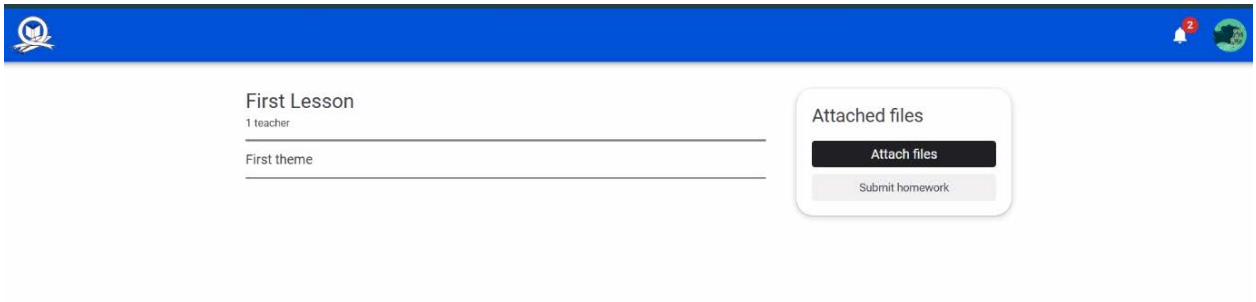

Рисунок 3.6 – Додавання навчальних матеріалів

Процес додавання лекційних матеріалів у систему відображено на рис. 3.7. Він включає вибір файлів через інтерфейс, їх завантаження на платформу для зберігання, а також можливість прив'язки до конкретних лекцій курсу. Інтерфейс забезпечує простоту використання, дозволяючи легко додавати

матеріали, що сприяє кращій організації навчального контенту для студентів та викладачів.

Рисунок 3.7 – Додавання лекційних матеріалів

Викладачі на платформі мають зручний інструмент для оцінювання роботи студентів. У розділі «Домашні завдання» або на сторінці конкретного уроку відображається список завдань, поданих студентами. Щоб виставити оцінку:

1. Виберіть завдання, яке потребує оцінювання.
2. Перегляньте прикріплені файли або текстові відповіді студента, використовуючи вбудований переглядач.
3. У полі для оцінки вкажіть числовий результат, наприклад, за шкалою від 0 до 100, залежно від налаштувань курсу.
4. Натисніть кнопку «Save», щоб виставити оцінку.

Оцінка одразу стає доступною студенту. Ця функція допомагає підтримувати прозорість навчального процесу, мотивувати студентів до покращення результатів і забезпечувати конструктивний зворотний зв’язок (рис. 3.8).

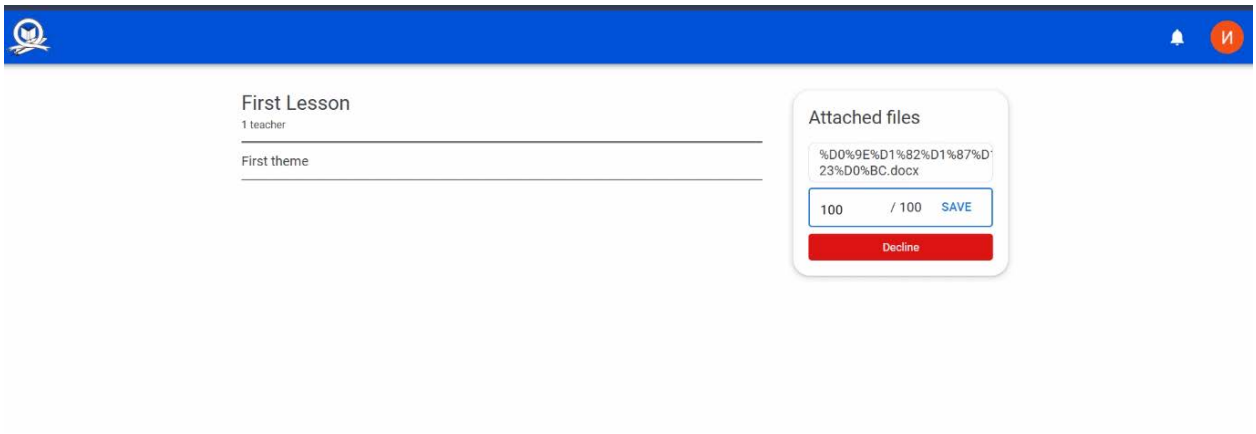

Рисунок 3.8 – Додавання лекційних матеріалів

Рисунок 3.9, що представляє назви навчальних курсів розробленої web-платформи для онлайн-навчання, зображує інтерфейс для перегляду курсів, створених викладачами.

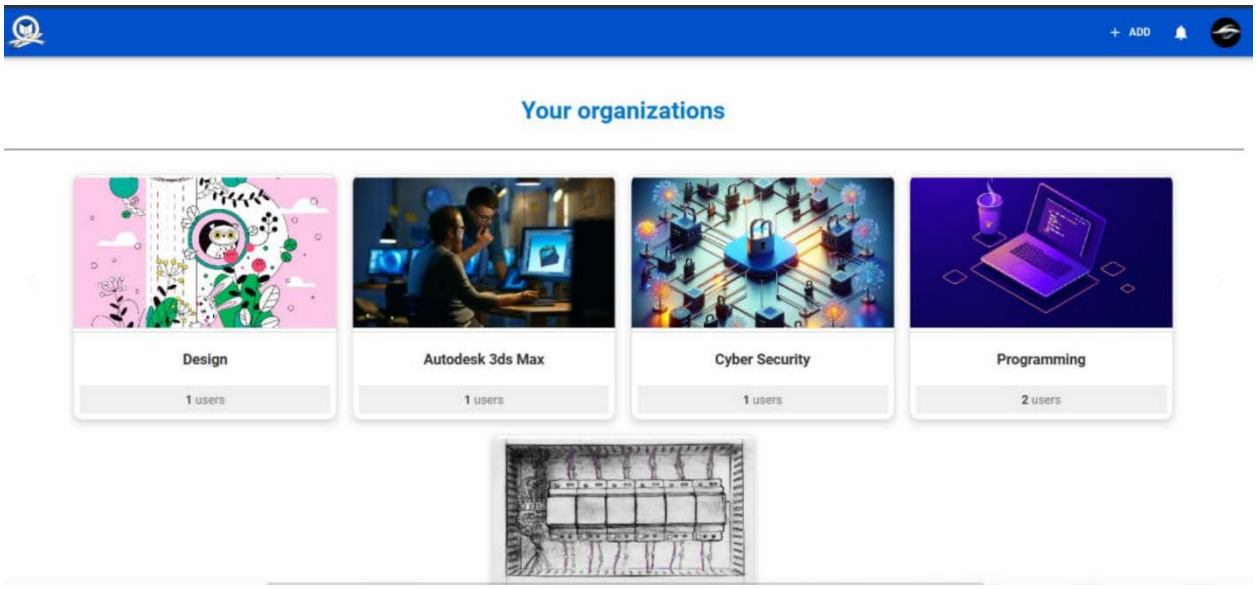

Рисунок 3.9 – Назви навчальних курсів розробленої web-платформи

Студенти можуть легко підключатися до класів і завантажувати виконані роботи через платформу (рис. 3.10). Після успішного приєднання студент отримує доступ до уроків, завдань і матеріалів цього класу. Для завантаження виконаних робіт (рис. 3.11) необхідно вибрати завдання, додати файл (наприклад, документ, презентацію або зображення).

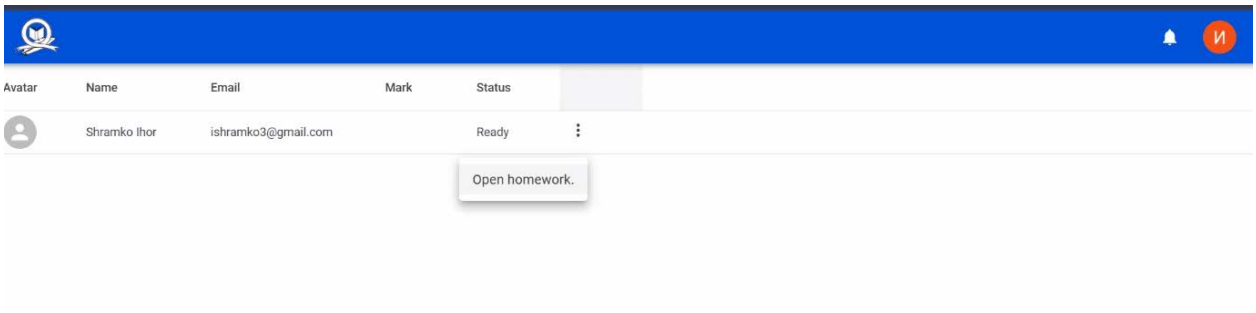

Рисунок 3.10 – Відкриття матеріалів для студента

Рисунок 3.11 – Завантаження робіт студентом

Завантажені роботи автоматично стають доступними для перевірки викладачем, а студент отримує можливість переглядати виставлені оцінки та коментарі до своїх робіт (рис. 3.12).

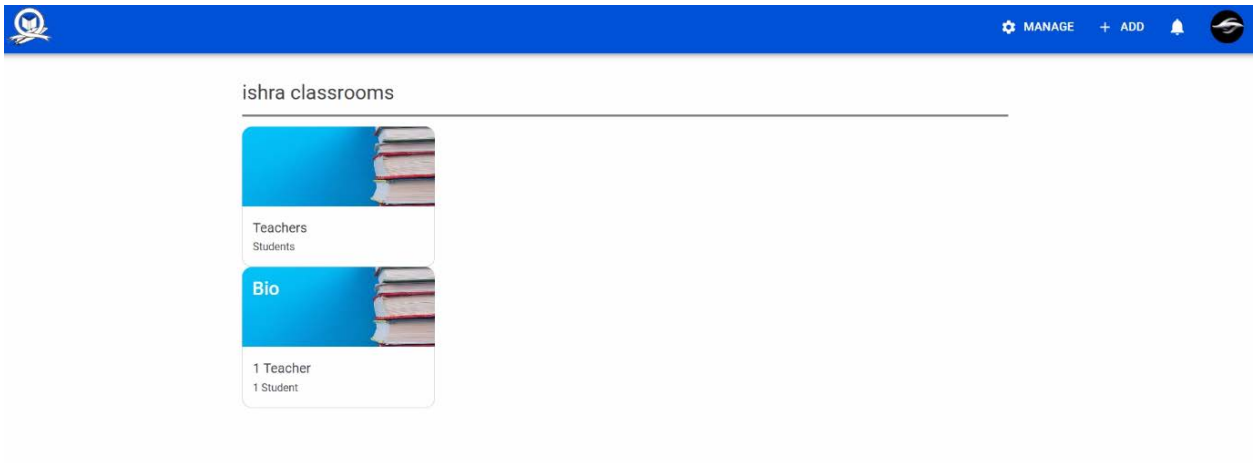

Рисунок 3.12 – Доступні класи

Платформа підтримує три основних ролі: менеджер, викладач і студент, кожна з яких має свій набір функцій і можливостей.

Менеджер відповідає за адміністративне управління платформою. У цьому режимі доступні функції створення та управління організаціями, призначення викладачів до класів, моніторинг активності користувачів і налаштування загальних параметрів системи (рис. 3.13).

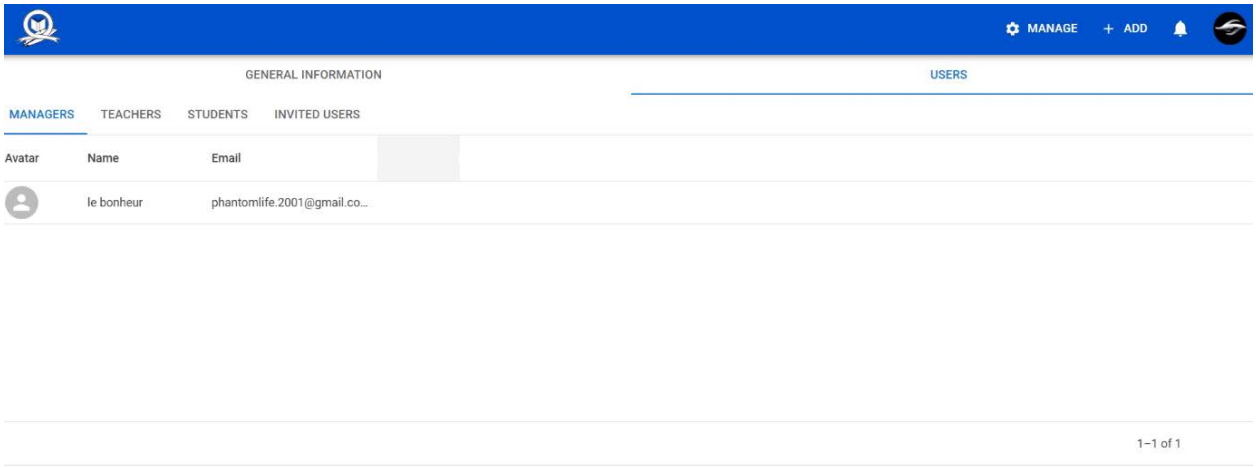

Рисунок 3.13 – Режим менеджера

Викладач працює з курсами, уроками та студентами. У цьому режимі доступні функції створення і налаштування класів, додавання навчальних матеріалів, завдань. Викладач також оцінює роботи студентів, залишає коментарі та спілкується зі студентами через обговорення. Крім того, викладачі можуть переглядати успішність студентів та створювати аналітичні звіти про прогрес у навчанні (рис. 3.14).

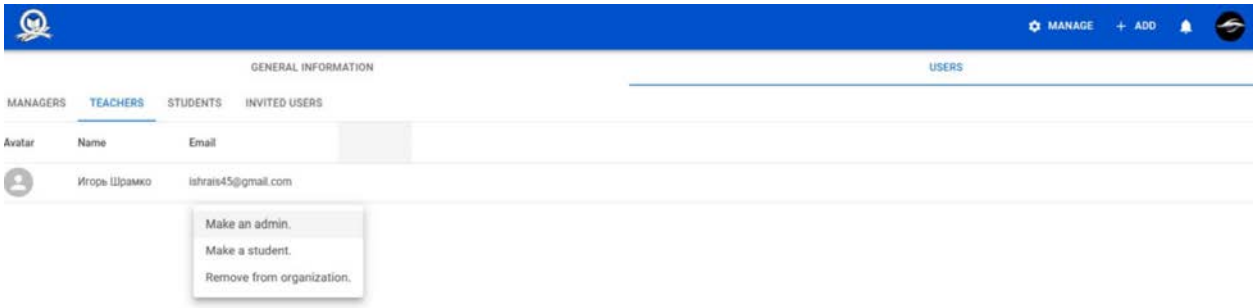

Рисунок 3.14 – Режим викладача

Студент зосереджений на доступі до навчальних матеріалів і виконанні завдань. У цьому режимі можна приєднуватися до класів за кодом, переглядати уроки, завантажувати виконані завдання та отримувати зворотний зв’язок від викладачів. Студенти також можуть брати участь в обговореннях, переглядати свої оцінки та стежити за власним прогресом у навчанні (рис. 3.15)

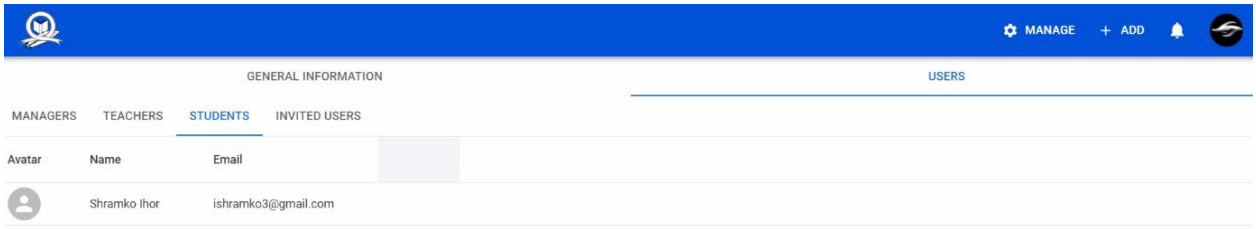

Рисунок 3.15 – Режим студента

Кожна роль забезпечує чіткий розподіл функціональності, що дозволяє оптимізувати процеси адміністрування, викладання та навчання на платформі.

Висновки до розділу:

У даному розділі випускної роботи було описано ключові етапи взаємодії з розробленою web-платформою для онлайн-навчання та спільної роботи. Інтерфейс системи був спроектований таким чином, щоб забезпечити зручність користування як для студентів, так і для викладачів та адміністраторів. Користувачі отримують доступ до всіх необхідних функцій через інтуїтивно зрозумілий інтерфейс, що включає реєстрацію, авторизацію, перегляд та виконання домашніх завдань.

Інструкція передбачає чітке пояснення кожного етапу взаємодії з платформою, починаючи від реєстрації на платформі до роботи з навчальними матеріалами та результатами. Окремо детально розглянуто інструменти для управління завданнями, завантаження матеріалів та організації взаємодії в команді для спільної роботи. Завдяки цим інструкціям користувачі можуть

швидко освоїти систему, що забезпечує ефективне навчання та спільну роботу без додаткових труднощів.

Таким чином, інструкція є важливим компонентом для забезпечення користувацької підтримки, що дозволяє кожному учаснику процесу навчання швидко адаптуватися до платформи та ефективно використовувати її функціонал. Це, у свою чергу, сприяє полегшенню навчального процесу та покращенню результативності роботи на платформі.

ВИСНОВКИ

У дипломній роботі на тему «Розробка web-платформи для онлайн-навчання та спільної роботи» виконано комплексне дослідження, проєктування, розробку та апробацію інформаційної системи, що забезпечує ефективне управління навчальними матеріалами, організацію дистанційного навчання та підтримку спільної роботи користувачів.

У першому розділі виконано аналіз сучасних підходів та програмних рішень для розробки web-платформ, які забезпечують можливості для онлайн-навчання та спільної роботи. Розглянуто існуючі рішення, їх функціональні можливості, переваги та недоліки, а також визначено вимоги до розроблюваної системи. Було обґрунтовано доцільність розробки власної web-платформи, яка відповідатиме потребам взаємодії між користувачами, ефективного управління навчальними процесами та зберігання даних у хмарному середовищі.

У другому розділі виконано вибір та обґрунтування технологій реалізації web-платформи для онлайн-навчання. Для розробки інтерфейсу користувача обрано бібліотеку React, для серверної частини та управління базою даних – Firebase, а також застосовано сучасні технології, такі як TypeScript, SCSS та JSON, які забезпечують гнучкість та масштабованість системи. Розроблено концептуальну модель сутностей предметної області, виконано проєктування структури бази даних, що включає таблиці для зберігання курсів, уроків, користувачів та їх взаємодій. Проєктування моделі виконано у вигляді UML-діаграми класів, яка відображає логічні зв’язки між компонентами системи.

У третьому розділі розроблено основний функціонал платформи, включаючи модулі для управління курсами, уроками, домашніми завданнями та їх оцінюванням. Проведено апробацію розробленого функціоналу, що підтвердило його відповідність поставленим вимогам. Надано детальний опис методики використання платформи, яка дозволяє студентам отримувати доступ до навчальних матеріалів та завдань, викладачам – створювати та

редагувати курси, а адміністраторам – ефективно управляти вмістом системи. Функціонал платформи забезпечує автоматизацію основних процесів онлайн-навчання та створює умови для інтерактивної взаємодії між усіма учасниками навчального процесу.

Розроблена web-платформа підтримує гнучке управління навчальними матеріалами, інтеграцію з хмарними сервісами для зберігання даних, а також забезпечує можливості для спільної роботи користувачів над завданнями та проектами. Апробація системи підтвердила її зручність у використанні як для викладачів, так і для студентів, а також її здатність масштабуватися під потреби освітніх закладів різного рівня.

Проєкт реалізує сучасний підхід до онлайн-навчання, що базується на інтерактивності, доступності та автоматизації навчальних процесів. Розроблена система сприяє підвищенню ефективності освітнього процесу, забезпечуючи доступ до якісних освітніх ресурсів незалежно від місця знаходження користувачів. Дана платформа може стати основою для впровадження нових інноваційних методів навчання та співпраці в освітній сфері.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. COURSERA [Електронний ресурс]. – Режим доступу: https://www.coursera.org/ – 21.09.2024р.
2. Udemy [Електронний ресурс]. – Режим доступу: https://www.udemy.com/ – 21.09.2024р.
3. Бурба, А. С. Основи веб-програмування: навчальний посібник. Київ: Видавничий дім «Кондор», 2020. 384 с.
4. Шилімов, Є. В., Василенко, Ю. В. Технології розробки веб-додатків: навчальний посібник. Харків: ХНУРЕ, 2019. 250 с.
5. Кенг, С. React. Сучасний посібник для початківців. Київ: Видавництво «Форс», 2021. 320 с.
6. Браун, М. TypeScript. Створення надійних веб-додатків. Львів: Видавництво «Новий Світ-2000», 2021. 280 с.
7. Firebase Documentation. Cloud Firestore Overview. [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/firestore (дата звернення – 01.10.24).
8. JavaScript Documentation. MDN Web Docs. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення – 12.08.24).
9. W3Schools. CSS Reference. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/cssref/ (дата звернення – 05.09.24).
10. Plant UML. url: https://plantuml.com// (дата звернення – 21.09.24).
11. Алексєєв, О. В., Кучеренко, М. В. Проєктування та розробка баз даних: навчальний посібник. Одеса: ОНУ, 2018. 272 с.
12. Гама, Е., Хелм, Р., Джонсон, Р., Влісідес, Дж. Паттерни проектування. Київ: Діалектика, 2020. 400 с.
13. Літвінов, М. В. Методології Agile у розробці програмного забезпечення: аналіз та практичне застосування. Вінниця: УНІВЕРСУМ, 2021. 198 с.

14. Google Developers. JSON Basics. [Електронний ресурс]. – Режим доступу: https://developers.google.com/json (дата звернення – 10.10.24).
15. React Documentation. React Official Website. [Електронний ресурс]. – Режим доступу: https://reactjs.org/ (дата звернення – 04.08.24).
16. SCSS Documentation. Sass: Syntactically Awesome Stylesheets. [Електронний ресурс]. – Режим доступу: https://sass-lang.com/ (дата звернення – 21.09.24).
17. Таненбаум, Е. С., Ветт, Т. Комп'ютерні мережі. Київ: Видавництво «Перо», 2020. 816 с.
18. Грабовський, П. О. Сучасні підходи до розробки інтерфейсів користувача. Наукові праці [Текст]. Київ: КНУ, 2022. № 2. С. 25–30.
19. Bass, L., Clements, P., Kazman, R. Software Architecture in Practice. Boston: Addison-Wesley, 2020. 572 p.
20. Kubecka, R. Practical Firebase: Building Scalable Serverless Apps. Berkeley: Apress, 2022. 298 p.
21. Jones R. Books: Measuring Research: What Everyone Needs to Know: Anyone For Donuts? Br J Gen Pract. 2018 Jul;68(672):337–8. doi: 10.3399/bjgp18X697793. PMCID: PMC6014411.
22. Ronald N. Kostoff. The Handbook of Research Impact Assessment. Edition 7. Summer 1997.
23. Yves Gingras. Bibliometrics and Research Evaluation: Uses and Abuses . Cambridge, MA : MIT Press , 2016 . 136 pp.
24. Портал розробників системи Clarivate. url: https://developer.clarivate.com/ (дата звернення – 12.09.24).
25. Elsevier Developer Portal url: https://dev.elsevier.com/ (дата звернення – 14.09.24).
26. Єгоров І. Ю. Оцінки результатів наукової діяльності: традиційні підходи та нові виклики // Наука та наукознавство. 2014. № 3. с. 42–47.

ДОДАТОК А

Лістинг сторінки HomeworkPage та ClassroomSettingsPage

```
import { doc, getDoc } from "firebase/firestore";
import React from "react";
import { useParams } from "react-router-dom";
import { useAppDispatch, useAppSelector } from "store";
import { Homework } from "types/Lessons";
import { firestoreDB } from "utils/firebase";
import formatDoc from "utils/helpers/formatDoc";
import { TextField, Button } from "@mui/material";
import "./styles.scss";
import { editHomework } from "store/slices/homework";
import Divider from "components/Divider";
import { modalsActions } from "store/slices/modals";
import DownloadIcon from "@mui/icons-material/Download";

const cssName = "homework-page";
const getNameFromUrl = (url = "") => {
  const s = url.split("?")[0].split("%2F");
  return s[s.length - 1];
};

function HomeworkPage() {
  const dispatch = useAppDispatch();
  const { organizationId, classroomId, lessonId, studentId } = useParams();
  const lessons = useAppSelector((state) => state.lessons.lessons);
  const lesson = lessons?.find((org) => org.id === lessonId);
  const classrooms = useAppSelector((state) => state.classrooms.classrooms);
  const classroom = classrooms?.find((x) => x.id === classroomId);
  const [homework, setHomework] = React.useState<Homework | null>(null);
  const [isLoading, setIsLoading] = React.useState(true);
  const [newMark, setNewMark] = React.useState("");

  React.useEffect(() => {
    setIsLoading(true);
    getDoc(
      doc(
        firestoreDB,
        `/organizations/${organizationId}/classrooms/${classroomId}/lessons/${lesso
nId}/homeworks/${studentId}`
      )
    )
      .then((res) => {
        const hom = formatDoc(res) as Homework;
        setHomework(hom);
        setNewMark(hom.mark);
```

```
      })
      .finally(() => setIsLoading(false));

    return () => {
      setHomework(null);
    };
  }, [lessonId]);

  const onMarkSave = () => {
    dispatch(
      editHomework({
        organizationId,
        classroomId,
        lessonId,
        userId: studentId,
        homework: { ...homework, mark: newMark },
      })
    );
    setHomework({ ...homework, mark: newMark });
  };

  const onDecline = () => {
    dispatch(
      editHomework({
        organizationId,
        classroomId,
        lessonId,
        userId: studentId,
        homework: { ...homework, isDone: false, isDeclined: true },
      })
    );
    setHomework({ ...homework, isDone: false, isDeclined: true });
  };

  const openFile = (url) => (e) => {
    e.stopPropagation();
    dispatch(modalsActions.openDocModal({ url }));
  };

  if (isLoading) return <span>Loading...</span>;

  if (homework === null) return <span>This student hasn't completed homework
yet</span>;

  const downloadFile = (url) => (e) => {
    e.stopPropagation();
    console.log(url);
    var link = document.createElement("a");
    link.setAttribute("download", getNameFromUrl(url));
    link.href = url;
    document.body.appendChild(link);
```

```
    link.click();
    link.remove();
  };

  return (
    <div className={cssName}>
      <div className={`${cssName}__main`}>
        <h2>{lesson?.topic}</h2>
        <h3>
          {classroom?.teachers?.length} teacher{classroom?.teachers?.length !== 1
&& "s"}
        </h3>
        <Divider />
        <h4>{lesson?.description}</h4>
        <Divider />
      </div>
      <div className={`${cssName}__aside`}>
        <h3>Attached files</h3>
        <div>
          {homework?.files.map((url) => (
            <div className="attached-file" onClick={openFile(url)} style={{
paddingRight: 12 }}>
              <span>{getNameFromUrl(url)}</span>
              <DownloadIcon onClick={downloadFile(url)} />
            </div>
          ))}
        </div>
        <div className="mark-input">
          <TextField value={newMark} onChange={(e) => setNewMark(e.target.value)}
type="number" />
          <span>/ 100</span>
          <Button title="Save" onClick={onMarkSave}>
            Save
          </Button>
        </div>
        <button className="decline-button" title="Decline" onClick={onDecline}>
          Decline
        </button>
      </div>
    </div>
  );
}

export default HomeworkPage;
```

```
import EditClassroomForm from "components/forms/EditClassroomForm";
import React from "react";
import Tabs from "@mui/material/Tabs";
import Tab from "@mui/material/Tab";
import TeachersTab from "./components/TeachersTab";
import StudentsTab from "./components/StudentsTab";

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

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

  return (
    <div role="tabpanel" hidden={value !== index} {...other}>
      {value === index && children}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    style: {
      width: "50vw",
      maxWidth: "unset",
    },
  };
}
function a11yInnerProps(index: number) {
  return {};
}

function ClassroomSettingsPage() {
  const [value, setValue] = React.useState(0);
  const [innerValue, setInnerValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  const handleInnerChange = (event: React.SyntheticEvent, newValue: number) => {
    setInnerValue(newValue);
  };
  return (
    <div>
      <Tabs value={value} onChange={handleChange} aria-label="basic tabs example">
        <Tab label="General Information" {...a11yProps(0)} />
        <Tab label="Users" {...a11yProps(1)} />
```

```
      </Tabs>
      <TabPanel value={value} index={0}>
        <EditClassroomForm />
      </TabPanel>
      <TabPanel value={value} index={1}>
        <Tabs value={innerValue} onChange={handleInnerChange} aria-label="basic
tabs example">
          <Tab label="Teachers" {...a11yInnerProps(0)} />
          <Tab label="Students" {...a11yInnerProps(1)} />
        </Tabs>
        <TabPanel value={innerValue} index={0}>
          <TeachersTab />
        </TabPanel>
        <TabPanel value={innerValue} index={1}>
          <StudentsTab />
        </TabPanel>
      </TabPanel>
    </div>
  );
}

export default ClassroomSettingsPage;
```

ДОДАТОК Б

Лістинг сторінки OrganizationSettingsPage та LessonPage

```
import React from "react";
import Tabs from "@mui/material/Tabs";
import Tab from "@mui/material/Tab";
import TeachersTab from "pages/OrganizationSettingsPage/components/TeachersTab";
import StudentsTab from "pages/OrganizationSettingsPage/components/StudentsTab";
import InvitedUsersTab from
"pages/OrganizationSettingsPage/components/InvitedUsersTab";
import ManagersTab from "./components/ManagersTab";
import EditOrganizationForm from "components/forms/EditOrganizationForm";

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

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

  return (
    <div role="tabpanel" hidden={value !== index} {...other}>
      {value === index && children}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    style: {
      width: "50vw",
      maxWidth: "unset",
    },
  };
}

function a11yInnerProps(index: number) {
  return {};
}

function OrganizationSettingsPage() {
  const [value, setValue] = React.useState(0);
  const [innerValue, setInnerValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
```

```
    setValue(newValue);
  };

  const handleInnerChange = (event: React.SyntheticEvent, newValue: number) => {
    setInnerValue(newValue);
  };

  return (
    <div>
      <Tabs value={value} onChange={handleChange} aria-label="basic tabs example">
        <Tab label="General Information" {...a11yProps(0)} />
        <Tab label="Users" {...a11yProps(1)} />
      </Tabs>
      <TabPanel value={value} index={0}>
        <EditOrganizationForm />
      </TabPanel>
      <TabPanel value={value} index={1}>
        <Tabs value={innerValue} onChange={handleInnerChange} aria-label="basic
tabs example">
          <Tab label="Managers" {...a11yInnerProps(0)} />
          <Tab label="Teachers" {...a11yInnerProps(1)} />
          <Tab label="Students" {...a11yInnerProps(2)} />
          <Tab label="Invited Users" {...a11yInnerProps(3)} />
        </Tabs>
        <TabPanel value={innerValue} index={0}>
          <ManagersTab />
        </TabPanel>
        <TabPanel value={innerValue} index={1}>
          <TeachersTab />
        </TabPanel>
        <TabPanel value={innerValue} index={2}>
          <StudentsTab />
        </TabPanel>
        <TabPanel value={innerValue} index={3}>
          <InvitedUsersTab />
        </TabPanel>
      </TabPanel>
    </div>
  );
}

export default OrganizationSettingsPage;
```

```
import Divider from "components/Divider";
import HomeworkUpload from "components/HomeworkUpload";
import React from "react";
import { useNavigate, useParams } from "react-router-dom";
import { useAppSelector } from "store";
import useIsClassTeacher from "utils/hooks/useIsClassTeacher";
import useIsOrgAdmin from "utils/hooks/useIsOrgAdmin";
import "./styles.scss";

const cssName = "lesson-page";

function LessonPage() {
  const navigate = useNavigate();
  const { lessonId, classroomId } = useParams();
  const lessons = useAppSelector((state) => state.lessons.lessons);
  const classrooms = useAppSelector((state) => state.classrooms.classrooms);
  const classroom = classrooms?.find((x) => x.id === classroomId);
  const lesson = lessons?.find((org) => org.id === lessonId);

  const isOrgAdmin = useIsOrgAdmin();
  const isClassTeacher = useIsClassTeacher() || isOrgAdmin;

  React.useEffect(() => {
    if (isClassTeacher) navigate(`homeworks`, { replace: true });
  }, [isClassTeacher]);

  return (
    <div className={cssName}>
      <div className={`${cssName}__main`}>
        <h2>{lesson?.topic}</h2>
        <h3>
          {classroom?.teachers?.length} teacher{classroom?.teachers?.length !== 1
&& "s"}
        </h3>
        <Divider />
        <h4>{lesson?.description}</h4>
        <Divider />
      </div>
      <div className={`${cssName}__aside`}>
        <h3>Attached files</h3>
        <HomeworkUpload />
      </div>
    </div>
  );
}
export default LessonPage;
```