

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»

УДК 004.031.42

До захисту допущено:

Завідувач кафедри

_____ Сергій СТИРЕНКО

«__» _____ 2023 р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Комп'ютерні системи та мережі» зі
спеціальності 123 «Комп'ютерна інженерія»**

на тему: «Клієнтська частина системи онлайн навчання»

Виконав:

студент II курсу, групи ІО-21мп

Востриков Нікіта Вячеславович _____

Керівник:

професор, д.т.н., Луцький Г.М. _____

Консультант з нормоконтролю:

проф. каф. ОТ, д.т.н., професор Кулаков Юрій Олексійович _____

Рецензент:

доц. каф. ІСТ, к.т.н., доц. Шимкович В. М. _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____

(підпис)

Київ – 2023 року

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Обчислювальної техніки
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 123. Комп'ютерна інженерія
(код і назва)

Спеціалізація Комп'ютерні системи та мережі
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТИПЕНКО
(підпис) (ініціали, прізвище)

« » _____ 2023 р.

**ЗАВДАННЯ
на магістерську дисертацію студенту
Вострикову Нікіті Вячеславовичу**
(прізвище, ім'я, по батькові)

1. Тема дисертації Клієнтська частина системи онлайн навчання

Науковий керівник дисертації професор, д.т.н., Луцький Г.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 07 » листопада 2023 р. № 5168-с

2. Строк подання студентом дисертації 05.01.2024р.

3. Об'єкт дослідження процес взаємодії компонентів клієнтської частини в системі управління навчанням

4. Предмет дослідження компоненти клієнтської частини системи, такі інтерфейс користувача, система навігації та авторизації, інструменти створення управління курсами, забезпечення безпеки даних

5. Перелік завдань, які потрібно розробити: дослідити предметну область та проаналізувати існуючі рішення, визначити особливості, переваги та недоліки існуючих рішень, визначити функціональні та нефункціональні вимоги до системи, визначити технології для розробки відповідної системи, розробити клієнтську частину системи, розробити стартап-проект

АНОТАЦІЯ

на магістерську дисертацію

виконану на тему: Клієнтська частина системи онлайн навчання

студентом: Востриковим Нікітою Вячеславовичем

Магістерська дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних літературних джерел (23 найменування) та 1 додатку. Загальний обсяг дисертації складає 107 с., 38 рисунків та 24 таблиці, перелік посилань на 2 сторінках.

Актуальність теми дослідження

Актуальність теми дослідження полягає в тому, що сучасний світ переживає період великих змін у сферах бізнесу та освіти, а віртуальні системи управління навчанням (LMS) набувають все більшого значення. Оскільки дистанційне навчання та цифрова трансформація стають все більш поширеними, створення та впровадження ефективної LMS стало стратегічним імперативом для бізнесу та навчальних закладів. На цьому тлі дослідження для аналізу та вдосконалення систем, таких як ETIMS, які інтегрують сучасні технології для навчання, є актуальними та доречними. Результати цього дослідження можуть стати вагомим внеском у розвиток систем LMS та сприяти покращенню навчального процесу та бізнес-перспектив користувачів ETIMS.

Мета і завдання дослідження

Метою даного дослідження є розробка та вдосконалення системи управління навчанням (Learning Management System, LMS, або ELMS), інтегрованої віртуальної платформи ETIMS (Education and Training Information Management System). Завдання полягає у створенні вискоелективної та функціональної системи, яка відповідає сучасним потребам і вимогам установ освіти та бізнес-середовища.

Об'єктом дослідження є процес взаємодії компонентів клієнтської частини в системі управління навчанням.

Предметом дослідження є основні компоненти клієнтської частини системи, такі як інтерфейс користувача, система навігації та авторизації, інструменти створення та управління курсами, забезпечення безпеки даних.

Методи досліджень включають аналіз вимог користувачів, актуальних рішень ринку, проектування та розробку програмного забезпечення, використання тестування, планування розвитку проекту для постійного вдосконалення.

Особистий внесок здобувача у цьому дослідженні полягає в проектуванні, розробці та інтеграції компонентів системи ETIMS, в аналізі вимог користувачів та забезпеченні відповідності їхнім потребам.

Практична цінність цього дослідження виявляється у створенні функціональної, ефективної та відповідної потребам системи управління навчанням, що може бути використана бізнесом в якості LMS-системи та будь-якими іншими освітніми установами для покращення та управління навчальним процесом. Розробка ETIMS відкриває нові можливості для цифрової трансформації освіти та бізнесу, а також допомагає оптимізувати процеси навчання та навчання на віддалі.

Ключові слова: LMS, ELMS, система управління навчанням, навчання, програмне забезпечення, віддалене навчання, Learning System, Angular, UX/UI, REST.

ABSTRACT

for a master's thesis

performed on the topic: Front-end of the Online Learning System

student: Vostrykov Nikita

The master's thesis consists of an introduction, four chapters, conclusions, a list of references (23 titles) and 1 appendix. The total volume of the thesis is 107 pages, 38 figures and 24 tables, a list of references on 2 pages.

Relevance of the research topic

The relevance of the research topic is that the modern world is going through a period of great change in the fields of business and education, and virtual learning management systems (LMS) are becoming increasingly important. As distance learning and digital transformation become more widespread, creating and implementing an effective LMS has become a strategic imperative for businesses and educational institutions. Against this backdrop, research to analyze and improve systems such as ETIMS that integrate modern technologies for learning is relevant and appropriate. The results of this study can make a significant contribution to the development of LMS systems and help improve the learning process and business prospects of ETIMS users.

Purpose and objectives of the study

The purpose of this study is to develop and improve the Learning Management System (LMS, or ELMS), an integrated virtual platform ETIMS (Education and Training Information Management System). The task is to create a highly efficient and functional system that meets the modern needs and requirements of educational institutions and the business environment.

The object of the study is the process of interaction between the components of the client part in the learning management system.

The subject of the study is the main components of the client part of the system, such as the user interface, navigation and authorization system, tools for creating and managing courses, and data security.

Research methods include analyzing user requirements, current market solutions, software design and development, using testing, and planning project development for continuous improvement.

The applicant's personal contribution to this research is to design, develop, and integrate ETIMS system components, analyze user requirements, and ensure that they meet their needs.

The practical value of this research is manifested in the creation of a functional, efficient and relevant learning management system that can be used by businesses as an LMS system and any other educational institutions to improve and manage learning. The development of ETIMS opens up new opportunities for digital transformation of education and business, and helps to optimize learning and distance learning processes.

Keywords: LMS, learning management system, training, software, remote learning, Learning System, Angular, UX/UI, REST.

ЗМІСТ

Перелік умовних позначень.....	10
Вступ.....	11
Розділ 1	13
LMS: теоретичний та практичний аналіз.....	13
1.1 Актуальність систем управління навчанням.....	13
1.2 Огляд існуючих LMS-систем.....	15
1.2.1 Moodle.....	15
1.2.2 Blackboard Learn	19
1.2.3 Google Classroom	22
1.2.4 Coursera.....	25
1.3 Порівняння систем.....	27
Висновки до розділу 1	30
Розділ 2	32
Вимоги, опис системи, підготовка до розробки системи.....	32
2.1 Функціональні вимоги.....	32
2.2 Нефункціональні вимоги.....	35
2.3 Архітектура системи та сценарії використання.....	37
2.4 Вибір типу застосунку.....	40
2.5 Вибір технологій для розробки Front-end частини застосунку	43
Висновки до розділу 2	49
Розділ 3	51
Розробка системи.....	51
3.1 Розробка користувацького дизайну застосунку	51
3.2 Розробка клієнтської частини застосунку	55
3.2.1 Основні налаштування.....	55
3.2.2 Визначення внутрішньої структури	56
3.2.3 Проєктування бази даних	58
3.2.4 Визначення пріоритетів розробки	59
3.2.5 Розробка базових компонент	60
3.2.6 Додавання маршрутизації.....	63
3.2.7 Розробка сторінок.....	64

3.2.8 Покращення якості коду	71
3.2.9 WebSocket	72
3.2.10 Авторизація Keycloak	74
3.2.11 Продуктивність системи	81
Висновки до розділу 3	83
Розділ 4	84
Розробка стартап-проекту	84
4.1 Опис ідеї стартапу	85
4.2 Технічний аудит ідеї стартапу	87
4.3 Аналіз ринкових можливостей запуску стартап-проекту	88
Висновки до розділу 4	93
Висновки	94
Перелік використаних джерел	96
Додатки	99
Додаток 1: Сценарії використання системи	99
Спільні сценарії використання	99
Сценарії використання системи вчителем	101
Сценарії використання системи студентом	105

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

2FA – Two-Factor Authentication

API – Application programming interface

BE – Back-end

CAS – Central Authentication Service

CD – Continuous Delivery

CI – Continuous Integration

CRUD – Create, Read, Update, Delete

DB – Database

FE – Front-end

IoC – Inversion of Control

JS – JavaScript

LDAP – Lightweight Directory Access Protocol

LMS – Learning Management System

OTP – one time password

REST – Representational State Transfer

SOLID – Single responsibility, Open–closed, Liskov substitution, Interface segregation та Dependency inversion

SSO – Single Sign-On

SSR – Server-side rendering

UI – User interface

UX – User experience

XSS – cross site scripting

ВСТУП

На початку 21 сторіччя, в 2019 році сталася подія, яка кардинально змінила парадигму освіти та навчання - пандемія COVID-19. Світ зіткнувся з масштабною пандемією, яка вимагала негайних заходів обмеження соціальних контактів та вводу глобального карантину. Однією з ключових областей, які зазнали великого впливу, була освіта. Закриття учбових закладів, обмеження відвідування публічних місць, введення суворих вимог для боротьби з пандемією створили необхідність забезпечити неперервність навчання та впровадити альтернативні методи навчання. Віддалене навчання стало головним рішенням для збереження здоров'я та безпеки учасників навчального процесу. Інші пандемії або глобальні кризи також можуть стати причиною подібних обмежень, саме тому необхідно розробляти стійкі та ефективні стратегії для навчання в умовах обмежених соціальних контактів.

У лютому 2022 року наша країна зіткнулась з новим серйозним викликом – напад росії на Україну. Перш за все, війна призвела до глобальних руйнувань енергетичної та іншої інфраструктури, окрім цього всі люди опинились під загрозою для життя та здоров'я, через регулярні обстріли, в тому числі навчальних закладів. Закриття навчальних закладів було необхідним кроком для збереження життя освітян та мінімізації ризику для їх життя та здоров'я, тому альтернативні методи навчання, такі як онлайн-освіта, стають життєво важливими для забезпечення доступу до освіти та неперервності навчання.

Війна створила нові виклики для громадян, необхідність переселення населення до більш безпечних регіонів, ізоляція та обмеження доступу до територій. В таких умовах онлайн-навчання надає можливість внутрішньо переміщеним та переселеним особам навчатися і розвиватися, незалежно від їхнього місця перебування. Окрім цього, онлайн-навчання під час війни може бути інструментом для підтримки і навчання захисників, військових та медиків, які захищають країну та надають медичну допомогу в умовах війни. Вони можуть

отримувати необхідні знання, навчальні матеріали та інструкції для ефективного виконання своїх обов'язків.

Віддалене навчання, як засіб забезпечення доступу до навчання в умовах неможливості відвідувати навчальні заклади фізично, стає актуальним завдяки своїй гнучкості та асинхронності. Онлайн-платформи та системи управління навчанням (LMS) надають можливість отримування знання незалежно від місця розташування та часу. Це робить навчання доступним для учасників.

Системи управління навчанням можуть бути реалізовані двома способами, в якості Мобільних та Веб-застосунків. Зручність використання таких застосунків для навчання ґрунтується на їхній можливості надати доступ до освітнього матеріалу та можливості навчання в зручний для користувачів спосіб. Веб-платформи та мобільні додатки надають можливість навчатися без територіальних та часових обмежень. Користувачі можуть обрати безпечне місце та час для навчання, що робить навчання безпечним для життя та надає гнучкість та можливість навчатися відповідно до власного розкладу та потреб.

Метою дослідження є розробка системи управління навчанням, розробленої для задоволення сучасних вимог до освітнього процесу і бізнесу в частині дистанційної освіти та розвитку компетенцій співробітників.

Дослідження спрямоване на створення гнучкої та ефективної платформи, що враховує потреби навчальних закладів, бізнесу та індивідуальних користувачів у навчанні та вдосконаленні власних знань. Дослідження охоплюватиме технічні аспекти розробки LMS, такі як мови програмування, фреймворки, вибір технологій, архітектурні рішення та методи безпеки.

Окрім цього, дослідження охоплює аналіз сучасних тенденцій в онлайн-навчанні та визначення найкращих практики використання LMS для навчання і розвитку. Також буде розглянута інтеграція з існуючими системами, такими як системи автентифікації та ідентифікації користувачів.

РОЗДІЛ 1

LMS: ТЕОРЕТИЧНИЙ ТА ПРАКТИЧНИЙ АНАЛІЗ

1.1 АКТУАЛЬНІСТЬ СИСТЕМ УПРАВЛІННЯ НАВЧАННЯМ

Learning Management Systems (LMS) є програмним рішенням, що спрямоване на управління, підтримку навчання і навчальних процесів [1]. LMS надають можливість організовувати, імплементувати, відстежувати і оцінювати навчання, як онлайн, так і в офлайн режимі, проте найбільш ефективно ці системи працюють в онлайн режимі. Такі системи включають в себе набір інструментів для створення та керування навчальним контентом, взаємодії зі студентами або учнями, ведення аналітики, журналів та оцінок, аналізу навчальних результатів. Функціонал таких систем може бути розширений до нескінченості.

У всьому світі ринок систем LMS зростає та розвивається. Він об'єднує багато галузей з різних секторів, включаючи освіту, корпоративне навчання, державні установи, спорт і охорону здоров'я. До світових лідерів у галузі LMS належать такі компанії, як Moodle, Blackboard, Canvas (частина Instruct), Adobe Captivate та інші. Ринок LMS оцінюється в мільярди доларів і продовжує зростати завдяки актуальності і популярності онлайн-навчання та зростаючому попиту на віддалене навчання.

Україна не стала винятком і відчула на собі вплив розвитку LMS. Національні та міжнародні університети, школи, підприємства та державні установи в Україні використовують LMS для покращення процесів навчання та розвитку співробітників. Більшість університетів, включаючи КПІ використовують Moodle та Google Classroom для віддаленого навчання. В Україні також є власні розробники систем LMS, які створюють рішення для освітніх та корпоративних потреб.

Системи LMS підходять для різних аудиторій та галузей. У сфері освіти вони дозволяють університетам, школам і навчальним центрам ефективно організовувати та контролювати освітні процеси. У бізнесі системи управління навчанням

сприяють підвищенню кваліфікації та розвитку співробітників, покращують професійні навички співробітників і допомагають підвищити продуктивність і ефективність компанії в цілому.

LMS в Україні допомагають університетам і навчальним закладам покращити навчальні процеси та надати можливості навчання студентам з різних регіонів, що є особливо актуально під час пандемій та війни. В українському бізнесі LMS допомагає ефективно та в короткі терміни підвищити кваліфікацію персоналу, скоротити витрати на навчання та підвищити ефективність бізнес-процесів.

Загалом системи LMS стають дедалі важливішими в сучасному освітньому та бізнес-середовищі, допомагаючи забезпечити доступ до якісної освіти та навчання незалежно від фізичної відстані та місця розташування. Вони підтримують персоналізоване навчання та роблять навчальний контент більш доступним та інтерактивним.

У сучасному світі, де мобільність та глобалізація стають все більш важливими, LMS-системи забезпечують зручний доступ до навчання за допомогою веб-браузерів та мобільних застосунків. Це робить навчання більш гнучким та доступним для тих, хто не має доступу до навчання «класичним», офлайн, способом та не має достатню кількість часу для цього.

Однак, важливо зауважити, що використання LMS-систем само по собі не здатно підвищити якість навчання. Для досягнення максимальної продуктивності такого навчання це вимагає якісного навчального контенту, продуманих педагогічних підходів, зручного UX та гарного UI, а також цілодобової підтримки для користувачів. Добре розроблені LMS можуть суттєво покращити якість, швидкість навчання та розвиток освітян, але їх ефективність залежить від якості контенту та підходів до навчання.

У майбутньому, розвиток та вдосконалення LMS-систем буде продовжуватися, зокрема, завдяки розвитку та поширенню штучного інтелекту,

аналітики даних та інновацій в галузі навчання. Вони залишаються ключовим інструментом для покращення освіти та розвитку у всьому світі, включаючи Україну, та сприяють забезпеченню доступу до знань та навчання для всіх.

1.2 ОГЛЯД ІСНУЮЧИХ LMS-СИСТЕМ

В сучасних реаліях LMS-системи відіграють ключову роль у віддаленому навчанні і розвитку освітян та працівників компаній. Із ростом онлайн-освіти та потребою в управлінні навчальним процесом, LMS-системи стали незамінними інструментами для освітніх установ, корпорацій, та інших організацій. Огляд ринку LMS-систем допомагає нам зрозуміти, які платформи є найбільш популярними та впливовими на глобальному та регіональному рівнях.

1.2.1 Moodle

Moodle є безкоштовним сервісом, що має відкритий програмний код, це означає, що користувачі можуть змінювати та адаптувати його відповідно до своїх потреб. Це надає багато можливостей для налаштування та розширення функціональності системи [2].



Рис. 1.1 Логотип Moodle

Moodle пропонує велику гнучкість у створенні навчального контенту та курсів. Він підтримує різні типи навчальних ресурсів, включаючи текст, відео, вікторини, форми, завдання тощо.

Moodle має велику й активну спільноту користувачів і розробників, які надають багато безкоштовних розширень і плагінів для системи. Це робить Moodle потужним інструментом для навчання використання в багатьох компаніях та навчальних закладах, зокрема українських університетах.

Moodle має зрозумілий інтерфейс, хоча і суттєво застарілий, що не відповідає сучасним вимогам ринку. Інтерфейс зрозумілий у використанні як для викладачів, так і для студентів. Це дозволяє швидко створювати та редагувати курси та навчальний контент. Проте інтерфейс вимагає певного часу на вивчення та адаптації до нього.

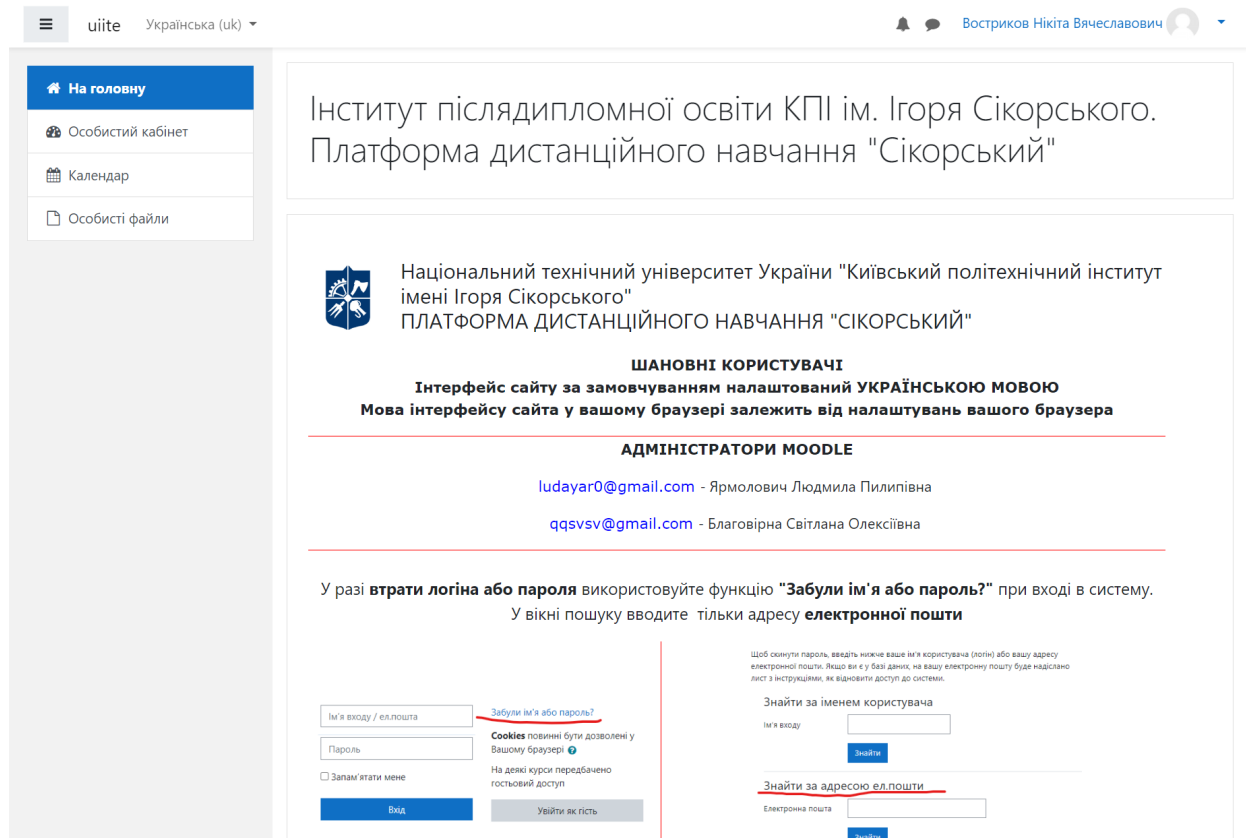


Рис. 1.2 Вигляд Веб-сторінки Moodle

Moodle має багато функцій, включаючи можливість створення та оцінювання завдань, проведення тестів, вікторин та опитувань, ведення журналів та обліку навчального прогресу студентів. Розглянемо основні можливості детальніше:

1. Курси та Категорії:

- Створення курсів: Викладачі можуть створювати курси для різних предметів.
- Категорії курсів: Курси можна групувати за категоріями, що полегшує організацію та пошук.

2. Матеріали курсів:

- Документи, відео, аудіо: Можливість додавати та обмінюватися різноманітними навчальними матеріалами.
- Завдання та тести: Використання різних форматів завдань та тестів для оцінювання.

3. Інтерактивність:

- Форуми та Обговорення: Можливість спілкування та взаємодії між учасниками курсу.
- Відео конференції: Інтеграція з іншими інструментами для відео-зустрічей та онлайн-конференцій.

4. Оцінювання та Звітність:

- Оцінювання завдань: Вчителі можуть створювати та оцінювати завдання.
- Генерація звітів: Забезпечення статистики та звітності про успішність студентів.

5. Адаптивний дизайн:

- Мобільна сумісність: Доступ до курсів та матеріалів через мобільні пристрої.

6. Зовнішні розширення:

- Магазин додатків та плагінів: Можливість розширювати функціональність через додатки та плагіни.

7. Підтримка SCORM та Tin Can API:

- Сумісність із стандартами e-learning

8. Співпраця та Групи:

- Можливість створення груп: Групи для спільної роботи та спілкування.

9. Адміністративні можливості:

- Керування користувачами та ролями: Адміністратори можуть керувати доступом та правами користувачів.

10. Безпека та Приватність:

- Контроль доступу: Налаштування прав доступу для забезпечення безпеки.

Moodle пропонує мобільні застосунки для Android та iOS, що дозволяють студентам та викладачам отримувати доступ до платформи з мобільних пристроїв.

Наступна важлива частина застосунку – авторизація та автентифікація. Moodle підтримує такі види автентифікації:

1. Основний механізм автентифікації:

- Moodle підтримує стандартні форми автентифікації, такі як введення логіну та пароля.

2. Зовнішні системи автентифікації:

- Інтеграція зовнішніх систем автентифікації, таких як LDAP, CAS або Shibboleth.

3. Соціальні мережі:

- Можливість автентифікації через облікові записи в соціальних мережах, таких як Google, Facebook або X.

4. Email-автентифікація:

- Відправлення спеціальних лінк для автентифікації через електронну пошту.

5. Механізми двоетапної автентифікації (2FA):

- Підтримка різних методів для двоетапної автентифікації, таких як введення коду, відправка SMS або використання аутентифікаторів.

6. Одноразові паролі (OTP):

- Використання одноразових паролів для додаткового рівня безпеки.

7. Гнучкі налаштування авторизації:

- Можливість адміністраторів налаштовувати правила та умови автентифікації для відповідності конкретним потребам.

8. Мультифакторна автентифікація (MFA):

- Підтримка різних методів MFA для забезпечення вищого рівня безпеки.

9. Використання сертифікатів:

- Можливість використання цифрових сертифікатів для автентифікації користувачів.

Підсумуємо особливості платформи Moodle:

- Відкритий код і гнучкість роблять Moodle особливо привабливим для організацій, які потребують індивідуалізації свого навчального середовища та не готові платити за сторонні рішення.
- Застарілий дизайн Moodle робить платформу не привабливою для сучасних, прогресивних компаній.
- Moodle має велику спільноту користувачів, що полегшує імплементацію системи та допомагає вирішувати різні проблеми.
- Можливість створювати як прості і складні курси, навчальні ресурси, включаючи інтерактивні елементи.

1.2.2 Blackboard Learn

Blackboard Learn пропонує багато варіантів створення курсів і керування ними [4]. Він підтримує різні типи вмісту, включаючи текст, відео, завдання, вікторини, спільноти, блоги тощо [5].



Рис. 1.3 Логотип Blackboard Learn

Blackboard Learn відомий своїм інтуїтивно зрозумілим інтерфейсом, що полегшує використання системи вчителями та учнями. Він забезпечує доступ до необхідної інформації та функціональності в декілька кліків. Інтерфейс якісно відрізняє Blackboard Learn від Moodle.

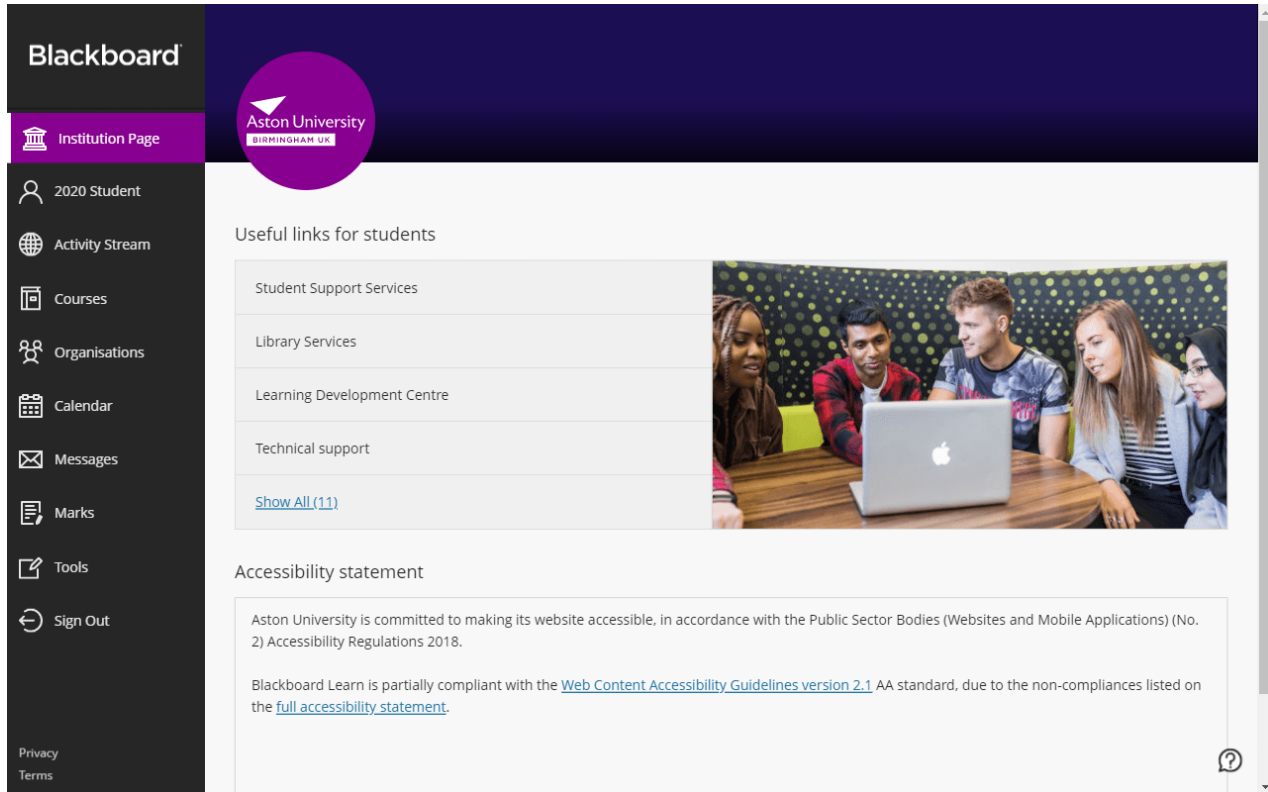


Рис. 1.4 Вигляд Веб-сторінки Blackboard Learn

Blackboard Learn має активну спільноту користувачів і розробників, які підтримують систему та створюють додаткові розширення (плагіни).

Blackboard Learn дозволяє користувачам, таким як вищим навчальним закладам та бізнесам адаптувати систему під свої потреби та вимоги.

Система також надає інструменти для збору й аналізу даних про навчання студентів, що дозволяє викладачам і адміністраторам приймати обґрунтовані рішення.

Blackboard Learn використовує комплексний підхід до автентифікації та авторизації користувачів, забезпечуючи безпеку та конфіденційність даних. Основні засоби включають:

1. Blackboard Learn Account:

- Користувачі створюють обліковий запис на платформі Blackboard Learn, використовуючи електронну пошту та пароль.

2. LDAP:

- В багатьох випадках, особливо в установах, де використовується централізований механізм автентифікації, Blackboard Learn може інтегруватися з LDAP для забезпечення єдиної точки автентифікації.

3. CAS:

- Blackboard Learn може підтримувати протокол CAS для централізованої автентифікації. Це дозволяє користувачам входити в систему за допомогою їхніх централізованих облікових записів.

4. SSO:

- Для зручності користувачів Blackboard Learn може підтримувати технологію SSO, яка дозволяє входити в систему один раз і мати доступ до різних ресурсів без повторного введення облікових даних.

5. Двофакторна автентифікація:

- Користувачі можуть налаштовувати двофакторну автентифікацію для додаткового рівня безпеки.

Особливості системи Blackboard Learn:

- Blackboard Learn відома своєю великою кількістю можливостей та розширень, що робить її конкурентоспроможною та привабливою для вищих навчальних закладів і корпорацій, де потрібні різні методи навчання та управління.
- Підтримка та надійність системи роблять Blackboard Learn популярною серед великих університетів і організацій, які вимагають постійної і стабільної роботи системи.
- Приємний дизайн додає позитивного користувацького досвіду.

1.2.3 Google Classroom

Google Classroom – це LMS, що має невеликий функціонал, мінімалістичний дизайн та потужні сервіси інтеграції.



Рис. 1.5 Логотип Google Classroom

Для розуміння потужності цієї системи розглянемо основні особливості цієї LMS.

- Інтеграція з сервісами Google: Google – найпопулярніша та одна з самих всеохоплюючих екосистем цифрових продуктів, це відкриває величезні можливості при використанні будь-якого продукту Google, Google Classroom не став виключенням. Classroom за замовчуванням інтегрується з іншими інструментами Google, такими як Google Drive, Google Forms, Google Docs, Google Sheets та Gmail. Це робить процес навчання більш ефективним і зручним, а всім учасникам навчального процесу дозволяє підтримувати постійну комунікацію між собою [6].
- Простий інтерфейс користувача: Інтерфейс Google Classroom є дуже простим і це робить його інтуїтивно зрозумілим, це дозволяє вчителям і учням легко розпочати використання системи без особливих знань.

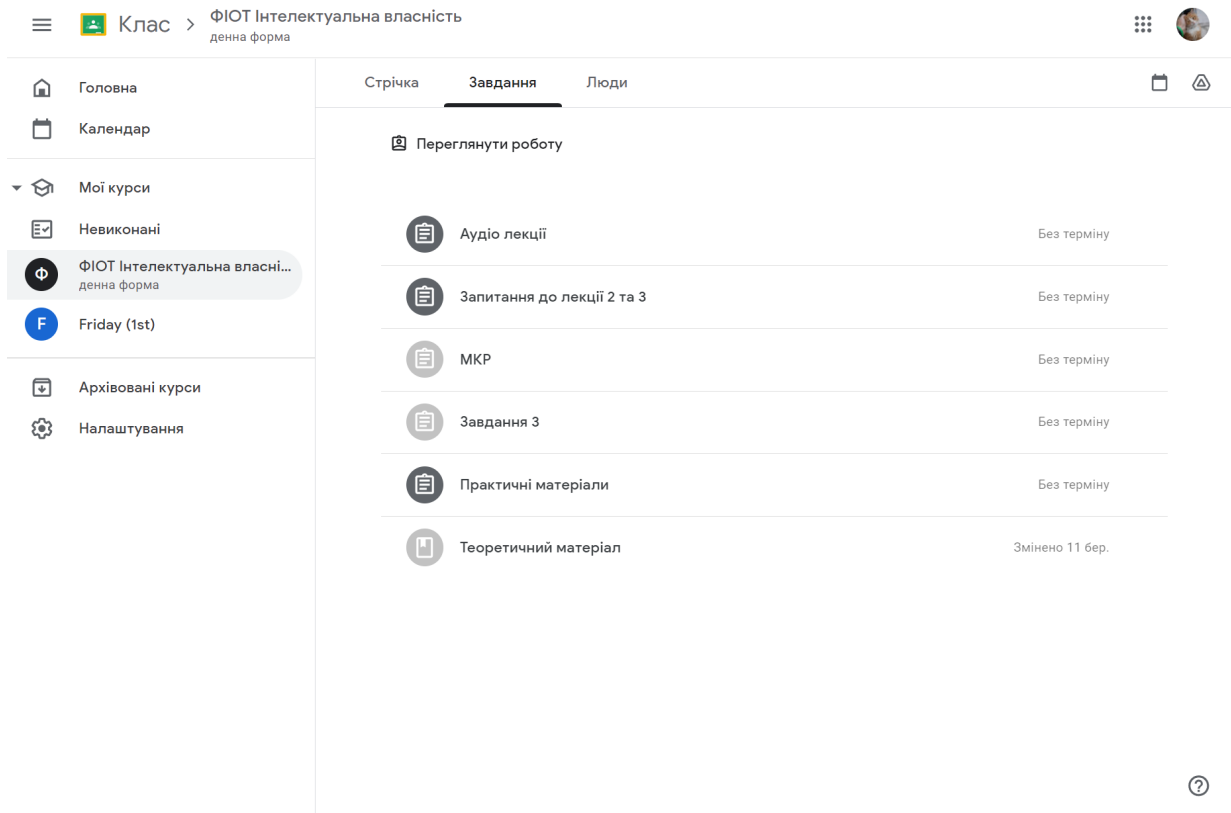


Рис. 1.6 Вигляд Веб-сторінки Google Classroom

- Можливість створювати та редагувати завдання і курси: Вчителі можуть легко створювати курси та завдання, редагувати їх та призначати учням, додавати дедлайни та оцінювати завдання онлайн.
- Можливість зберігати та спільно редагувати документи: У Google Classroom вчителі можуть створювати документи та давати учням доступ до них, після чого дозволяти учням редагувати їх, а викладач бачить зміни в реальному часі.
- Контроль навчального процесу: Вчителі мають можливість відслідковувати прогрес учнів, переглядати зроблені завдання та надавати зворотний зв'язок [7].
- Можливість спільно спілкуватися та обговорювати завдання: Учні та вчителі можуть легко спілкуватися через коментарі, обговорювати матеріали та завдання.
- Наявність сервісів для голосового/відео-зв'язку: Одна з найпотужніших особливостей Google Classroom це наявність вбудованої можливості

створювати безкоштовні відео та голосові конференції для учасників навчального процесу і не тільки.

Google Classroom використовує систему автентифікації та авторизації, що базується на облікових записах Google. Основні засоби авторизації та автентифікації включають:

1. Google Account:

- Кожен користувач, який хоче користуватися Google Classroom, повинен мати обліковий запис Google. Це може бути поштова скринька Gmail або інший обліковий запис Google, наприклад піддомен Ill.kpi.ua.

2. OAuth 2.0:

- Google Classroom використовує протокол авторизації OAuth 2.0 для забезпечення безпеки під час обміну даними між різними службами та додатками.

3. Автентифікація через Google Apps for Education:

- В навчальних закладах, де використовується Google Apps for Education, учні та вчителі можуть використовувати свої шкільні облікові записи для автентифікації в Google Classroom.

4. Єдиний вхід (SSO):

- Якщо дистрибутив Google Classroom інтегровано з системою єдиного входу, користувачі можуть автоматично увійти в систему, використовуючи свої облікові записи.

5. Багаторівнева авторизація:

- Google Classroom надає багаторівневу систему авторизації, яка дозволяє вчителям та учням отримувати різні рівні доступу залежно від їхніх ролей.

6. Захист даних:

- За допомогою механізмів автентифікації та авторизації Google, забезпечується захист особистої інформації та конфіденційних даних користувачів.

Загальний принцип полягає в тому, що користувачі повинні увійти в систему, використовуючи свій обліковий запис Google, а додатки та служби отримують доступ до необхідних даних лише після отримання відповідних дозволів через протокол OAuth 2.0.

1.2.4 Coursera

Coursera - це онлайн-платформа для навчання, що пропонує доступ до курсів та спеціалізацій великої кількості університетів і навчальних закладів.



Рис. 1.7 Логотип Coursera

Розглянемо основні особливості Coursera, це допоможе визначити конкурентоспроможність цієї системи на ринку та її спеціалізацію і цільову аудиторію [8].

- Широкий вибір курсів і спеціалізацій: Coursera співпрацює з більш ніж 200 університетами та навчальними закладами і надає доступ до десятків тисяч курсів та спеціалізацій у різних галузях, починаючи від будівництва і садівництва, закінчуючи тайм-менеджментом і програмуванням на найсучасніших фреймворках.
- Відомі викладачі та університети: Курси на Coursera проводять викладачі з таких всесвітньо відомих університетів, таких як Йель, Стенфорд, Гарвард і багато інших, на цій платформі також представлено багато курсів від Microsoft, Google, NVIDIA та інших, це забезпечує високий рівень кваліфікації та освіти [9].
- Відеолекції та інтерактивні завдання: Більшість курсів Coursera мають відеолекції, інтерактивні завдання, матеріали для самостійного вивчення та тести, які допомагають студентам засвоювати матеріал.

- **Форуми та спільноти:** Студенти Coursera можуть обговорювати матеріали та завдання через вбудовані механізми комунікації, форуми і спільноти, що створює сприятливі умови для легкого вирішення проблем, співпраці та обміну досвідом.
- **Сертифікати та дипломи:** Після успішного завершення курсу або спеціалізації студенти можуть отримати офіційний сертифікат або навіть диплом від університету.
- **Дизайн:** Дизайн Coursera приємний і мінімалістичний, отримувати доступ до різних елементів просто, дістатися до будь-якої сторінки можна за декілька кліків.

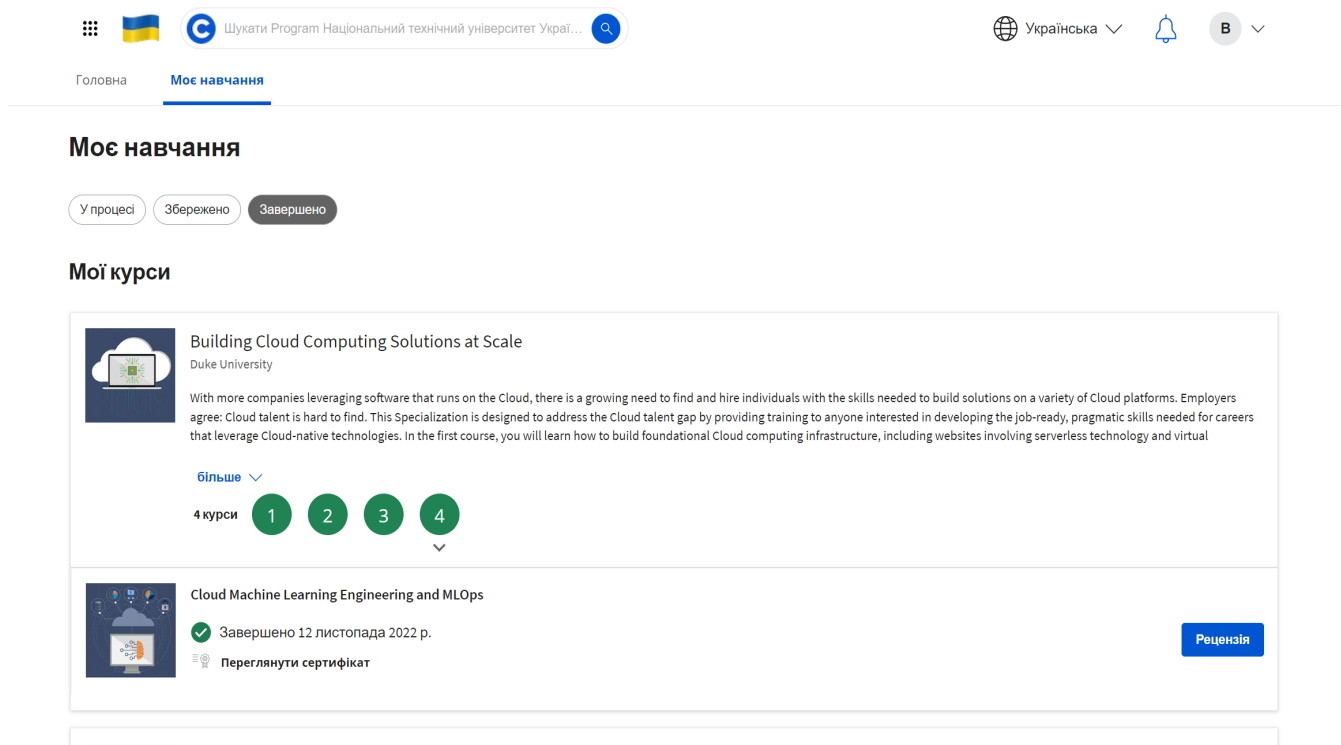


Рис. 1.8 Вигляд Веб-сторінки Coursera

Coursera використовує систему автентифікації та авторизації, яка забезпечує безпеку та конфіденційність даних користувачів. Основні засоби включають:

1. Coursera Account:

- Користувачі повинні створити обліковий запис на платформі Coursera за допомогою пошти та паролю.

2. OAuth 2.0:

- Coursera надає можливість взаємодії з іншими системами та додатками через протокол авторизації OAuth 2.0. Це дозволяє користувачам безпечно обмінюватися даними та використовувати зовнішні ресурси.

3. Інтеграція з Google та Facebook:

- Користувачі можуть використовувати свої облікові записи Google або Facebook для швидкого та зручного входу на платформу.

4. Двофакторна аутентифікація (2FA):

- Для підвищення безпеки користувачі можуть налаштовувати двофакторну аутентифікацію, яка включає додатковий етап перевірки особи.

Особливості даної LMS-системи:

- Coursera виділяється своїм підходом до навчання, пропонуючи курси від університетів і викладачів світового рівня. Великий вибір курсів і спеціалізацій робить його особливо привабливим для самостійної освітньої діяльності.
- Модель Coursera заснована на відкритому доступі до багатьох курсів, але також пропонує платні курси і спеціалізації для отримання сертифікатів і дипломів, що робить її придатною для різних аудиторій.

1.3 ПОРІВНЯННЯ СИСТЕМ

Для визначення необхідних характеристик майбутньої системи важливо розуміти можливості існуючих систем. Для цього порівняймо важливі характеристики систем для визначення первинних вимог.

Таблиця 1.1

Порівняння основних характеристик

	MOODLE	Blackboard	Classroom	Coursera
Дизайн, UI	Застарілий	Застарілий	Сучасний	Сучасний
Користувацький досвід, UX	Громіздкий, вимагає часу на адаптацію	Незрозумілий	Зручний, мінімалістичний	Мінімалістичний, вимагає часу на адаптацію
Актуальність продукту	Розробляється та вдосконалюється волонтерами	Підтримка припинена	Підтримується, не оновлюється	Не оновлюється
OpenSource	+	-	-	-
Статистика	Відвідувань, успішність завдань, успішність студентів, час виконання завдання / курсу	Відвідувань, успішність студентів	Прогрес виконання завдань	Успішність студентів, витрачений час, відвідувань
Підтримка xAPI	+	+	-	-
Аналітика	-	Взаємодія з учнями, успішність студентів	-	Розташування студентів, успішність студентів, динаміка взаємодії викладача та студента, ефективність викладачів
Інноваційні рішення				Заглушка з попередженням про кодекс доброчесності

Проаналізувавши та порівнявши LMS-системи видно, що всі вони мають свої сильні та слабкі сторони. Найоптимальнішою з наведених систем є Coursera, в неї найбільше позитивних якостей в порівнянні з іншими системами, найгіршою системою виявилась Blackboard, що очікувано, оскільки система не розвивається.

Таблиця 1.2

Порівняння характеристик безпеки

	MOODLE	Blackboard	Classroom	Coursera
Авторизація за Логін + Пароль	+	+	-	+
LDAP [23]	+	+	-	-
SSO [22]	-	+	+	-
CAS	+	+	-	-
2FA	+	+	Google Auth	+
Social Net Auth	Google, Facebook або X	-	Google	Google, Facebook
OAuth 2.0	+	+	+	+
OTP	Email	-	-	-
Використання сертифікатів	+	-	-	-

Табл. 1.2 Порівняння характеристик безпеки

Проаналізувавши та порівнявши особливості систем сформуємо короткі та попередні вимоги до майбутньої системи: вона повинна мати сучасний дизайн, мінімалістичний та зручний UX, система має збирати та надавати статистику успішності студентів, успішність виконання завдань, витрачений час на кожне завдання і на курс цілком, статистика проходження курсів (успішно / невдало). Окрім цього система також має надавати таку аналітику: розташування студентів, час проходження курсів, успішність студентів, курсів та завдань, швидкість відповіді на завдання у вигляді графіків, таблиць та діаграм.

З огляду безпеки система повинна підтримувати такі методи авторизації: логін + пароль, LDAP, SSO, CAS, через соціальні мережі, підтримувати протокол OAuth 2.0 та мати можливість отримувати OTP через SMS та Email.

ВИСНОВКИ ДО РОЗДІЛУ 1

У цьому розділі детально досліджено концепцію систем управління навчанням (LMS-систем) і визначено, що ця система є ключовим інструментом для організації та управління навчальними процесами та ресурсами. LMS дозволяє забезпечувати ефективне навчання незалежно від середовища, будь то вищий навчальний заклад чи бізнес-компанія.

У бізнесі системи управління навчанням допомагають навчати та розвивати працівників, забезпечують дистанційне навчання та відстежують прогрес працівників. У навчальних закладах системи управління навчанням допомагають організувати навчання, надають навчальні матеріали та допомагають вчителям вести журнали та оцінювання, а учням відстежувати прогрес навчання.

Аналіз приводить до важливого висновку: системи управління навчанням відіграють ключову роль у сучасній освіті, допомагаючи забезпечити доступність, ефективність і якість освіти. Системи LMS відрізняються за своєю функціональністю та характеристиками, що робить їх важливими об'єктами подальших досліджень і розробок.

Порівнюючи LMS-системи, такі як Moodle, Blackboard Learn, Google Classroom, Coursera, можна визначити, що кожна з них має свої унікальні характеристики і функціонал, спрямований на вирішення конкретних завдань у сфері навчання та управління освітнім процесом, проте у всіх систем є певні недоліки.

Moodle вражає своєю гнучкістю та відкритістю, дозволяючи розширювати функціонал за допомогою плагінів і модулів. Blackboard Learn надає рішення для вищих навчальних закладів і підтримує різні види навчання. Google Classroom пропонує інтуїтивно зрозуміле інтерфейс та інтеграцію з іншими Google-сервісами. Coursera вигідний для масштабного вивчення та онлайн-курсів, забезпечуючи доступ до великої кількості курсів від провідних університетів та організацій світу.

Таким чином розробка таких систем є актуальною через стрімкий розвиток онлайн-освіти та необхідністю ефективного управління освітнім процесом в сучасних умовах. Інтеграція технологій у навчання полегшує доступ до знань і надає можливість вивчати матеріали в будь-якому місці і в будь-який час. Розробка LMS-систем важлива для створення зручного та ефективного середовища для навчання, що відповідає потребам сучасної освіти.

РОЗДІЛ 2

ВИМОГИ, ОПИС СИСТЕМИ, ПІДГОТОВКА ДО РОЗРОБКИ СИСТЕМИ

У сучасному інформаційному суспільстві, де навчання стає все більш цифровим та дистанційним, системи управління навчанням набувають надзвичайно важливого значення. Однак, перш ніж впроваджувати систему LMS в освітні та бізнес-процеси, необхідно визначити вимоги до програмного забезпечення для цієї системи. Вимоги до програмного забезпечення визначають, якими технічними параметрами і функціями повинна володіти система LMS, щоб задовольнити потреби користувачів і забезпечити ефективність навчання та управління.

Вимоги до програмного забезпечення дуже різноманітні і залежать від використовуваної системи LMS. Щонайменш, вони визначають технічні характеристики системи, такі як необхідне апаратне забезпечення, операційна система та база даних, а також функціональні вимоги, які визначають функції, якими повинна володіти система для задоволення потреб користувачів і бізнесу.

У цьому розділі детально розглядаються різні типи вимог до програмного забезпечення для системи LMS та їхній вплив на вибір і впровадження конкретних технологій, що будуть використані в системі. Тут також розглядаються питання сумісності, масштабованості та безпеки, які відіграють ключову роль в успішному функціонуванні системи LMS. Визначення та розуміння вимог до програмного забезпечення є важливим кроком у впровадженні системи LMS та забезпеченні її ефективної роботи для користувачів.

2.1 ФУНКЦІОНАЛЬНІ ВИМОГИ

Функціональні вимоги є частиною вимог до програмного забезпечення системи і визначають, які функції та можливості повинна мати система, щоб задовольнити потреби користувачів і досягти конкретних цілей. Ці вимоги

описують бажану функціональність системи і визначають, як система повинна взаємодіяти з користувачами та іншими системами.

Функціональні вимоги можуть бути докладно описані і включати в себе такі елементи, як:

- Опис функцій: Це опис конкретних функцій, які повинна виконувати система. Наприклад, LMS має надавати можливість створювати, редагувати та видаляти навчальні курси.
- Вимоги до взаємодії з користувачем: Описують, як користувачі взаємодіють з системою. Наприклад, повинен бути користувацький інтерфейс, що дозволяє зареєстрованим користувачам входити в систему і переглядати доступні курси.
- Вимоги до обробки даних: Вони визначають, як система повинна обробляти, зберігати та відображати дані. Наприклад, система повинна вміти завантажувати, зберігати та відображати навчальні матеріали.
- Вимоги до безпеки та доступу: Вони визначають, як система повинна забезпечувати захист даних та обмеження доступу до різних функцій системи.
- Вимоги до продуктивності та масштабованості: Вони визначають, як система повинна працювати в умовах великої навантаженості та забезпечувати швидку реакцію на запити користувачів.

Для LMS-системи визначено такі функціональні вимоги:

- Авторизація користувачів;
 - За логіном та паролем;
 - За допомогою Google, Facebook, LinkedIn [10] та ін.
- Реєстрація користувачів;
- Відновлення паролю;
- Підтримка імплементації в бізнес-системи та навчальні заклади;
- Можливість переглядати календар з поточними /майбутніми подіями та курсами;
- Інтеграція з Google Calendar;

- Можливість переходити на різні сторінки застосунку;
- Можливість переглядати курси в різних режимах відображення;
- Захист даних користувачів від несанкціанованого доступу;
- База даних має бути індексована для швидкого отримання даних;
- Надійне зберігання авторських прав та видалення контенту що його порушує;
- Можливість зміни інформації користувача.

Розглянемо вимоги для реалізації вчителя / навчальної організації:

- Можливість створювати курси;
- Можливість завантажити і прикріпити файли, відео;
- Можливість створювати завдання в курсах;
- Можливість призначити завдання конкретному студенту;
- Можливість згенерувати посилання-запрошення на курс;
- Можливість отримати посилання на навчальну організацію / викладача;
- Можливість дати і обмежити доступ до курсу;
- Можливість оцінити роботу студента у випадку, якщо завдання не перевіряється автоматично;
- Можливість подивитись журнал оцінок всіх студентів;
- Можливість подивитись аналітику успішності студентів, що включає різноманітні графіки та діаграми;
- Можливість залишати повідомлення учню;
- Можливість встановити подію для учнів в календар;

Розглянемо вимоги для реалізації учня:

- Можливість під'єднуватись та покидати курси;
- Можливість проходити курси і виконувати завдання в них;
- Можливість переглядати оцінки за виконані завдання;
- Можливість перегляду всіх / поточних / пройдених курсів;
- Можливість пошуку курсу;

- Можливість подивитись аналітику проходження навчальних матеріалів;
- Отримувати сповіщення про поточні та майбутні дедлайни за допомогою SMS / Email / Push сповіщень;
- Можливість написати та отримати відповідь від вчителя всередині системи;
- Можливість оцінки курсу після проходження;
- Можливість попереднього перегляду та завантаження файлів і відео;
- Трекінг витраченого часу;

2.2 НЕФУНКЦІОНАЛЬНІ ВИМОГИ

Нефункціональні вимоги – це частина програмних вимог до системи, яка визначає якісні характеристики та обмеження системи, які не стосуються конкретної функціональності. Такі вимоги визначають параметри, що впливають на ефективність, надійність та інші аспекти системи.

Нефункціональні вимоги можуть включати в себе такі елементи, як:

- Вимоги до продуктивності: Ці вимоги визначають швидкість обробки та відповіді системи на запити користувачів. Наприклад, система повинна завантажувати головну сторінку і сторінку навчального курсу не довше певного проміжку часу.
- Вимоги до безпеки: Вони визначають рівень захисту даних та доступу до системи. Наприклад, система повинна забезпечувати надійне шифрування даних та аутентифікацію користувачів.
- Вимоги до надійності: Ці вимоги стосуються стійкості системи до відмов та відновлення роботи у випадку критичних збоїв. Наприклад, система повинна мати регулярні резервні копії даних та самовідновлюватись у випадку критичних збоїв.
- Вимоги до масштабованості: Вони визначають, як система може розширюватися та обробляти зростаючий обсяг даних та користувачів.
- Вимоги до сумісності: Вони стосуються можливості системи співпрацювати з іншими програмами та платформами. Наприклад, система повинна бути

сумісною з всіма популярними веб-браузерами (підтримка старих версій не є обов'язковою) та операційними системами.

Після визначення поняття нефункціональних вимог створюємо список нефункціональних вимог для нашої системи:

- Система повинна забезпечувати швидке завантаження сторінок та матеріалів для користувачів. Користувач повинен побачити перше відображення сторінки менше ніж за 2 секунди. (контент всередині сторінки може бути завантажений пізніше);
- Захист конфіденційності та цілісності користувацьких даних;
- Можливість автентифікації користувачів та керування правами доступу;
- Захист від зловмисних атак, таких як SQL-ін'єкції та крос-сайтові скрипти (XSS);
- Можливість автоматичного відновлення роботи системи після відмов або аварій;
- Регулярне створення резервних копій даних, не рідше одного разу в годину. В кращому випадку мати два, або більше, дублюючих сервери;
- Здатність системи розширюватися для обробки зростаючого обсягу даних та кількості користувачів;
- Сумісність з популярними веб-браузерами та операційними системами. В розрахунок береться підтримка версій останніх 2-х років;
- Здатність інтегруватися з іншими додатками та сервісами, такими як Google Calendar, Google Auth, Apple ID та ін.;
- Підтримка різних мов та локалізованих версій, мінімальна кількість підтримуваних мов – 2: Англійська та Українська;
- Здатність коректно працювати в різних часових зонах та з локальними форматами (наприклад дати, валюти);

- Використання мінімально допустимих обсягів пам'яті (до 300мб ОЗП без врахування відтворення контенту, наприклад, відео) та обчислювальних ресурсів;
- Здатність підтримувати велику кількість курсів та одночасно активних користувачів;
- Інтуїтивний та зручний інтерфейс користувача (UX);
- Приємний дизайн без різкої зміни кольорів, приємні пастельні кольори домінують (UI);
- Здатність користуватися системою на смартфонах та планшетах;
- Підтримка адаптивного дизайну;
- Можливість завантаження та відтворення різних форматів навчального контенту, таких як відео, аудіо, тексти та інше;
- Засоби для відстеження активності користувачів та оцінки ефективності навчання;
- Здатність бази даних масштабуватися для забезпечення швидкого доступу до інформації;
- Здатність системи оновлювати та розширювати функціональність без великих перерв у роботі.

2.3 АРХІТЕКТУРА СИСТЕМИ ТА СЦЕНАРІЇ ВИКОРИСТАННЯ

Архітектура системи — це концептуальний склад компонентів та їхніх взаємозв'язків в межах програмного забезпечення. Вона визначає основні принципи організації та взаємодії всіх елементів системи. Це важливий момент планування перед початком розробки, оскільки вона визначає стратегічні аспекти проєкту та визначає основи для подальшого розвитку.

Розробка архітектури передуює фазі кодування і дозволяє команді розробників зрозуміти всю картину системи. Це включає в себе розуміння ключових компонентів, моделювання взаємодій, визначення структурних принципів.

Архітектурне проектування допомагає уникнути проблем, таких як невідповідність вимогам, непрофесійна інтеграція, або низька масштабованість, що можуть виникнути на етапі реалізації.

Зокрема, важливість архітектури полягає в тому, що вона створює фундамент для надійної, розширюваної та легко змінюваної системи, спрощуючи роботу з усією командою розробників та забезпечуючи ефективне управління ресурсами.

Застосунок було розподілено на функціональні частини, певні домени – великі програмні частини, які містять в собі менші частини з функціоналом.

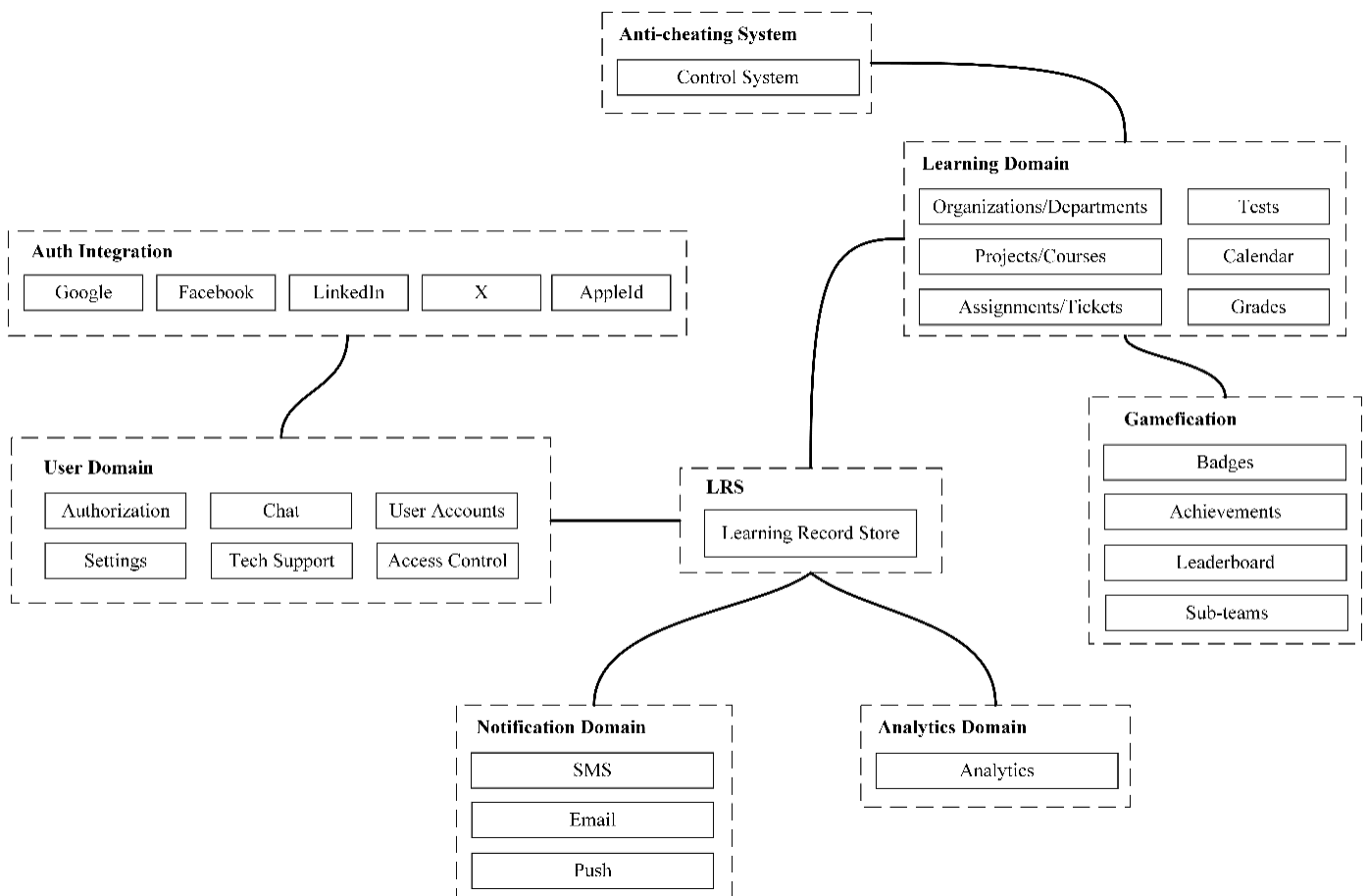


Рис. 2.1 Архітектура застосунку

- LRS – Головний концентратор, що відповідає за взаємодію всіх інших доменів
- User Domain – Домен, що відповідає за основний користувацький функціонал, такий як авторизація, чат, управління акаунтами, управління доступом.

- Learning Domain – Домен, що відповідає за основний функціонал застосунку – частину, що пов’язана з навчанням. Цей домен відповідає за курси, завдання, призначення їх студентам, оцінювання, календар, робота системи в організаціях. Цей домен містить також два модулі – Анти-Чітинг система та Геймефікація.
- Analytics Domain – Домен відповідає за збір та обробку аналітики.
- Notification Domain – Домен, що відповідає за оповіщення користувачів.

Для реалізації системи оберемо класичну клієнт-серверну архітектуру. Клієнт-серверна архітектура – це підхід до побудови програмних застосунків, при якому загальна функціональність програми поділяється на дві основні складові: клієнт і сервер. Клієнт – це програмна частина, що працює на пристрої користувача – сторінка в веб-браузері, і надає інтерфейс для взаємодії з додатком. Сервер – це віддалений комп’ютер або система, яка зберігає, обробляє і передає дані між клієнтами. Сервер в свою чергу встановлює зв’язок з базою або базами даних, там записує, оновлює дані в них. Це стандартні CRUD операції для роботи з даними в БД.

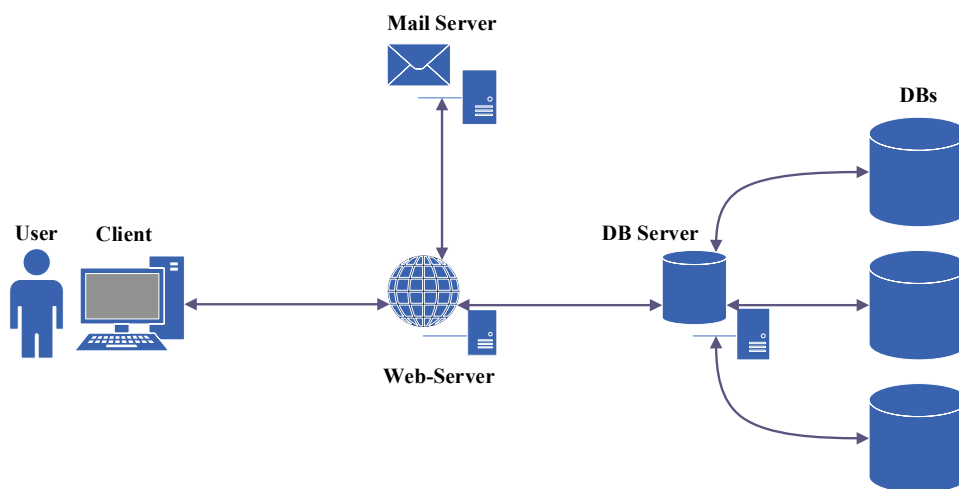


Рис. 2.2 Клієнт-серверна модель

Сценарії використання системи - це набір конкретних ситуацій, умов або набір дій, що описують, як система буде використовуватися користувачами або сторонніми зацікавленими сторонами для досягнення певної мети або завдання [11]. Сценарії використання показують, як функціональні та нефункціональні вимоги системи взаємодіють з користувачами і як система реагує на дії користувачів.

Ці сценарії створюються для того, щоб визначити різні можливі варіанти використання системи і забезпечити чітке розуміння того, як система повинна функціонувати в реальному середовищі. Вони допомагають розробникам, менеджерам проєктів та іншим зацікавленим сторонам зрозуміти, як система буде функціонувати в реальних ситуаціях, і визначити, чи відповідає вона потребам користувачів.

Сценарії використання використовуються для створення тестових кейсів, проектування користувацьких інтерфейсів і загалом для уточнення вимог до системи. Варіанти використання системи є важливою частиною процесу розробки програмного забезпечення і відіграють ключову роль у забезпеченні відповідності системи потребам та очікуванням користувачів. Основні сценарії використання системи наведені в Додатку 1.

2.4 ВИБІР ТИПУ ЗАСТОСУНКУ

У сучасному світі інформаційних технологій дуже важливо обрати тип додатку, який дозволить реалізувати ідею стартап-проєкту. Існує три основні типи додатків: десктопні, мобільні та веб-застосунки. Для розробки обрано веб-застосунок як основний майданчик, пізніше буде реалізовано також і мобільний застосунок. Нижче будуть наведені конкретні переваги веб-застосунків та чому це є найкращим вибором для нашого стартап-проєкту [12].

- **Доступність:** веб-додатки пропонують значні переваги доступності для користувачів. Веб-додаток можна використовувати на будь-якому пристрої з веб-браузером, включаючи комп'ютери, ноутбуки, смартфони, планшети, телевізори та смарт-годинники. Це робить платформу доступною для

абсолютної більшості користувачів, без обмежень за типом пристрою або операційної системи.

- Зручність: веб-додатки можна використовувати без встановлення на пристрої користувача додаткових програмних засобів, оскільки веб-браузер є у всіх користувачів інтернету. Користувачам не потрібно завантажувати та оновлювати програмне забезпечення на своїх пристроях, що робить використання нашої системи LMS максимально зручним і простим.
- Сумісність: веб-додатки сумісні з широким спектром операційних систем – немає необхідності розробляти окремі версії для Windows, MacOS, Android, iOS та ін. Один веб-додаток буде коректно працювати на будь-якій платформі (без врахування особливостей роздільної здатності дисплеїв).
- Оновлення та підтримка: однією з суттєвих переваг веб-додатків є те, що оновлення та підтримка можуть надаватися централізовано. Зміни та покращення можуть вноситися на сервері, в реальному часі, та автоматично ставати доступними для користувачів без необхідності перевстановлення або оновлення застосунку.
- Масштабованість: веб-додатки можуть легко розширювати інфраструктуру та значно збільшувати кількість користувачів без необхідності будь-яких дій з боку користувача.

Обираючи веб-версію для нашого стартап-проекту ETIMS, ми надаємо користувачам можливість доступу до нашої LMS-системи в будь-який час і з будь-якого пристрою за наявності інтернет-підключення, забезпечуючи зручність, доступність і простоту використання. Хоча також буде додана підтримка мобільних застосунків для систем iOS та Android, ключовим представленням залишається веб-версія.

Використання архітектури клієнт-сервер для стартап-проекту ETIMS є стандартним, вдалим вибором з наступних причин [13]:

- Обробка та обчислювальна складність розділені: сервер відповідає за складну обробку та зберігання даних, в той час як клієнт відповідає за відображення користувацького інтерфейсу та взаємодію з користувачем. Це дозволяє ефективно використовувати ресурси та розподіляти завдання між компонентами. Хоча в проекті буде реалізовано SSR, більшість обчислювань з відмальовування даних буде лежати на користувачах, що не буде мати великого впливу на користувачів, проте суттєво зменшить навантаження на сервери.
- Зручність і доступність: клієнтські програми можуть працювати на різних типах пристроїв і операційних систем. Навіть при створенні всіх видів застосунків, буде зберігатись крос-платформеність та дані з серверу будуть отримані на будь-якій платформі. Це дає користувачам широкий вибір варіантів доступу до навчальної системи.
- Безпека і контроль: сервер може контролювати доступ до даних і функцій, забезпечуючи таким чином безпеку і захист інформації. Це унеможливорює «злом» та «піратство» таких застосунків, це є вагомим аргументом для вибору такого підходу.
- Масштабованість: клієнтські програми можна легко оновлювати, а оновлені версії автоматично стають доступними для всіх користувачів. Сервери можуть бути безшовно розширені без будь-яких наслідків для користувачів.

Вибір архітектури клієнт-сервер забезпечує гнучкість, масштабованість і простоту використання, що є ключовими факторами в реалізації стартап-проекту ETIMS.

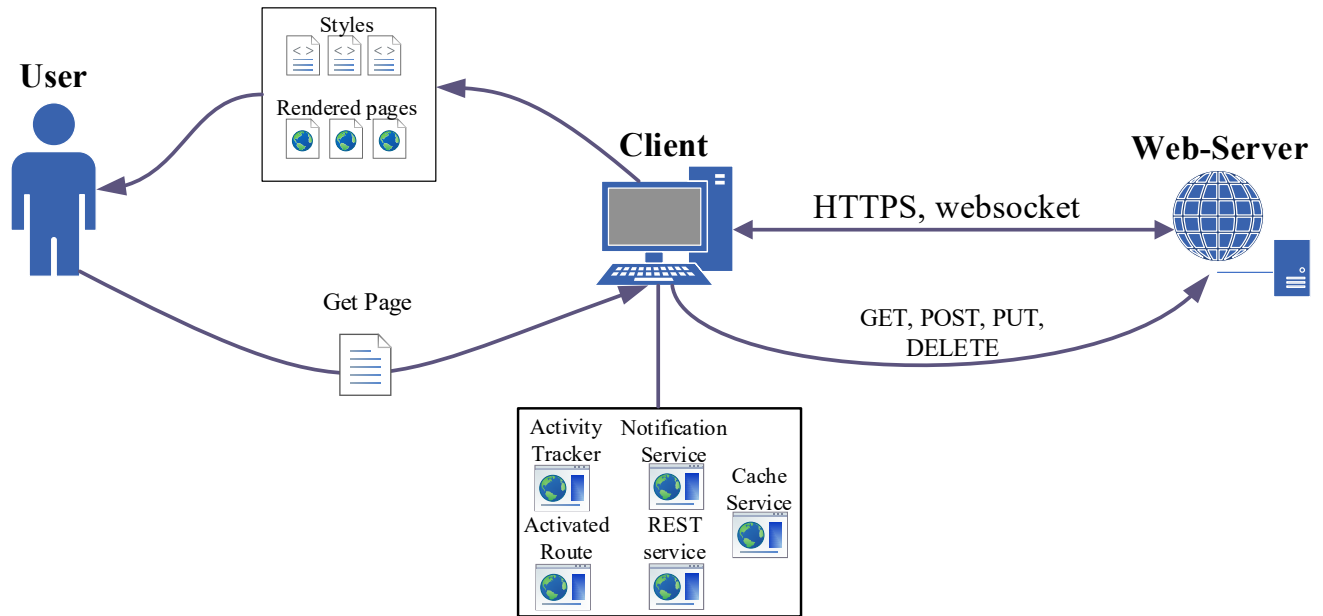


Рис. 2.3 Архітектура клієнтської частини застосунку

На даному рисунку зображено архітектуру клієнтської частини застосунку. Користувач взаємодіє з клієнтом у вигляді веб-сторінки, виконуючи певні дії (actions), які обробляє клієнт та звертається до сервера за необхідності отримання даних.

Клієнт може кешувати запити, наприклад, можна ввімкнути кешування якихось умовно статичних даних, наприклад списку tabs, або списку полів форми для створення форми, що пришвидшить відображення сторінки та зменшить навантаження на мережу та сервер. Клієнт також містить модулі для відстеження активності користувача, надсилання повідомлень, взаємодії з сервером та зміни і відстеження url.

2.5 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ FRONT-END ЧАСТИНИ ЗАСТОСУНКУ

Фронтенд-частина застосунку – це частина програмного продукту, що відповідає за інтерфейс користувача та взаємодію з ним. Це та частина, яку бачить користувач і з якою взаємодіє на екрані свого пристрою. Інтерфейс включає в себе певну структуру елементів на веб-сторінці, стилі, анімації та логіку взаємодії з цими елементами, що дозволяє користувачеві користуватись програмним продуктом.

Окрім цього фронтенд-частина відповідає за взаємодію між користувачем та сервером за допомогою API.

Фреймворки та інструменти для розробки інтерфейсу допомагають розробникам створювати ефективні, інтерактивні та зручні інтерфейси. Візуальна частина програми відповідає за відображення інформації, обробку подій, перевірку вхідних даних та багато інших аспектів, пов'язаних із взаємодією з користувачем.

Для веб-додатків інтерфейс зазвичай пишеться за допомогою таких інструментів, як HTML, CSS і JavaScript / TypeScript. Проте це не суворі обмеження, фронтенд частина може бути написана за допомогою мов програмування PHP, Python та C#. Для реалізації застосунків використовують різні фреймворки та бібліотеки, щоб суттєво спростити розробку, уніфікувати програмні рішення та забезпечити кращу продуктивність.

Angular, React і Vue - це три найпопулярніші фреймворки для розробки фронтенд-частини веб-застосунків, кожен з яких має свої особливості та переваги. Нижче коротко розглянемо всі три та оберемо оптимальніший варіант [14].

Angular

Angular - це сучасний та потужний фреймворк для розробки веб-застосунків, який має свої особливості та характеристики. Angular має величезне ком'юніті, безліч бібліотек, які задовільняють будь-яким потребам.

Переваги:

- Використання мови програмування TypeScript дозволяє писати більш надійний та чистий код, типізація допомагає розробникам виявляти безліч помилок ще на етапі написання коду, це також полегшує підтримку проєкту в майбутньому.
- Angular пропагандує та спонукає використовувати модульну структуру, це допомагає розробникам легко організовувати та підтримувати код,

робить модулі незалежними, та легкими для будь-яких міграцій, що особливо актуально в великих застосунках.

- Фреймворк надає різноманітні можливості для створення складних застосунків, включаючи роботу з HTTP-запитами, маршрутизацію, автентифікацію та особливо роботу з формами, це є killer-feature даного фреймворку.
- Angular має велику та активну спільноту розробників, яка надає безліч ресурсів, бібліотек, модулів та інструкцій для полегшення розробки.

Мінуси:

- Angular може здаватися складним для новачків або тих, хто раніше не працював з TypeScript, не мав досвіду та уявлення модулярності, або сучасними фронтенд-технологіями.
- Зазвичай, Angular генерує більше boilerplate коду в порівнянні з іншими фреймворками, що може призвести до збільшення розміру застосунку.
- Перехід на нові версії Angular або міграція існуючих проєктів може бути непростим завданням через зміни в API та архітектурі. Такі випадки особливо актуальні при переході одразу на декілька версій, наприклад з 12 на 17 версію.

Не зважаючи на мінуси, Angular залишається популярним вибором для створення великих та потужних веб-застосунків завдяки своїм можливостям та активній спільноті розробників.

React

React – це потужний фреймворк для розробки фронтенд застосунків, він має багато схожого з Angular, проте відрізняється структура компонентів, React має менше розділення на темплейти, логіку та стилі.

Переваги:

- React сприяє структурованому та легкому читанню коду завдяки використанню компонентів.
- Віртуальний DOM допомагає покращити продуктивність застосунка та зменшити завантаження сервера.
- Існує величезна кількість сторонніх бібліотек, розширень та інструментів для React, що робить його дуже потужним і розширюваним.
- React сприяє тестуванню компонентів та логіки застосунка.

Недоліки:

- Навчання React може бути більш вимогливим для новачків у веб-розробці, в порівнянні з іншими фреймворками.
- Велика кількість сторонніх бібліотек і розширень ускладнює вибір підходящої бібліотеки та наражає на ризик, що розробник перестане підтримувати бібліотеку, що викличе серйозні проблеми з міграцією на іншу бібліотеку.
- Управління станом додатку може стати складним у великих проєктах.

Загалом, React є потужним і популярним фреймворком для розробки фронтенд-частини веб-застосунків.

Vue

Vue – це фреймворк, що використовує MVVM шаблон для створення інтерфейсів користувача беручи за основу моделі даних за допомогою реактивності.

Переваги:

- Vue має дуже низький поріг входження і дозволяє швидко розпочати роботу з ним, навіть для новачків.
- Реактивна система Vue робить роботу з даними і відображенням простою та ефективною.

- Vue має активну спільноту розробників, що означає наявність безлічі розширень, бібліотек і інструментів.

Недоліки:

- Незважаючи на активну спільноту, Vue все ж не має такої кількості бібліотек та інструментів, як у React або Angular.
- Обмежена великість проєктів: В деяких випадках Vue може бути менш практичним для великих та складних проєктів у порівнянні з Angular або React.

Вибір фреймворку

Обираючи фреймворк для розробки фронтенд частини стартап-проєкту ETIMS, було обрано фреймворк Angular. Це рішення ґрунтувалося на таких факторах та особливостях.



Рис. 2.4 Логотип Angular

1. Однією з головних причин вибору Angular був досвід розробки односторонніх додатків з використанням цього фреймворку. Досвід роботи з Angular дозволить створювати якісний стартап, враховуючи знання кращих підходів та методологій розробки. Це дозволить пришвидшити розробку, зробити її ефективною і мінімізувати час, витрачений на вивчення нових інструментів.
2. Angular має модульну структуру, що дозволяє організувати код проєкту у легко зрозумілі блоки. Angular має класичну архітектуру MVC, в компонентах це відображається у вигляді файлу з виконуваним кодом, файлу стилів та шаблону. Angular дозволяє розподіляти обов'язки в команді та робить підтримку коду простішою через низьку зв'язність компонент при правильному використанні модульності.

3. Angular має величезне, активне ком'юніті розробників і має регулярні оновлення, раз в 6 місяців. Це означає, що існує безліч бібліотек та інструментів, що допомагають у розробці, підтримці проектів та зменшують необхідність писати свої рішення для всіх задач.
4. Angular пропонує багато можливостей «з коробки» для створення складних додатків та пропонує такі можливості: виконання HTTP-запитів, потужне використання форм та гнучкої маршрутизації.

Обґрунтовуючи свій вибір, поєднання досвіду розробки, модульності та широкого спектру можливостей робить Angular найкращим вибором для розробки ETIMS.

Вибір додаткових технологій для розробки

Для успішної розробки LMS ETIMS було використано не тільки фреймворк Angular та мову програмування TypeScript, а також додаткові інструменти, які сприяють ефективній роботі. У процесі створення застосунку команда розробників використовувала систему контролю версій Git для контролю версій, управління кодом та спільної роботи над проектом.

Для налагодження та тестування невід'ємною частиною розробки стало використання веб-браузера Google Chrome та Opera GX, що дозволило досліджувати та налагоджувати фронтенд частину. Окрім цього, для розробки використовується інтегроване середовище розробки IntelliJ IDEA для легкого написання та налагодження коду. Ці інструменти сприяють розробці якісної та надійної системи управління навчанням.

ВИСНОВКИ ДО РОЗДІЛУ 2

У цьому розділі детально розглянуті функціональні та нефункціональні вимоги до системи. Функціональні вимоги визначають, якими функціями та можливостями повинна володіти система для задоволення потреб користувачів навчальних інститутів та бізнесу, тоді як нефункціональні вимоги встановлюють якісні характеристики та обмеження системи, що впливають на її ефективність та надійність.

Після аналізу вимог було визначено, що система LMS повинна відповідати високим стандартам з точки зору продуктивності, безпеки та надійності. Вимоги до зручності використання та підтримки мобільних пристроїв також є важливими, оскільки вони впливають на користувацький досвід.

Крім того, система повинна бути гнучкою і масштабованою, щоб відповідати мінливому навчальному середовищу і зростаючим вимогам та потребам. Не слід також забувати про підтримку різних мов та локалей.

Загалом, правильне визначення вимог є важливим кроком у розробці LMS. Вони визначають майбутню функціональність системи і впливають на її успішне впровадження та використання. У наступних розділах розглядаються деталі проектування та впровадження цієї системи з урахуванням усіх наведених вимог.

Також розглянуто концепцію сценаріїв використання системи управління навчанням та їх важливість у контексті освіти та навчання. Описано основні варіанти використання LMS для загальних випадків і для різних ролей користувачів, таких як викладачі та студенти описано специфічні варіанти використання для кожної з цих груп. Аналіз сценаріїв використання дає чітке уявлення про те, як має працювати LMS-система, з чіткими end-to-end сценаріями.

Приклади використання LMS розкривають можливості, які система надає для доступу до навчальних ресурсів, взаємодії викладачів за студентами та з різними частинами системи, такими як інтерфейс, сторінки та ін. Вони показують, як

користувачі можуть отримати бажану інформацію або результат покроково, що робить користувацький досвід більш повним та приємнішим.

Розглянуто загальну архітектуру системи, де описано основні її складові частини LRS, User Domain, Learning Domain, Analytics Domain, Notification Domain та частини, які від них залежать.

Вивчення прикладів та сценаріїв використання LMS-системи є важливим кроком у розумінні та аналізі впливу цих систем на викладання, навчання та взаємодію користувачів у системі. Вони демонструють потенціал для подальшого розвитку та вдосконалення LMS з метою покращення навчального процесу та підвищення якості освіти.

Було проведено підготовку до розробки фронтенд-частини застосунку, цей етап був критично важливим для подальшого успішного втілення проєкту. Почавши з інсталяції необхідних утиліт та інструментів розробки, таких як IDEA, Git, Angular CLI, а також налаштування використання stylelint і eslint, забезпечено оптимальні умови для ефективної розробки та підтримки коду.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ

У даному розділі магістерської роботи буде проведено детальний огляд процесу створення та реалізації запропонованої системи. Даний етап дослідження визначає ключові аспекти розробки, включаючи створення основних компонентів, розробку сторінок, та впровадження важливих елементів, таких як авторизація за допомогою Keycloak і продуктивність системи.

Детально буде описано процес розробки дизайну застосунку, включаючи вибір та використання інструментів для дизайну. Крім того, буде звернуто увагу на важливість планування та визначення пріоритетів розробки для забезпечення успішності проекту. Буде розроблено систему.

3.1 РОЗРОБКА КОРИСТУВАЦЬКОГО ДИЗАЙНУ ЗАСТОСУНКУ

Для створення дизайну було використано інструмент Figma. Figma – це інструмент для створення прототипів веб-сервісів та дизайну, який дозволяє користувачам створювати та візуалізувати ідеї сторінок та інтерфейсів додатків.

Figma також дозволяє створювати інтерактивні прототипи, які можна тестувати та вдосконалювати до початку розробки. Цей інструмент відіграє важливу роль у розробці дизайну ETIMS, сервіс дозволяє отримувати CSS код елементів дизайну [15].

Для розробки дизайну було досліджено сучасні тенденції в розробці функціональних сторінок, з врахуванням кращих UX та UI практик.

Першим і найважливішим кроком у процесі створення дизайну для LMS був пошук референсів (початкових вхідних даних) та аналіз найкращих рішень у галузі. Цей етап включав в себе дослідження останніх тенденцій дизайну, вивчення інтерфейсів подібних систем та зберігання кращих рішень, в вигляді скріншотів, для подальшого використання в якості прикладу.

Під час пошуку вивчено як дизайн інших популярних систем, так і подібних веб-додатків. Такий підхід дозволив визначити ключові аспекти, які слід врахувати при розробці інтерфейсу ETIMS, а також отримати уявлення про ефективні дизайнерські рішення.

За допомогою зібраних джерел та найкращих практик визначено напрямки для подальшого розвитку дизайну, для відповідності сучасним стандартам та очікуванням користувачів.

Було зібрано з десяток різних варіантів та більше сотні скріншотів для різних випадків використання системи. Починаючи з головної сторінки, хедера і закінчуючи кнопками та формами для скидання паролю.

Після зібрання достатньої кількості першоджерел було почато процес дизайну. Дизайн зайняв суттєву кількість часу, було витрачено більше 30 годин часу для створення власного, продуманого, мінімалістичного та простого дизайну, визначено загальний дизайн для проєкту та створено макети для основних сторінок.

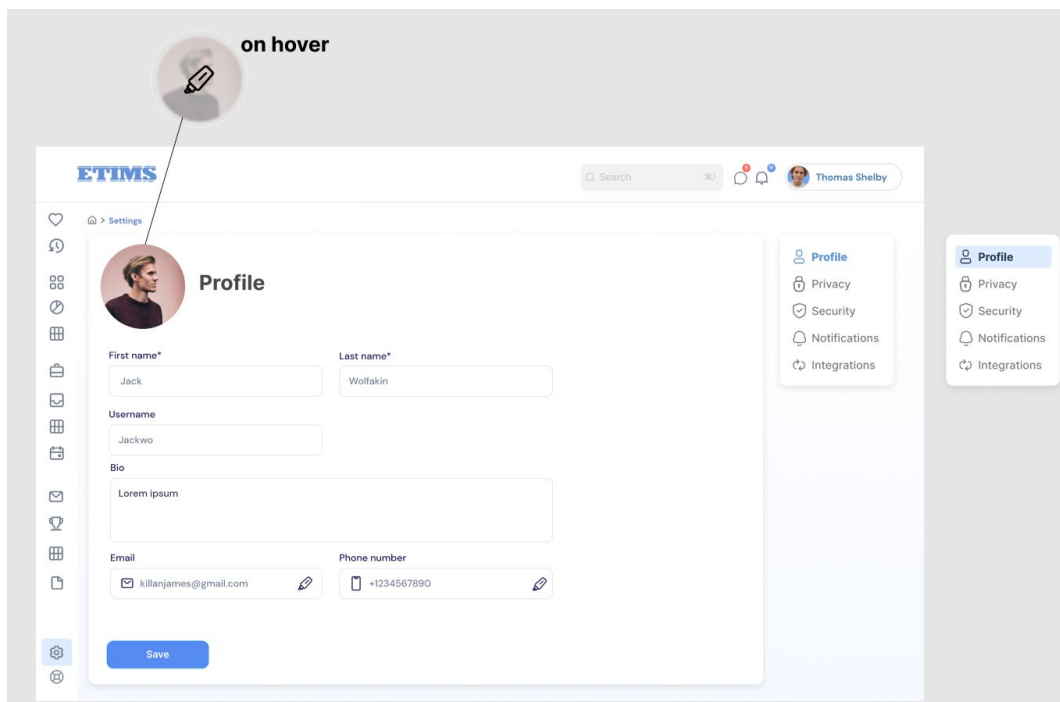


Рис. 3.1 Дизайн сторінки налаштувань

Сторінка з курсами повинна мати мінімалістичний і достатньо зрозумілий дизайн, оскільки це та частина застосунку, якою користувачі будуть користуватися найбільше.

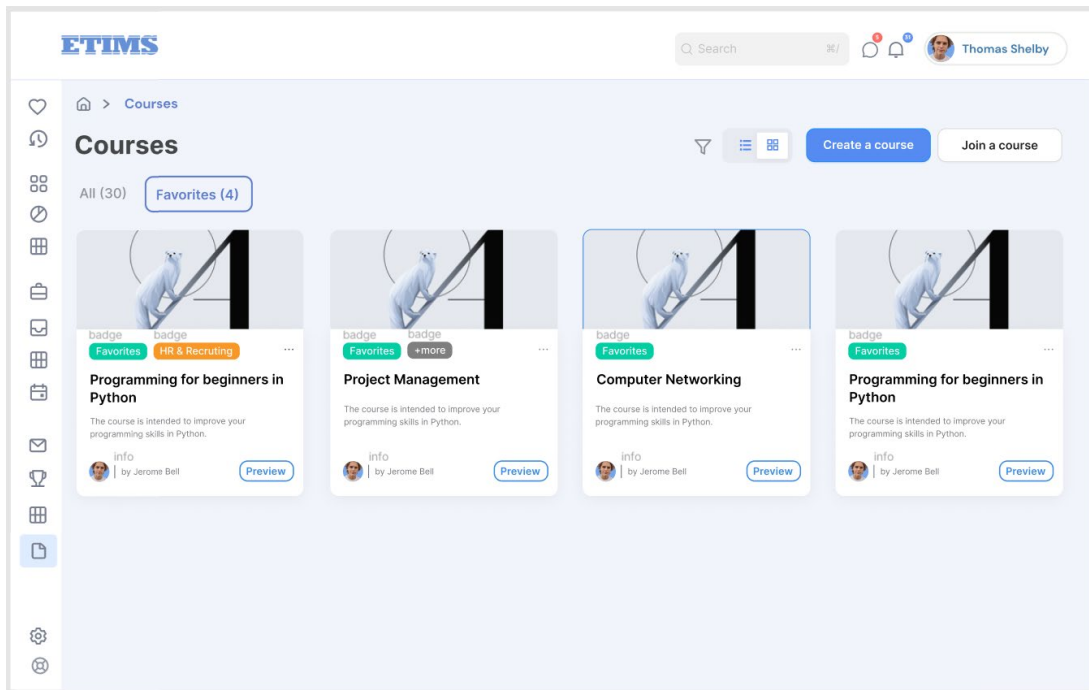


Рис. 3.2 Дизайн сторінки з списком поточних курсів. Плитки

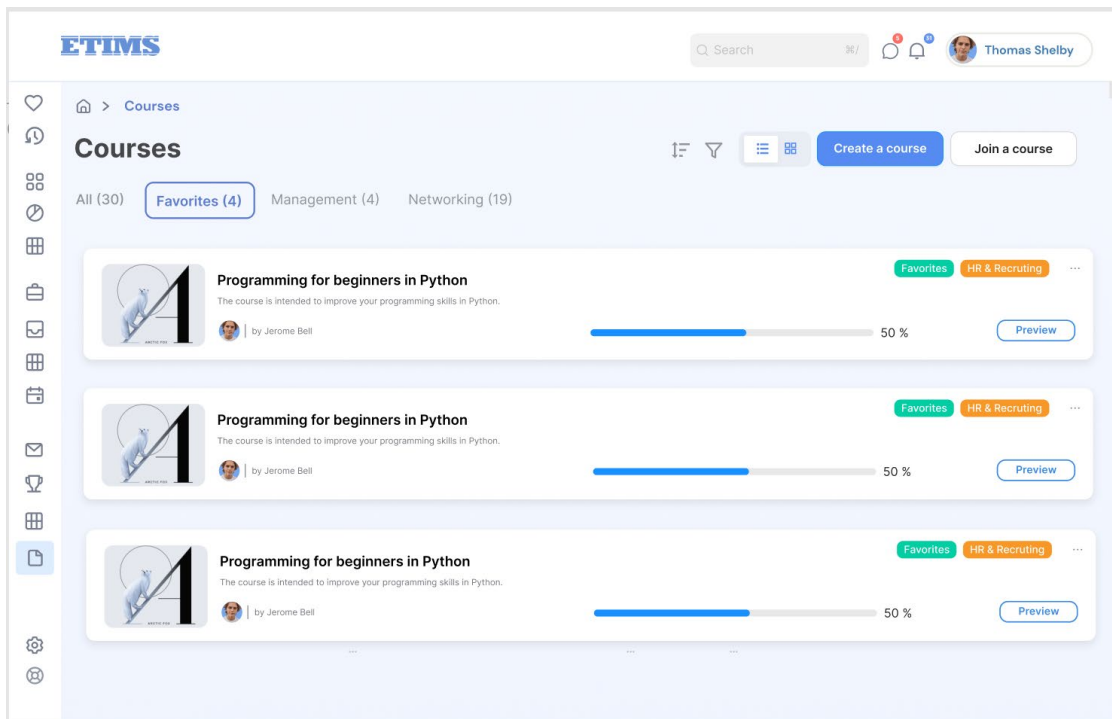


Рис. 3.3 Дизайн сторінки з списком поточних курсів. Рядки

Дрібні частини застосунку також дуже важливі, оскільки полегшують навігацію та загальний досвід користування застосунком.

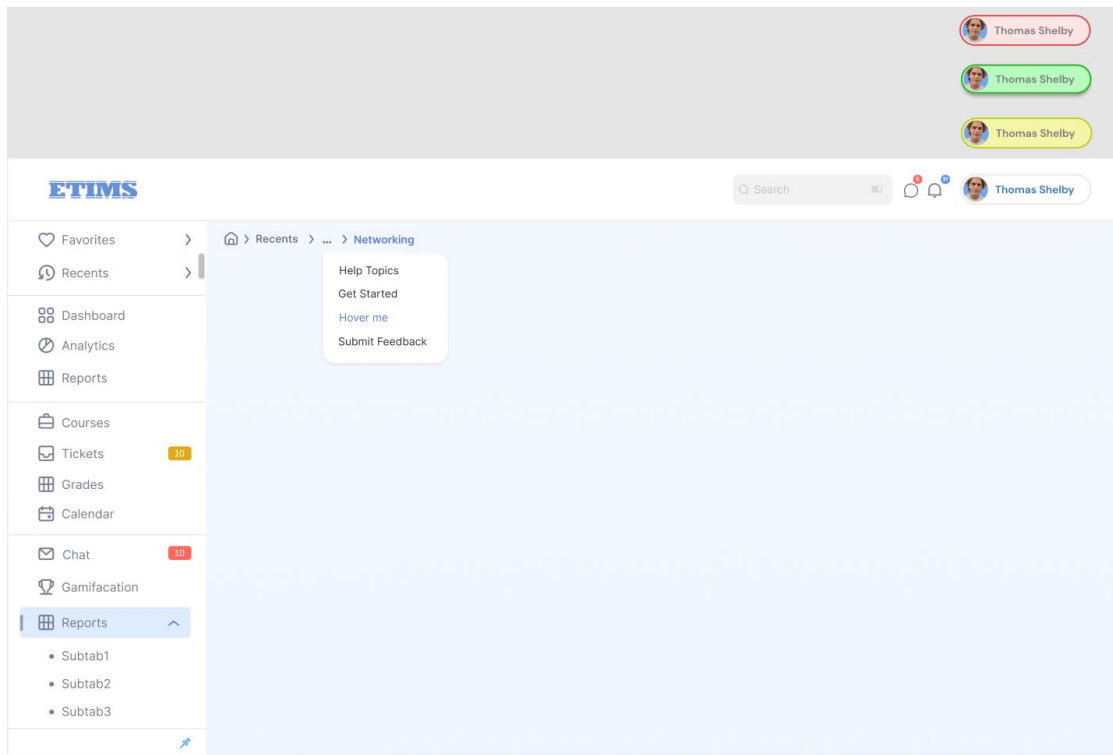


Рис. 3.4 Дизайн sidebar, breadcrumbs, header

На цьому етапі розробки зроблено великий крок вперед у створенні дизайну проекту. Було розроблено дизайн більшості ключових сторінок системи, включаючи головну сторінку, головну панель управління, сторінку входу, індивідуальні профілі користувачів та сторінки курсів.

Дизайн базується не лише на естетиці, але й на функціональності та користувацькому досвіді. Дизайн визначив кольорову палітру, шрифти та структуру сторінок. Він також підкреслив унікальний стиль і бренд ETIMS, що відповідає його цілям і цільовій аудиторії.

Створені дизайни лягли в основу подальшої розробки функціонального інтерфейсу системи, а також визначили основну концепцію дизайну та напрямок проекту. Це забезпечило єдиний візуальний стиль та ефективну розробку системи на наступному етапі.

3.2 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

Для початку розробки клієнтської частини застосунку необхідно встановити деяке додаткове ПЗ.

- Node.js – середовище виконання JS
- Angular CLI – інструмент командного рядка для роботи з Angular
- IntelliJ IDEA – IDE
- LESS – препроцесор для CSS
- NPM – пакетний менеджер

3.2.1 Основні налаштування

Одразу тримаємо курс на модулярність, згідно філософії Angular. Для полегшення розробки одразу локально налаштовуємо Ангуляр в файлі `angular.json`, Ангуляр автоматично згенерував цей файл при створенні проєкту, також він заповнив його стандартними налаштуваннями. Звертаємо увагу на наш проєкт, наразі це єдиний проєкт в даному репозиторії, переходимо до схематиків (користувачьких шаблонів генерації коду).

Schematics в Angular – це потужний інструмент, що дозволяє автоматизувати та розширювати процес генерації коду в Angular-додатках, Ангуляр схематики дозволяють також створювати власні генератори, проте зараз в цьому немає необхідності, обмежимося налаштуваннями стандартних генераторів. Налаштуємо Ангуляр застосунок для використання «strict» режиму. Це встановлює строгі правила до використання певних анотацій та посилює перевірки, які заблокують запуск застосунку при недотриманні цих вимог. Це дозволить покращити якість коду та майбутню підтримку, в ряді випадків це може допомогти уникнути зайвих помилок. Для генерації компонентів встановлюємо стандартний формат стилів – LESS. Для пришвидшення роботи застосунку встановлюємо `changeDetection` в `OnPush`, хоча це може бути непомітно в невеликих застосунках, проте в великих системах це суттєвий момент. В даній роботі можуть детально не розглядатися та не

пояснюватися незначні аспекти реалізації, які не суттєво впливають на процес реалізації програмних рішень. Встановимо також за замовчуванням standalone, це зробить компоненти чистішими та зменшить кількість залежностей. Встановимо також префікс для наших програмних файлів – «ets».

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "etims-web": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:application": {
          "strict": true
        },
        "@schematics/angular:component": {
          "style": "less",
          "changeDetection": "OnPush",
          "standalone": true
        }
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "ets",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
```

Рис. 3.5 Частина конфігурації проєкту

3.2.2 Визначення внутрішньої структури

Здебільшого застосунок має класичну для модульного застосунку структуру, в корені проєкту є папка .git для системи Git-контролю версій, .idea з локальними налаштуваннями IDE в даному проєкті, node_modules – папка де зберігаються всі бібліотеки та src – це папка в якій знаходяться всі файли розробки, окрім цього в корні проєкту знаходяться також конфігураційні файли.

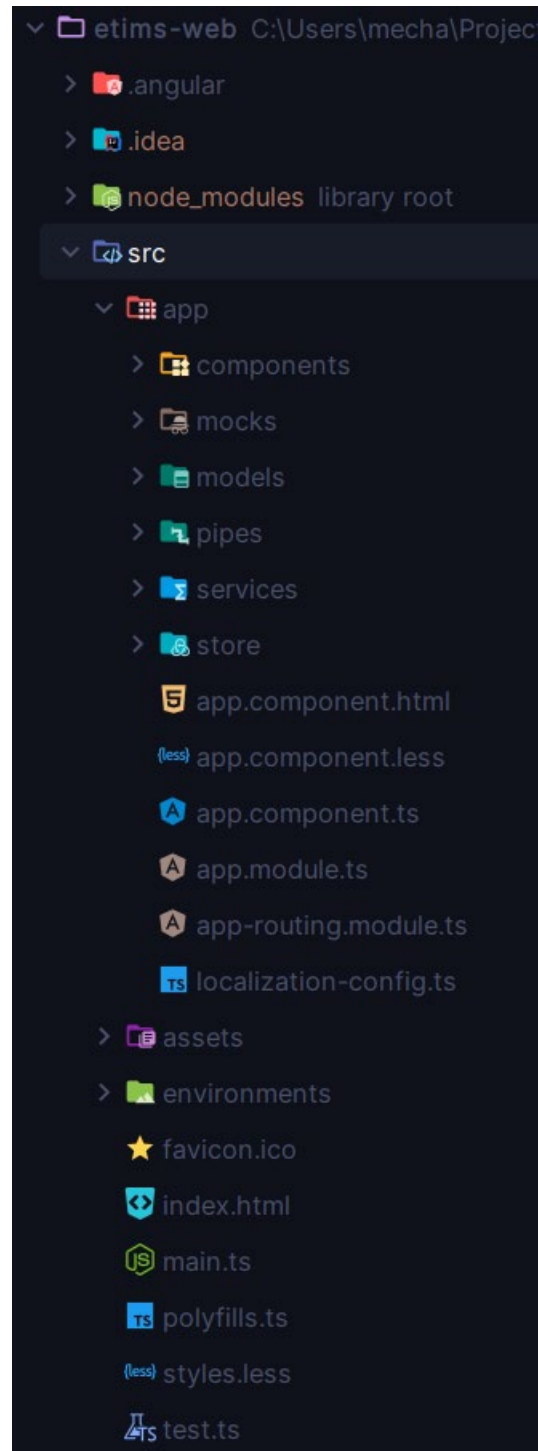


Рис. 3.6 Структура проекту

Для розробки незалежно від бекенд частини застосунку (тимчасовий варіант для тестування), ми можемо використовувати mock дані, тобто передвстановлені дані, що визначаються в якості json-об'єктів, масивів, рядків, що містять певні дані, що у майбутньому мають бути отримані з серверу.

3.2.3 Проектування бази даних

Для розробки застосунку критично важливо на ранніх етапах спроектувати та узгодити моделі даних, це прискорить розробку всіх частин застосунку та убезпечить від конфліктів при імплементації API та від витрати зайвого часу на адаптацію готового коду.



Рис. 3.7 Базові моделі БД

На цьому етапі ми спланували та узгодили базові моделі для основних сутностей, таких як оцінки, курси, коментарі, користувачі, проєкти, акаунти, категорії курсів.

3.2.4 Визначення пріоритетів розробки

Визначення пріоритетів у розробці є ключовим елементом для успішної реалізації проєкту. Це допомагає ефективно розподілити ресурси, фокусуватися на ключових завданнях і досягати стратегічних цілей. Крім того, цей процес сприяє оптимізації робочих процесів, зменшенню ризиків та забезпеченню відповідності вимогам стейкхолдерів, що важливо для успішного впровадження та подальшого розвитку проєкту.

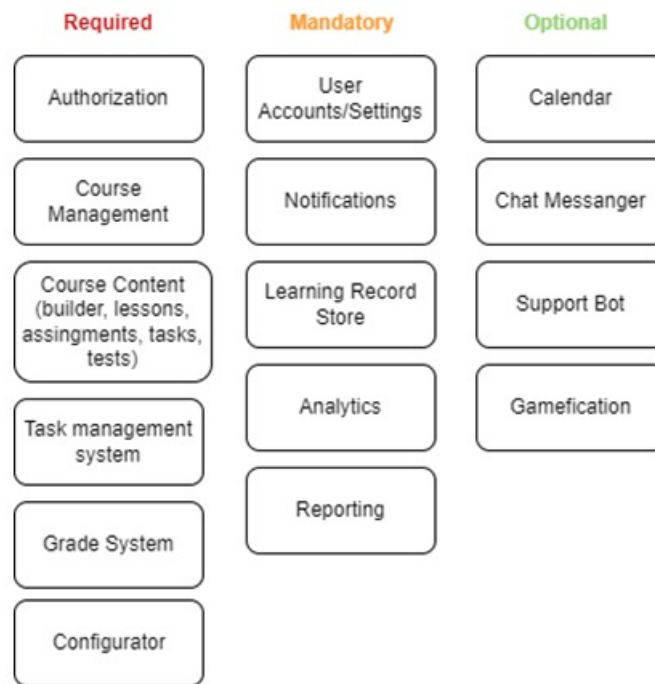


Рис. 3.8 Пріоритети розробки

Визначаємо головні пріоритети для розробки застосунку, без яких система не має «життєвого сенсу», так званий MVP – мінімальний робочий функціонал.

Пріоритети розділені на 3 частини – критичні, мандаторні та мінорні. В такому ж порядку вони будуть імплементуватись в проєкт. Без опціонального функціоналу система не зазнає кардинальних змін, а от без критичного функціоналу система не буде мати сенсу.

3.2.5 Розробка базових компонент

В проєкті використовується потужна Open-Source бібліотека PrimeNG, вона також включає в себе деякий платний функціонал, проте нам вистачить безкоштовного. Хоча ця бібліотека має багато різних компонент, зазвичай цього недостатньо для великих проєктів. Компоненти з PrimeNG будуть основою для створення власних компонентів, якщо в цьому буде необхідність.

В будь-якому проєкті є необхідність використання іконок. Процес додавання та використання іконок необхідно уніфікувати та автоматизувати з самого початку, оскільки міграція великих проєктів є дуже затратним з огляду на ресурси.

Для цього створюємо компонент `icon` за допомогою швидкої команди створення компонентів `ng g c icon`. В нас є необхідність змінювати у іконок розмір та колір. В якості іконок звісно використовуються векторні `svg` файли для якісного відображення в будь-яких розмірах.

Динамічне редагування `svg` іконок є дуже проблематичним, оскільки всі властивості зберігаються всередині іконок, в `xml`-форматі. Для можливості редагувати іконки необхідно вирізати всі атрибути, що впливають на колір та розмір зсередини іконок. Angular та інші фронтенд фреймворки не підходять для вирішення таких задач, проте є чудово бібліотека, що може впоратись з цією задачею – Gulp.

Створимо скрипт, який зможе працювати з файлами. Не варто забувати, що іконки можуть бути умовно двох видів – стандартні з певних фігур та з ліній, для який встановлюється ширина. Для обох випадків необхідно розробити окремі алгоритми, оскільки підходи обробки будуть відрізнятись. Перший крок – обрати всі файли та по черзі модифікувати їх, для «звичайних» іконок необхідно вирізати атрибути `fill` та `style`, для другого випадку: `stroke`, `stroke-width` та `fill`. Для вирізання використаємо іншу бібліотеку, що може працювати з `xml` – “cheerio”, ця бібліотека

має баг та екранізує символ “>”, тож повернемо попередній вигляд та замінимо “>” на “>”.

Тепер ми маємо файли іконок, буде добрим рішенням оптимізувати та мінімізувати їх. Для цього можемо використати бібліотеку `svgrmin`, передамо туди всі файли та налаштуємо плагіни для оптимізації. Для іконок необхідно реалізувати `Lazy-loading` для зменшення навантаження на систему, окрім цього необхідно не надсилати запит на кожну іконку, така поведінка буде негативно відображатись на користувацькому досвіді, коли сторінка буде відображена, а іконки будуть поступово з’являться.

Для реалізації такої задумки необхідно створити спрайт, який будить містити всі іконки та за необхідності буде можливість використати будь-яку іконку з цього спрайту. Для створення спрайту можемо використати бібліотеку `svgSprite`, налаштуємо мод та селектор. Після цього ми можемо додати цей спрайт в `html` документ, який ми додаємо в головну сторінку застосунку, таким чином іконки будуть доступні з будь-якого місця в застосунку. Також оброблено випадок використання кольорових іконок, в яких не треба видаляти колір.

Окрім іконок також створимо інші корисні компоненти. Їх розробка також включає пошук кращих рішень, підтримка адаптивності, динамічної зміни згідно контенту та підтримка використання змінних стилів. Створено й інші базові компоненти, що будуть використовуватися для створення та побудови інших компонентів застосунку. Кожна з цих компонент має визначену структуру та стилі, що дозволить забезпечити єдність та консистентність в дизайні та відображенні даних. Всі компоненти створені згідно принципів модульності та мінімізації залежностей.

- **card:** Цей «тупий» компонент, що не має жодних інпутів та аутпутів. Він дозволяє створювати інші блоки, просто передавши темплейт в середину компоненти. Це компонента-обгортка, що додасть «підложку» з тінями та фоном.
- **chip:** Використовується для створення "чіпів" або "тегів," які можна прикріплювати до об'єктів або категоризувати інформацію, чіп приймає будь-який текст та відображає його з фоном. Якщо колір фону не вказано – він буде автоматично згенерований на основі переданого тексту.
- **dot:** Компонент для створення крапок або індикаторів, що можуть бути корисними для відображення стану або різних значень. Dot приймає колір.
- **form:** Цей компонент – це розширення можливостей Ангуляр з створення форм, ця компонента імплементує велику кількість полів з PrimeNG, таких як поле пароллю, емайл, текстове поле, чекбокс та текст ареа. Це дозволяє створювати форми просто передаючи FormGroup, який в свою чергу може бути створений просто з масиву полів, для цього також створено відповідну утиліту.
- **label:** Компонент для відображення міток або заголовків, що можуть бути використані в багатьох компонентах.
- **nav-item:** Елемент навігації, який використовується в сайдбарі та меню навігації.
- **nav-menu:** Меню навігації, що імплементує nav-item'и, ця компонента допомагає організовувати різні сторінки та реалізовувати роути.
- **progress-bar:** Простий компонент індикатор прогресу, обгортка над прогрес-баром з PrimeNG, який розширює його функціонал.

Ці базові компоненти спрощують розробку та дозволяють швидко створювати інші компоненти для застосунку, використовуючи вже готові блоки. Вони є частиною функціонального контенту, з яким буде взаємодіяти користувач.

3.2.6 Додавання маршрутизації

У цьому розділі буде розглянуто процес створення сторінок з використанням маршрутизації (роутингу) в веб-застосунку. Роутинг є важливою частиною будь-якого веб-додатку, оскільки він дозволяє користувачам отримувати доступ до різних частин застосунку шляхом переходу по різних сторінкам застосунку. Це стандартна практика для полегшення доступу до елементів та покращення користувацького досвіду. У цьому розділі буде розглянуто, як налаштувати роутинг, створити сторінки для застосунку та як можна генерувати модулі автоматично з роутингом за допомогою Angular CLI, забезпечуючи зручну навігацію та відображення вмісту [16].

Роутинг дозволить структурувати застосунок і забезпечити користувачам зручний спосіб переходу між різними частинами сервісу. У цьому розділі розглянемо ключові концепції роутингу та використаємо їх для створення функціональних сторінок веб-застосунку.

Для полегшення роботи з роутами створимо окремий файл `app-routing.module` поруч з `app.module`, в цьому файлі будуть розміщуватись всі загальні роути (до сторінок) застосунку. Для створення цього файлу також можемо використати команду Ангуляр CLI: `ng g m app-routing`. Після створення модуля одразу додамо його імпорт в `app.module`.

Тепер можемо приступити до визначення роутів. Необхідно оптимізувати роботу додатка та не завантажувати всі сторінки при запуску застосунку. Для цього допоможе `Lazy-loading`, це такий підхід, що забезпечує завантаження ресурсів, сторінок в тому числі, тільки за необхідності – при запиті користувача. Для реалізації такого підходу будемо використовувати `loadChildren`.

Додамо обробку стандартної помилки – 404, сторінка не знайдена. Необхідно щоб користувач бачив спеціальну сторінку, якщо затребуваний шлях не було знайдено. Якщо не додати обробку – користувач просто буде бачити стандартне

повідомлення браузера, це виглядає некрасиво та геть не функціонально. Створимо масив для not-found роутів, пояснення буде пізніше. Ми хочемо обробляти всі невідомі сторінки і редіректити користувача на сторінку 404, для цього додамо роутер, який буде обробляти всі запити, та встановимо для нього редірект: `{path: "**", redirectTo: "/404"}`. Тепер можемо додати роут на саму сторінку: `{path: "404", loadChildren: () => import("./components/pages/not-found-page/not-found-page.module").then(m => m.NotFoundPageModule)}`.

Після додавання роутів на сторінку 404, можемо додати роути для основних сторінок, для цього також створюємо масив роутів, де так само визначаємо роути для всіх сторінок, не забуваємо про loadChildren, це необхідно для Lazy-loading.

Для встановлення роутів в застосунок необхідно в імпортах `app-routing.module` вказати імпорт `RouterModule`, куди передати наші роутери за допомогою методу `forRoot`. У нас всього вийшло 2 масиви, проте метод `forRoot` приймає всього один масив, для вирішення цієї проблеми ми можемо просто сконкатенувати ці масиви та передати в вищезгаданий метод. Таке ускладнення використовується не просто так, це необхідно для безшовної генерації нових роутів за допомогою Angular CLI. За замовченням ця утиліта шукає масив роутів, який використовується в методі `forRoot` та додає до нього нові роути, проблема в тому, що нові роути додаються в кінець масиву і при використанні спільного масиву для звичайних роутів та роутів для обробки 404, ми отримуємо ситуацію, коли обробка перехоплює роути на нові сторінки, та вони стають просто недоступними. Для вирішення цієї проблеми і використовується такий підхід.

3.2.7 Розробка сторінок

Після того, як ми розібрались з роутами, можемо перейти до розробки сторінок. Після вивчення можливостей командного рядка Ангуляр можемо використовувати наступну команду для створення сторінок: `ng g m name-page --route the-address-you-want --module app-routing.module`. На даному кроці може виникнути питання, чому тут все ще використовуються модулі, окрім standalone компонент, відповідь на це питання буде проста та трохи обнадійлива. Angular CLI на момент

написання роботи не підтримує можливість генерації компонент з роутингом. Цією нетривіальною задачею можна займатися і вручну, проте є ще одне обмеження яке змушує відмовитись від використання стандалон компонент в якості сторінок – дочірній роутинг. Стандалон компоненти не підтримують вкладений роутинг, це може стати серйозною проблемою, якщо одна сторінка буде мати декілька вкладених адрес, наприклад, сторінка налаштувань, де є меню з різними категоріями, кожна з яких відповідає за відображення різного контенту.

Створимо модулі з компонентами для всіх базових сторінок, використовуючи вищезгадану команду для генерації. Всі сторінки розмістимо в директорії pages.

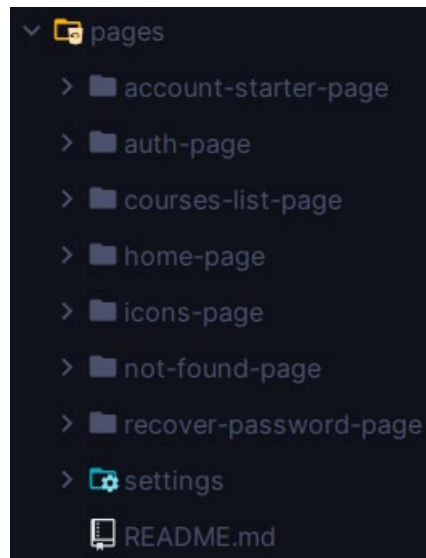


Рис. 3.9 Структура компонентів сторінок застосунку

Після створення необхідних сторінок, необхідно реалізувати функціонал та дизайн, які були обговорені на етапі проектування. В якості даних використовуються стабільні mock-дані, які були згадані раніше. В процесі створення сторінок може виникнути необхідність в створення інших компонент для розбиття логіки та можливості перевикористання деяких компонент. Локальні компоненти, які планується використовувати тільки на якійсь одній сторінці слід створювати в директорії самої сторінки, глобальні компоненти, які будуть використані як основа для всіх / багатьох сторінок слід розташовувати в директорії components. Якщо ми говорити про якість невеликі, або великі компоненти, які

можуть бути багато разів перевикористані в різних інших компонентах, то їх слід розташовувати в окремій папці lib.

При розробці може виникнути необхідність в створенні тимчасових сторінок, так званих шоукейсів, які можуть показувати як працюють певні компоненти, сервіси, директорії та пайпи. Ці сторінки можуть допомогти зрозуміти як працюють ті чи інші частини коду, іноді розробники можуть давати доступ до API цих програмних частин, зазвичай за допомогою input полів.

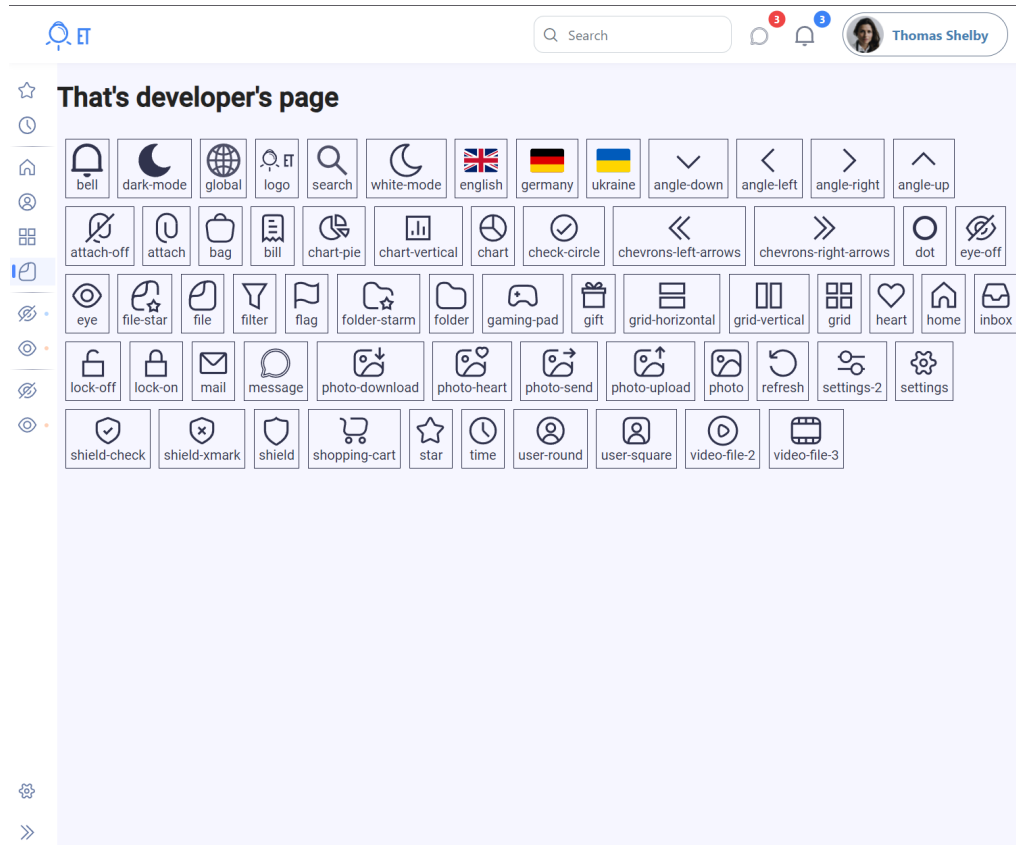


Рис. 3.10 Приклад сторінки для розробників

При першому відкритті сайту, користувач повинен бачити сторінку-вітання, або ж landing. На цій сторінці відображається базова інформація про систему або продукт. Це і є самореклама, ця сторінка є важливою, оскільки система має надавати позитивне перше враження про себе та зацікавити користувача з перших моментів.

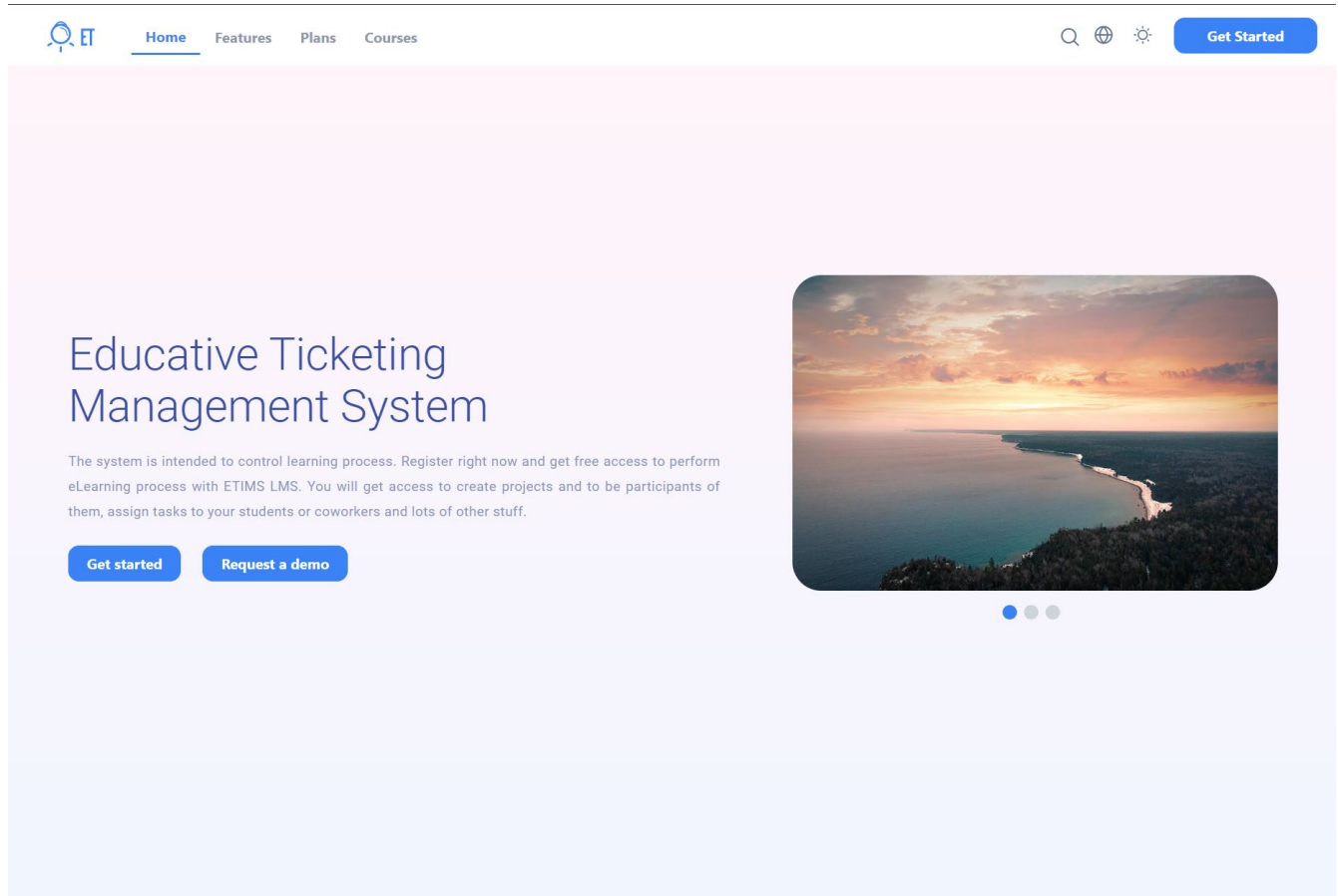


Рис. 3.11 Сторінка вітання користувачів

Імплементація сторінки налаштування може здатися тривіальною задачею, проте необхідно продумати необхідні налаштування користувача та пов'язати це з бекенд частиною системи.

В цій частині застосунку використовується потужний функціонал Angular – це форми, за допомогою них можна створити свої компоненти-обгортки під свої потреби, що дозволить досить легко імплементувати складні системи полів у будь-яку сторінку, просто наконфігурувавши базові їх властивості.

The image shows a web application interface for user profile management. On the left is a sidebar with a search icon and 'ET' logo, followed by navigation items: Favorites, Recents, Home, Account, Courses, Icons, Temp1 (5), Temp2 (2), 404, Settings (2), Recover password (2), and Settings (selected). The main area is titled 'Profile' and contains a form with fields for First Name* (User), Last Name* (Lastoniuk), Username, Bio, and Email* (user@email.com). A blue 'Submit' button is at the bottom. On the right is a secondary sidebar with links: Profile (selected), Privacy, Security, Notifications, and Integrations. The top header features a search bar, notification icons, and a user profile for Thomas Shelby.

Рис. 3.12 Сторінка налаштувань користувача
з розгорнутим сайдбаром

Головною частиною застосунку є система навчання, основним напрямком є курси, тож, створимо сторінку для відображення списку курсів. Сторінка повинна підтримувати два моди відображення – рядками та плитками, реалізуємо їх обидва. Окрім цього необхідно, щоб курси можна було відображати в табах, тобто розділяти курси за категоріями. Наразі є 2 стандартні таби – це Всі та Обрані, куди користувачі самі можуть додати будь-які курси. Інші таби генеруються згідно головного тегу курсу.

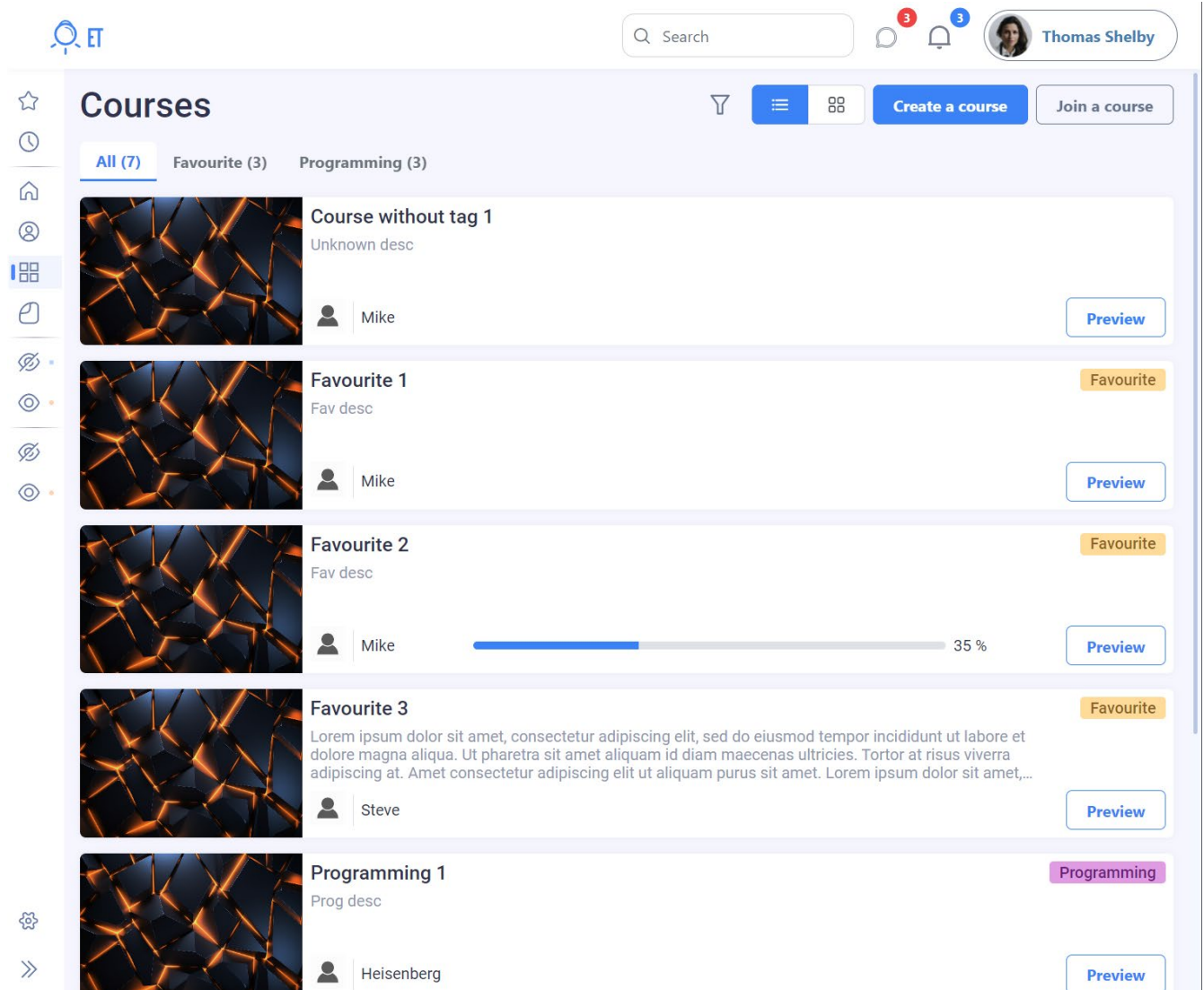


Рис. 3.13 Сторінка списку курсів користувача

Для перегляду базової інформації про курс треба додати сторінку або попап з детальним оглядом курсу, щоб користувач міг ознайомитись з складом курсу та визначитись, чи треба його проходити. Додамо сторінку з інформацією про курс. На даній сторінці розмістимо інформацію про курс, відгуки учасників курсу, питання по курсу, ресурси курсу (матеріали, які будуть представлені в курсі), на цій сторінці буде відображена ціна за курс, рівень складності курсу, моменти, які будуть вивчені в курсі та яка мета вивчення даного курсу.

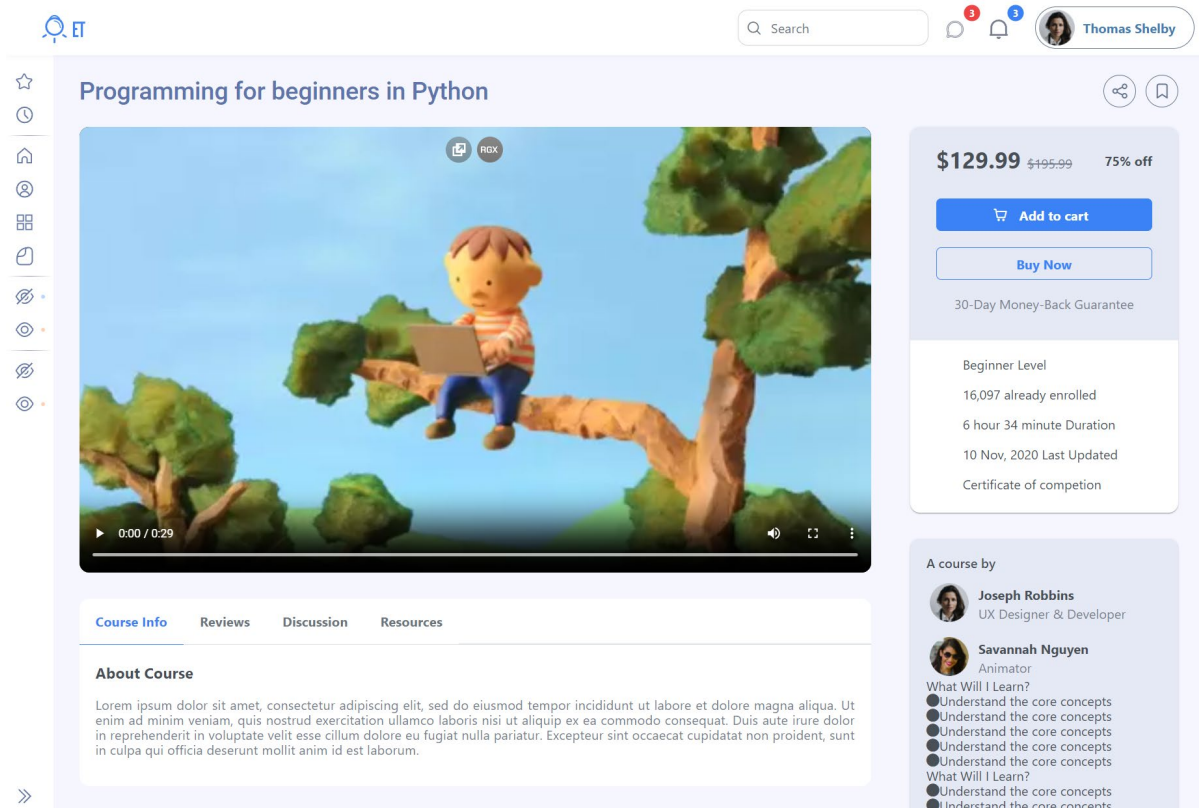


Рис. 3.14 Сторінка Preview курсу

Додамо також сторінки для авторизації в системі, створимо дві сторінки: Логіну та Реєстрації, саме в цих сторінках реалізовані iFrame, які будуть необхідні для імплементації Keycloak в майбутньому.

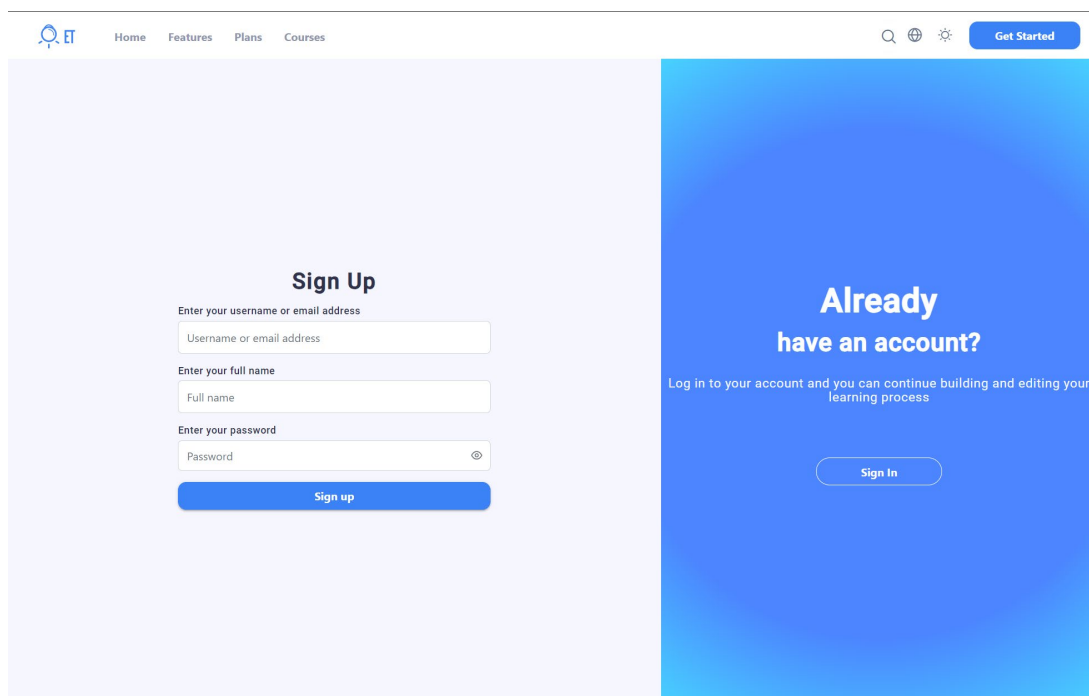


Рис. 3.15 Сторінка реєстрації

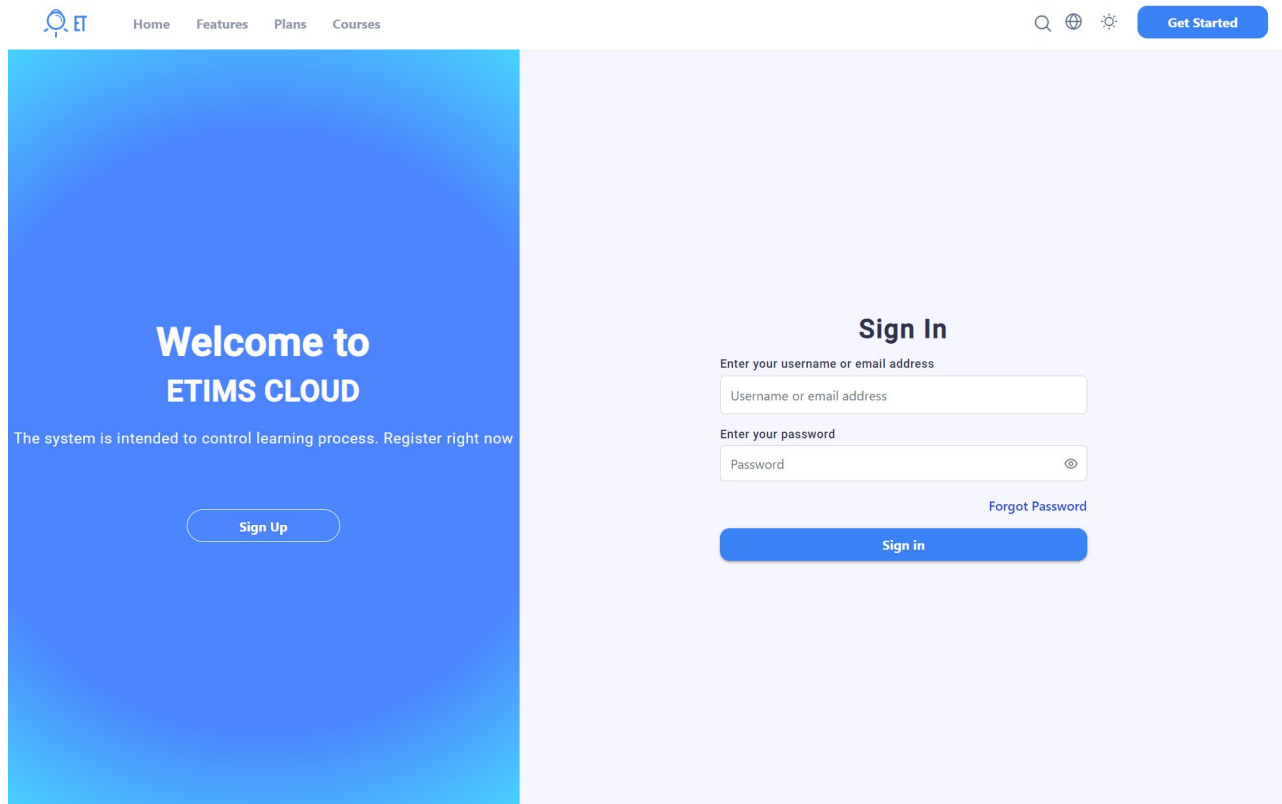


Рис. 3.16 Сторінка авторизації

3.2.8 Покращення якості коду

Для дотримання максимально можливої якості коду необхідно налаштувати також деякі корисні бібліотеки, які дадуть змогу автоматично, ще на етапі компіляції знаходити зауваження по коду. Наступні утиліти допомагають уніфікувати код у всьому проєкті, що є особливо актуальних, якщо на проєкті більше ніж один розробник.

Stylelint і ESLint – це два інструменти для аналізу та валідації коду для файлів стилів та виконуваного коду (TypeScript в нашому випадку) відповідно. Вони допомагають підтримувати високий стандарт коду, виявляти та виправляти потенційні помилки та підвищувати читабельність коду.

Stylelint – це лінтер для CSS (з повною підтримкою SCSS, SASS, LESS). Він перевіряє CSS-код на відповідність конкретним, визначеним розробниками в конфігураційному файлі, стандартам та правилам. Stylelint допомагає забезпечити послідовність та консистентність в коді стилів. Він виявляє потенційні помилки та

недоліки, такі як неочікувані властивості, невірне форматування та інші стилістичні проблеми.

ESLint – це лінтер для JavaScript та TypeScript, який дозволяє виявляти та виправляти помилки в коді, а також впроваджувати стандарти коду та рекомендації з щодо стилю програмування. Він допомагає забезпечити правильну структуру та стиль коду TypeScript. Він виявляє помилки та недоліки коду, такі як невизначені змінні, невикористані функції, несумісність зі стандартами коду та багато іншого.

Важливість використання цих утиліт полягає в тому, що вони сприяють створенню чистого, консистентного коду, мінімізуючи «механічні» помилки, полегшують спільну роботу розробників в команді та забезпечують високу якість програмного забезпечення.

3.2.9 WebSocket

WebSocket – це протокол зв'язку між веб-браузером та веб-сервером, який дозволяє встановлювати і підтримувати постійне з'єднання між ними. Його основні особливості і призначення включають:

1. **Двостороннє з'єднання:** WebSocket дозволяє обмінювати даними між браузером і сервером в режимі реального часу. Він підтримує повідомлення як від браузера до сервера, так і в зворотньому напрямку без необхідності постійно створювати нові HTTP-запити.
2. **Постійне підключення:** WebSocket дозволяє підтримувати постійне і активне з'єднання між клієнтом і сервером. Він не вимагає повторних підключень для кожного запиту, що дозволяє значно зменшити затрати на з'єднання та полегшити взаємодію клієнта і сервера в реальному часі.
3. **Низька латентність:** WebSocket дозволяє передавати повідомлення без зайвих затримок, що робить його ідеальним для задач, які вимагають низької латентності, таких як онлайн-гра чи чат.

4. **Динамічна комунікація:** WebSocket ідеально підходить для сценаріїв, де виникає потреба у миттєвому оновленні інформації на сторінці без перезавантаження.
5. **Підтримка різних протоколів:** WebSocket може використовувати різні протоколи, такі як ws:// та wss:// для забезпечення безпеки (WebSocket Secure).

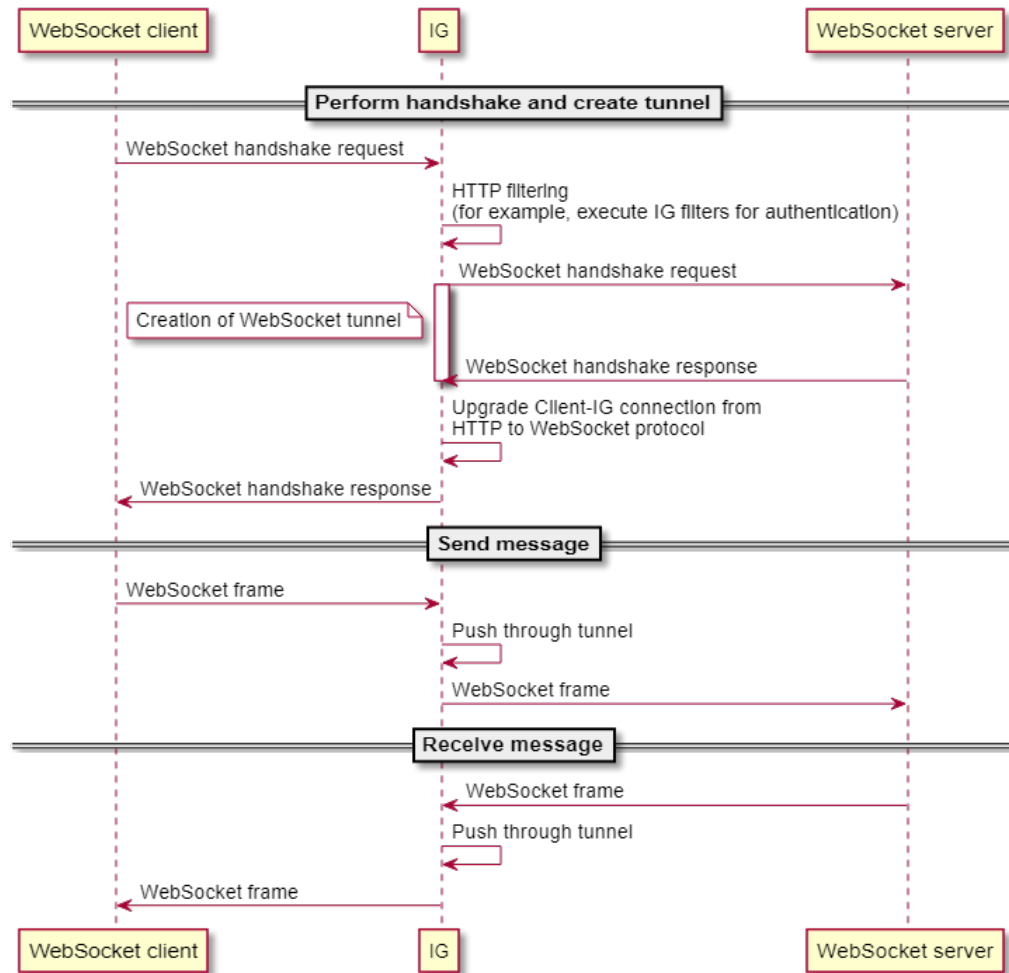


Рис. 3.17 Схема взаємодії WebSocket з клієнтом та сервером [21]

Релізуємо чат, з використанням WebSocket, за допомогою цього протоколу ми встановлюємо постійне з'єднання між клієнтом і сервером, що дозволяє обмінюватися повідомленнями в режимі реального часу. При цьому клієнти та сервер можуть надсилати та отримувати повідомлення без необхідності оновлення сторінки чи виконання додаткових HTTP-запитів.

Окрім цього використання WebSocket-роутера чи каналів дозволяє ідентифікувати різні чат-кімнати або групи. Взаємодія з WebSocket забезпечена через API, що дозволяє обробляти події, таких як надходження нових повідомлень. WebSocket дозволяє ефективно передавати різні дані в режимі реального часу, такі як текстові повідомлення, емодзі, файли чи інші медіа-дані.

Загалом, використання WebSocket в чаті застосунку дозволяє створити інтерактивне та миттєве спілкування між користувачами без необхідності постійного оновлення сторінок та додаткових HTTP-запитів.

3.2.10 Авторизація Keycloak

Keycloak – це open-source ідентифікаційний та управлінський сервіс, який надає рішення для автентифікації та авторизації в розподілених системах. Використовується для створення безпечних та надійних застосунків, де важливо забезпечити захист конфіденційності та доступу до ресурсів[17].

Для імплементації цього сервісу необхідно спочатку створити проєкт з Keycloak, після чого необхідно наконфігурити сам сервіс та білд для нього.

Конфігурація Keycloak є великим JSON файлом, в якому визначено безліч властивостей автентифікації. Цей файл називається реалмом.

```
{
  "id": "0d0534a9-2fcf-4303-8581-8c8c145c7e86",
  "realm": "etims",
  "displayName": "ETIMS",
  "displayNameHtml": "ETIMS",
  "notBefore": 0,
  "defaultSignatureAlgorithm": "RS256",
  "revokeRefreshToken": true,
  "refreshTokenMaxReuse": 0,
  "accessTokenLifespan": 900,
  "accessTokenLifespanForImplicitFlow": 900,
  "ssoSessionIdleTimeout": 1800,
  "ssoSessionMaxLifespan": 36000,
  "ssoSessionIdleTimeoutRememberMe": 0,
  "ssoSessionMaxLifespanRememberMe": 0,
  "offlineSessionIdleTimeout": 2592000,
  "offlineSessionMaxLifespanEnabled": false,
  "offlineSessionMaxLifespan": 5184000,
  "clientSessionIdleTimeout": 0,
  "clientSessionMaxLifespan": 0,
  "clientOfflineSessionIdleTimeout": 0,
  "clientOfflineSessionMaxLifespan": 0,
  "accessTokenLifespan": 60,
  "accessTokenLifespanUserAction": 300,
  "accessTokenLifespanLogin": 1800,
  "actionTokenGeneratedByAdminLifespan": 43200,
  "actionTokenGeneratedByUserLifespan": 86400,
  "oauth2DeviceCodeLifespan": 600,
  "oauth2DevicePollingInterval": 5,
  "enabled": true,
  "sslRequired": "external",
  "registrationAllowed": true,
  "registrationEmailAsUsername": false,
  "rememberMe": true,
```

Рис. 3.18 Частина конфігурації Keycloak

Для збірки Keycloak будемо використовувати Docker, для нього створимо конфігурацію docker-compose та Dockerfile.

```

version: '3.8'
services:
  keycloak:
    container_name: keycloak
    build: ./keycloak/
    command:
      - start-dev --import-realm
    ports:
      - "${KEYCLOAK_PORT}:8080"
    volumes:
      - ../keycloak/conf:/opt/keycloak/conf
      - ../keycloak/data:/opt/keycloak/data
      - ../keycloak/themes:/opt/keycloak/themes
    environment:
      - KEYCLOAK_ADMIN=${KEYCLOAK_ADMIN:-admin}
      - KEYCLOAK_ADMIN_PASSWORD=${KEYCLOAK_ADMIN_PASSWORD:-admin}
      - KEYCLOAK_PORT=${KEYCLOAK_PORT:-8081}
      - KEYCLOAK_DB=${KEYCLOAK_DB:-auth}
      - KEYCLOAK_DB_USERNAME=${KEYCLOAK_DB_USERNAME}
      - KEYCLOAK_DB_PASSWORD=${KEYCLOAK_DB_PASSWORD}
      - KEYCLOAK_DB_URL=${KEYCLOAK_DB_URL}
      - KEYCLOAK_SMTP_PASSWORD=${KEYCLOAK_SMTP_PASSWORD}
  db:
    container_name: pg
    image: postgres:alpine3.17
    ports:
      - "${DB_PORT}:5432"
    environment:
      - POSTGRES_USERNAME=${DB_USERNAME:-postgres}
      - POSTGRES_PASSWORD=${DB_PASSWORD:-postgres}
      - POSTGRES_DB=${DB_NAME:-auth}

```

Рис. 3.19 Docker-compose

```

FROM quay.io/keycloak/keycloak:latest as > builder

WORKDIR /opt/keycloak

# for demonstration purposes only, please make sure to use proper certificates in production instead
RUN keytool -genkeypair -storepass password -storetype PKCS12 -keyalg RSA -keysize 2048 -dname "CN=server" -alias server -ext "SAN:c=DNS:localhost,IP:127.0.0.1" -keystore conf/server.keystore
RUN /opt/keycloak/bin/kc.sh build

FROM quay.io/keycloak/keycloak:latest
COPY --from=builder /opt/keycloak/ /opt/keycloak/

ENTRYPOINT ["/opt/keycloak/bin/kc.sh"]

```

Рис. 3.20 Dockerfile

Запустимо білд та запуск за допомогою команди *docker compose -f docker/docker-compose.yaml --env-file=env.dist up --force-recreate -d*, куди також передамо файл змін оточення.

Реалм в Keycloak – це ізольований простір, в якому визначаються правила аутентифікації, авторизації, користувачі, клієнти та інші параметри, необхідні для обслуговування конкретної системи чи додатку.

1. **realm:** Це кореневий елемент, який представляє собою реалм. У цьому елементі визначаються основні параметри реалму, такі як його ім'я.
2. **users:** Цей розділ містить інформацію про користувачів, яких слід створити в рамках цього реалму. Кожен користувач описується своїм ідентифікатором (username), паролем та іншими атрибутами.
3. **roles:** Тут визначаються ролі, які можуть бути призначені користувачам у межах реалму. Ролі використовуються для контролю доступу.
4. **clients:** Визначаються параметри для клієнтів, які взаємодіють з реалмом. Клієнти можуть бути веб-додатками, мобільними додатками чи іншими системами.
5. **identityProviders:** Цей розділ дозволяє налаштовувати зовнішні постачальники ідентичності, такі як Google або Facebook.
6. **authenticationFlows:** Визначається набір потоків аутентифікації, які використовуються для аутентифікації користувачів.
7. **clientScopes:** Тут визначаються зовнішні області, які можуть бути використані клієнтами.

В адмін-панелі є можливість простіше налаштовувати всю систему автентифікації [18].

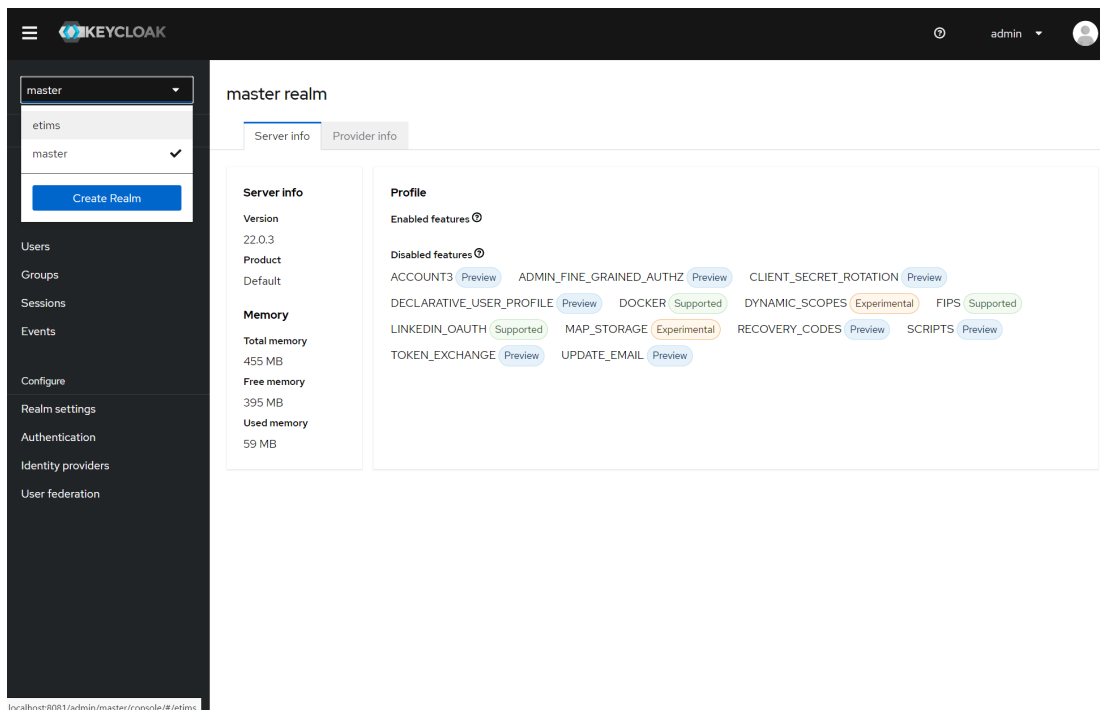


Рис. 3.21 Адмін-панель Keycloak

В адмін-панелі можна переглядати всіх користувачів, що зареєструвалися в системі, окрім цього можемо створити своїх користувачів, налаштувати їм певні ролі, статуси та можливості.

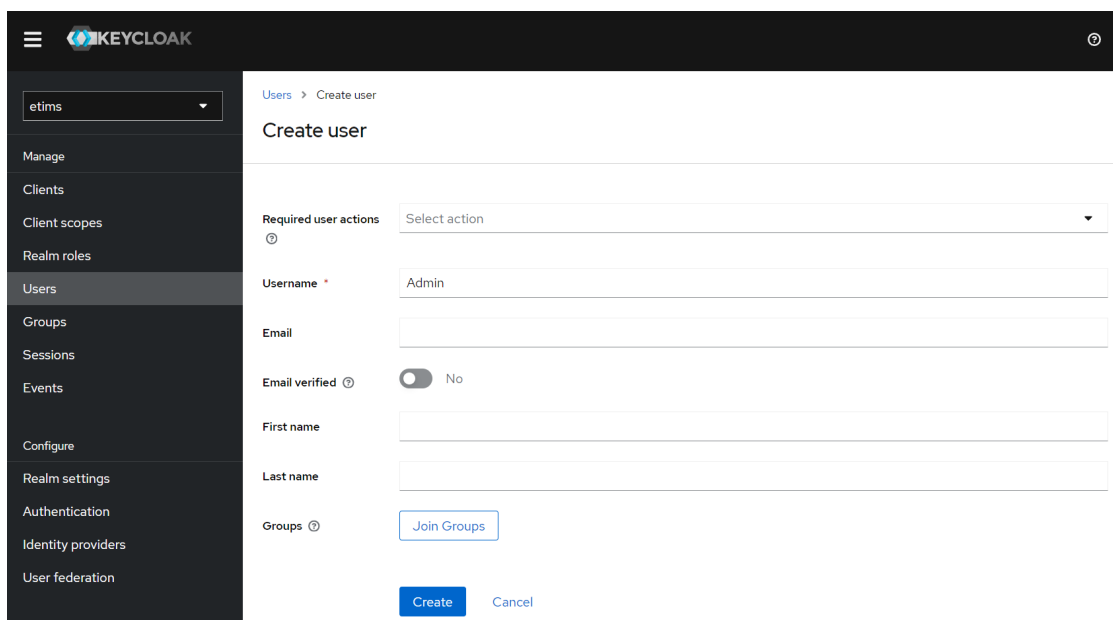


Рис. 3.22 Створення користувача в Адмін-панелі

Тут же можемо налаштувати інші проперті конфіденційності, безпеки. На етапі розробки все запускається локально, тому конфігурується для локальних адрес.

Headers	Brute force detection
X-Frame-Options ②	SAMEORIGIN
Content-Security-Policy ②	frame-src 'self' http://127.0.0.1 http://localhost:4200 http://127.0.0.1:4200 http://localhost localhost; frame-anc...
Content-Security-Policy-Report-Only ②	
X-Content-Type-Options ②	nosniff
X-Robots-Tag ②	none
X-XSS-Protection ②	1; mode=block
HTTP Strict Transport Security (HSTS) ②	max-age=31536000; includeSubDomains
Referrer Policy ②	no-referrer

Рис. 3.23 Налаштування Request-Headers

Сторінки для відображення в keusloak зберігаються в форматі ftl, ще дещо модифікований html з підтримкою деяких функціональних властивостей мови програмування.

```

1  <#import "template.ftl" as layout>
2  <@layout.registrationLayout displayMessage=messagesPerField.existsError('username','password') displayInfo=realm.password && realm.registrationAllowed && !registrationDisabled?> section>
3  <#if section = "header">
4      ${msg("loginAccountTitle")}
5  <#elseif section = "form">
6      <div id="kc-form">
7          <div id="kc-form-wrapper">
8              <#if realm.password>
9                  <form id="kc-form-login" onsubmit="login.disabled = true; return true;" action="${url.loginAction}"
10                     method="post">
11                      <#if !usernameHidden?>
12                          <div class="field-container ${properties.kcFormGroupClass!}">
13                              <label for="username"
14                                  class="${properties.kcLabelClass!}"><#if !realm.loginWithEmailAllowed>${msg("username")}
15                                  <#elseif !realm.registrationEmailAsUsername>${msg("usernameOrEmail")}<#else>${msg("email")}</#if></label>
16
17                              <input tabIndex="1" id="username" class="${properties.kcInputClass!}" name="username"
18                                  value="${(login.username!)}" type="text" autofocus autocomplete="off"
19                                  aria-invalid="<#if messagesPerField.existsError('username','password')>true</#if>"
20                                  />
21
22                              <#if messagesPerField.existsError('username','password')>
23                                  <span id="input-error" class="${properties.kcInputErrorMessageClass!}"
24                                      aria-live="polite">
25                                      ${kcSanitize(messagesPerField.getFirstError('username','password'))?no_esc}
26                                  </span>
27                              </#if>
28                          </div>
29

```

Рис. 3.24 Частина коду конфігурації сторінки логіну

Keycloak використовує відкритий стандарт авторизації та аутентифікації, відомий як OpenID Connect (OIDC). OIDC є протоколом, який додає шар авторизації на основі веб-стандартів для протоколу аутентифікації OAuth 2.0.

Основні етапи авторизації в Keycloak за допомогою протоколу OpenID Connect:

1. **Реєстрація клієнта:** Перш ніж користувачі можуть авторизуватися через Keycloak, розробник повинен зареєструвати свій веб-сервіс як "клієнта" у Keycloak. Це дозволяє Keycloak визначити, які ресурси та інформацію може запитати та отримувати цей клієнт.
2. **Запит авторизації:** Коли користувач вибирає увійти на сайт або здійснити іншу дію, що вимагає авторизації, клієнт направляє користувача на Keycloak разом із запитом на авторизацію.
3. **Вхід користувача:** Якщо користувач не авторизований, його перенаправлять на сторінку входу Keycloak для введення логіна та пароля.
4. **Видача токенів:** Після успішної аутентифікації Keycloak видає два основних види токенів: ID Token та Access Token. ID Token містить інформацію про користувача, таку як ім'я та ідентифікатор. Access Token використовується для доступу до ресурсів на серверах ресурсів, які захищені Keycloak.
5. **Повернення до клієнта:** Здійснивши успішний вхід, Keycloak повертає користувача на сайт клієнта з токенами.
6. **Запит на отримання ресурсів:** Клієнт може використовувати Access Token для отримання доступу до ресурсів, які захищені Keycloak. При цьому токен може передаватися як частина запиту або у заголовку.

Використання стандарту OpenID Connect у Keycloak дозволяє надійно та безпечно забезпечити процес аутентифікації та авторизації в додатках та сервісах, які використовують цю систему.

Для імплементації Keycloak у систему використаємо `iframe`, в якому в якості сорсів вкажемо адресу на сторінки логіну, реєстрації, скидання паролю. За цими адресами Keycloak поверне відрендерені `ftl` сторінки вже в форматі `html`, який в свою чергу вставиться в сам `iframe`.

```
1 <iframe height="400"
2     id="inlineFrameExample"
3     src="http://localhost:8081/realms/etims/protocol/openid-connect/auth"
4     style="border-style: none;"
5     title="Inline Frame Example"
6     width="500"
7 ></iframe>
```

Рис. 3.25 Iframe для сторінки логіну Keycloak

На стороні Ангуляру нам необхідно відстежувати запити, та при успішній автентифікації корентно розпарсити токен, та встановити відповідного користувача. Для локального зберігання даних автентифікації користувача використовується `Local Storage` браузера.

3.2.11 Продуктивність системи

Ще одним важливим моментом є продуктивність системи. Від швидкості відображення сторінки, а особливо, від первинного рендерингу залежить загальне враження від всієї системи. Це є ще один дуже важливий момент, який необхідно постійно тримати в фокусі і розраховувати, як зміни можуть вплинути на користувацький досвід.

Для перевірки перфоменсу системи використаємо браузерну утиліту `DevTools` та відповідну вкладку [19].

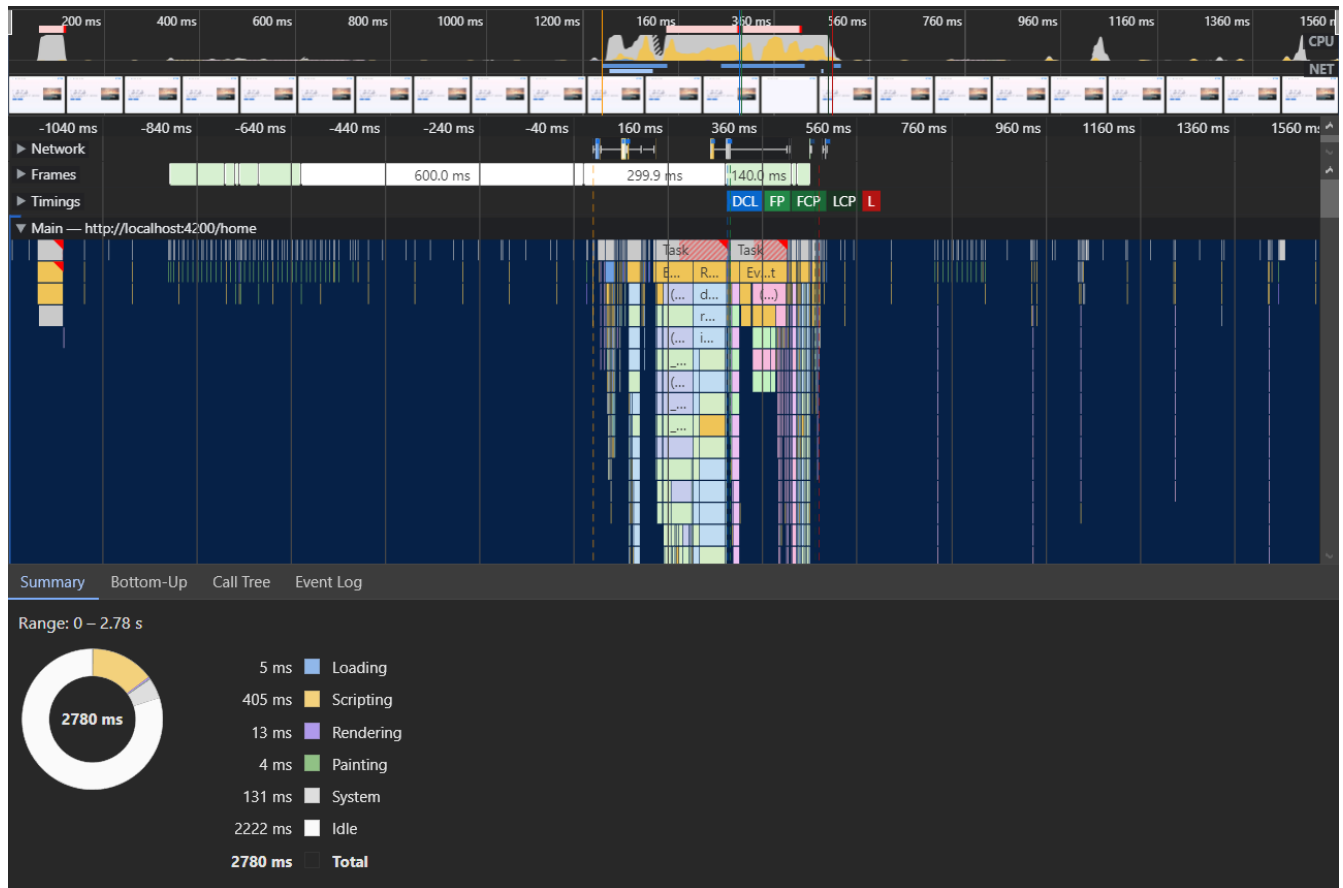


Рис. 3.26 Продуктивність системи

Аналіз показує, що повне відображення системи без кешування відбувається за 558мс, що є дуже швидким. Цей показник можна суттєво зменшити при використанні сервісів кешування.

На даний момент розробки результати цілком гарні. За належної уваги до перфоменсу, сервіс може пропонувати користувачам найкращий досвід використання, що добре відображаються на загальному враженні від продукту та відчутті швидкодії системи.

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі магістерської роботи було спроектовано дизайн застосунку, на основі якого було імплементовано різні частини застосунку, такі як сторінки, маршрутизація, утиліти покращення коду. Також було визначено пріоритети розробки системи [20].

Було додано систему автентифікації Keycloak, що є вкрай важливим кроком у забезпеченні безпеки та надійності системи. Використання протоколу OpenID Connect у поєднанні з Keycloak дозволяє ефективно управляти аутентифікацією та авторизацією користувачів. Keycloak спрощує процеси управління ідентифікацією, забезпечуючи централізовану систему управління правами доступу. Його можливості щодо створення та управління користувачами, надання ролей та налаштування прав доступу дозволяють ефективно адаптувати систему до різних потреб проєкту. Додавання Keycloak є критично важливим кроком для забезпечення безпеки, і створення надійної системи управління доступом.

Завершивши цей етап, ми готові до подальших етапів розробки, включаючи імплементацию запланованої функціональності, реалізацію дизайну, який був ще не реалізований та покращення існуючих компонент. Усі попередні кроки були вкрай важливими для створення надійної та продуктивної фронтенд-частини застосунку.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП-ПРОЄКТУ

Сучасний світ інновацій та технологій характеризується високою швидкістю змін та постійними трансформаціями у всіх сферах життя. Особливо це стосується сфери освіти та навчання, де впровадження новітніх технологій та платформ є важливим аспектом підвищення якості та доступності навчання. У цьому контексті стартап-проекти та розробка нових систем навчання в освіті стають все більш актуальними та важливими.

Стартап-проект в освіті – це інноваційний бізнес-проект, що спрямований на вирішення актуальних проблем та викликів у сфері освіти. Це може бути розробка нових педагогічних підходів, створення навчальних платформ, розробка навчальних інструментів або реалізація інших інноваційних ідей, спрямованих на вдосконалення навчального процесу та розширення доступу до освіти для всіх.

Цей розділ магістерської роботи описує процес розробки стартап-проекту – системи управління навчанням. Дуже важливо детально проаналізувати процес створення та розвитку цього проекту, оскільки це допоможе зрозуміти підґрунтя вибору конкретних технологій та рішення певних задач. Розвиток даного проєкту в перспективі може мати значний вплив на покращення якості освіти, вдосконалення процесів викладання та навчання, пропонуючи інноваційні та ефективні рішення для студентів, викладачів, навчальних закладів та бізнесів. Технологічна зручність та інноваційні підходи стають ключовими факторами для покращення якості освіти.

У цьому розділі розглядається процес створення освітнього стартап-проекту та визначаються ключові аспекти цього процесу. Вивчення та розуміння цих аспектів має важливе значення для успішної розробки та впровадження інноваційних навчальних рішень.

4.1 ОПИС ІДЕЇ СТАРТАПУ

Ідея стартап-проекту полягає у створенні зручної, функціональної, зручної та доступної навчальної платформи. ETIMS прагне впроваджувати новітні технології, унікальні можливості та інструменти, які допомагають покращити процес навчання, забезпечити доступ до якісної освіти, підвищити продуктивність та впроваджувати новітні технології та інструменти, що допомагають зробити навчальний процес простішим та будуть заохочувати до навчання.

Даний стартап-проект має охоплювати широкий спектр закладів, від шкіл та університетів до бізнесів. Кожен з цих секторів може використовувати нашу LMS-систему для покращення навчального процесу, організації та управління навчальними ресурсами і підвищення якості освіти з мінімізацією часу, витраченого на здобування певної кваліфікації, що є особливо важливим для бізнес-клієнтів.

ETIMS має імплементувати кращі практики (best practices) інших LMS систем, вдало поєднувати їх та пропонувати унікальні функції, які не мають інші проекти. Переваги використання стають ETIMS очевидними, користувачі отримують гнучку систему з поєднаним функціоналом інших систем, що робить даний проект привабливим для всіх видів користувачів. Для вчителів та викладачів це покращує їхню можливість створювати будь-які курси в декілька кліків та контролювати процес навчання студентів. Студенти отримують доступ до високоякісних навчальних матеріалів з будь-якої точки планети, що допоможе покращити успішність та зробити навчання більш привабливим для всіх категорій людей. Для компаній ETIMS надає особливі можливості для інтеграції з внутрішніми системами бізнесу, можливість навчати та розвивати навички своїх співробітників в онлайн форматі, що сприяє підвищенню кваліфікації співробітників. Для зручної інтеграції пропонується SSO, LDAP та CAS.

Таким чином, наш стартап-проект має потенціал стати інструментом, який кардинально змінює підхід до навчання та розвитку компетенцій, надаючи користувачам якісну освіту та сучасні можливості для навчання на віддалі.

Таблиця 4.1

Визначення характеристик ідеї проєкту

№	Характеристики ідеї	ETIMS	Moodle	BlackBoard	Google Classroom	Coursera
1	Ролі користувачів в системі	+	+	+	+	+
2	Профіль користувача	+	+	+	-	+
3	Список студентів	+	+	+	+	+
4	xAPI	+	+	+	-	-
5	Гарний UI	+	-	-	+	+
6	Зручний UX	+	-	-	+	+
7	Авторизація за логін + пароль	+	+	+	+	-
8	LDAP	+	+	+	-	-
9	SSO	+	-	+	+	-
10	CAS	+	+	+	-	-
11	2FA	+	+	+	+	+
12	OTP	+	+	-	-	-
13	Використання сертифікатів	-	+	-	-	-

4.2 ТЕХНІЧНИЙ АУДИТ ІДЕЇ СТАРТАПУ

В даному підрозділі буде оглянуто можливість реалізувати ідеї проєкту, використовуючи певні можливості та ПЗ. Дані аудиту наведені в таблиці нижче.

Таблиця 4.2

Технологічна здійсненність ідеї проєкту

№	Можливості застосунку	Технології реалізації	Наявність технологій	Доступність
1	Авторизація	KeyCloak	+	Free
2	Фреймворк FE	Angular v17	+	Free
3	Препроцесор стилів	LESS	+	Free
4	Взаємодія з сервером	REST	+	Free
5	Робота з підписками	RxJs	+	Free
6	Підтримка постійного зв'язку та обміну даними	WebSocket	+	Free
7	Реалізація логіки застосунку	TypeScript	+	Free
8	Реалізація відображення застосунку	Html, CSS	+	Free

В результаті проведеного дослідження було визначено, що кожна з необхідних технологій є доступною, а отже реалізація ідеї проєкту є можливою.

4.3 АНАЛІЗ РИНКОВИХ МОЖЛИВОСТЕЙ ЗАПУСКУ СТАРТАП-ПРОЕКТУ

Наступним кроком є визначення можливостей запуску стартап-проекту, враховуючи рішення, що наявні на ринку, буде визначено ринкові загрози, що можуть завадити реалізації ідеї проекту.

Першим кроком визначено чи присутній попит на ринку, його обсяг та динаміка розвитку. Ці дані наведено у таблиці нижче.

Таблиця 4.3

Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, дол/ум.од	0 у випадку Moodle, система є OpenSource, Coursera – ~500млн. дол / рік; інші проекти – мільйони дол на рік
3	Динаміка ринку (якісна оцінка)	Зростає, з поширенням тенденції автоматизації всіх процесів та тренду здорового способу життя.
4	Наявність обмежень для входу (вказати характер обмежень)	Для повної реалізації проекту необхідно юридично зареєструвати підприємницьку діяльність, а також врахувати нюанси законів про зберігання користувацьких даних для всіх країн, де планується діяльність
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	60%

Середній відсоток вкладень в банк на депозит складає 15% річних, у вигляді облігацій може складати 18%. Через війну в країні також присутня суттєва інфляція, через це вкладення в проект є рентабельним.

Наступним етапом було визначення характеристик потенційних клієнтів

проекту, що наведена у таблиці нижче.

Таблиця 4.4

Характеристика потенційних клієнтів

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в асинхронному навчанні	Навчальні заклади, бізнеси	Для бізнесів потрібна можливість надати працівникам зручний час для навчання, в навчальних закладах необхідно забезпечити навчання незалежно від офлайн-навчання.	Можливість проводити навчання зручно, без прив'язки до конкретного часу.
2	Потреба в онлайн навчанні	Навчальні заклади, бізнеси	В умовах війни та епідемій онлайн навчання особливо актуально для навчальних закладів, коли немає можливості відвідувати відповідні заклади. При віддаленій роботі бізнеси потребують навчання співробітників онлайн.	Можливість навчатись за допомогою підключення інтернет та без необхідності відвідувати офіс, або інші заклади.
3	Потреба в покращенні спеціалізацій	Вищі навчальні заклади, бізнеси	Вищі навчальні заклади підвищують якість надання освіти, бізнеси покращують продуктивність співробітників.	Можливість впроваджувати курси та спеціалізації для покращення обізнаності і навичок.

Після цього було проведено аналіз ринку на предмет факторів ризику та сприяння розвитку проекту.

Таблиця 4.5

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Законна реєстрація	Будь-яку економічну діяльність необхідно зареєструвати юридично.	Офіційно оформити стартап-проект задля співпраці з юридичними особами.
2	Конкуренти	Поява нових конкурентів, що надають схожі можливості, або суттєвий розвиток існуючих.	Розробка унікальних функцій. Програма лояльності.
3	Стрімке розширення кількості користувачів	Різде велике збільшення клієнтів системи.	Впровадити тимчасове рішення, задля пристосування існуючого. Постійне покращення існуючого продукту. Масштабування систем для підтримки навантаження.
4	Хакерські атаки	Безпекові, навантажувальні атаки на систему	Впровадження періодичний курсів з безпеки для співробітників. При масованій атаці можливе тимчасове відключення серверів з даними користувачів для унеможливлення витоку інформації. Імплементация існуючих рішень для покращення безпеки. Встановлення фільтрів на поштові скриньки для блокування фішингу.

Таблиця 4.6

Фактори можливостей

№	Фактор	Зміст можливостей	Можлива реакція компанії
1	Збільшення лояльності клієнтів	При покращенні якості надаваних послуг, клієнти з більшою вірогідністю продовжуватимуть використання продукту.	Покращення існуючих функцій та бізнес-процесів в систему. Залучення нових клієнтів.

Продовження таблиці 4.6

№	Фактор	Зміст можливостей	Можлива реакція компанії
2	Зростання прибутку	Збільшення кількості надходжень коштів	Розробка нового функціоналу. Покращення та розвиток існуючого функціоналу.
3	Покращення роботи штату компанії	З наявною системою штат зможе підвищити ефективність роботи, а отже зменшити витрати.	Впровадження премій, підвищення заробітних плат, розвиток нового функціоналу або продукту.
4	Реклама	Можливість просувати проєкт за допомогою реклами.	Купівля реклами і релевантних блогерів, використання Google Ads та Facebook для просування продукту.

Наступним кроком було визначення стратегії позиціонування стартап-проєкту

Таблиця 4.7

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформулювати комплексну позицію власного проєкту
1	Демпінгова ціна на старті	Безкоштовна розробка та зменшення вартості обслуговування серверів	Використання безкоштовних технологій та безкоштовних розробників	Доступність
2	Легкість імплементації в існуючі системи авторизація для бізнесів	Підтримка сучасних протоколів та легка інтеграція з системами замовників	Наявність великої кількості можливих інтеграцій та легкість імплементації	Легкість інтеграції
3	Наявність статистики та аналітики	Впровадження розширеного функціоналу з отримання аналітики та статистики	Збір великої кількості даних для аналітики та статистики та використання спеціальних бібліотек для відображення графіків та діаграм	Наглядність прогресу навчання

Продовження таблиці 4.7

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформулювати комплексну позицію власного проекту
4	Зручність використання	Аналіз потреб користувачів та ринку	Зручний, сучасний та зрозумілий інтерфейс	Зрозумілість

В результаті було обрано наступну стратегію ринкової поведінки: базова стратегія – стратегія диференціації, стратегія конкурентної поведінки – наслідування лідеру. Дані стратегії визначатимуть в якому напрямку працюватиме стартап-компанія на ринку.

В даному розділі було проаналізовано можливості розвитку розробленої інформаційної системи як стартап-проекту. Аналіз показав, що в проекті є можливість ринкової комерціалізації, оскільки наявний попит, зростаюча динаміка ринку та висока рентабельність. Також було визначено потенційні групи клієнтів, конкурентоспроможність та бар'єри входження, після чого було визначено наявність перспектив впровадження. Є доцільною подальша імплементація проекту.

ВИСНОВКИ ДО РОЗДІЛУ 4

При розробці стартап-проекту було визначено та описано напрямки застосування, головну ідею, користь для кінцевого користувача системи. Стартап-проект представляє з себе інформаційну систему для управління навчанням та менеджменту навчальних матеріалів.

Проведено технологічний аудит стартапу, в якому було оглянуто необхідні технології при реалізації проекту. Всі необхідні технології доступні для використання та є безкоштовними. Вони є сумісними одні з одним та підходять для комплексного використання в одній системі. Аналіз поточних та можливих конкурентів на ринку показали, що на даний момент система з запланованим функціоналом буде краще всіх інших наявних рішень. У випадку появи майбутніх конкурентів розроблено план розвитку системи.

Аналіз факторів загроз показує, що у системи є певні загрози різного характеру, але всі вони можуть бути вирішені методами, які були описані. Аналіз фактору можливостей показує, що при сприятливих умовах система може стрімко розвиватися, описано можливі наслідки обох факторів.

ВИСНОВКИ

Результатом виконання роботи є розроблена клієнтська частина системи керування навчанням.

Для розробки даної системи було ретельно досліджено і проаналізовано предметну область застосування системи в сучасному середовищі. Дослідження актуальності таких систем дозволяє зрозуміти раціональність розробки системи.

Було детально розглянуто ринок подібних систем, проаналізовано основні та найпопулярніші системи. Проаналізовано функціональні можливості та недоліки таких систем, аналіз систем дав розуміння необхідного функціоналу системи.

Результатом попереднього аналізу існуючих систем було визначено та сформовано конкретні функціональні та нефункціональні вимоги до системи. Визначення таких вимог дає змогу приступити до визначення сценаріїв використання системи.

В третьому розділі було визначено основні сценарії, які необхідно імплементувати в систему. Визначено 3 загальних типи сценаріїв – загальні, студента та вчителя. Для кожного типу було детально розписано сценарії користування системою. Результатом створення сценаріїв використання та вимог до системи є чітке розуміння необхідного функціоналу системи.

В четвертому розділі було оглянуто існуючі фреймворки для імплементатії системи, серед яких було обрано Angular для клієнтської частини. Було ретельно продумано процес розробки системи, зокрема, визначені пріоритети розробки, внутрішня архітектура, структура бази даних та основні компоненти. Використання клієнт-серверної архітектури та модульності забезпечила гнучкість та легкість підтримки.

Було додано основні компоненти клієнтської частини, реалізовано коректну роботу застосунку, включаючи роутинг, різні сторінки з контентом. Для забезпечення належної безпеки та автентифікації було імплементовано систему

Keusloak. Keusloak було інгровано через імплементацію у вигляді iframe, що робить процес авторизації невидимим для кінцевого користувача та забезпечує зручність використання.

Було розглянуто продуктивність системи, що підтверджує відповідність сучасним вимогам до швидкості відображення та задоволення користувацьких потреб.

Розроблена система відповідає визначеним вимогам щодо функціональних та нефункціональних можливостей та сучасним вимогам ринку. Система може і буде покращена за потреби.

Після розробки застосунку було проаналізовано можливість виходу на ринок і вигляді стартап-проєкту. Дослідження довело, що даний програмний продукт може бути комерціаналізований та може мати високий коефіцієнт рентабельності. Було досліджено можливі ризики та можливості, обрано стратегії базового розвитку і конкурентної поведінки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is an LMS? | iSpring. *Explore the eLearning world with us.*
URL: <https://www.ispringsolutions.com/blog/what-is-lms> (date of access: 10.09.2023).
2. Online Learning With The World's Most Popular LMS - Moodle. *Moodle.*
URL: <https://moodle.com> (date of access: 13.12.2023).
3. Lab O. Blackboard Learn. *Open Library Publishing Platform – Pressbooks for Ontario's Postsecondary Educators.*
URL: <https://ecampusontario.pressbooks.pub/techinthecurriculum/chapter/blackboard-learn/> (date of access: 11.09.2023).
4. Educational Technology Services | Blackboard | North America. *Educational Technology Services | Blackboard | North America.*
URL: <https://www.blackboard.com> (date of access: 13.09.2023).
5. Blackboard Learn. *Anthology.* URL: <https://www.anthology.com/en-emea/products/teaching-and-learning/learning-effectiveness/blackboard-learn> (date of access: 15.09.2023).
6. Google Classroom | Google for Developers. *Google for Developers.*
URL: <https://developers.google.com/classroom> (date of access: 16.09.2023).
7. Excuse assignments in Google Classroom. *Google Workspace Updates.*
URL: <https://workspaceupdates.googleblog.com/2023/12/excuse-assignments-in-google-classroom.html> (date of access: 18.09.2023).
8. Coursera's Mission, Vision, and Commitment to Our Community | Coursera. *Coursera's Mission, Vision, and Commitment to Our Community | Coursera.* URL: <https://about.coursera.org> (date of access: 20.09.2023).
9. Coursera Features – Режим доступу до ресурсу: <https://www.g2.com/products/coursera/features> (date of access: 20.09.2023)..
10. How to we implement social authentications like facebook, linkedin, google,etc. *Strapi Community Forum.* URL: <https://forum.strapi.io/t/how-to-we>

- implement-social-authentications-like-facebook-linkedin-google-etc/25238/5 (date of access: 28.09.2023).
11. User: stories, scenarios, journey, flow. У чому різниця?. *Telegraf – журнал дизайнерів*. URL: <https://telegraf.design/user-stories-scenarios-journey-flow-u-chomu-riznytsya/> (дата звернення: 02.10.2023).
 12. Contributor T. What is Web Application (Web Apps) and its Benefits. *Software Quality*. URL: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app> (date of access: 05.10.2023).
 13. Benefit of using MVC - GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/benefit-of-using-mvc/> (date of access: 06.10.2023).
 14. Angular vs React vs Vue: Core Differences | BrowserStack. *BrowserStack*. URL: <https://www.browserstack.com/guide/angular-vs-react-vs-vue> (date of access: 10.10.2023).
 15. What is Brand Design and Why is it Important. *Passionate Design Agency*. URL: <https://passionates.com/importance-of-brand-design/> (date of access: 15.10.2023).
 16. Angular. *Angular Router*. URL: <https://angular.io/guide/router> (date of access: 23.10.2023).
 17. Keycloak SSO – advantages of Single Sign-On and a ready-made access management system - Pretius. *Pretius*. URL: <https://pretius.com/blog/keycloak-sso/> (date of access: 08.11.2023).
 18. Configuring Keycloak - Keycloak. *Keycloak*. URL: <https://www.keycloak.org/server/configuration> (date of access: 10.11.2023).
 19. QAInsights. Chrome DevTools for Performance Engineers - Performance, 2021. *YouTube*. URL: <https://www.youtube.com/watch?v=LGT4Qe7v0So> (date of access: 21.11.2023).
 20. GitHub - ETIMS-Inc/etims-web: ETIMS front-end. *GitHub*. URL: <https://github.com/ETIMS-Inc/etims-web> (date of access: 22.11.2023).

21. ForgeRock Identity Gateway 7 > Gateway Guide > Proxying WebSocket Traffic. *ForgeRock*. URL: <https://backstage.forgerock.com/docs/ig/7/gateway-guide/websocket.html> (date of access: 23.11.2023).
22. Social vs. SSO: What's the Difference and How Do They Work?. *CrypTr*. URL: <https://www.crypTr.co/blog/social-vs-sso-whats-the-difference-and-how-do-they-work> (date of access: 22.12.2023).
23. LDAP vs SSO - What's the Difference? (Explained with Use Cases). *Cloud Infrastructure Services*. URL: <https://cloudinfrastructureservices.co.uk/ldap-vs-sso/> (date of access: 22.12.2023).

ДОДАТКИ

ДОДАТОК 1: СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Спільні сценарії використання

Таблиця 2.1

Сценарій реєстрації користувача

Назва	Реєстрація
Опис	Реєстрація користувача в системі
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку реєстрації за допомогою кнопки або відповідного посилання 3. Введення email-адреси, паролю, підтвердження паролю та додаткових полів 4. Натискання кнопки реєстрації 5. Користувач отримує листа з посиланням для активації акаунта 6. Користувач переходить за посиланням та активує акаунт
Результат	Користувач успішно зареєструвався в системі

Таблиця 2.2

Сценарій авторизації користувача

Назва	Авторизація
Опис	Ідентифікація користувача системою
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку авторизації за допомогою кнопки або відповідного посилання 3. Введення логіну та паролю 4. Натискання кнопки авторизації
Результат	Користувач увійшов в систему та успішно ідентифікований

Таблиця 2.3

Сценарій отримання довідкової інформації

Назва	Отримання загальної інформації про продукт
Опис	Отримання загальної інформації про систему для ознайомлення з особливостями та можливостями
Порядок дій	1. Відкриття сторінки LMS-системи 2. Скрол сторінки та перегляд інформації
Результат	Користувач успішно переглянув та ознайомився з інформацією про систему

Таблиця 2.4

Сценарій створення курсу

Назва	Створення навчального курсу
Опис	Вчителі можуть створювати та налаштовувати навчальні курси в системі LMS. Вони додають матеріали, завдання, тести та інші ресурси, які студенти можуть використовувати для навчання.
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсів 3. Натискання кнопки створення курсу 4. Введення даних про курс 5. Додавання файлів та інших ресурсів 6. Натискання кнопки «Створити»
Результат	Користувач успішно створив курс

Таблиця 2.5

Сценарій створення завдання

Назва	Створення завдання студенту
Опис	Вчителі можуть створювати завдання для студентів через LMS, надавати доступ до них та збирати роботи в електронному форматі.
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсу 3. Натискання кнопки створення завдання 4. Заповнення завдання контентом 5. Натискання кнопки «Зберегти»
Результат	Користувач успішно створив завдання

Таблиця 2.6

Сценарій оцінення завдання

Назва	Оцінення завдання студента
Опис	Вчителі можуть оцінювати здані завдання студентами.
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсу 3. Натискання кнопки перегляду відповідей 4. Перегляд відповідей та оцінювання 5. Натискання кнопки «Зберегти»
Результат	Користувач успішно оцінив завдання

Таблиця 2.7

Сценарій перегляду аналітики

Назва	Перегляд аналітики курсу
Опис	Вчителі можуть переглядати аналітику по курсу, скільки часу займають завдання, середні оцінки та ін.
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсу 3. Натискання кнопки перегляду аналітики 4. Ознайомлення з аналітикою: діаграми, графіки та таблиці
Результат	Користувач переглянув аналітику курсу

Таблиця 2.8

Сценарій обмеження доступу до курсу

Назва	Обмеження доступу до курсу
Опис	Вчителі можуть встановлювати певні обмеження щодо доступу до курсу, наприклад, зробити його публічним, дати доступ тільки за посиланням або тільки для людей, які отримують запрошення.
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсу 3. Натискання кнопки редагування курсу 4. Перехід в налаштування конфіденційності та доступу 5. Зміна доступу згідно потреб
Результат	Користувач змінив доступність курсу

Таблиця 2.9

Сценарій редагування курсу

Назва	Редагування курсу
Опис	Вчителі можуть редагувати наявні курси
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсу 3. Натискання кнопки редагування курсу 4. Редагування відповідних полей або інших даних 5. Натискання кнопки збереження курсу
Результат	Користувач оновив курс

Сценарій використання системи студентом

Таблиця 2.10

Сценарій перегляду інформації про курс

Назва	Ознайомлення з контентом курсу
Опис	Студенти можуть перейти на сторінку будь-якого доступного (що не обмежено налаштуваннями доступності) курсу
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсів 3. Натискання на кнопку деталей будь-якого курсу 4. Ознайомлення з даними про курс
Результат	Користувач переглянув дані про курс

Таблиця 2.11

Сценарій приєднання до публічного курсу

Назва	Приєднання до відкритого курсу
Опис	Студенти можуть приєднуватись до курсів, до яких є доступ, базовий випадок – це відкритий курс для всіх
Порядок дій	1. Відкриття сторінки LMS-системи 2. Перехід на сторінку курсів 3. Натискання на кнопку деталей будь-якого курсу 4. Натискання кнопки приєднання до курсу
Результат	Користувач приєднався до курсу

Таблиця 2.12

Сценарій приєднання до приватного курсу

Назва	Приєднання до закритого курсу за допомогою посилання-запрошення
Опис	Студенти можуть отримати від викладача посилання на курс задля приєднання до нього
Порядок дій	1. Студент переходить / відкриває посилання-запрошення на курс від викладача 2. Відкривається сторінка курсу 3. Натискання кнопки приєднання до курсу
Результат	Користувач приєднався до курсу

Таблиця 2.13

Сценарій відстеження прогресу проходження курсу

Назва	Відстежування прогресу проходження курсу
Опис	Студенти можуть отримати інформацію про прогрес проходження курсу у відсотковому значенні
Порядок дій	1. Студент переходить на сторінку всіх курсів 2. Студент дивиться на плитку курсу та отримує інформацію про прогрес проходження (на кожному початку курсу є Progress-Bar з прогресом)
Результат	Користувач отримав інформацію про прогрес проходження курсу

Таблиця 2.14

Сценарій отримання дод. Інформації від викладача

Назва	Отримання додаткової інформації від викладача
Опис	Студенти можуть отримати будь-яку додаткову інформацію від викладача в курсі або завданні
Порядок дій	1. Студент переходить на сторінку конкретного курсу або завдання 2. Студент вводить запитання до викладача в спеціальному полі на сторінці курсу / завдання 3. Студент натискає кнопку відправлення повідомлення 4. Викладач надсилає відповідь студенту на його запитання
Результат	Користувач отримав затребувану інформацію

Таблиця 2.15

Сценарій перегляду і завантаження файлів курсу

Назва	Перегляд / завантаження файлів, які прикріплені до курсу / завдання
Опис	Студенти можуть переглядати та завантажувати певні навчальні матеріали зі сторінки курсу та завдання (якщо такі матеріали були надані викладачем)
Порядок дій	1. Студент переходить на сторінку конкретного курсу або завдання 2. Студент натискає на файл, який прикріплений до відповідного курсу / завдання 3. За можливості студент бачить передогляд файлу, в інакшому випадку виводиться повідомлення, що неможливо відобразити цей файл 4. Студент натискає на іконку завантаження для завантаження файлу на пристрій
Результат	Користувач завантажив файл на пристрій