НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.9_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інтегровані інформаційні системи»

спеціальності 126 «Інформаційні системи та технології»

на тему: «Освітня платформа з елементами гейміфікації»

Виконав:
студент 2 курсу, групи ІА-31мп
Вознюк Микола Володимирович _____

Керівник:
доцент кафедри ІСТ ФІОТ, к.т.н., доцент,
Жураковська Оксана Сергіївна _____

Рецензент:
доцент кафедри ІПІ, к.т.н., доцент,
Баклан Ігор Всеволодович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти — другий (магістерський)

Спеціальність — 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Вознюку Миколі Володимировичу

1. Тема проєкту «Освітня платформа з елементами гейміфікації», науковий керівник дисертації Жураковська Оксана Сергіївна, к.т.н. доц., затверджені наказом по університету від «08» 11 2024 р. № 5016-с
2. Термін подання студентом дисертації: “9” 12 2024 р.
3. Об’єкт дослідження: процес організації навчання із застосуванням технік гейміфікації, який включає створення та публікацію навчальних курсів, забезпечення їх доступності для користувачів, а також підтримку проходження та засвоєння матеріалів.
4. Вихідні дані: освітня платформа з елементами гейміфікації має бути реалізована із забезпечення високої продуктивності системи з можливістю обробки до 100 запитів на хвилину, швидкість відповіді сервера не більше 500 мілісекунд, підтримкою масштабованості для забезпечення надійності, а також гарантувати безпеку даних користувачів.
5. Перелік завдань, які потрібно зробити: провести детальний аналіз існуючих освітніх платформ та їх підходів до гейміфікації, спроєктувати архітектуру системи з урахуванням багаторольності та мультифункціональності. Визначити оптимальний технологічний стек. На його основі реалізувати бекенд та фронтенд частини платформи. Розробити інтерактивний конструктор курсів з відібраними в ряді дослідження гейміфікаційними техніками. Провести всебічне модульне та інтеграційне тестування системи

для забезпечення її надійності та відповідності технічним вимогам. Підготувати повну текстову та графічну частину пояснювальної записки, що детально описує всі аспекти розробки та функціонування платформи.

6. Перелік графічного матеріалу: діаграма діяльності викладача, діаграма діяльності студента, діаграма варіантів використання платформи студентом, діаграма варіантів використання платформи викладачем, ER-діаграма сутностей освітньої платформи, структура бази даних освітньої платформи, діаграма класів освітньої платформи, діаграма послідовностей дій в системі для викладача, діаграма послідовностей дій в системі для студента, діаграма компонентів, структурна схема елементів картки уроку, приклад гнучкої структури.

7. Орієнтовний перелік публікацій: публікації не плануються.

8. Дата видачі завдання "02" 09 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	09.09.2024	
2	Аналіз предметної області	13.09.2024	
3	Постановка та формалізація задачі	20.09.2024	
4	Огляд наявних аналогів	27.09.2024	
5	Аналіз вимог до програмного забезпечення	01.10.2024	
6	Обґрунтування використовуваних технічних засобів	04.10.2024	
7	Розробка архітектури програмного забезпечення	11.10.2024	
8	Розробка програмного забезпечення	01.11.2024	
9	Тестування та налагодження програми	08.11.2024	
10	Виконання графічних документів	15.11.2024	
11	Оформлення пояснювальної записки	29.11.2024	

Студент Микола ВОЗНЮК

Науковий керівник Оксана ЖУРАКОВСЬКА

РЕФЕРАТ

Освітня платформа з елементами гейміфікації: 169с., 17 рис., 53 табл., 10 дод., 26 джерел.

ОСВІТНЯ ПЛАТФОРМА, ГЕЙМІФІКАЦІЯ, КОНСТРУКТОР КУРСІВ, СИСТЕМА ДИСТАНЦІЙНОГО НАВЧАННЯ.

Актуальність. Актуальність теми обумовлена зростаючим попитом на ефективні онлайн-освітні рішення та проблемою мотивації і продуктивності навчання студентів, що є характерним для систем дистанційного навчання.

Зв'язок дослідження з науковими програмами, планами, темами. Робота виконана в рамках НДР "Інформаційні технології і системи підтримки прийняття рішень. Високопродуктивні комп'ютерні системи та мережі" (державний реєстраційний номер 0124U002078).

Мета і задачі дослідження. Метою даного дослідження є підвищення ефективності процесу навчання шляхом імплементації гейміфікаційних технік у навчальний процес довільної тематики. Для досягнення мети необхідно виконати завдання: провести аналіз існуючих освітніх платформ та застосованих ними технік гейміфікації; визначити найбільш ефективні гейміфікаційні практики для інтеграції в навчальний процес; визначити вимоги, характерні модифікованому гейміфікацією навчальному процесу; спроєктувати архітектуру прототипу освітньої платформи з урахуванням визначених вимог та технік; реалізувати прототип платформи та провести його тестування для оцінки впливу на ефективність навчального процесу; оцінити можливості впровадження платформи.

Об'єктом дослідження є процес організації навчання з використанням гейміфікаційних технік. Предметом дослідження виступають методи імплементації гейміфікаційних технік у навчальний процес.

Практичне значення. Розроблена платформа та її модулі можуть бути впроваджені як самостійне рішення, або як підтримуючий інструмент для освітніх закладів та програм дистанційного навчання, сприяючи підвищенню мотивації та продуктивності навчального процесу.

ABSTRACT

Educational platform with gamification elements: 169 p., 17 draw., 53 tab., 10 app., 26 sources.

EDUCATIONAL PLATFORM, GAMIFICATION, COURSE CONSTRUCTOR, DISTANCE LEARNING SYSTEM.

Relevance. The relevance of the topic is due to the growing demand for effective online educational solutions and the issues of student motivation and productivity characteristic of distance learning systems.

Connection of the work with scientific programs, plans, and themes. This work was carried out within the framework of the research project "Information Technologies and Decision Support Systems. High-Performance Computer Systems and Networks" (State registration number 0124U002078).

Aim and objectives of the research. The aim of this research is to enhance the efficiency of the learning process by implementing gamification methods into educational processes of any subject matter. To achieve this goal, the following tasks need to be accomplished: analyze existing educational platforms and the gamification techniques they employ; identify the most effective gamification practices for integration into the educational process; determine the requirements for a gamified educational process; design the architecture of a prototype educational platform considering the defined requirements and techniques; implement the platform prototype and conduct testing to evaluate its impact on the efficiency of the educational process; assess the feasibility of implementing gamification techniques.

The object of the research is the process of organizing learning using gamification methods. The subject of the research is the methods of implementing gamification techniques into the educational process.

Practical significance. The developed platform and its modules can be implemented as a standalone solution or as an auxiliary tool for educational institutions and distance learning programs, contributing to increased motivation and productivity in the learning process.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

СУБД – система управління базою даних

API – Application Programming Interface – певний контракт, що визначає, якіз застосунком можуть взаємодіяти інші

JPA – Java Persistence API – специфікація Java для збереження, доступу та управління даними між об'єктами Java та реляційними базами даних

NPM – Node Package Manager – менеджер пакетів для NodeJS

DAO – Data Access Object – абстрактний інтерфейс до бази даних або іншого сховища

DTO – Data Transfer Object – об'єкт, який використовується для передачі даних між шарами системи

JWT – JSON Web Token – стандарт відкритого доступу для передачі безпечної інформації у вигляді JSON-об'єкта

OAuth – Open Authorization – стандарт авторизації

REST – Representational State Transfer – стиль написання веб-застосунків, описує як користувачу слід організовувати написання коду веб-застосунку

SQL – Structured Query Language – мова написання запитів, за допомогою якої можна взаємодіяти із базами даних

MVC – Model-View-Controller – архітектурний шаблон для побудови застосунків

ВСТУП

У сучасному світі освітні процеси зазнали суттєвих змін. Під впливом пандемії коронавірусу 2020-2021 та, водночас, за стрімкого розвитку інформаційних технологій відбувся різкий перехід навчальних процесів на дистанційний режим та у бік онлайн навчання, що базується на лекціях та завданнях тестового характеру розміщених у хмарному середовищі та доступних через інтернет.

Такі системи часто не забезпечують необхідного рівня мотивації та залученості учнів, що призводить до зниження ефективності навчання, високої плинності студентів та недостатнього засвоєння матеріалу. Відповідно, виникає потреба у впровадженні новітніх підходів, які можуть зробити навчальний процес більш інтерактивним, цікавим та адаптивним до індивідуальних потреб учнів.

Одним із перспективних напрямків у цій сфері є гейміфікація – використання ігрових елементів або методик у неігрових контекстах. Гейміфікація дозволяє модифікувати навчальний процес, створивши умови, в яких учні при навчанні мають простір для вибору, можливість обирати стратегію, чи навчатись за певними «ігровими» правилами. Досягаючи певного прогресу, студенти отримують винагороди. Таким чином сприйняття навчального курсу як певної системи з власними правилами та стратегіями дослідження підсилює вплив емоційного та психологічного характеру, що, у свою чергу, посилює мотивацію та відчуття задоволення від навчального процесу.

Освітня платформа з ефективними техніками гейміфікації сприятиме покращенню користувацького досвіду викладачів та учнів. Зробить процес створення складних та якісних курсів зручним та швидким, що у свою чергу надасть викладачам більше можливостей зосередитися на проєктуванні змісту курсу, а не на технічних аспектах його створення. Для студентів це стане важливим фактором, адже якісний контент із гейміфікаційними елементами підвищить їхній інтерес і мотивацію, стимулюватиме їх до активного засвоєння навчального матеріалу. А завдяки інтерактивним завданням і ігровим механікам, які

перетворюють навчання на захопливий процес, студенти зможуть легше засвоювати матеріал і практичні навички.

Незважаючи на наявність освітніх платформ, що активно застосовують гейміфікацію, вони, у більшості, зосереджуються на конкретній тематиці та реалізовують власний продукт як не гнучкий застосунок, що не дає можливості змінити тематику без втрати ефективності навчального процесу. Відповідно, існують невирішені питання щодо уніфікації процесу створення курсів довільної тематики.

Проведення дослідження, зосередженого на систематизації методик гейміфікації як частини навчального процесу, вивченні можливостей їх застосування до тем різної спрямованості та їх впливу на навчання, дозволить створити гнучку методологію, яка забезпечить застосування гейміфікації незалежно від тематики навчального матеріалу та покращить досвід учасників навчання.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис процесу діяльності

Предметною областю даної роботи є організація навчального процесу з впровадженням технік гейміфікації. Для цього пропонується розробити онлайн-застосунок — платформу для хостингу навчальних курсів, яка спрямована на підтримку навчального простору, що інтегрує гейміфікаційні техніки для підвищення ефективності освітнього процесу.

Для таких систем можна виділити кілька основних компонентів:

– навчальний курс: об’єкт створений певним користувачем, що має визначену структуру, наповнений навчальними матеріалами та поділений на уроки. Має певні умови успішного завершення та градації оцінок.

– урок: структурна одиниця курсу, що визначає матеріал, який відповідає певній темі. Має метрики оцінювання, часу та спроб затрачених на завершення завдання визначеного уроком.

– конструктор: система створення навчальних курсів, що дозволяє викладачу створювати, наповнювати та структурувати навчальний матеріал для його подальшої публікації.

– система підписки: механізм отримання студентом доступу до навчального матеріалу шляхом підпису на навчальний курс через функціонал платформи.

– елементи гейміфікації: механізми або правила що надають навчальному процесу характерних ігрових елементів.

Подані компоненти формують наступні основні процеси системи:

– побудова навчального курсу: викладач за допомогою конструктора створює власний курс, у якому структурує визначені ним матеріали по урокам, які в свою чергу також розподілені за типами завдань (лекції або оціночні роботи), задає для них градації оцінок та умови проходження. Також конструктор надає функціонал для заповнення уроків згідно їх типу, з урахуванням усіх потрібних деталей.

– публікація курсів: за бажанням викладача система автоматизує процес переведення курсу у активну фазу. У якій курс стає доступним для підписки студентам що дає їм можливість проходити курс. Важливо відмітити що на етапі проходження всі дані про цей процес збираються та передаються на огляд викладачу.

– процес навчання: здійснюється студентом, завданням якого є проходження усього обов'язкового матеріалу із достатнім обсягом успішності визначеним оціночними метриками.

1.2 Опис процесу діяльності

Основний бізнес-процес – надання послуг з навчання користувачів за обраними ними курсами. Для цього необхідно забезпечити взаємодію між системою та її основними користувачами: викладачами, студентами.

Для реалізації власних процесів користувачам необхідно бути аутентифікованими у системі, а, відповідно, мати персоналізовані акаунти, що давали б можливість контролювати власні дані або об'єкти, створені та розміщені на платформі. Акаунти мають бути захищені відповідними даними входу, при введені яких у систему, користувачу надавався б доступ до власного облікового запису. Після входу користувач отримує право доступу до особистої теки, у якій він може редагувати як власні дані так і зберігати потрібні йому об'єкти для роботи у системі.

Діаграми що описують процеси діяльності користувача при реєстрації та авторизації зображено на рисунках 1.1 та 1.2 відповідно.

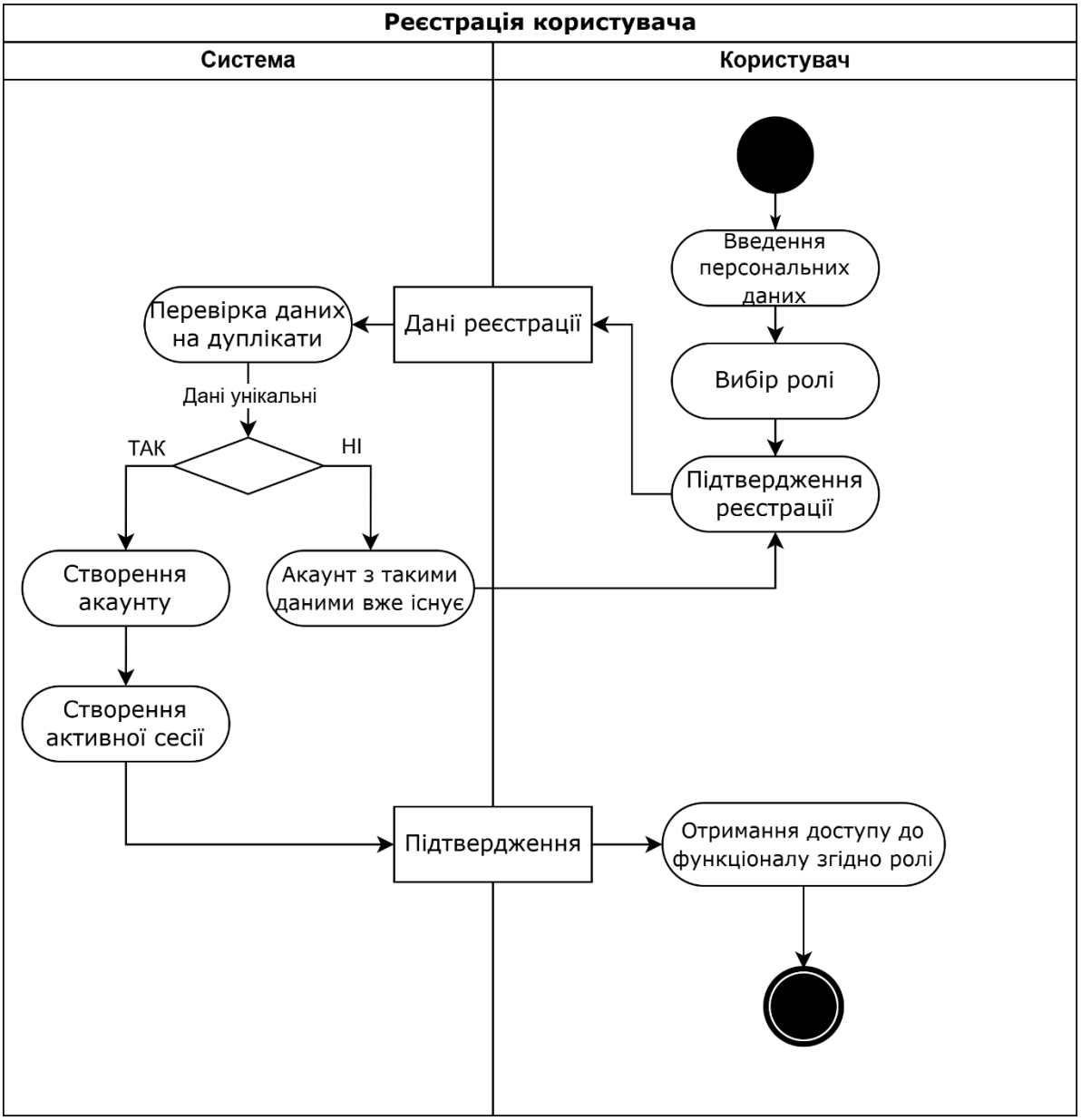

Рисунок 1.1 — Діаграма діяльності користувача при реєстрації

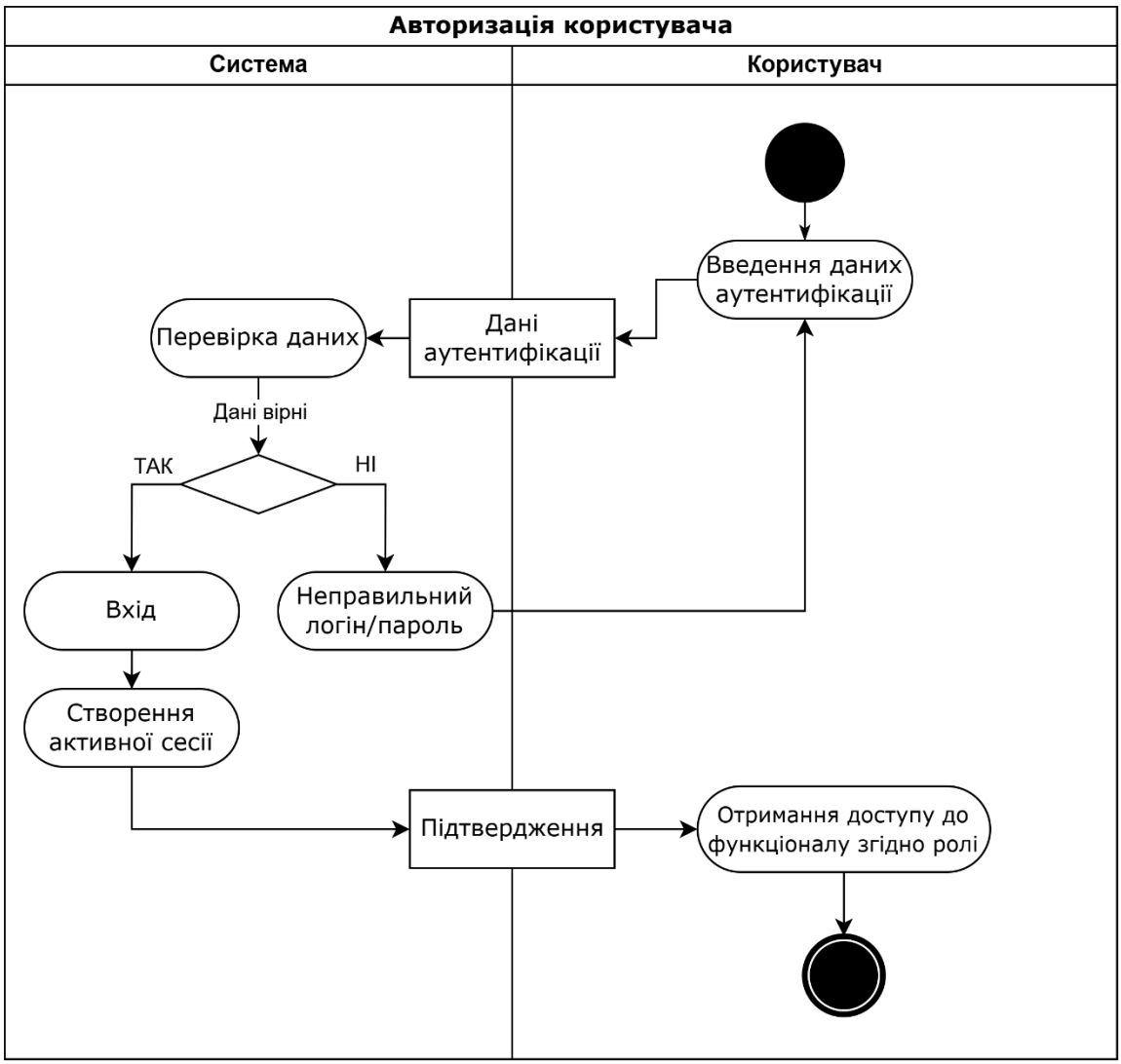

Рисунок 1.2 — Діаграма діяльності користувача при авторизації

Платформа реалізує розподіл функціоналу за трьома ролями: викладачі, студенти та адміністратори. Викладачі мають доступ до конструктора курсів, можуть створювати навчальні матеріали, організовувати їх у певні структури та публікувати їх для доступу студентів. Студенти проходять навчальні курси, створені викладачами, вивчають матеріал у поданих викладачем структурах, можуть обирати курси через теку курсів та підписуватись на них для проходження. Адміністратори відповідають за керування профілями користувачів, їх ролями та сесіями.

1.2.1 Опис процесу діяльності викладача

Як вже було сказано вище викладачі відіграють роль «надавача» матеріалів. Ця роль розширює базовий функціонал користувача можливостями створення, редагування, видалення, публікації та огляду активного курсу з детальною інформацією про його використання студентами.

Найважливішим процесом діяльності у полі можливостей ролі викладача є створення курсу. Цей процес включає в себе підготовку структури курсу, задання можливостей студента у полі цього курсу та наповнення його матеріалом згідно власної програми. Саме у цьому моменті платформа має забезпечити можливість інтеграції гейміфікованих принципів, що розширить спектр можливостей взаємодії студентів з навчальним матеріалом.

Діаграму що описує процеси діяльності викладача наведено у додатку Б. Там наведено основну діяльність викладача, а саме конструювання курсів з інтеграцією додаткових елементів.

1.2.2 Опис процесу діяльності студента

Роль студентів на платформі – кінцевий споживач. Його основною можливістю є використання створених викладачем курсів задля досягнення освітніх цілей. Також, враховуючи те що платформа має забезпечувати гейміфіковані принципи, студент може взаємодіяти з ними як у межах курсу так і глобально на платформі. Діаграму діяльності процесів студента наведено у додатку В. На ньому детально зображено що студент здійснює навчальну діяльність за яку отримує винагороди та може з ними взаємодіяти.

1.3 Постановка задачі

1.3.1 Призначення системи

Призначенням платформи є уніфікація процесу створення навчальних курсів довільної теми з імплементацією визначених гемифікаційних технік, які можна було б вважати базовими. Це дозволить використовувати різні прийоми для

підвищення ефективності навчального процесу та покращення особистого досвіду як викладачів так і студентів.

Види діяльності, що підлягають дослідженню:

а) підхід до конструювання курсів:

1) систематизація процесу побудови курсу: Розробка чіткої методології створення курсу, що включала б в себе усі етапи що потрібні для повного визначення курсу;

2) структуризація курсу: встановлення структури, яка б в повній мірі відображала курс та його особливості незалежно від обраної теми;

б) гейміфікаційні техніки:

1) відбір технік: визначення декількох найбільш застосовуваних на практиці технік гейміфікації, які мають найбільший вплив на освітній процес;

2) систематизація даних: визначення метрик які допомагають у інтеграції гейміфікаційних технік;

в) принципи проходження курсів:

1) систематизація процесу проходження курсів: визначення умов проходження курсу, можливостей до різних варіантів його завершення, розширення курсу додатковими матеріалами;

2) гнучкість процесу: надання студентами можливості адаптувати процес навчання під свої потреби.

1.3.2 Цілі та задачі розробки

Метою даної роботи визначено підвищення ефективності процесу навчання шляхом імплементації гейміфікаційних технік у навчальний процес довільної тематики.

Для того, що б досягнути визначеної мети, необхідно виконати наступний ряд завдань:

а) провести аналіз, щодо застосування гейміфікації в освіті та її впливу на ефективність навчального процесу;

б) визначити техніки, які можна застосовувати з найбільшим ефектом незалежно від тематики навчання;

в) спроєктувати платформу з урахуванням визначених технік:

1) розробити механізм авторизації в системі;

2) розробити функціонал CRUD операцій для курсів та уроків з правами доступу за ролями;

3) систему оцінювання завдань;

4) систему запису студентів та проходження ними курсів;

5) розробити інтерфейс:

– авторизації на платформі;

– персонального робочого простору, що відрізняється в залежності від ролі користувача та демонструє різні відображення навчальних елементів, таких як курс, урок тощо;

г) спроєктувати конструктор курсів:

1) розробити функціонал конструктора:

– спроєктувати інтуїтивно зрозумілий інтерфейс для викладачів, який дозволяє швидко створювати курси незалежно від їхньої тематики;

– інтегрувати CRUD-операції для роботи з елементами курсу, для уніфікації з платформою;

2) на основі дослідження впровадити найкращі практики гейміфікації, з відповідними їм інтерфейсами до конструктора;

Висновки до розділу 1

У даному розділі було детально розглянуто предметне середовище що досліджується та заснована на ньому освітня платформа, яка являє собою комбінацію конструктора курсів та простору для їх розміщення з можливістю доступу до них користувачами для проходження та засвоєння матеріалів що подані в них.

Така платформа має бути орієнтовна на підтримку діяльності двох ключових ролей – викладачів та студентів. Відповідно до цих ролей сформовані основні бізнес-процеси: створення, налаштування та управління курсами для викладачів та проходження цих курсів, шляхом виконання завдань, із використанням інтегрованих технік гейміфікації для студентів.

Детальний аналіз потреб користувачів дозволив окреслити ключові функції платформи, які сприяють освітньому процесу. Також було визначено основні сценарії використання системи, починаючи від реєстрації користувача з певною функціональною роллю та створення курсів, до їх застосування.

Це дало змогу сформулювати цілі й задачі, які спрямовані на дослідження можливостей гейміфікації та створення робочого прототипу платформи з інтегрованим у нього конструктором, що забезпечує повний та інтуїтивний інструментарій створення навчальних курсів, та застосуванням визначених технік гейміфікації, які мають покращити досвід учасників платформи та підвищити ефективність навчального процесу.

Завдяки сформульованим цілям і визначеним завданням було створено план дослідження, що дозволив впорядкувати подальшу роботу і оптимізувати процес дослідження та інтеграції результатів у прототип.

2 АНАЛІЗ ГОТОВИХ РІШЕНЬ

Розробка нової платформи передбачає глибокий аналіз та порівняльну характеристику вже наявних на ринку аналогів та дослідження їх найбільш вдалих практик. Важливо розуміти, якими характерними рисами вони наділені, які переваги та недоліки притаманні обраному їм набору рішень. Це дозволить визначити вимоги до нашого продукту, сформувати набір найкращих практик і на основі них сформувати базис технік, які можуть бути інтегровані у будь-яку навчальну програму та задовольняють умовам покращення досвіду учасників платформи.

У цьому розділі буде проведено детальний аналіз найбільш популярних освітніх платформ, та їх спроб застосування гейміфікацій, таких як Khan Academy, Duolingo, Coursera, Udemy, Edx.

2.1 Огляд освітніх платформ

2.1.1 Khan Academy

Академія Хана була створена як некомерційна організація, що спрямовувала свою діяльність на забезпечення освіти «для будь-кого і будь-де». Проєкт став ініціативою випускника MIT педагога Салмана Хана, який хотів створити платформу де можна буде навчитись різним навичкам або наукам виключно безкоштовно для самого студента [1].

2.1.1.1 Бізнес модель проєкту:

Проєкт позиціонує себе як відкриту для всіх бажаючих базу знань. Підтримується за допомогою пожертвувань з різних благодійних фондів та корпорацій. Має опцію малих пожертвувань від користувачів.

2.1.1.2 Особливості платформи:

а) широкий спектр предметів: Khan Academy пропонує курси з математики, природничих наук, економіки, історії, комп'ютерних наук та багатьох інших дисциплін;

б) вправи та тести: Платформа забезпечує учнів завданнями, які дозволяють закріплювати знання та відстежувати прогрес по проходженню ними курсу. Платформа пропонує наступний перелік завдань: тести, письмові вправи, перевірки програмного коду, спільні чати та відео сесії з викладачем, курсові роботи (великі роботи що об’єднують тести та письмові вправи) [2];

в) система балів та бейджів: Платформа надає можливість здобувати бейджі (відзнаки) за певні дії учасників на платформі. Це використовуються для мотивації учнів до регулярного навчання та досягнення певних цілей і сприймається ними як здобуті трофеї. Вони відмічаються у профілі та демонструють майстерність його власника що також впливає на мотивацію студента[3].

2.1.2 Duolingo

Duolingo була заснована в 2011 році Луїзі Піроль та Сергієм Бриномз метою зробити вивчення іноземних мов доступним для всіх. Платформа швидко здобула популярність завдяки своєму унікальному підходу до навчання через ігрові елементи. [4]

2.1.2.1 Бізнес модель проєкту:

На поточний момент сервіс працює у форматі, який компанія називає «фриміум», пропонуючи як безкоштовний, так і платний доступ до платформи, що розрізняються розширеними можливостями платного режиму, але при цьому безкоштовний режим не обмежує основний функціонал.[5]

2.1.2.2 Особливості платформи:

а) фокус на мовах: Duolingo спеціалізується на вивченні іноземних мов, пропонуючи курси для більш ніж 30 мов, включаючи популярні (англійська, іспанська, французька) та менш поширені (наприклад, валлійська чи хінді);

б) ігрові механіки: Платформа активно використовує елементи гейміфікації, такі як: рівні знання мови, нагород у вигляді бейджів, таблиці лідерів за набраними балами у навчанні та серії вправ, що мотивують до щоденного навчання. Користувачі отримують бонуси за безперервну серію днів навчання;

в) доступність: Duolingo пропонує доступ до своїх сервісів через вебплатформу, а також мобільні застосунки для Android, iOS та Windows. Це забезпечує зручність навчання незалежно від пристрою. Крім того, додатки мають офлайн-режим, що дозволяє займатися навіть без доступу до Інтернету;

г) адаптивне навчання: Система автоматично адаптується до рівня знань користувача завдяки: вхідним тестам, які визначають початковий рівень та динамічному підбору щоденних завдань, які відповідають поточному рівню знань, враховують слабкі місця учня;

2.1.3 Coursera

Coursera була заснована у квітні 2012 року Ендрю Ин та Дефною Коллер з Стенфордського університету з метою забезпечити доступ до якісної освіти від провідних університетів та компаній світу. З моменту свого створення Coursera розширила свій асортимент курсів та партнерств, ставши однією з найбільших онлайн-платформ для навчання[6].

2.1.3.1 Бізнес модель проєкту:

Coursera використовує модель монетизації, яка поєднує платний і безкоштовний доступ до навчальних матеріалів:

а) система сертифікатів: користувачі можуть безкоштовно отримати доступ до матеріалів багатьох курсів, але для отримання сертифіката про завершення необхідно сплатити встановлену суму. Сертифікати є одним із ключових джерел доходу платформи, за рахунок їх цінності як документу підтверджуючого здобуті навички;

б) навчальні програми: Coursera пропонує користувачам варіант доступу до серії взаємопов'язаних курсів, спрямованих на здобуття спеціалізації. Такий варіант є вигіднішим за придбання окремих курсів і надає знижку за придбання цілої програми;

в) Coursera Plus: Це модель підписки, яка надає користувачам необмежений доступ до більшості курсів і програм на платформі за фіксовану щорічну або щомісячну плату. Coursera Plus пропонує зручність для тих, хто планує пройти кілька курсів, дозволяючи заощадити в порівнянні з оплатою окремих курсів[7];

г) Coursera for Business: Цей напрямок орієнтований на корпоративних клієнтів і пропонує рішення для професійного навчання співробітників компаній. Організації можуть підписатися на спеціалізовані програми для підвищення кваліфікації своїх працівників. Coursera for Business забезпечує індивідуальні рішення для компаній, включаючи відстеження прогресу працівників, звіти про ефективність навчання та персоналізований підбір курсів;

д) партнерства з університетами та компаніями: Coursera співпрацює з провідними університетами та компаніями з усього світу для створення високоякісних навчальних курсів і програм. Прибуток від таких курсів ділиться між Coursera і партнерами, стимулюючи розробників контенту до створення актуальних і практичних програм. Університети отримують можливість залучати глобальну аудиторію, а компанії — популяризувати свої знання та технології.

2.1.3.2 Особливості платформи:

а) партнерство з провідними університетами: Coursera співпрацює з університетами, такими як Стенфорд, Йель, Прінстон, Caltech та багатьма іншими. що гарантує високу якість контенту, створеного експертами з різних галузей;

б) широкий спектр курсів: Курси охоплюють різні сфери знань, включаючи технології, бізнес, мистецтво, гуманітарні науки та багато іншого;

в) сертифікати: Coursera надає користувачам можливість отримувати офіційні сертифікати після успішного завершення курсу. Ці сертифікати підтверджують здобуті навички та знання і є важливим документом, який може бути доданий до резюме або професійного портфоліо. Ці сертифікати дуже часто мають академічну акредитацію, вони користуються попитом серед роботодавців, оскільки створюються у партнерстві з провідними університетами та організаціями;

г) спільнота та форуми: Платформа забезпечує можливість спілкування з іншими учасниками курсів, що сприяє обміну знаннями та досвідом;

д) інтерактивні завдання та тести: Забезпечують практичне закріплення знань та оцінювання прогресу. Щоправда варто відзначити, платформу часто критикують за систему оцінювання, де студентів оцінюють інші студенти, які не завжди мають достатній рівень фаховості; крім того, анонімність оцінювача дає йому змогу в коментарях до роботи писати будь-що, навіть виходячи за межі етичних норм;

е) стандартний навчальний процес: На відміну від інших платформ Coursera вирішила притримуватись стандартної моделі викладання навчального матеріалу, обмежуючись викладенням матеріалів які розбиті на тижні без додаткових ускладнень структури або послідовностей. Не враховуючи систему сертифікатів, яку можна визначити як систему нагород, платформа не наділена ігровими елементами.

2.1.4 Udemy

Udemy була заснована у 2010 році Ереном Балі, Октаєм Чагларом і Гаганом Біяні як платформа для онлайн-освіти. Ідея виникла у 2007 році, коли засновники створили програмне забезпечення для живого віртуального класу в Туреччині. У 2010 році вони запустили Udemy — "Вашу академію" — як платформу, доступну для всіх, і всього за кілька місяців вона зібрала понад 10 000 користувачів, 1000 викладачів та 2000 курсів.

29 жовтня 2021 року Udemy провела IPO, ставши публічною компанією під символом UDMY, що закріпило її позицію як однієї з провідних платформ для онлайн-освіти у світі [8].

2.1.4.1 Бізнес модель проєкту:

а) система маркетплейсу: Udemy надає можливість для викладачів створювати та продавати власні курси. Вони можуть бути безкоштовні, або надаватись безоплатно в певному обсязі для залучення студентів. Доходи від продажів поділяються між автором та компанією у пропорціях в залежності від того чи знайшов студент курс на платформі, чи через рекламу самої Udemy, чи по спеціальному реферальному посиланню. Дохідність варіюється від 3% до 75% від вартості курсу[9];

б) Udemy Personal Plan: персональний план за яким доступ до більшості курсів платформи за умови оплати місячної або річної підписки на даний пакет;

в) Udemy Business: Udemy пропонує підписку для. Ця послуга надає компаніям доступ до вибраного каталогу курсів для навчання співробітників. Компанії сплачують щорічну підписку, що забезпечує стабільний дохід для платформи.

2.1.4.2 Особливості платформи:

а) доступність: Платформа пропонує курси за доступними цінами, часто з великими знижками, що робить навчання доступним для широкої аудиторії;

б) мобільний додаток: Udemy пропонує мобільні додатки для iOS та Android, що дозволяє навчатися на ходу та завантажувати матеріали для офлайн-доступу.

в) доступність публікації: Udemy дозволяє користувачам реєструватись як інструкторам і самостійно створювати власні курси та викладати їх;

г) довічний доступ: Після придбання курсу користувач отримує необмежений доступ до його матеріалів, що дозволяє повторно переглядати уроки у будь-який час;

д) акредитація: Сертифікати Udemy не мають академічної акредитації або професійного визнання, що відрізняє їх від сертифікатів Coursera, які мають більшу вагу у професійних колах.

2.1.5 Edx

edX — це онлайн-платформа для навчання, заснована у травні 2012 року Массачусетським технологічним інститутом (MIT) та Гарвардським університетом. Перший курс, присвячений схемам та електроніці, залучив 155 000 студентів із 162 країн. У 2013 році edX співпрацювала зі Стенфордським університетом, і до червня того ж року кількість студентів досягла 1 мільйона. У 2021 році компанія 2U придбала edX за 800 мільйонів доларів, зобов'язавшись зберегти та розвивати місію платформи.

2.1.5.1 Бізнес модель проєкту:

а) edX пропонує безкоштовний доступ до більшості своїх курсів, але для отримання сертифікатів про завершення курсу користувачі повинні сплатити певну суму;

б) платформа також пропонує професійні сертифікати та мікро-магістерські програми, які є платними та надають більш глибокі знання та офіційні документи про завершення;

в) edX співпрацює з університетами та компаніями, пропонуючи корпоративні навчальні програми.

2.1.5.2 Особливості платформи:

а) партнерство з провідними університетами та організаціями: edX співпрацює з понад 160 установами, включаючи таких світових гігантів як MIT, Гарвард, Берклі та інші, що забезпечує високу якість навчальних матеріалів [10].

б) академічність: курси платформи є повноцінними університетськими програмами з певних дисциплін, що робить їх академічно цінними, але через це не дає широкого спектра курсів, орієнтованих на хобі або загальні навички;

в) акредитація: платформа пропонує різні сертифікати, зокрема професійні, які визнаються як академічними установами, так і роботодавцями, підтверджуючи здобуті знання та навички. А деякі програми акредитуються як магістерські та можуть бути використанні при подальшому навчанні у відповідних університетах.

2.2 Аналіз гейміфікаційних практик

Для визначення базових технік гейміфікації, які мають найкращий вплив на навчальний процес, визначимо характерні для платформ техніки та оцінимо їх застосування. Результати подані у таблиці 2.1.

Таблиця 2.1 — Аналіз застосування технік гейміфікації

Платформа	Техніка Гейміфікації	Застосування
Khan Academy	Бали	Нараховуються за виконання завдань,

Платформа	Техніка Гейміфікації	Застосування
		стимулюючи прогрес у навчанні. Є ефективною мірою успішності проходження завдань.
	Бейджі (Значки)	Надаються за досягнення певних цілей, сприймаються як трофеї, що спонукає до дослідження навчальних матеріалів для збору та колекціонування якнайбільшої кількості значків. Прихована мотивація до детального вивчення матеріалів.
	Стрічка прогресу	Дозволяє візуально відстежувати прогрес, розуміти поточний рівень вивчення матеріалу.
	Інтерактивні завдання	Включають в себе відео, аудіо та текстові матеріали, підкріплені тестовими та практичними письмовими завданнями.

Платформа	Техніка Гейміфікації	Застосування
	Сертифікати	Неофіційні, підтверджують завершення курсу, можуть сприйматись аналогічно до бейджів як трофеї..
Duolingo	Рівні	Розподіляють навчання на певні групи уроків за темою або рівнем володіння мови, що краще сигналізує про рівень навичок студента. Також дозволяє студенту бачити якими навичками він вже володіє і може краще розподілити час навчання.
	Нагороди (Значки)	Система досягнень на Duolingo дозволяє користувачам отримувати значки за виконання різних навчальних завдань. Усі нагороди та досягнення збираються у профілі користувача, створюючи враження "полиці з трофеями".

Платформа	Техніка Гейміфікації	Застосування
		Вона включає нагороди за ключові досягнення, такі як досягнення нових рівнів, успіхи у лігах або встановлення рекордів.
	Рівні нагород	Деякі нагороди доступні для початківців (наприклад, за додавання друзів), тоді як інші призначені для досвідчених користувачів (наприклад, значки за рік безперервної серії навчання). Система рівнів мотивує як новачків, так і постійних користувачів, пропонуючи різний рівень викликів залежно від їхньої активності.
	Система особистих рекордів	Новий вид досягнень, який фіксує персональні рекорди, такі як найбільша кількість XP за день або найкраще місце у лізі.

Платформа	Техніка Гейміфікації	Застосування
		Відзначення персональних досягнень допомагає користувачам відчувати власний прогрес і сприяє бажанню покращити результати.
	Таблиця Лідерів	Створює конкурентне середовище, заохочуючи змагатися з іншими студентами за позицію у рейтингу. (Конкурентність не завжди може нести мотиваційний характер, а навіть іноді демотивувати)
	Серії	Мотивують регулярне навчання через нагороди за збереження послідовності днів. Мотивація підсилюється коли серія досягає великого значення і студент відчуває себе одним з найкращих і така серія цінується іншими студентами.

Платформа	Техніка Гейміфікації	Застосування
	Динамічна адаптація	Завдання адаптуються до рівня знань студента, забезпечуючи ефективніше навчання. Також якщо студент володіє певними знаннями він може пропустити цей рівень через систему «модульного» завдання, яке протестує ваші навички по цьому розділу і перевірить чи дійсно студенту варто пропустити цей модуль.
Coursera	Сертифікати	Офіційно визнані, підтверджують професійні та академічні знання.
	Форумна система	Створена для обговорення завдань. Не є дуже вдалою через обраний підхід до завдань інтерактивного характеру, де студенти оцінюють своїх колег по курсу. Через це форум перетворюється на чат з

Платформа	Техніка Гейміфікації	Застосування
		коментарями або проханнями про перевірку.
	Інтерактивні завдання	Мають базовий набір для онлайн курсів: тести, письмові завдання, відео, аудіо та текстові матеріали. Додатково інтегрований Jupiter Notebook що дозволяє виконувати завдання з програмування прямо на платформі і бачити результати їх виконання. Варто відмітити що є завдання які перевіряються самими студентами і це дуже часто приводить до конфузних ситуацій спричинених некомпетенцією та невмотивованістю студентів до якісного оцінювання робіт.
	Дедлайни	Чіткі терміни виконання завдань стимулюють до

Платформа	Техніка Гейміфікації	Застосування
		організованого навчання. Порушення дедлайнів закриває доступ до завдання і вимагає оновлення дедлайнів студентом вручну (не позбавляє студента права доступу на придбані курси)
Udemy	Сертифікати	Надаються після завершення курсу, підтверджуючи його проходження, хоч і не є акредитованими.
	Форумна система	Дозволяє задавати питання інструкторам та спілкуватися з іншими студентами в рамках курсу.
	Інтерактивні завдання	Мають базовий набір для онлайн курсів: тести, письмові завдання, відео, аудіо та текстові матеріали.
	Гнучкий графік	Відсутність дедлайнів дозволяє студентам навчатися у власному темпі.

Платформа	Техніка Гейміфікації	Застосування
edX	Сертифікати	Академічно акредитовані, визнаються університетами та роботодавцями.
	Форумна система	Сприяє обговоренню навчальних матеріалів між студентами та викладачами. (Найктивніша та найбільш корисна форумна система серед розглянутих)
	Інтерактивні завдання	Містить власну платформу для написання коду, вправи та тести для глибшого засвоєння матеріалу. Великий обсяг відеозаписів лекцій з провідних вузів світу.
	Змішаний підхід до графіку	Мають вільний графік але курс може деактивуватись у випадку довгої відсутності студента на платформі. (Курс не втрачається студентом а лише архівується,

Платформа	Техніка Гейміфікації	Застосування
		оплачені сертифікати зберігаються назавжди)

Висновки до розділу 2

У даному розділі було розглянуто найбільші освітні платформи та їх підхід до оптимізації навчального процесу. Особливий акцент було зроблено саме на дослідженні того як компанії застосовують гейміфікацію.

Розглянуті сервіси включають Khan Academy, Duolingo, Coursera, Udemy та edX. В результаті аналізу було визначено ключові особливості кожної платформи, їхні бізнес моделі. Можна стверджувати, що незважаючи на велику схожість у основних принципах, а саме у надаванні матеріалів у формі курсів з тестовими та письмовими завданням та викладенням матеріалів у відео, аудіо та текстових форматах, платформи сильно розрізняються у відкритості до викладачів та підходах до конструювання курсів і їх гейміфікації.

Деякі платформи, такі як Khan Academy та Duolingo, активно інтегрують гейміфікаційні елементи, які суттєво підвищують мотивацію користувачів за допомогою нагород, рівнів та інтерактивних завдань. Водночас, академічно орієнтовані сервіси, як-от Coursera та edX, зосереджуються на офіційній сертифікації та високоякісному контенті, що правда вносить обмеження для інструкторів які мають отримати дозвіл від платформи і пройти перевірку на якість створеного контенту, а у випадку платформи edX, взагалі, сторонні від університетів учасників інструктори не можуть мати доступ до створення курсів. Також через доволі жорстку академічність учасники платформ можуть відчувати дефіцит різних механік, які могли б зробити навчальний процес більш захоплюючим.

Udemy займає проміжне становище, пропонуючи широкий вибір практично орієнтованих курсів без значного акценту на гейміфікації. Попри це, Udemy відрізняється доступністю та гнучкістю, що приваблює студентів із різними

потребами. Курси створюються незалежними викладачами, які мають свободу в підході до їх структурування, однак це іноді призводить до різної якості матеріалів, що може вплинути на сприйняття платформи користувачами.

У підсумку, проведений аналіз дозволив визначити основні критерії, які є ключовими для побудови ефективної освітньої платформи. Зокрема, важливо забезпечити доступність для викладачів, що дозволить незалежним інструкторам легко створювати курси без зайвих бар'єрів, так як це розширить можливості дохідності платформи. При цьому платформа повинна містити зручний конструктор, що має покращити загальний досвід викладачів і позитивно відобразитись на якості курсів. Конструктор має забезпечити можливість створювати адаптивні курси з інтерактивними елементами, і доповнювати їх механіками досліджень.

Впровадження гейміфікаційних елементів, таких як значки, рівні й мітки прогресу, а з ними і можливість створення курсів довільної теми із гнучким форматом навчання, стане основою для підвищення зацікавленості як студентів, так і викладачів. Інтеграція системи нагород сприятиме підвищенню мотивації учнів, стимулюючи їх виконувати завдання та покращувати власні навички. Об’єднання цих критеріїв дозволить створити платформу, яка забезпечить захоплюючий, ефективний і зручний навчальний процес для всіх учасників.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Після проведення детального аналізу існуючих освітніх платформ, стало зрозуміло, що об’єднавши найкращі практики гейміфікації платформ та, розробивши унікальний конструктор, можна позбутись певних характерних недоліків таких як вузьконаправленість платформи, або строга академічність без ігрових елементів.

Наступним кроком є визначення вимог до прототипу освітньої платформи, чітке формулювання яких стане основою подальшого проєктування та розробки системи.

3.1 Загальні вимоги

Для ефективної роботи освітньої платформи вона має бути виконана з урахуванням стандартів та принципів розробки програмного забезпечення, гарантувати безпеку даних та надійність при функціонуванні.

а) SOLID принципи: допомагають забезпечити якісну об’єктно-орієнтовану архітектуру, яка легко підтримується та масштабується [11];

б) мікросервісна архітектура: використання мікросервісів довзолить розподілити функціонал на незалежні сервіси, кожен з яких відповідальний за певний функціонал. Сервіси спілкуються між собою через спеціальні API. Кожен мікросервіс має власну базу даних або сховище, що зменшує залежність між модулями. Для нашої системи на даному етапі доцільно було розділення за функціональним підходом:

1) застосунок для роботи з сутностями курсів, уроків та гейміфікаційних елементів;

2) застосунок для забезпечення захисту даних, контролю сесій та обробки даних користувачів.

3.2 Вимоги до надійності

Надійність є одним із ключових аспектів розробки будь-якої системи, особливо в умовах багатокористувацького середовища. Система повинна гарантувати безперебійну роботу, захищене зберігання даних і швидке відновлення після можливих збоїв. Наведемо вимоги до надійності платформи у таблиці 3.1.

Таблиця 3.1 — Вимоги до надійності

ID	Назва вимоги	Опис вимоги	Пріоритет
RR-1	Стійкість до збоїв	Платформа повинна залишатися доступною при відмові окремих компонентів (наприклад, через використання кластерної архітектури). Автоматичне резервне копіювання даних із можливістю відновлення за останні 24 години.	Високий
RR-2	Доступність	Система повинна бути доступною не менше ніж 99,9% часу на місяць. Повідомлення користувачів про планові технічні роботи за 24 години.	Високий
RR-3	Відновлення після збоїв	Автоматичне відновлення роботи при незначних збоях. План реагування на серйозні інциденти із	Високий

		середнім часом відновлення не більше 1 години.	
RR-4	Надійність зберігання даних	Гарантоване зберігання даних із перевіркою їхньої цілісності. Реплікація даних на декілька серверів для запобігання втраті інформації.	Високий

3.3 Функціональні вимоги

Функціональні вимоги визначають основні можливості та функції платформи. Вони описують, що саме система повинна виконувати для забезпечення потреб користувачів та досягнення поставлених цілей. У таблиці 3.2 наведені основні функціональні вимоги, які мають бути реалізовані в системі.

Таблиця 3.2 — Функціональні вимоги платформи

ID	Назва вимоги	Опис вимоги	Пріоритет
FR-1	Управління користувачами	Система повинна забезпечувати реєстрацію та аутентифікацію користувачів із використанням захищених протоколів (OAuth 2.0 або JWT).	Високий
FR-1.1	Реєстрація користувачів	Користувачі повинні мати можливість реєструватися через електронну пошту або соціальні мережі.	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
FR-1.2	Аутентифікація користувачів	Система повинна підтримувати безпечну аутентифікацію користувачів за допомогою OAuth 2.0 або JWT.	Високий
FR-1.3	Ролі та права доступу користувачів	Система повинна підтримувати різні ролі користувачів: студенти, викладачі, з відповідними правами доступу.	Високий
FR-1.4	Редагування профілю користувача	Користувачі повинні мати можливість редагувати свій профіль, включаючи зміну особистих даних та аватара.	Середній
FR-2	Управління курсами	Викладачі повинні мати можливість створювати, редагувати та видаляти курси, а також додавати різні типи навчальних матеріалів.	Високий
FR-2.1	Створення курсів	Викладачі повинні мати змогу створювати нові курси, та заповнювати інформацію про них.	Високий
FR-2.2	Редагування курсів	Викладачі повинні мати можливість редагувати існуючі курси, змінювати їхній зміст та структуру.	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
FR-2.3	Видалення курсів	Викладачі повинні мати змогу видаляти курси, які більше не потрібні.	Високий
FR-2.4	Додавання навчальних матеріалів	Система повинна підтримувати додавання різних типів навчальних матеріалів, таких як відео, текстові документи, інтерактивні завдання та тести.	Високий
FR-2.5	Конструювання адаптивних курсів	Викладачі повинні мати змогу створювати адаптивні курси з інтерактивними завданнями та системою нагород для мотивації студентів.	Високий
FR-3	Навчальний процес	Система повинна забезпечувати можливість проходження курсів студентами, тестування та перевірку знань, відстеження.	Високий
FR-3.1	Проходження курсів	Студенти повинні мати змогу проходити курси, переглядати матеріали та виконувати завдання.	Високий
FR-3.2	Тестування та перевірка знань	Система повинна підтримувати створення	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
		тестів та завдань з автоматичним оцінюванням відповідей студентів.	
FR-3.3	Система прогресу	Система повинна відстежувати досягнення користувача, надаючи нагороди та значки за виконання завдань та курсів.	Високий
FR-4	Адміністративні функції	Система повинна забезпечувати моніторинг активності користувачів для адміністрації платформи, управління ролями та правами доступу, а також надання аналітики та звітності викладачам.	Середній
FR-4.1	Моніторинг активності користувачів	Адміністратори повинні мати змогу переглядати сесії користувачів платформи. А викладачі активність студентів на власних курсах.	Середній
FR-4.2	Управління ролями та правами доступу	Адміністратори повинні мати змогу призначати та змінювати ролі користувачів, визначаючи	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
		їхні права доступу до різних функцій системи.	
FR-5	Керування контентом	Система повинна дозволяти викладачам та адміністраторам керувати навчальним контентом, включаючи додавання, редагування та видалення матеріалів.	Високий
FR-5.1	Додавання контенту	Викладачі повинні мати змогу додавати нові навчальні матеріали до курсів, включаючи відео, документи та інтерактивні завдання.	Високий
FR-5.2	Редагування контенту	Викладачі повинні мати можливість редагувати існуючі матеріали, вносячи зміни та оновлення за потребою.	Високий
FR-5.3	Видалення контенту	Викладачі та адміністратори повинні мати змогу видаляти непотрібні або застарілі матеріали з курсів.	Високий

3.4 Нефункціональні вимоги

Нефункціональні вимоги забезпечують якість платформи, визначаючи загальні характеристики системи та її поведінку під різними умовами експлуатації.

Вони визначають, як система повинна працювати, забезпечуючи її ефективність, безпеку, масштабованість та інші важливі аспекти. У таблиці 3.3 представлені основні нефункціональні вимоги до системи.

Таблиця 3.3 — Нефункціональні вимоги платформи

ID	Назва вимоги	Опис вимоги	Пріоритет
NFR-1	Продуктивність	Система повинна обробляти щонайменше 500 транзакцій на хвилину та забезпечувати середній час відгуку не більше ніж 2 секунди при пікових навантаженнях.	Високий
NFR-1.1	Час відповіді сервера	Середній час відповіді сервера не повинен перевищувати 200 мс для стандартних запитів при навантаженні до 300 одночасних користувачів. При піковому навантаженні до 1000 одночасних користувачів час відповіді не повинен перевищувати 1 секунди.	Високий
NFR-1.2	Обробка одночасних користувачів	Система повинна одночасно обробляти не менше 300 користувацьких запитів без зниження продуктивності.	Високий
NFR-2	Масштабованість	Система повинна бути спроектована таким чином, щоб підтримувати	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
		горизонтальне та вертикальне масштабування, забезпечуючи можливість розширення функціоналу та збільшення обсягу даних.	
NFR-2.1	Горизонтальне масштабування	Архітектура повинна підтримувати додавання нових серверів для розподілу навантаження без необхідності зупинки системи, забезпечуючи безперервну роботу.	Високий
NFR-2.2	Вертикальне масштабування	Можливість збільшення ресурсів існуючих серверів (CPU, RAM, дисковий простір) для покращення продуктивності без зміни архітектури системи.	Середній
NFR-3	Безпека	Система повинна забезпечувати високий рівень захисту даних та запобігання несанкціонованому доступу, атак та витоків інформації.	Високий
NFR-3.1	Захист від атак	Захист від SQL-ін'єкцій, XSS-атак, CSRF-атак та інших поширених вразливостей веб-додатків, використовуючи	Високий

ID	Назва вимоги	Опис вимоги	Пріоритет
		сучасні методи та інструменти безпеки.	
NFR-3.2	Шифрування даних	Всі чутливі дані повинні зберігатися у зашифрованому вигляді та передаватися через захищені протоколи (HTTPS, TLS).	Високий

Висновки до розділу 3

У даному розділі було досліджено вимоги до освітньої платформи. Їх було розподілено на функціональні і нефункціональні та додатково виділено вимоги до надійності системи, так як для систем такого типу особливо важливою є безперебійна діяльність.

Для забезпечення ефективної роботи платформи було визначено загальні вимоги, що включають дотримання стандартів та принципів розробки програмного забезпечення, таких як SOLID-принципи. Впровадження мікросервісної архітектури дозволяє розподілити функціонал на незалежні сервіси, що сприяє легкості підтримки та масштабування системи. Крім того, особлива увага приділяється безпеці даних та надійності функціонування системи.

Вимоги до надійності включають забезпечення високої доступності системи (не менше 99,9% часу на місяць), захисту даних через шифрування та реплікацію, а також швидкого відновлення після можливих збоїв. Ці вимоги гарантують безперебійну роботу платформи та захищеність інформації користувачів.

Функціональні вимоги визначають основні можливості та функції платформи, такі як управління користувачами, курсами, навчальний процес, адміністративні функції та інтеграції з зовнішніми сервісами. Деталізація цих вимог дозволяє забезпечити відповідність системи потребам різних категорій користувачів, а також гнучкість у розширенні її функціональності. Функціональні вимоги охоплюють такі аспекти, як реєстрація та аутентифікація користувачів,

створення та управління курсами, проведення тестувань, відстеження прогресу студентів та адміністрування системи.

Нефункціональні вимоги спрямовані на забезпечення якості платформи, визначаючи її загальні характеристики та поведінку в різних умовах експлуатації. Основні категорії нефункціональних вимог включають продуктивність, масштабованість, безпеку. Ці вимоги визначають, як система повинна працювати, забезпечуючи її ефективність, захищеність даних та доступність для користувачів.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

Вибір технологічного стеку є важливим етапом, проєктування систем, від нього, у більшій мірі, залежить базова продуктивність та масштабованість об’єкту розробки. Також технологічне забезпечення впливає на зручність розробки та подальшої підтримки програмних продуктів. Тому важливо встановити чітку відповідність між обраною технологією та задачею що вона дозволить вирішити.

При виборі технологій для розробки платформи було враховано наступні ключові критерії:

– масштабованість: можливість системи ефективно обробляти зростаючу кількість користувачів та даних;

– продуктивність: забезпечення високої швидкодії при обробці запитів;

– безпека: захист даних користувачів та запобігання несанкціонованому доступу;

– гнучкість та адаптивність: легкість у внесенні змін та розширенні функціональності;

– підтримка спільноти та доступність ресурсів: наявність великої спільноти розробників та якісної документації.

На основі цих критеріїв було обрано наступний технологічний стек:

– фронтенд: React, React Flow;

– бекенд: Java, Spring Framework, Lombok;

– збереження даних: MySQL;

– безпека та контроль доступу: KeyCloak;

– контейнеризація: Docker.

4.1 Фронтенд

4.1.1 React

Для розробки інтерфейсу користувача основним засобом розробки було обрано JavaScript бібліотеку React [12]. З основних переваг що вплинули на вибір можна виділити:

– компонентно-орієнтована архітектура: дозволяє створювати інтерфейс із незалежних компонентів, що сприяє повторному використанню коду та спрощує підтримку і масштабування додатку;

– висока продуктивність: завдяки використанню Virtual DOM, React забезпечує ефективне оновлення інтерфейсу без перезавантаження сторінки, що покращує користувацький досвід;

– широка підтримка та спільнота: React має велику спільноту розробників та багату екосистему додаткових бібліотек і інструментів, що полегшує вирішення проблем та прискорює розробку;

– сумісність із сучасними веб-технологіями: React легко інтегрується з іншими бібліотеками та фреймворками, що дозволяє створювати складні та інтерактивні інтерфейси.

4.1.2 ReactRouter

Як вже було сказано вище, React має велику кількість додаткових бібліотек що пришвидшують розробку, вирішуючи цілий розділ проблем. Однією з таких бібліотек є ReactRouter. Серед його переваг найбільший вплив на його вибір мали наступні показники:

– SPA-архітектура: React Router дозволяє створювати односторінкові додатки з динамічними маршрутами без перезавантаження сторінки, що покращує швидкодію та UX;

– гнучкість у налаштуванні маршрутів: підтримуються складні маршрути з параметрами, вкладеними маршрутами та умовною навігацією;

– широке використання в спільноті: React Router є де-факто стандартом для маршрутизації в React-додатках, що забезпечує надійність та підтримку у майбутньому і доступність коду широкому колу розробників.

4.1.3 ReactFlow

Перед нами стоїть задача реалізації конструктора курсів. Згідно вимог він має бути зручним, інтуїтивним для використання та повно-функціональним тобто таким, що здатний виконати усі завдання по створенню курсів без сторонніх застосунків.

Для цього було обрано бібліотеку ReactFlow, що має наступні позитивні для нас аспекти:

– інтуїтивно зрозумілий інтерфейс: React Flow дозволяє створювати інтерактивні діаграми та графи, що складаються з вузлів та зв’язків між ними. Це нам дозволяє інтерпретувати курс як певний набір уроків з’єднаних у послідовності або складні структури направленими зв’язками, що в свою чергу спрощує процес побудови структури курсів для викладачів;

– гнучкість та кастомізація: бібліотека дозволяє налаштовувати вигляд вузлів та зв'язків, що важливо для відображення специфічних елементів навчального процесу;

– зручний набір готових об’єктів: ресурс надає готовий набір атрибутів які можна використати при побудові власних об’єктів для конструктора курсів.

4.2 Бекенд

4.2.1 Java

Для проєктів що можуть характеризуватись як Enterprise найпопулярнішою мовою програмування є Java. Використання цієї мови обумовлено наступними причинами:

– надійність та масштабованість: Java є мовою для створення великих корпоративних додатків, що потребують високої надійності та можливості масштабування;

– велика екосистема: наявність великої кількості бібліотек, фреймворків та інструментів розробки полегшує створення складних систем;

– мультиплатформність: Java-додатки можуть працювати на різних операційних системах без зміни коду, що спрощує розгортання.

4.2.2 Spring Framework

Spring [13] є найпопулярнішим фреймворком для розробки веб та корпоративних додатків. Більшість банківських систем світу працюють саме на технічній зв’язці Java-Spring, що свідчить про високу надійність та безпечність такого рішення. Для нас можна виділити наступні найважливіші характеристиками для використання цієї технології на бекенді:

– модульність та гнучкість: Spring надає широкий спектр модулів які виконують різні завдання, що дозволяє нам вибирати лише необхідні, що робить розроблюваний застосунок легким та гнучким;

– підтримка залежності ін'єкції: спрощує управління залежностями між компонентами, підвищуючи модульність та тестованість коду;

– Spring Boot: забезпечує швидку конфігурацію та запуск додатків, що прискорює процес розробки;

– Spring Security: модуль що надає інструментарій для аутентифікації, авторизації та валідації користувачів та даних;

– підтримка мікросервісної архітектури: Spring Cloud спрощує розробку розподілених систем, що важливо для масштабованості платформи.

4.2.3 Lombok

Оскільки в будь-якій системі є код, що повторюється, наприклад, геттери і сеттери, що призводить до збільшення обсягів повторюваного і необов'язкового коду, який не додає нічого нового до бізнес-логіки застосунку, і додає часу розробці. Тому існує можливість використання бібліотек та інструментів, які допомагають підвищити продуктивність і уникнути цієї рутини. Саме це забезпечує бібліотека Lombok[14]. Вона надає наступні переваги:

– скорочення шаблонного коду: автоматична генерація гетерів, сетерів та інших стандартних методів по типу валідаторів зменшує кількість коду та підвищує читабельність;

– покращення продуктивності розробки: за рахунок антоцій, розробники можуть легко використовувати згенеровані методи. Також варто відмітити що бібліотека надає всі методи згідно стандартів найменувань у мові Java, та генерує їх на основі назв полів даних які розробник використовує. Це дозволяє зосередитися на бізнес-логіці замість написання рутинного коду;

– зниження ймовірності помилок: чим менше ручного коду, тим менша ймовірність помилок при написанні і неочікуваних результатів у роботі. Також це полегшує тестування оскільки згенеровані методи не вимагають перевірки.

4.3 Збереження даних

Головним питанням у збережені даних є вибір між реляційними та не реляційними СУБД. Нереляційні СУБД (NoSQL) більше підходять для зберігання неструктурованих або слабо структурованих даних, горизонтального масштабування та високих навантажень на запис. Оскільки основні вимоги платформи пов'язані зі структурованими даними та складними транзакціями, реляційна СУБД є більш відповідним вибором. Тому було прийнято рішення обрати саме реляційну систему управління базами даних, серед переваг яких є:

– структурованість даних: дані платформи мають чітку структуру та взаємозв'язки, що добре моделюються;

– підтримка транзакційності: реляційні СУБД забезпечують підтримку ACID-транзакцій, що важливо для збереження цілісності та консистентності даних, особливо при операціях реєстрації, оплати (яка у майбутньому стане важливим елементом платформи) тощо.

– складні запити та агрегування: SQL дозволяє виконувати складні запити, об'єднання та агрегування даних;

– широка підтримка та зрілість технології: реляційні СУБД є перевіреними рішеннями з багаторічною історією використання в корпоративних системах;

– сумісність з ORM-фреймворками: інтеграція з такими інструментами, як Hibernate та Spring Data JPA, спрощує роботу з базою даних на рівні об'єктів.

Для вибору конкретної реляційної СУБД були розглянуті дві найпопулярніші опції: MySQL та PostgreSQL [15-16]. Обидві СУБД є відкритими, потужними та широко використовуються. Для обґрунтованого вибору, порівняємо їх характеристики та внесемо дані порівняння до таблиці 4.1.

Таблиця 4.1 — Порівняння MySQL та PostgreSQL СУБД

Критерій	MySQL	PostgreSQL
Продуктивність	Висока швидкість читання та запису при простих операціях. Оптимізований для веб-додатків з високим навантаженням.	Відмінна продуктивність при складних запитах та операціях з великими обсягами даних. Краще справляється зі складними транзакціями.
Підтримка стандартів SQL	Часткова відповідність стандартам SQL, має власні розширення.	Висока відповідність стандартам SQL, підтримує багато

Критерій	MySQL	PostgreSQL
		розширень та додаткових функцій.
Функціональність	Набір базових функцій, достатній для більшості застосувань. Обмежена підтримка складних типів даних.	Широкий набір можливостей: розширені типи даних, повнотекстовий пошук, підтримка CTE, вбудовані процедури тощо.
Масштабованість	Підтримує горизонтальне та вертикальне масштабування, реплікацію. Легке налаштування кластерів.	Також підтримує масштабування, але налаштування може бути складнішим. Має розширені можливості для складних систем.
Простота використання	Проста установка та конфігурація. Менша крива навчання для адміністраторів та розробників.	Більш складна установка та конфігурація. Вимагає глибшого розуміння для оптимального використання.
Сумісність з інструментами	Широко підтримується багатьма інструментами та ORM-фреймворками, такими як Hibernate, Spring Data JPA.	Також добре підтримується, але може вимагати додаткових налаштувань для повної сумісності з деякими інструментами.
Спільнота та підтримка	Велика та активна спільнота, багато	Активна спільнота, але трохи менша за

Критерій	MySQL	PostgreSQL
	ресурсів, документації та готових рішень.	розміром. Більш спеціалізована підтримка.
Безпека	Надає основні механізми безпеки, такі як SSL, управління правами доступу.	Більш розширені можливості безпеки, включаючи RLS (Row-Level Security), політики доступу, шифрування даних на рівні СУБД.
Ліцензування	Відкрита ліцензія GPL для спільноти, комерційна підтримка доступна через Oracle.	Ліцензія PostgreSQL, яка є вільною та менш обмежувальною.
Реплікація та відмовостійкість	Підтримує майстер-слейв реплікацію, асинхронну реплікацію. Легке налаштування кластерів для високої доступності.	Підтримує як синхронну, так і асинхронну реплікацію. Розширені можливості для налаштування відмовостійких кластерів.

Враховуючи порівняння, було прийнято рішення обрати MySQL для нашої освітньої платформи з наступних причин:

– продуктивність для веб-додатків: MySQL оптимізований для швидкої обробки простих запитів, що є типовим для веб-додатків з високим навантаженням. Оскільки платформа очікує значну кількість одночасних користувачів, швидкість відповіді бази даних є критичною;

– широка підтримка та сумісність: MySQL має велику спільноту та широко підтримується різними інструментами та ORM-фреймворками, такими як Hibernate

та Spring Data JPA, які використовуються в нашому бекенді на Java. Це спрощує інтеграцію та забезпечує доступ до великої кількості ресурсів та готових рішень;

– масштабованість та реплікація: MySQL надає можливості більш легкого горизонтального та вертикального масштабування, а також налаштування реплікації. Це забезпечує можливість адаптувати систему до зростаючих потреб без значних змін в архітектурі.

4.4 Безпека та контроль доступу:

Для забезпечення безпеки та контролю доступу було вирішено не створювати власне рішення, а використати готове рішення – KeyCloak. Це відкрита платформа управління ідентифікацією та доступом. Keycloak підтримує сучасні протоколи, такі як OpenID Connect, OAuth 2.0 та SAML 2.0, що забезпечує безпечну аутентифікацію та авторизацію, також він легкий у налаштуванні та здатний повністю покрити усі потреби платформи в захисті даних та контролі сесій [17].

Основні переваги використання Keycloak для нашої системи:

– підтримка сучасних протоколів: Keycloak підтримує OpenID Connect, OAuth 2.0 та SAML 2.0, забезпечуючи безпечну аутентифікацію та авторизацію;

– Single Sign-On (SSO): дозволяє користувачам використовувати одні облікові дані для доступу до різних сервісів, підвищуючи зручність, що дозволить, наприклад викладачам бути студентами на курсах інших викладачів не створюючи окремого аккаунту;

– гнучке управління ролями та політиками доступу: забезпечує детальне налаштування прав доступу для різних груп користувачів, таких як студенти, викладачі та адміністратори;

– інтеграція з Spring Security: наявність готових адаптерів спрощує інтеграцію з бекендом, зменшуючи час розробки;

– можливість кастомізації: дозволяє налаштовувати інтерфейс аутентифікації відповідно до стилю та вимог нашої платформи;

– розширюваність та масштабованість: підтримує кластеризацію та може обслуговувати велику кількість користувачів, що робить його придатним для проєктів різного масштабу.

4.5 Контейнеризація

Важливою для легкої розробки і подальшого розгортання у хмарі є контейнеризація – оформлення програмних систем у контейнери з власним оптимальним оточенням та інструментарієм, що дозволяє розгортати та запускати платформу на комп'ютерах, незалежно від встановленого програмного забезпечення. Для цієї задачі був обраний Docker [18]. Він надає наступні переваги:

– портативність: контейнери Docker забезпечують однакове середовище на різних платформах, що усуває проблеми з сумісністю;

– швидке розгортання: контейнери дозволяють швидко розгортати та масштабувати додатки;

– ізоляція сервісів: забезпечує ізоляцію компонентів системи, що підвищує безпеку та стабільність.

Висновки до розділу 4

У цьому розділі було детально обґрунтовано вибір технологій та інструментів, використаних при розробці освітньої платформи. Ретельний аналіз потреб проєкту, технічних вимог та доступних технологічних рішень дозволив сформувати оптимальний для нас стек, який відповідає критеріям продуктивності, масштабованості, безпеки та гнучкості.

Фронтенд частина платформи була реалізована за допомогою React у поєднанні з бібліотеками, React Router та React Flow. Вибір React був зумовлений його компонентно-орієнтованою архітектурою, високою продуктивністю завдяки Virtual DOM та великою спільнотою розробників. React Router забезпечив ефективну маршрутизацію в односторінковому додатку, покращуючи

користувацький досвід без перезавантаження сторінки. React Flow став ключовим інструментом для реалізації конструктора курсів, надаючи викладачам інтуїтивно зрозумілу схему та інтерфейс для візуального моделювання структури навчальних матеріалів.

Бекенд платформи був побудований на основі Java з використанням Spring Framework та бібліотеки Lombok. Java була обрана за її надійність, масштабованість та високий рівень безпеки, що є критичним для систем, які обробляють дані користувачів. Spring Framework забезпечить гнучкість та модульність розробки, спростить управління залежностями та прискорить загальний процес створення додатку завдяки Spring Boot. Використання Spring Security надало потужні засоби для реалізації обробки аутентифікації та авторизації. Lombok дозволить зменшити кількість шаблонного коду що стосується сутностей та роботи з даними, підвищуючи продуктивність розробки та знижуючи ризик помилок.

При виборі системи управління базами даних було прийнято рішення використовувати реляційну СУБД з огляду на структурований характер даних платформи та необхідність підтримки транзакцій як гарантування надійності виконання запитів та безпечності даних при невдалих запитах. Провівши порівняння між MySQL та PostgreSQL, було обрано MySQL через її високу продуктивність у веб-додатках, простоту використання, широку підтримку та сумісність з існуючими інструментами розробки на Java. MySQL забезпечує достатню функціональність для потреб платформи та має можливості масштабування та реплікації, що важливо для подальшого зростання системи.

Безпека та управління доступом були реалізовані за допомогою відкритою програмної системи Keycloak. Вона забезпечує підтримку сучасних протоколів аутентифікації та авторизації, таких як OpenID Connect та OAuth 2.0. А також Keycloak легко інтегрується з Spring Security, що спрощує розробку та підтримку системи безпеки.

Використання Docker для контейнеризації додатків дозволяє забезпечити портативність та ізоляцію середовищ розробки, тестування та розгортання готового продукту, що сприяє стабільності та відтворюваності системи.

У підсумку, обраний технологічний стек дозволяє задовольнити функціональні та нефункціональні вимоги і вимоги надійності, визначені для платформи. Такий набір забезпечує:

– продуктивність: Швидка обробка запитів та ефективне управління ресурсами завдяки оптимізованим технологіям як на фронтенді, так і на бекенді;

– масштабованість: Можливість горизонтального та вертикального масштабування як додатку, так і бази даних, що дозволяє адаптувати систему до зростаючих навантажень;

– безпеку: Захист даних користувачів через використання перевірених інструментів та протоколів безпеки, а також належне управління доступом;

– гнучкість та розширюваність: Модульна архітектура та використання популярних фреймворків дозволяють легко додавати новий функціонал та адаптувати систему до змінних вимог;

– підтримка та спільнота: Використання технологій з великою спільнотою розробників забезпечує доступ до ресурсів, документації та готових рішень, що знижує ризики та прискорює розробку.

5 ПРОЄКТУВАННЯ ОСВІТНЬОЇ ПЛАТФОРМИ

5.1 Структура системи

При проєктуванні освітньої платформи ключовим завданням є розробка структури системи, яка забезпечить ефективну реалізацію всіх функціональних та нефункціональних вимог. Структура повинна бути гнучкою, масштабованою та надійною, що дозволить системі адаптуватися до зростаючих потреб та забезпечити високу якість обслуговування користувачів.

В своєму базовому функціональному варіанті платформа складається з 5-х частин:

– клієнтська частина (фронтенд): реалізована на базі бібліотеки React і призначена для забезпечення взаємодії з користувачами, відображення інтерфейсу та обробку подій на стороні клієнта. Що дозволить викладачам та студентам ефективно взаємодіяти як між собою так і з навчальними елементами;

– серверна частина (бекенд): розроблена на Java з використанням Spring Framework. Інкапсулює бізнес-логіку платформи. Забезпечує реалізацію обробки запитів користувача з клієнтської частини та взаємодіє з базою даних;

– система аутентифікації: розгорнута як окремий сервіс на базі KeyCloak. Забезпечує контроль сесій, аутентифікацію та авторизацію користувачів на платформі. Інтегрується з бекендом для валідації запитів. Зберігає дані про користувачів;

– база даних: відповідає за зберігання даних про елементи освітньої платформи;

– сховище файлів: зберігає різного набору матеріали: фотографії, відео тощо у доступному для бекенду місці.

Зобразимо структуру подану вище на рисунку 5.1.

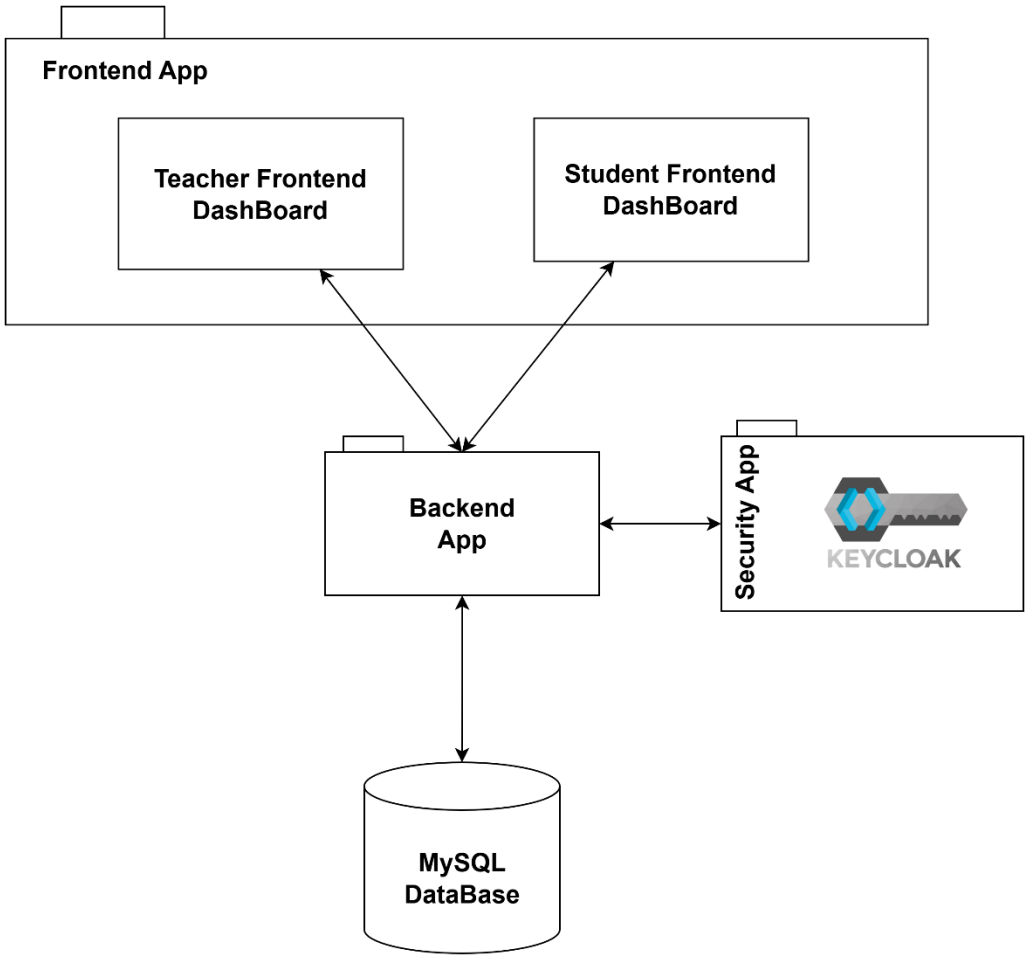

Рисунок 5.1 — Схема структурна основної частини освітньої платформи

Для забезпечення масштабованості та надійності освітньої платформи необхідно впровадити механізми балансування навантаження, регіональне масштабування та резервування основних компонентів системи. Запропонована архітектура побудована на основі найкращих практик дизайну систем, що дозволяє ефективно вирішувати задачі високого навантаження та доступності, що визначені у нефункціональних вимогах [19], та має наступну структуру:

а) фронтенд (React):

1) збирається як статичні файли (HTML, CSS, JavaScript);

2) розміщується в холодному сховищі (наприклад Amazon S3) з активацією статичного веб-хостингу;

3) інтегрується з мережею доставки контенту (CDN), такою як Amazon CloudFront, для швидкої доставки контенту користувачам по всьому світу;

4) масштабування фронтенду відбувається без необхідності керувати серверами та балансувальниками навантаження;

б) система аутентифікації (KeyCloak):

1) розгортається в кластері для забезпечення високої доступності;

2) перед KeyCloak встановлюється балансувальник навантаження для розподілу запитів між вузлами;

3) можна використовувати Kubernetes або інші оркестраційні системи для управління кластерами KeyCloak;

в) бекенд (Java, Spring Framework):

1) розгортається в контейнерах (наприклад, Docker) і керується за допомогою оркестраційних систем (наприклад, Kubernetes);

2) перед бекендом встановлюється балансувальник навантаження для розподілу запитів між інстанціями.

3) забезпечує горизонтальне масштабування за рахунок додавання нових інстанцій при зростанні навантаження;

г) база даних:

1) використовується масштабована база даних, така як Amazon RDS або аналогічна служба;

2) реплікація та шардінг застосовуються для підвищення продуктивності та надійності;

3) регулярне резервне копіювання даних для запобігання втрат;

д) Сховище файлів:

1) зберігається в розподіленому файловому сховищі, такому як Amazon S3;

2) доступ до файлів здійснюється через бекенд або безпосередньо з фронтенду з використанням захищених URL;

3) фронтенд можна розгорнути на декількох вузлах, де балансувальник навантаження буде перенаправляти запити користувачів на менш завантажені інстанції. Це забезпечить швидку реакцію інтерфейсу та стійкість до збоїв окремих вузлів.

Зобразимо масштабовану архітектуру на рисунку 5.2.

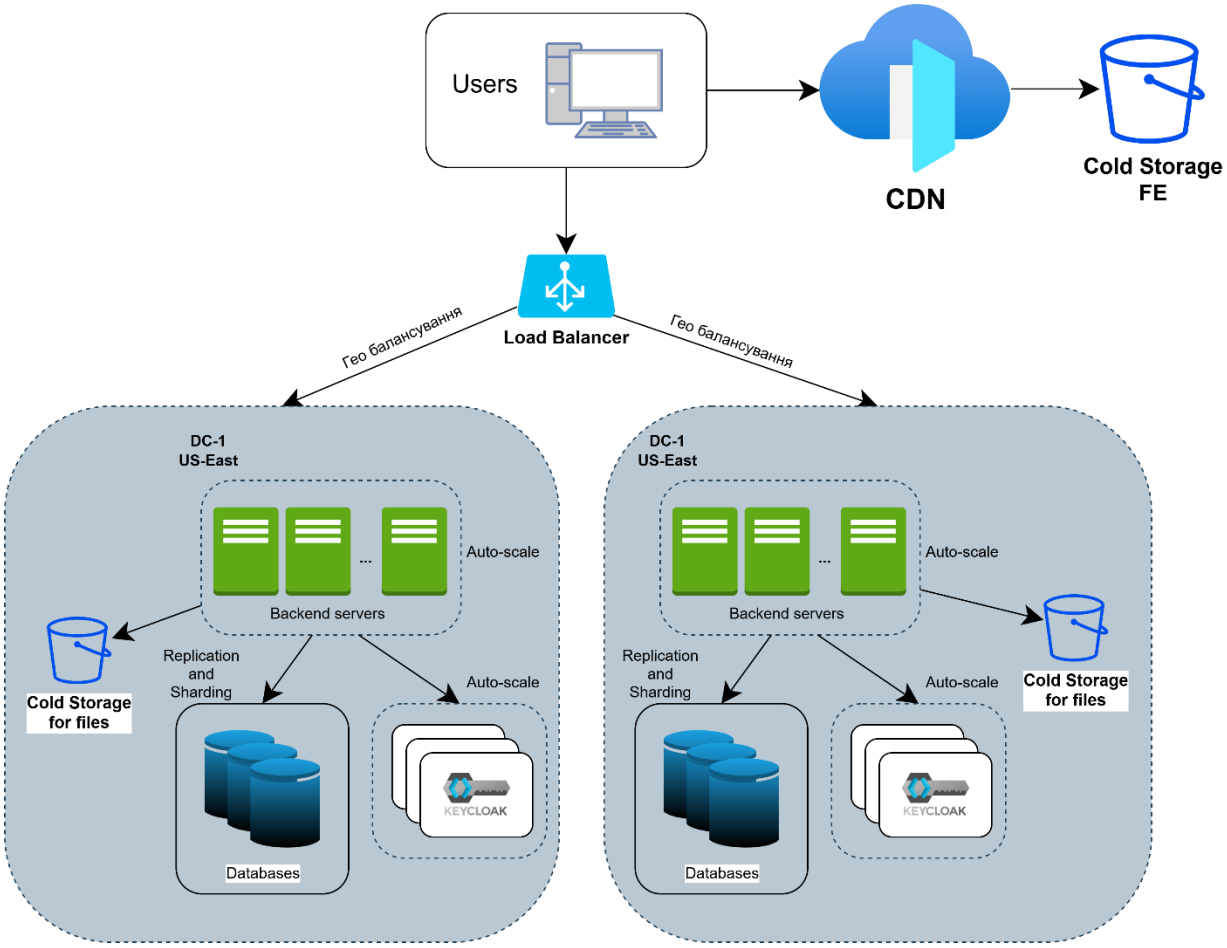

Рисунок 5.2 — Схема структурна масштабована освітньої платформи

5.2 Моделювання бізнес-процесів

Освітня платформа призначена для забезпечення ефективної взаємодії між двома основними акторами системи: викладачем та студентом з самою платформою. Управління їхніми робочими процесами є ключовим завданням платформи, що спрямована на оптимізацію навчального процесу та підвищення його якості.

5.2.1 Загальний бізнес-процес

Платформа має забезпечувати зручне середовище створення курсів для викладачів, для того щоб студенти мали можливість отримувати якісний матеріал з цікавим механізмом його засвоєння. Цей підрозділ буде зосереджено на описі основної ідеї побудови курсів та того як студенти будуть його виконувати.

З розділу 2, де ми досліджували аналоги, стало зрозуміло що більшість платформ не реалізовують гейміфікацію у спосіб, в якому студент потрапляє у середовище с певними правилами та винагородами [20]. А якщо гейміфікація реалізована, то сфера навчальних матеріалів певним полем знань, наприклад знання мови.

Підкресливши недоліки та переваги, було розроблено власний ігровий принцип, який дозволить обгорнути будь-який навчальний курс у нього.

5.2.1.1 Модель курсу викладача

Для викладача пропонується реалізовувати курси через спеціальний графічний конструктор. Він утворений комбінацією табличних форм та графового інтерфейсу, де кожний урок це вузол орієнтованого графу, який має вхідні та вихідні зв’язки та приносить студенту певну кількість балів за успішне проходження. Схематичний вигляд такого вузла поданий на рисунку 5.3.

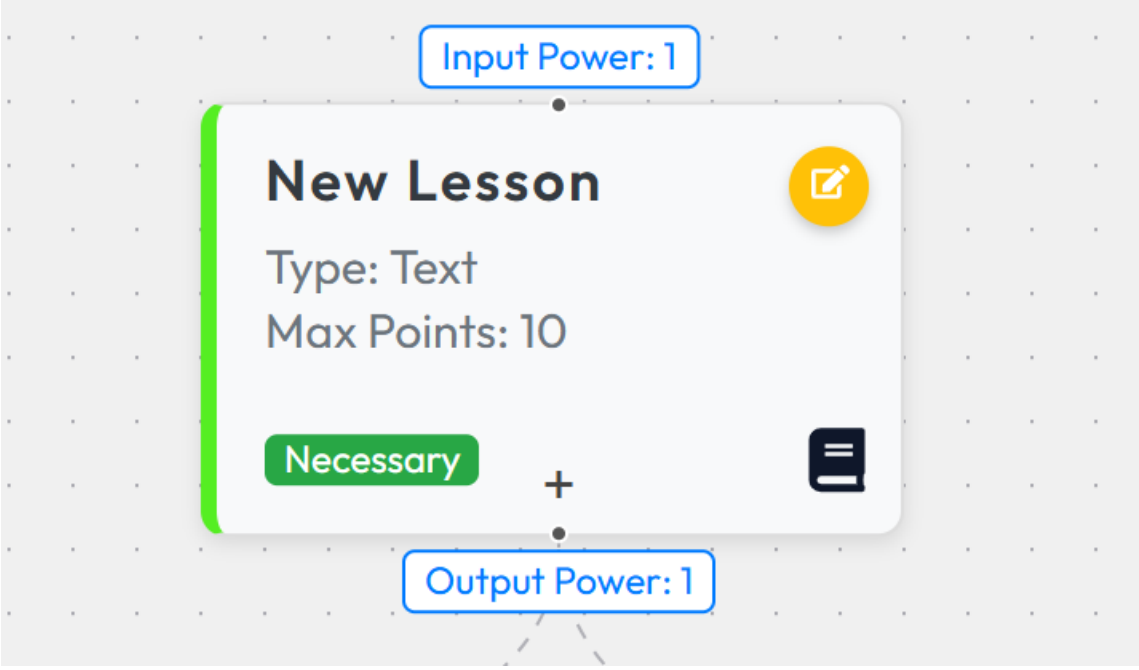

Рисунок 5.3 — Схема структурна уроку у конструкторі курсів

Для успішного завершення курсу більшість освітніх платформ пропонують схему успішності 80% — якщо студент набрав за курс 80% від загально-можливої оцінки, то він вважається завершеним. Ми також використаємо цю схему але вдосконалимо її двома правилами:

– обов'язковість: викладач має помітити певні уроки маркером обов'язковості, що означає що без завершення цього завдання курс не буде завершено незалежно від набраної кількості балів. У свою чергу інші завдання викладач може визначити як необов'язкові до виконання, але дати за них значну винагороду тим самим мотивуючи студента до поглиблення власних знань;

– вдосконалене правило 80%: пропонується визначати курс як завершений тільки, якщо виконані усі обов'язкові завдання і набрана кількість балів не менша за 80% від загально доступної кількості. Таким чином ми не обмежуємо студента у виборі завдань, але спонукаємо його до дослідження та вибору тих необов'язкових завдань, які, на його думку, будуть більш корисні для нього.

Додатково до цього вводиться можливість гнучкої структури. Ця ідея полягає у тому, що студент, розуміючи свій рівень знань, може виконати «модульне»

завдання посиленої складності і пропустити певний фрагмент курсу, що стосувався цього модульного тестування.

Реалізація гнучкої структури буде забезпечена через вагові коефіцієнти зв’язків між уроками. Ця ідея натхненна системами машинного навчання, а саме багатошаровим перцептроном, де кожний шар нейронів був сполучений з нейронами інших шарів зв’язками. Ці зв’язки мали вагові коефіцієнти, що впливали на вихідний сигнал і відповідно на результат. Для нашої платформи це буде мати наступний вигляд:

– вихідна потужність (величина впливу): кожен урок при завершені створює вихідний сигнал певної величини і передає усім суміжним з ним урокам;

– вхідна потужність (величина активації): якщо всі вхідні в урок зв'язки сумарно передають величину не меншу за задану величину активації урок стає доступним для проходження студентом. Таким чином цей механізм дозволяє будувати структури які можуть мати різну значимість, перекривати один одного, надавати опціональність.

Приклад пропуску певних вправ за допомогою ускладненого завдання з використанням принципу гнучкої структури поданий на рисунку 5.4.

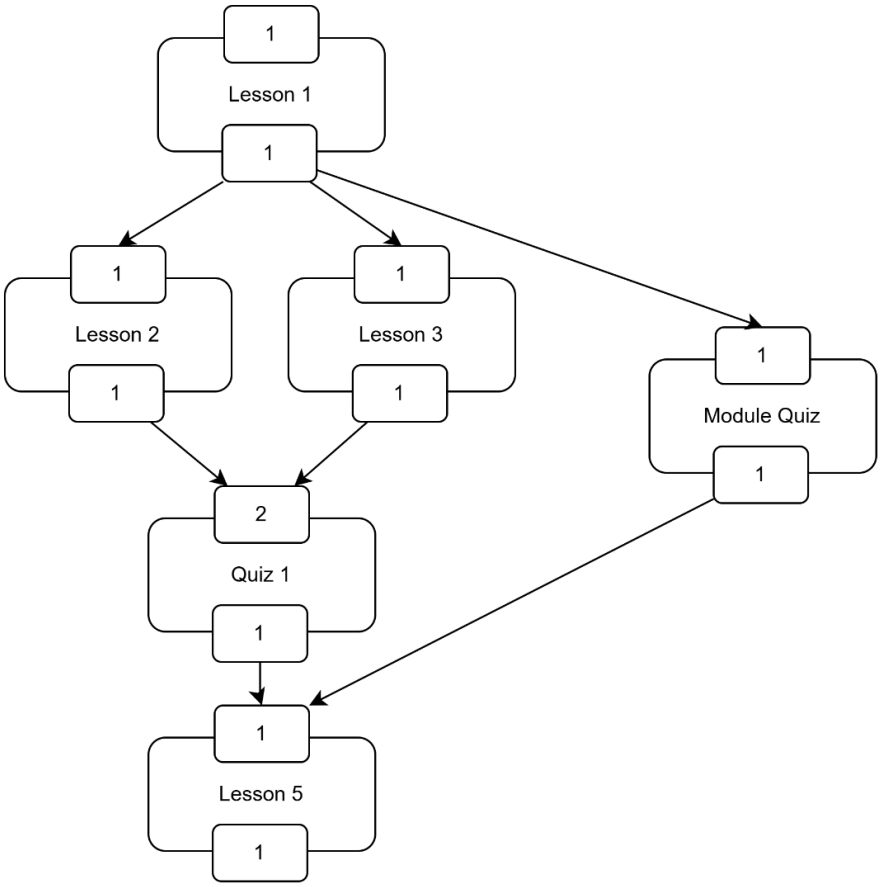

Рисунок 5.4 — Схема гнучкої структури курсу

Також викладачу надається можливість конструювання нагород. Цей процес був утворений з допомогою наступних критеріїв:

– кожен урок має нагороду у вигляді балів;

– платформа контролює час затрачений на спробу виконати завдання студентом;

– платформа має доступ до даних відвідування платформи студентом;

– платформа має доступ до усіх деталей процесу проходження курсу студентом: час, кількість днів, бал.

Згідно до практик застосованих на освітніх платформах та ігрових принципів зарахування досягнень [21], нагороди можуть даватись за наступні дії:

– першість у проходженні: якщо опублікований курс або завдання ще не були ніким пройдені — викладач може видавати першим виконавцям за це нагороду у вигляді бейджа (значка);

– серія відвідувань: якщо викладач фіксує що студент щоденно відвідує його курс, або платформа фіксує щоденне відвідування довгий період часу, то студент може отримати певну винагороду за стабільне та неперервне навчання;

– оцінка: викладач може встановити нагороду за відмінне проходження студентом курсу або уроку який має велику складність.

Відповідно до цього можна описати наступний алгоритм для побудови правил видачі нагород за сформовані досягнення, схема якого наведено нижче.

Крок 1. Вибір об’єкту, який перевіряється.

Крок 2. вибір критерію за яким відбувається перевірка об’єкту.

Крок 3. Вибір умови за якою критерій перевіряється.

Крок 4. Вибір величини з якою порівнюється критерій об’єкту.

5.2.1.2 Модель курсу студента

Для студента пропонується модифікація навчального процесу без урахування тих обмежень, що задані конструктором курсів, через введення системи добових спроб. Це принцип, за яким кожному студенту надається по дві спроби на день для виконання завдань. Таким чином, студент не зможе проходити завдання без підготовки, але водночас отримує можливість розробляти власну стратегію прогресу в навчанні.

У випадку, якщо студенту потрібно швидко просуватись у матеріалах курсу, він може скористатися гнучкою системою та спробувати виконувати інші завдання, якщо з якимось виникла складність.

Окрім того, обмеження кількості спроб підвищує якість навчання, оскільки не дозволяє студенту за один день мати надто багато спроб і проходити завдання методом підбору відповідей. Це спонукає до більш зваженого підходу та ретельнішої підготовки перед кожною спробою.

5.2.2 Функціональна модель діяльності викладача

Спираючись на діаграму діяльності викладача, можемо стверджувати що його основні функції включають:

– створення навчальних курсів: викладач створює окрему сутність курс, визначає його структуру та зміст, до цього списку можна віднести: назву курсу, опис, послідовності уроків, їх параметри;

– управління навчальним матеріалом: додавання лекцій, тестів та відео матеріалів, оформлення їх в уроки, визначення винагороди за них, налаштування взаємозв’язків між уроками;

– управління навчальним процесом: особа викладача може самостійно вносити зміни до курсу, у разі невідповідності очікуваним нормам, додаючи правки до вже опублікованого курсу;

– оцінювання: викладач має оцінювати студентів за виконання завдань або делегувати це на автоматичну систему платформи.

В об’єднанні ці функції створюють середовище, яке задовольняє викладача у своїх функціональних потребах.

Для кращого розуміння, зобразимо відношення між викладачем та варіантами його взаємодії із платформою, за допомогою діаграми прецедентів, яка наведена у додатку Г.

Дана діаграма використовується для опису послуг, які система пропонує викладачу, а точніше варіантів використання викладачем системи, за рахунок яких він може здійснювати власну діяльність. Таким чином, викладач має наступні варіанти використання:

– авторизація на платформі шляхом вводу особистого логіну та паролю у відповідні форму авторизації, для доступу до функціоналу;

– реєстрація на платформі для створення власного акаунту з доступом до функцій платформи;

– вихід із системи після завершення роботи;

– відкриття списку курсів для перегляду та вибору необхідного курсу;

– створення нового курсу для оформлення навчального контенту в єдину структуру;

– створення уроку в рамках визначеного курсу для додавання нового навчального матеріалу;

– відкриття існуючого курсу для перегляду або редагування його вмісту;

– редагування курсу або уроку з метою оновлення або вдосконалення навчальних матеріалів;

– видалення курсу або уроку у разі, якщо вони не відповідають цілям чи намірам викладача;

– відкриття панелі керування активним курсом для перегляду аудиту (статистики курсу по студентам) та можливості редагувати курс якщо аудит показує певні проблеми;

– перегляд таблиці статистики курсу, яка містить інформацію про прогрес студентів та їхню успішність;

– створення нагород для мотивування студентів при досягненні ними певних умов або результатів;

– відкриття існуючої нагороди для перегляду деталей або внесення змін;

– зміна нагороди, що дозволяє оновити її умови, опис або інші характеристики;

– видалення нагороди, якщо вона більше не актуальна або потрібна.

5.2.3 Функціональна модель діяльності студента

Для студента основними функціями, заснованими на його діяльності, можна вважати:

– запис на навчальні курси: студент може підписатись на доступні на платформі курси, прив’язавши власний обліковий запис;

– перегляд навчальних матеріалів: підписавшись на курс, студент отримує доступ до усіх матеріалів, створених викладачем для цього курсу. Цей набір

включає: лекції, тести та відео матеріали, структура курсу на яку він може орієнтуватись;

– виконання завдань: студент може використовувати матеріал у навчальних цілях, та виконувати завдання таким чином проходячи курс до моменту його завершення;

– відстеження прогресу: студент може бачити власні оцінки та завершені завдання.

Зобразимо відношення між студентом та варіантами його взаємодії із платформою, за допомогою діаграми прецедентів, яка наведена у додатку Д.

Ця діаграма ілюструє можливості системи, доступні студенту для використання в процесі навчання. Вона демонструє, як студент може взаємодіяти з платформою, виконувати навчальні завдання, слідкувати за своїм прогресом та отримувати винагороди за досягнення. Таким чином, користувач має наступні варіанти використання:

– авторизація для доступу до теки курсів шляхом вводу особистого логіну та паролю у відповідні форму авторизації;

– реєстрація аккаунту студента на платформі з доступом до функціоналу відповідного студентській ролі;

– вихід із системи після завершення роботи;

– пошук доступних курсів для перегляду пропозицій системи;

– перегляд опису курсу для отримання детальної інформації про курс перед підпискою;

– підписатися на курс для доступу до його матеріалів і завдань;

– відкриття курсу через студентську версію вигляду курсу (конструктор курсів без інструментарію та прав до змін, лише графічний інтерфейс) для початку навчання роботи із завданнями;

– перегляд завдань курсу для розуміння вимог і умов їх виконання;

– спроба виконання завдання із можливістю досягнення успішного або неуспішного результату;

– отримання винагороди за успішне виконання завдання;

– перегляд прогресу у курсі, щоб оцінити свій поточний статус і досягнення;

– перегляд дошки винагород, де студент може переглядати всі свої нагороди за виконані курси та завдання;

– перегляд детальної інформації про конкретну винагороду для аналізу її умов та отримання більш детальних даних.

5.3 Модель бази даних

Для збереження даних було використано реляційну систему управління базами даних MySQL. До її структурних одиниць відносять бази даних, таблиці, поля та записи, що дозволяє ефективно організувати та зберігати великі обсяги інформації, необхідні для функціонування платформи та забезпечення надійності зберігання даних.

5.3.1 Сутності платформи

Розглянемо сутності, які є в нашій освітній платформі:

– User — користувач платформи, що має роль студента або викладача (незалежна сутність);

– Course — курс створений викладачем (незалежна сутність, але з можливістю видалення за бажанням викладача, якщо викладач покидає платформу);

– Lesson — урок (навчальна картка), містить деталі потрібні для графічної складової конструктора (залежна сутність);

– Lesson Content — наповнення уроку, яке залежить від типу картки: тест, текстова або відео лекція (залежна сутність);

– Edge — зв'язок між уроками, направлений від одного уроку до іншого, що визначає послідовність навчання (залежна сутність);

– Achievement — досягнення студента, які він отримує за виконання певних умов або завдань (залежна сутність).

Зобразимо сутності та зв’язки між ними за допомогою ER-діаграми сутностей на рисунку 5.5.

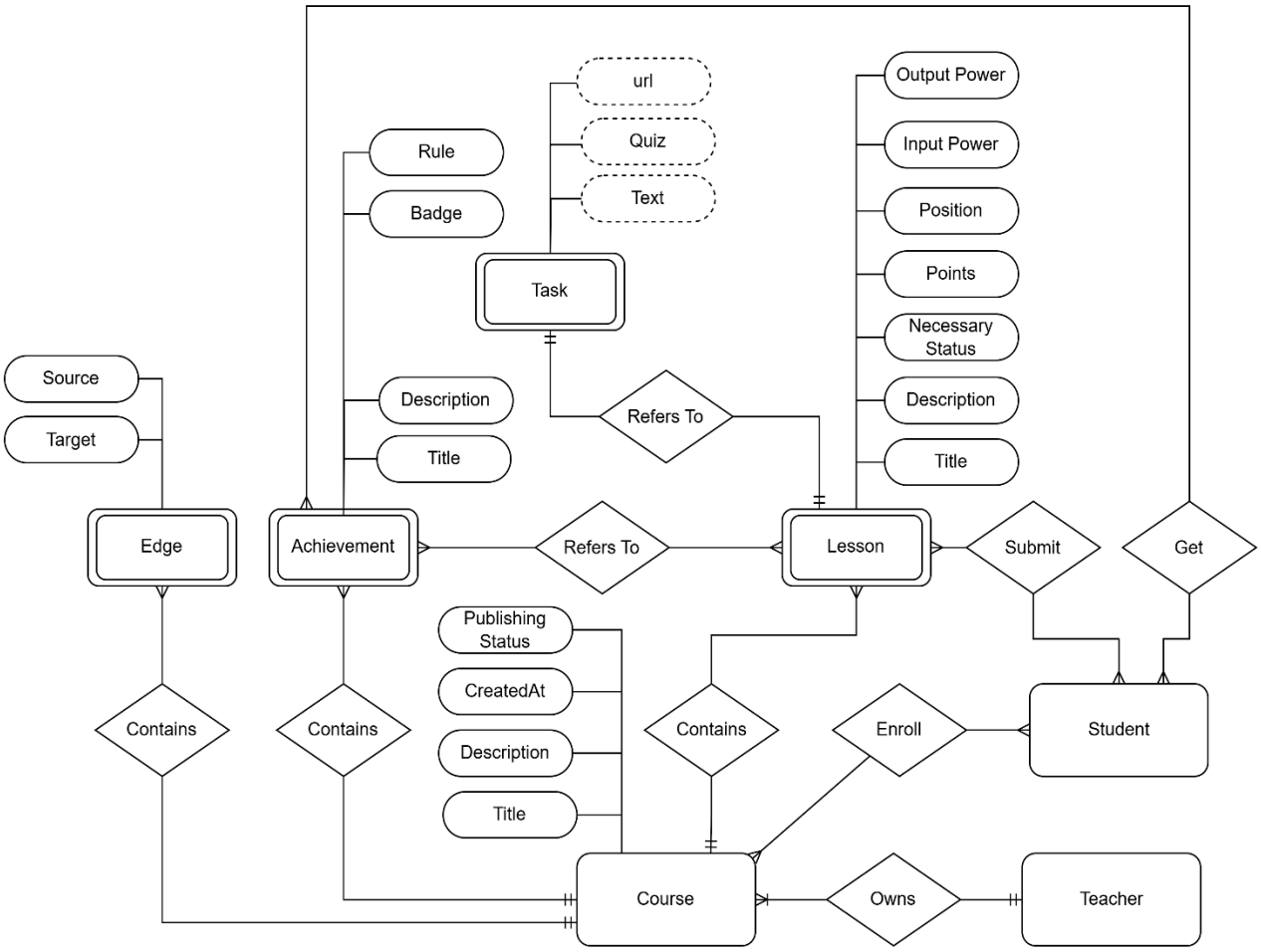

Рисунок 5.5 — ER діаграма сутностей

Додатково побудуємо схему бази даних, що дасть нам більш детальне розуміння зв’язків між даними, на рисунку 5.6.

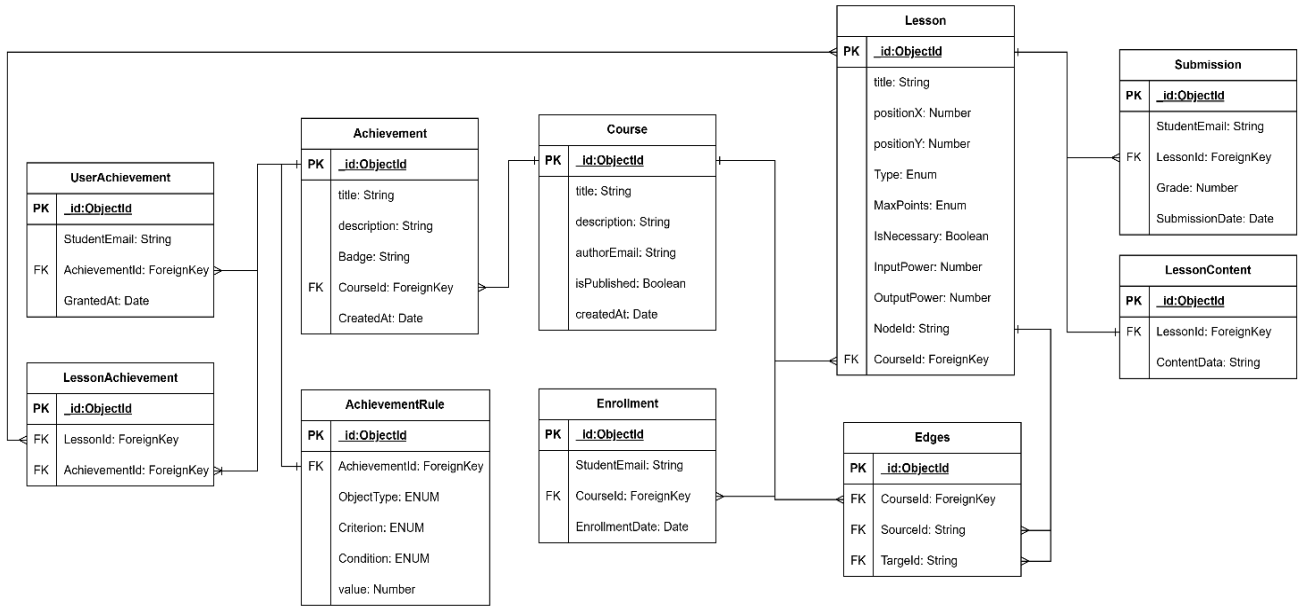

Рисунок 5.6 — Схема бази даних

Деталізуємо структуру через наведення таблиць сутностей бази даних [22], представляючи їх атрибути та відповідні значення за допомогою таблиць 5.1-5.8

Таблиця 5.1 — Структура таблиці User з відповідними типами даних

Таблиця	Атрибут	Значення	Тип даних
User	_id	унікальний ідентифікатор користувача	String, PRIMARY KEY — унікальний ідентифікатор запису
	email	електронна пошта користувача	String, рядкове значення
	hash_pass	хеш персонального пароль користувача	Hash String, захешоване рядкове значення
	role	роль користувача	ENUM [TEACHER, STUDENT]
	firstName	ім’я користувача	String, рядкове значення

Таблиця	Атрибут	Значення	Тип даних
	lastName	прізвище користувача	String, рядкове значення
	avatar	посилання на зображення профілю користувача	String, рядкове значення
	registration_date	Дата реєсттрації користувача	DATETIME формат дати та часу за UTC форматом

Таблиця 5.2 — Таблиця структури таблиці Sessions з відповідними даними

Таблиця	Атрибут	Значення	Тип даних
Sessions	_id	унікальний ідентифікатор запису про сесію користувача	String, PRIMARY KEY — унікальний ідентифікатор запису
	user_id	ідентифікатор користувача, сесія якого міститься в записі	String, FOREIGN KEY — посилання на User._id
	login_date	дата та час входу користувача на платформу	DATETIME формат дати та часу за UTC форматом
	logout_date	дата та час виходу користувача з платформи	DATETIME формат дати та часу за UTC форматом

Важливо, що вище подані таблиці не відносяться до СУБД MySQL, а зберігаються у власній СУБД системи KeyCloak, що відповідає нашому розподілу на окремі сервіси.

Таблиця 5.3 — Структура таблиці Course з відповідними типами даних

Таблиця	Атрибут	Значення	Тип даних
Course	_id	унікальний ідентифікатор курсу	String, PRIMARY KEY — унікальний ідентифікатор запису
	title	назва курсу	String, рядкове значення
	description	детальний опис курсу	String, рядкове значення
	author_email	електронна пошта автора (author_id не використовується, так як користувачі зберігаються в СУБД KeyCloak)	String, рядкове значення
	createdAt	дата та час створення курсу	DATETIME, часовий формат представлення дати
	is_published	статус курсу, що відмічає чи курс опубліковано чи ще знаходиться на стадії проектування викладачем	Boolean

Таблиця 5.4 — Таблиця структури таблиці Lesson з відповідними даними

Таблиця	Атрибут	Значення	Тип даних
Lesson	_id	унікальний ідентифікатор уроку	String, PRIMARY KEY — унікальний ідентифікатор запису
	course_id	ідентифікатор курсу, до якого належить урок	String, FOREIGN KEY — посилання на Course._id
	node_id	ідентифікатор картки уроку на фронтенді для бібліотеки React Flow	String, рядкове значення
	title	назва уроку	String, рядкове значення
	position_x	координата X розташування уроку в конструкторі	INT, числове значення
	position_y	координата Y розташування уроку в конструкторі	INT, числове значення
	type	тип уроку (наприклад, lecture, test, video)	ENUM
	maxPoints	максимально доступна кількість балів за завдання	INT, числове значення
	isNecessary	визначає чи обов’язковий урок	BOOLEAN
	inputPower	величниа активації	INT, числове значення
	outputPower	величина впливу	INT, числове значення

Таблиця 5.5 — Структура таблиці LessonContent з відповідними типами даних

Колекція	Атрибут	Значення	Тип даних
Lesson Content	_id	унікальний ідентифікатор контенту уроку	String, PRIMARY KEY — унікальний ідентифікатор запису
	lesson_id	ідентифікатор уроку, до якого належить контент	String, FOREIGN KEY — посилання на Lesson._id
	content_data	посилання на папку з матеріалами уроку	String, рядкове значення

Таблиця 5.6 — Структура таблиці Edges з відповідними типами даних

Колекція	Атрибут	Значення	Тип даних
Edges	_id	унікальний ідентифікатор зв’язку	String, PRIMARY KEY — унікальний ідентифікатор запису
	course_id	ідентифікатор курсу, до якого належить урок	String, FOREIGN KEY — посилання на Course._id
	source_id	ідентифікатор вихідного уроку	String, FOREIGN KEY — посилання на Course.node_id
	target_id	ідентифікатор вхідного уроку	String, FOREIGN KEY — посилання на Course.node_id

Таблиця 5.7 — Структура таблиці Enrollment з відповідними типами даних

Колекція	Атрибут	Значення	Тип даних
Enrollment	_id	унікальний ідентифікатор запису про підписку на курс	String, PRIMARY KEY — унікальний ідентифікатор запису
	user_email	електронна пошта студента	String, рядкове значення
	course_id	ідентифікатор курсу, до якого належить урок	String, FOREIGN KEY — посилання на Course._id
	enrollment_date	дата та час зарахування на курс	DATETIME, часовий формат представлення дати

Таблиця 5.8 — Структура таблиці Submission з відповідними типами даних

Колекція	Атрибут	Значення	Тип даних
Submission	_id	унікальний ідентифікатор запису про виконання завдання	String, PRIMARY KEY — унікальний ідентифікатор запису
	user_email	електронна пошта студента (оскільки користувачі зберігаються в Keycloak)	String, рядкове значення
	lesson_id	ідентифікатор уроку, в якому знаходиться завддання	String, FOREIGN KEY — посилання на Lesson._id

Колекція	Атрибут	Значення	Тип даних
	grade	оцінка за завдання	DECIMAL, числове значення
	submission_date	дата та час подання завдання	DATETIME, формат дати та часу за UTC

Таблиця 5.9 — Структура таблиці Achievement

Колекція	Атрибут	Значення	Тип даних
Achievement	_id	унікальний ідентифікатор бейджу (нагороди)	String, PRIMARY KEY — унікальний ідентифікатор запису
	name	назва нагороди	String, рядкове значення
	description	опис нагороди	String, рядкове значення
	badge	оцінка за завдання	String, рядкове значення
	created_at	дата та час подання завдання	DATETIME, формат дати та часу за UTC
	course_id	ідентифікатор курсу, до якого належить досягнення	String, FOREIGN KEY — посилання на Course._id

Таблиця 5.10 — Структура таблиці AchievementRule

Колекція	Атрибут	Значення	Тип даних
Achievement Rule	_id	унікальний ідентифікатор контенту уроку	String, PRIMARY KEY — унікальний ідентифікатор запису

	achievement_id	ідентифікатор, що зв’язує правило з відповідним бейджем	String, FOREIGN KEY — посилання на Achievement._id
	object_type	тип об'єкта (курс, урок)	ENUM [Lesson, Course]
	criterion	критерій правила наприклад “TopByFirst” (перший з Value проходження)	ENUM [“TopByFirst”, “Grade”, “Time”, “Attempts”]
	condition	умова правила	ENUM [“>”; “<”; “=”; “≥”; “≤”;]
	value	величина (наприклад, число днів, оцінка)	DECIMAL, числове значення

Таблиця 5.11 — Структура таблиці зв’язків UserAchievement

Колекція	Атрибут	Значення	Тип даних
User Achievement	_id	унікальний ідентифікатор контенту уроку	String, PRIMARY KEY — унікальний ідентифікатор запису
	user_email	електронна пошта студента (оскільки користувачі зберігаються в Keycloak)	String, рядкове значення

Колекція	Атрибут	Значення	Тип даних
	achievement_id	ідентифікатор уроку, в якому знаходиться завддання	String, FOREIGN KEY — посилання на Achievement._id
	granted_at	дата та час подання завдання	DATETIME, формат дати та часу за UTC

Таблиця 5.12 — Структура таблиці зв’язків LessonAchievement

Колекція	Атрибут	Значення	Тип даних
Lesson Achievement	_id	унікальний ідентифікатор контенту уроку	String, PRIMARY KEY — унікальний ідентифікатор запису
	lesson_id	ідентифікатор уроку, в якому знаходиться досягнення	String, FOREIGN KEY — посилання на Lesson._id
	achievement_id	ідентифікатор уроку, в якому знаходиться завддання	String, FOREIGN KEY — посилання на Achievement._id

5.4 Архітектура програмного забезпечення

Визначення архітектури програмного забезпечення є фундаментальною складовою створення інформаційної системи. У цьому розділі буде детально розглянуто об'єктно-орієнтовану модель освітньої платформи, включаючи діаграму класів, опис основних класів та їх взаємодію, а також використані патерни проєктування.

5.4.1 Патерни проєктування

Для платформи, враховуючи технологічний стек, було застосовано низку патернів проєктування:

5.4.1.1 Singleton

Патерн Singleton гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до нього.

У системі патерн Singleton використовується для забезпечення єдиного екземпляра деяких сервісів, які повинні бути унікальними протягом усього часу роботи додатку. Наприклад:

– підключення до бази даних: Створення єдиного екземпляра класу, який відповідає за взаємодію з базою даних, щоб уникнути перевантаження системи зайвими підключеннями;

– конфігураційні налаштування: Зберігання налаштувань програми в єдиному екземплярі класу.

5.4.1.2 DAO (Data Access Object)

Патерн DAO абстрагує доступ до джерела даних, надаючи інтерфейс для виконання CRUD-операцій без необхідності знати деталі реалізації бази даних.

Використовується для ізоляції рівня бізнес-логіки від деталей взаємодії з базою даних. Наприклад, для кожної сутності (User, Course, Lesson) створюється відповідний DAO-клас.

5.4.1.3 DTO (Data Transfer Object)

Патерн проєктування, що використовується для передачі даних між різними рівнями системи або між її компонентами. Об'єкти DTO інкапсулюють дані та, як

правило, не містять бізнес-логіки. Вони дозволяють зменшити кількість викликів між компонентами системи, передаючи необхідну інформацію у вигляді цілісного об’єкта.

Наприклад, патерн використовується для отримання повного курсу з усіма уроками та нагородами, що є різними об'єктами але компонуються разом у один для легшої передачі між компонентами (контролерами, сервісами тощо)

5.4.1.4 MVC (Model-View-Controller):

MVC є архітектурним патерном, який розділяє додаток на три основні компоненти:

– Model: Модель даних, бізнес-логіка.

– View: Представлення, інтерфейс користувача.

– Controller: Контролер, який обробляє запити та координує роботу між моделлю та представленням.

У системі, заснованій на Spring Framework, патерн MVC є фундаментальним. Контролери обробляють HTTP-запити, взаємодіють з моделями та повертають представлення або дані у форматі JSON для клієнтського додатку.

5.4.2 Діаграма класів

Діаграма класів представляє статичну структуру системи, показуючи класи, їх атрибути, методи та взаємозв'язки між ними. Для освітньої платформи можна виділити наступні класи:

5.4.2.1 Бекенд

Бекенд архітектура складається з наступних класів:

а) класи сутностей (DAO-класи);

1) Course;
2) Lesson;
3) LessonContent;
4) Edge;
5) Enrollment;
6) Submission;
7) Achievement;

б) класи сервісів;

1) CourseService;
2) LessonService;
3) StorageService;
4) EdgeService;
5) EnrollmentService;
6) SubmissionService;
7) AchievementService;

в) класи контролерів;

1) CourseController;
2) LessonController;
3) EdgeController;
4) EnrollmentController;

5) SubmissionController;

6) AchievementController;

г) DTO класи;

1) CourseDTO;

2) LessonDTO;

3) EdgeDTO;

4) EmpttyLessonDTO;

5) LessonDataDTO;

6) PositionDTO;

7) QuizDTO;

8) SaveCourseDTO;

9) SaveCourseResponseDTO;

д) класи помилок;

1) NotFoundException;

2) BadRequestException;

3) UnauthorizedException;

4) ForbiddenException;

5.4.2.2 Фронтенд

Для фронтенд частини проєкту визначені наступні класи:

а) клас роутера;

1) RouterProvider;

б) класи помилок;

1) NotFoundError;

2) BadRequestError;

3) UnauthenticatedErrror;

4) UnauthorizedError;

Детальний опис основних класів освітньої платформи наведено у таблиці 5.12

Таблиця 5.12 — Опис основних класів програмного застосунку

Клас	Опис
Course	Клас, що представляє курс в освітній платформі. Містить атрибути, такі як назва, опис, автор (викладач), статус публікації та дату створення.
Lesson	Клас, що представляє урок у курсі. Містить інформацію про урок, включаючи назву, тип уроку (лекція, тест, завдання), позицію в структурі курсу, вміст уроку.
Edge	Клас, що представляє зв'язок між уроками в курсі. Визначає послідовність переходу між уроками, дозволяючи будувати навчальний шлях студента. Містить інформацію про початковий та кінцевий уроки зв'язку.
Achievement	Клас, що представляє досягнення, які можуть отримувати користувачі. Містить назву досягнення, опис, умови отримання, посилання на зображення.
Enrollment	Клас, що представляє запис про зарахування студента на курс. Містить інформацію про студента, курс, дату зарахування та статус навчання. Відповідає за управління зарахуваннями та відстеження прогресу студентів у курсах.
Submission	Клас, що представляє запис про виконання завдання. Містить інформацію про завдання, дату подання, оцінку та статус.
UserAchievement	Клас, що представляє зв'язок між користувачем та отриманими ним досягненнями. Містить інформацію про користувача, досягнення та дату отримання. Відповідає за відстеження та відображення досягнень, отриманих користувачами.

Клас	Опис
CourseController	Клас контролера, який обробляє HTTP-запити, пов'язані з курсами. Реалізує методи для створення, редагування, отримання та видалення курсів включно з уроками, якщо це стосується усіх уроків курсу. Забезпечує взаємодію між клієнтським додатком та бізнес-логікою курсів.
LessonController	Клас контролера, що обробляє HTTP-запити, пов'язані з уроками. Реалізує методи для управління уроками в курсах: додавання, редагування, видалення та отримання вмісту уроків. Забезпечує взаємодію між клієнтським додатком та бізнес-логікою уроків.
EdgeController	Клас контролера, який відповідає за управління зв'язками між уроками. Відповідає за отримання інформації про існування зв’язків між певними уроками.
EnrollmentController	Клас контролера, що керує зарахуванням студентів на курси. Обробляє запити на зарахування, відписку від курсу, перегляд списку студентів, записаних на курс, та курсів, на які записаний студент. Забезпечує контроль доступу та перевірку прав користувача.
SubmissionController	Клас контролера, що керує статусом виконання завдань. Обробляє запити на подання завдань студентами.
AchievementController	Клас контролера, що керує досягненнями користувачів. Обробляє запити на отримання списку досягнень, додавання нових досягнень в систему, призначення досягнень користувачам. Забезпечує відображення досягнень та умов їх отримання.
CourseService	Клас сервісу, що містить бізнес-логіку для управління курсами. Відповідає за операції створення, редагування,

Клас	Опис
	публікації та видалення курсів, збереження курсів разом з уроками, взаємодію з базою даних через DAO.
LessonService	Клас сервісу, що містить бізнес-логіку для управління уроками. Відповідає за операції додавання, редагування, видалення уроку, управління вмістом уроку, взаємодію з базою даних через DAO.
EdgeService	Клас сервісу, що реалізує бізнес-логіку для управління зв'язками між уроками. Відповідає за оновлення зв’язків у курсі.
EnrollmentService	Клас сервісу, що містить бізнес-логіку зарахування студентів на курси. Відповідає за перевірку можливості зарахування, управління записами про зарахування, взаємодію з базою даних.
SubmissionService	Клас сервісу, що містить бізнес-логіку для управління завданнями. Відповідає за операції подання завдань студентами.
AchievementService	Клас сервісу, що реалізує бізнес-логіку для управління досягненнями. Відповідає за визначення та перевірку умов отримання досягнень, призначення досягнень користувачам, відображення статистики досягнень, взаємодію з базою даних.
RouterProvider	Клас компонент бібліотеки ReactRouter за допомогою якого реалізовано переходи між сторінками та перевірки доступу.
PrivateRoutes	Власний клас за допомогою якого реалізовано перевірку доступу до платформи за ролями.

Структурна архітектура бекенду представлена за допомогою діаграми класів, яка подана у додатку Е.

5.4.3 Діаграма послідовності

Опишемо послідовності дій у системі, проаналізувавши як об'єкти взаємодіють між собою в рамках визначених сценаріїв, відображаючи порядок дій компонентів та користувачів і обмін повідомленнями між ними. Враховуючи інформацію отриману з діаграм використання, маємо що в системі існують наступні основні процеси:

– реєстрація та авторизація: для початку роботи на платформі будь-якому користувачу потрібно авторизуватись у системі. Для цього користувач натискає кнопку “Sign-in”, або “Sign-up” якщо не має персонального акаунту. Система перенаправляє його у форму системи KeyCloak, де користувач вводить дані і отримує доступ у випадку коректного введення даних;

Даний процес подано на діаграмах послідовностей реєстрації та авторизації на рисунках 5.7-5.8.

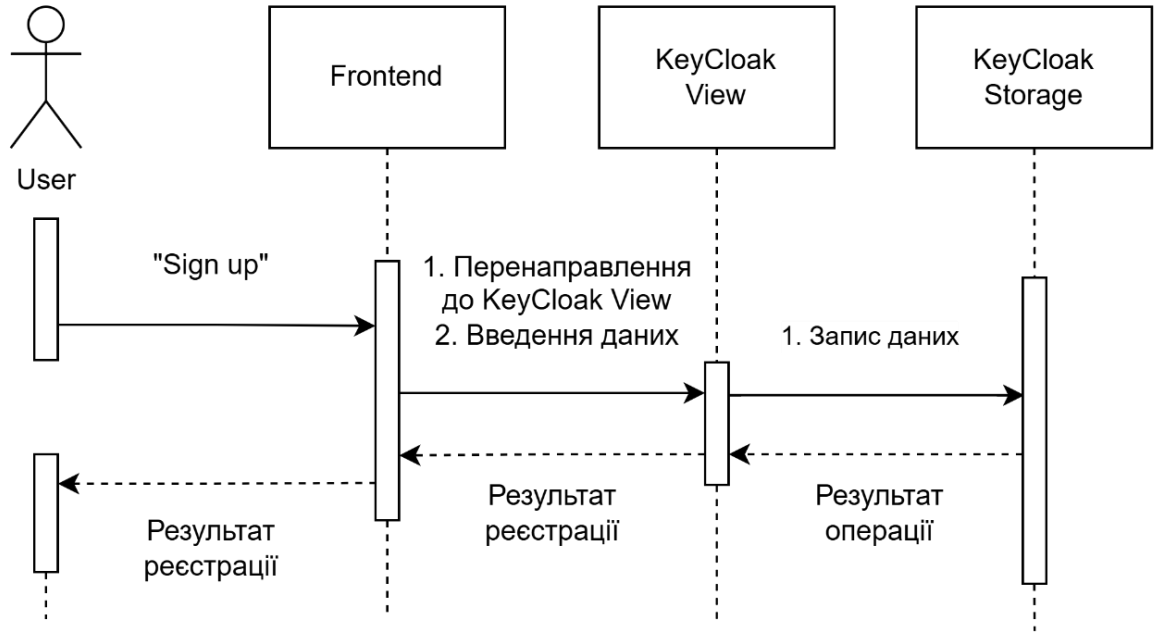

Рисунок 5.7 — Послідовність процесів під час реєстрації користувача на платформі

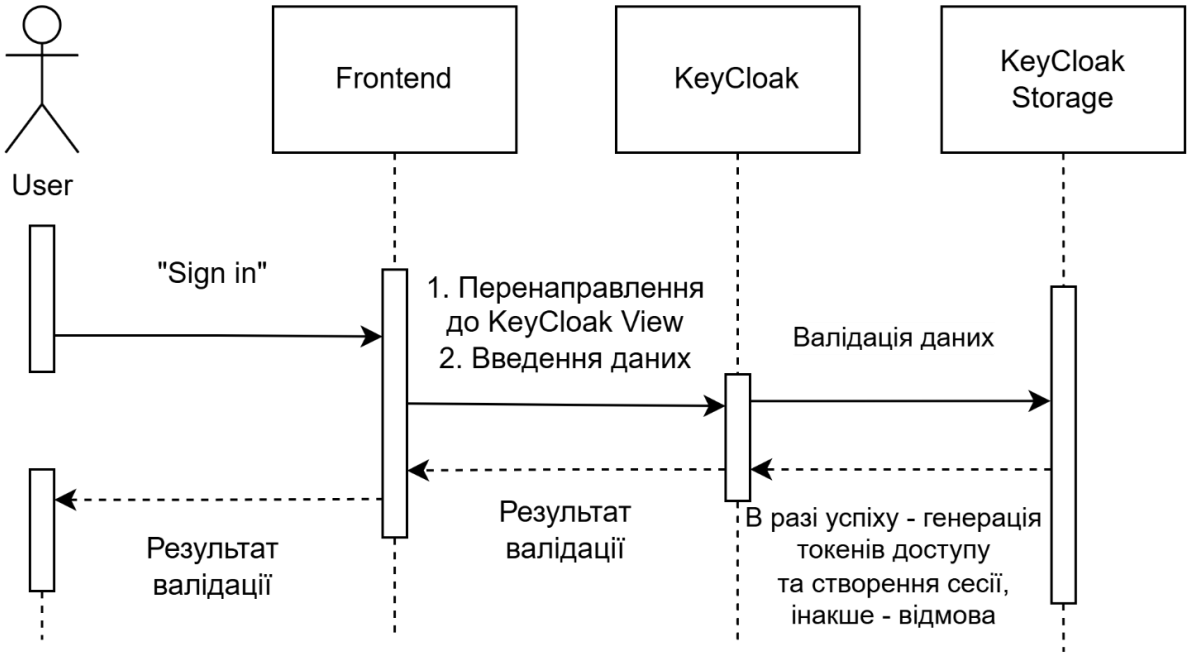

Рисунок 5.8 — Послідовність процесів під час авторизації користувача на платформі

– конструювання курсів викладачами: якщо викладач успішно авторизувався у системі – він отримує доступ до панелі викладача. У цій панелі користувач може бачити власні курси, натиснувши на які може перейти до їх модифікації, або перейти до конструктора та створити новий;

– публікація курсів викладачами: якщо викладач вирішив опублікувати курс для доступу студентам він може натиснути кнопку “Publish” та опублікувати його. Аналогічно викладач може призупинити дію курсу натиснувши “Unpublish”;

– підпис на курси студентами;

– проходження курсу студентами.

Процеси 2-3 подано на діаграмі послідовностей викладача зображеній на рисунку 5.9.

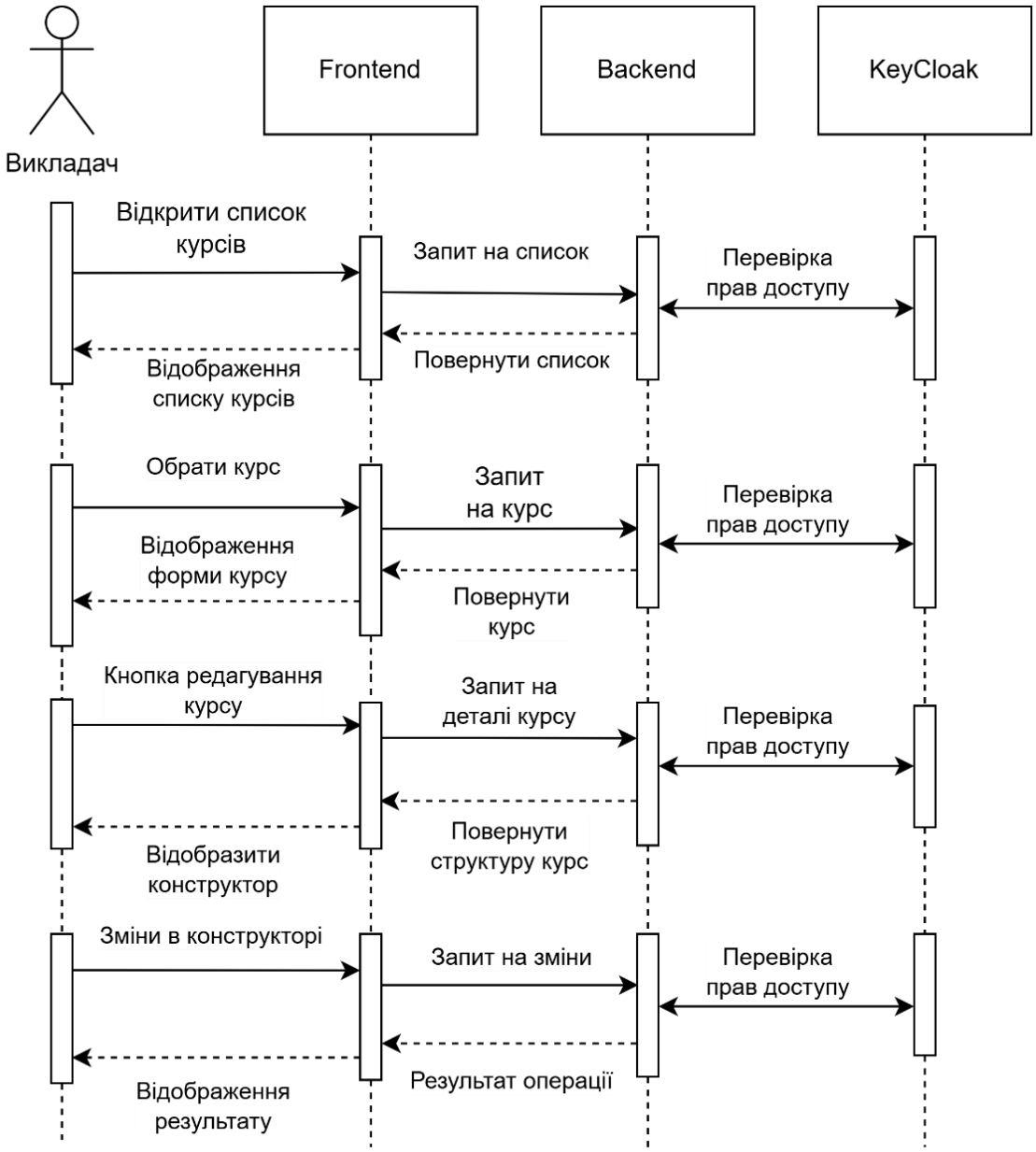

Рисунок 5.9 — Послідовність основних процесів викладача на платформі

Процеси 4-5 подано на діаграмі послідовностей студента зображеній на рисунку 5.10.

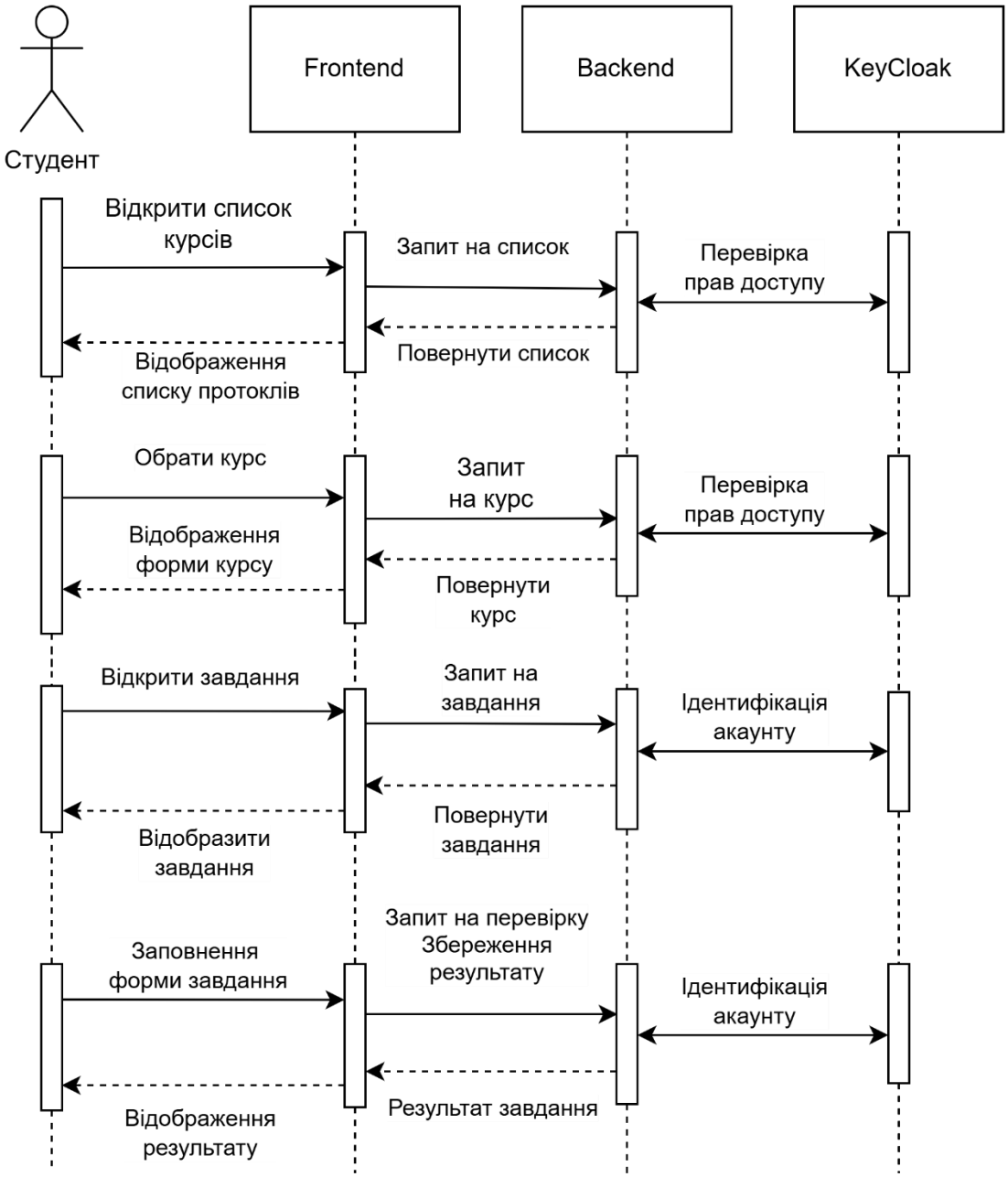

Рисунок 5.10 — Послідовність основних процесів студента на платформі

Детальні послідовності процесів для викладацької та студентської частин платформи описані на діаграмах послідовностей, які наведені у додатках Ж та К відповідно.

5.4.4 Діаграма компонентів

Програмно наша платформа складається з 4-х головних компонентів:

– фронтенд (клієнтський застосунок): компонент, який відповідає за надання інструментарію користувачу, забезпечує взаємодію з платформою через обмін запитами по API з серверним застосунком і відображає дані;

– бекенд (серверний застосунок): компонент, що виконує роль сервісу посередника між клієнтським застосунком і базою даних. Відповідає за валідацію та обробку запитів, які отримує від фронтенду. Обробка здійснюється відповідно до бізнес-логіки системи, після чого відбуваються запити до бази даних для отримання необхідної інформації, внесення змін до таблиць або збереження даних;

– KeyCloak (сервіс аутентифікації та контролю сесій): компонент, що виконує роль фаєрвола (мережевого екрану), який перевіряє валідність запитів шляхом перевірки токенів, які надсилаються разом з запитами до бекенду. Бекенд, у свою чергу, надсилає отриманий токен через API до KeyCloak-у на перевірку. Оскільки KeyCloak зберігає дані про користувачів, ролі та сесії у власній базі даних, він швидко перевіряє отримані токени і повертає результат з підтвердженням прав доступу, або відмовою;

– база даних: компонент, що зберігає дані, що стосуються об’єктів платформи. Він приймає запити, які надійшли від сервера, обробляє їх, і виконує екстракцію, оновлення, збереження чи видалення даних згідно отриманого SQL-запиту. Після чого, надсилає дані назад на сервер.

Для кращого розуміння наведено діаграму компонентів програмного забезпечення у додатку Л.

Додатково, клієнтський застосунок (Frontend) структурно та функціонально можна поділити на дві менших підсистеми (модулі), які виконують власні функції. До модулів клієнтської частини відносяться:

– Course Constructor: є головним модулем платформи, об’єднує в собі функціонал по створенню курсів та графічний інтерфейс з варіативним керуванням. Складається з двох компонентів:

1) Interface (інтерфейс): система відображення графічних компонентів з обробкою керування дій користувачів;

2) Constructor Tools (інструментарій): набір функціональних елементів, відповідальних за роботу з елементами курсу, їх структуризації та впорядкування у об'єкти для відправки на сервер;

– Educational Platform: модуль, який використовує конструктор для надання викладачам можливості створення курсів (використання обох під-модулів конструктора) та їх відображення з можливістю інтерактивної взаємодії для студентів (застосовується під-модуль Interface). Також цей модуль реалізовує взаємодію користувачів з матеріалами, їх перегляд та обмін даними з серверною частиною.

Спільно з бекендом ці компоненти взаємодіють за допомогою архітектури клієнт-сервер з використанням методології REST (Representational State Transfer), призначенням якої є забезпечення ефективної, масштабованої та гнучкої комунікації між клієнтським додатком та сервером.

RESTful API дозволяє клієнтському додатку взаємодіяти з сервером через чітко визначені ендпойнти, використовуючи HTTP-методи для виконання CRUD-операцій (Create, Read, Update, Delete) без вимоги до контролю стану клієнта – уся потрібна для виконання запиту інформація міститься у ньому [23].

Кінцеві точки доступу API були спроектовані з урахуванням методології REST, що відповідає наступним вимогам:

– використання назв сутностей у множині, які підлягають обробці: наприклад/courses, /lessons, /edges;

– вкладені ресурси для відображення ієрархії: /courses /{courseId} /lessons – отримати всі уроки курсу.

Деталізована таблиця стуктури нашого API наведена у таблиці 5.13

Таблиця 5.13 — Структура API освітньої платформи

Ресурс	Метод	Точка доступу	Опис
Курс (Course)	GET	/courses	Отримати список курсів.

Ресурс	Метод	Точка доступу	Опис
	POST	/courses	Створити новий курс (для викладачів).
	GET	/courses/{courseId}	Отримати інформацію про конкретний курс.
	PUT	/courses/{courseId}	Оновити інформацію про курс.
	DELETE	/courses/{courseId}	Видалити курс.
Урок (Lesson)	GET	/courses/{courseId}/lessons	Отримати список уроків курсу.
	POST	/courses/{courseId}/lessons	Додати новий урок до курсу.
	GET	/lessons/{lessonId}	Отримати інформацію про урок.
	PUT	/lessons/{lessonId}	Оновити урок.
	DELETE	/lessons/{lessonId}	Видалити урок.
Enrollment	POST	/courses/{courseId}/enroll	Запис на курс (для студентів).
	GET	/enrollmets Body params: user email	Отримати всі записи на курси студента

Ресурс	Метод	Точка доступу	Опис
Зв’язок Edge	GET	/courses/{courseId}/lessons /{lessonId}/edges	Отримати всі зв'язки для уроку.
	GET	/courses/{courseId}/lessons /{lessonId}/edges/source	Отримати всі зв'язки для уроку де він вихідний об’єкт.
	GET	/courses/{courseId}/lessons /{lessonId}/edges/target	Отримати всі зв'язки для уроку де він вхідний об’єкт.
Подання завдання Submission	PUT	courses/{courseId}/lessons /{lessonId}/submit	Подати виконання завдання для уроку (для студентів)
	GET	/submissions/{submissionId}	Отримати інформацію про конкретне подання.
	GET	/lessons/{lessonId}/submissions	Отримати всі подання для конкретного уроку.

Ресурс	Метод	Точка доступу	Опис
Досягнення Achievements	POST	/achievements/{user_email}	Присвоїти досягнення користувачу
	GET	/achievements/{user_email}	Отримати список досягнень користувача.
	POST	/course/{courseId}/achievements	Створити нове досягнення для курсу (для викладачів)
	GET	/achievements/{achievementId}	Отримати інформацію про конкретне досягнення.
	PUT	/achievements/{achievementId}	Оновити досягнення.
	DELETE	/achievements/{achievementId}	Видалити досягнення.

Висновки до розділу 5

У даному розділі було розглянуто проєктування освітньої платформи, включаючи структуру та функціонал системи. Структура платформи визначена як 4-х компонентна система, елементи якої взаємодіють для виконання основних завдань згідно з бізнес-логікою. Чіткий розподіл додатку, забезпечує надійність за рахунок розмежування обов'язків кожної з них і спрощує процес розробки та підтримки системи.

Важливим етапом було визначення загальних бізнес процесів системи. Разом з ними, було описано механізми гейміфікації що застосовуються до конструктора та навчального процесу. З допомогою цих технік буде поліпшено визначені бізнес процеси навчальної платформи.

Також було проаналізовано функціональні моделі акторів системи. Побудовані моделі варіантів використання, що відображають які функції можуть виконувати актори у системі, як вони взаємодіють з нею та які процеси є важливими для роботи платформи.

На основі отриманої інформації, було визначено сутності які існують у системі та розроблено модель бази даних, яка реалізована за допомогою реляційної системи управління базами даних MySQL. Використання даної СУБД, дозволило організувати залежні сутності через реляційні зв'язки та забезпечити цілісність даних за допомогою зовнішніх ключів. У результаті було спроєктовано наступні таблиці сутностей: Course, Lesson, Edge, Achievement, Enrollment, Submisson, UserAchievement та AchievmentRule. Важливо додати що усі сутності які будуть створені мають каскадну залежність від курсу. Тобто при видалені курсу з бази – будуть видалені уроки, досягнення, зв'язки. Але записи про проходження студентами завдань та курсів лишаться.

Орієнтуючись на базу даних та методологію MVC, було побудовано основну діаграму класів, що дозволяє глибше зрозуміти внутрішню логіку системи, включаючи процес обробки запитів на серверній частині та зв'язки між класами. На основі цього було проведено аналіз послідовності дій користувачів у системі та відповідних її реакцій, було побудовано діаграми на відповідні сценарії для відображення порядку, у якому ця взаємодія відбувається. Додатково було наведено схеми архітектури програмного забезпечення, компонентів та ключових модулів що наявні на платформі. Показано механізми їх взаємодії, на основі яких було описано структуру REST API.

6 КОНСТРУКТОР КУРСІВ

6.1 Загальні положення

Конструктор курсів це основоположний компонент освітньої платформи. Уся платформа будується під реалізований конструктор курсів. Саме тому його детальний опис є важливим для визначення платформи як системи.

Конструктор курсів призначений для забезпечення викладачів інтуїтивно зрозумілим та потужним інструментом для створення, налаштування та управління навчальними курсами. Він дозволяє створювати структуровані курси, додавати уроки, встановлювати послідовність навчання через зв'язки між уроками. Конструктор спроектований з урахуванням модульної архітектури, що забезпечує гнучкість, масштабованість та легкість у підтримці та розширенні його функціональності.

6.2 Структура даних

Конструктор курсів є основним інструментом для створення та управління структурами курсів на платформі, тому його функціонал і набір даних повинні повністю відповідати вимогам системи. Він має організовувати всі дані в єдину структуру у форматі JSON (можна вважати приблизним виглядом об’єкту SaveCourseDTO), яка виглядає наступним чином:

а) курс:

1) Title – назва курсу;

2) Lessons – масив об’єктів уроків;

– node_id – ідентифікатор картки уроку для бібліотеки ReactFlow;

– type – тип картки (вказується так як ми створюємо власний об’єкт під потреби платформи);

– isNecessary – прапорець, що визначає обов’язковість уроку;

– inputPower – вхідна величина, величина активації, мінімальне значення суми вихідних величин інших уроків, що потрібна для розблокування уроку;

– outputPower – вихідна величина, що передається усім з'єднаним на вхід урокам;

– Position (x,y) – об'єкт для збереження позиції картки на графічному полі;

– achievements – масив досягнень які застосовані до даного уроку;

– data – об'єкт для збереження даних уроку:

1. title – назва уроку;
2. quiz – об'єкт для збереження тестового завдання;
3. text – об'єкт для збереження текстової лекції;
4. url – об'єкт для збереження посилання на відео;

Важливо що ми зберігаємо усі варіанти уроку до моменту відправки його у базу даних. Це вирішено було зробити для надійності та зручності, оскільки викладач може мати декілька варіантів уроку і може його змінювати аж до того моменту поки не вирішить зберегти усі зміни.

3) Edges – окремий масив що показує зв'язки між картками уроків (це зроблено для спрощення роботи з даними курсу, так як витягувати зв'язки з кожного уроку було б занадто витратним процесом):

– edge_id – ідентифікатор об'єкту зв'язку уроку для бібліотеки ReactFlow;

– source – ідентифікатор картки з якої виходить зв'язок;

– target – ідентифікатор картки в яку направлений зв'язок.

6.3 Функціональні можливості

Конструктор курсів надає широкий спектр функцій, спрямованих на полегшення процесу створення та управління навчальним курсом. Детальний опис основних функціональних можливостей конструктора курсів:

– створення навчальних карток: конструктор відображає та створює шаблонну картку (елемент у просторі конструктора), що позначає урок. На цьому моменті урок заповнений шаблонними даними та потребує редагування;

– редагування контенту: після створення навчальної картки викладач має можливість редагувати контент уроку. Це включає додавання текстового матеріалу, відео, зображень, тестового завдання;

– створення дочірнього елементу: конструктор може створювати “дочірні” уроки, які також шаблонні до моменту редагування, але вже з'єднані. Таким чином можна швидше створювати послідовності уроків;

– видалення уроку: конструктор надає функціональність видалення уроків з курсу. Це дозволяє викладачам корегувати структуру курсу, видаляти непотрібні або застарілі уроки з відповідними йому зв’язками, без порушення загальної організації навчального матеріалу. Процес видалення є безпечним та не впливає на інші компоненти курсу.

– створення зв’язку: надається можливість з’єднати два уроки орієнтованим зв’язком;

– видалення зв’язку: у випадку якщо зв’язок непотрібний між двома картками його можна розірвати;

– зміна зв’язку: зв’язок можна перевизначити перетягнувши один з кінців на іншу картку, це повністю перевизначить зв’язок (тобто зміниться не просто ідентифікатор картки, а відбудеться повна переініціалізація зв’язку з, відповідно, новим ідентифікатором);

– створення досягнень: викладач може створити досягнення з певним правилом здобуття винагороди за нього;

– закріплення досягнень: для того аби досягнення було активне воно має бути закріплене за елементом (урок, група уроків або сам курс для якого досягнення було створене);

– визначення вхідних та вихідних величин уроків: для реалізації гнучкої структури курсу мають бути визначені вхідні та вихідні величини для кожного з уроків, ці вагові коефіцієнти впливають на доступність уроку до проходження.

6.4 Графічний інтерфейс

Графічний інтерфейс конструктора розроблений за мінімалістичною схемою. Більшість елементів приховані в окремих вікнах або відкриваються після відповідних дій мишкою. Основним принципом який дотримувався при проєктуванні інтерфейсу була інтуїтивність використання.

Робоче поле конструктора розділено на дві зони: робочий простір, та панель керування.

– Панель керування дає можливість зберегти курс , змінити його назву, або перейти на інші сторінки платформи;

– Робочий простір – площина в які розміщуються створені об’єкти.

Надихаючись такими графічними конструкторами як Figma, було вирішено створити адаптивне керування. Головною ідеєю є надання можливості використовувати конструктор як через мишку та клавіатуру так і через жести на сенсорній панелі. Для цього було опрацьовано наступні варіанти керування:

– збільшення та зменшення (Zooming): функція, яка дозволяє користувачам змінювати масштаб робочого простору для більш детального перегляду або загального огляду курсу. Реалізовано як через колесо мишки, так і через жести масштабування на сенсорній панелі. Ця функція забезпечує гнучкість у роботі з великими або деталізованими курсами;

– пересування (Panning): дозволяє користувачам переміщувати об’єкти робочого простору, щоб оглянути різні частини курсу. Реалізовано через натискання та перетягування мишкою або використання двох пальців для

пересування на сенсорній панелі. Це забезпечує зручність у навігації по великому робочому простору без необхідності постійного масштабування.

Додатково для керування була реалізована підтримка клавіш:

– контекстне меню: при натисканні ПКМ у робочому просторі буде відкриватись контекстне меню яке має різний набір варіантів в залежності від місця де було натиснуто кнопку (якщо натиснути на картці то збуде запропоновано її видалити, якщо на групі карток – видалити групу, якщо у пустому місці – створити новий урок);

– видалення: надається можливість видалити виділені об'єкти через контекстне меню або через клавішу Delete;

– групове виділення (Multi-Selection): дозволяє користувачам вибирати окремі об’єкти або групи об’єктів на робочому просторі для їх редагування, видалення або переміщення. Виділені об’єкти підсвічуються, що забезпечує чітке візуальне підтвердження вибору. Забезпечується одночасним натисканням клавіші Shift та ЛКМ;

6.5 Система планування

Оскільки завдання мають складні залежності, важливо забезпечити студентам можливість легко зрозуміти, які саме завдання потрібно виконати для доступу до бажаного. Зважаючи на складність цих залежностей, студентам буває важко зрозуміти, які саме кроки необхідно зробити для доступу до конкретного завдання. Було б корисно відображати набір необхідних завдань, які потрібно розв'язати, щоб активувати обране завдання, враховуючи всі зв’язки та умови їх активації.

6.5.1 Постановка задачі

Задано множину завдань, які студент може виконати в рамках навчального курсу. Між цими завданнями існують зв'язки, що означають залежності: виконання

одного завдання може вимагати попереднього виконання інших. Ці залежності визначають послідовність та умови, за яких завдання стають доступними для студента.

Для кожного завдання задано величину активації. Завдання стає доступним для виконання (активованим), якщо його величина активації менша або рівна величині впливу сусідніх завдань, які студент виконав. Кожне виконане завдання передає власну величину "впливу" суміжним з нею завданням.

Поточний стан студента характеризується множиною завдань, які він уже виконав або які доступні для виконання без додаткових умов. Студент бажає отримати доступ до конкретного завдання, яке наразі недоступне через існуючі зв’язки які утворюють послідовності завдань.

Необхідно визначити мінімальний набір завдань, які студент повинен виконати, щоб отримати доступ до бажаного завдання, враховуючи всі існуючі залежності, умови активації та поточний прогрес курсу.

6.5.2 Формалізація задачі

Для точного визначення проблеми використаємо математичну модель на основі орієнтованого графа $G(V, E)$, де V – множина вершин, кожна з яких відповідає окремому завданню, E – множина орієнтованих ребер, які відображають залежності між завданнями. Ребро (u,v) означає, що виконання завдання v залежить від виконання завдання u.

Кожна вершина має певний поріг активації $e(v)$, який визначає мінімальну сумарну величину впливу попередніх відвіданих вершин, необхідну для активації завдання v. Якщо цей поріг не досягнуто, вершина залишається заблокованою, тобто недоступною для відвідування.

Крім того вершини мають певну величину впливу на сусідні вершини $x(v)$, яка активується після відвідування вершини v.

Активація (доступність до відвідування) вершини v відбувається за формулою:

$$\sum_{u \in N^-(v) \cap A} x(u) \geq e(v), \quad (6.1)$$

де $N^-(v)$ – множина безпосередніх попередників (вхідних сусідів) завдання v у графі G, A – множина вершин які відвідані (завдань які виконані) - активовані.

Потрібно знайти мінімальну підмножину завдань $T \subseteq V \setminus S$, яку студент повинен виконати, щоб бажане завдання $X \in V$ стало доступним (активованим). Множина S – це множина стартових завдань які вже активовані (виконані або доступні до виконання через те що умови активації виконані попередньо).

6.5.3 Типологія задачі

Поставлена задача є варіантом задачі вибору цільового набору (Target Set Selection Problem, TSSP), яка є відомою в теорії графів та комбінаторній оптимізації. TSSP полягає в знаходженні мінімальної множини вершин у графі, активація яких призведе до активації певної цільової множини вершин або всього графа згідно з заданими правилами поширення активації. Нашу задачу можна вважати частковою задачею вибору цільового набору, де цільовим набором виступає одна обрана вершина.

TSSP є NP-складною задачею, оскільки немає відомих поліноміальних алгоритмів, які б розв'язували всі випадки цієї задачі за прийнятний час. А кількість можливих комбінацій завдань для активації експоненційно зростає з кількістю завдань у системі, що робить повний перебір всіх можливих варіантів обчислювально неможливим для великих графів. Хоча для деяких класів графів, наприклад у повних графах або графах без циклів з однаковими порогами активації, задача може бути розв’язана за поліноміальний час, але наш варіант задачі не відноситься до цих класів і, ймовірно, не має рішення за поліноміальний час.

Практичні застосування TSSP:

– соціальні мережі: моделювання поширення інформації, впливу або епідемій; визначення ключових осіб для максимального поширення інформації.

– вірусний маркетинг: вибір оптимальної групи клієнтів для запуску рекламної кампанії.

– мережеві системи: аналіз надійності мереж; виявлення вразливих місць.

– біологічні системи: розуміння поширення сигналів у біологічних мережах.

6.5.4 Методи розв’язання

Розглянемо детально можливі підходи до розв'язання поставленої задачі та подамо їх у вигляді таблиці 6.1

Таблиця 6.1 — Алгоритми розв’язання задачі TSSP

Назва	Опис	Переваги	Недоліки
Повний перебір	Перевірка всіх можливих комбінацій; практично непридатний для великих графів через експоненційну складність.	– Гарантовано знаходить мінімальну множину вершин для активації цільової вершини.	– Експоненційна обчислювальна складність; – Великі ресурсні затрати.
Метод гілок та меж	Метод будує дерево, де кожна вершина (вузол) представляє часткове рішення, тобто певну множину активованих вершин. Рішення, які не можуть привести до кращого рішення,	– Гарантує знаходження оптимального рішення; – Відсікання значної кількості неефективних гілок зменшує обсяг обчислень.	– У найгіршому випадку метод може вимагати перебору всіх можливих комбінацій; – Ресурсні затрати менші ніж в повного перебору але - суттєві.

Назва	Опис	Переваги	Недоліки
	ніж уже знайдене, відсікаються.		
Жадібні алгоритми	На кожному кроці алгоритм робить вибір, який здається найкращим у даний момент, без урахування глобальних наслідків.	– Швидкодія; – Простота реалізації; – Придатний для роботи з великими графами.	– Не гарантує знаходження оптимального рішення; – Результат залежний від обраної евристики.
Метаевристики	Алгоритми, такі як генетичні алгоритми, імітації відпалу, табу-пошук, які можуть знаходити близькі до оптимальних рішень. Використовують біологічно натхненні процеси для пошуку близьких до оптимальних рішень.	– Придатні для великих графів; – Можуть бути адаптовані під специфіку задачі; – Гарантують результат наближений до оптимального (не найгірший з можливих).	– Рішення все ще можуть бути далекими від оптимальних; – Складні у налаштуванні під специфіку задачі.

6.5.5 Обґрунтування методу розв’язання

Враховуючи порівняння методів розв’язання TSSP задач, було обрано метод «гілок та меж». На це вплинули наступні фактори:

– оптимальність рішення: метод гарантує знаходження оптимального розв'язку, що є важливим у нашому контексті, де оптимальний план буде мати великий вплив на ефективність навчального процесу. Також важливим фактором є те, що додавання такої опції графічних підказок по плануванню втрачає сенс якщо план буде видавати хаотичні набори, або набори з зайвими завданнями;

– відсікання неефективних варіантів: надає нам можливість не розглядати варіанти, які вже гірші за поточне рішення;

– придатність для графів малого та середнього розміру: у навчальному процесі поява курсів з кількістю уроків більшою за 100 малоймовірна. Також варто враховувати що алгоритм планування не обов’язково може застосовуватись на максимальну дистанцію - від початкового до кінцевого уроку, а не певні субмаршрути, які будуть значно коротші за максимально можливий.

Далі детально розглянемо використання алгоритму «гілок та меж» у нашій задачі. Його основна ідея полягає в систематичному переборі можливих рішень задачі у вигляді дерева рішень, де кожна вершина (вузол) представляє часткове рішення – множину активації. Знаючи розмір найкращого рішення можемо використати його як межу, що дозволяє відсікати гілки, які не можуть привести до кращого рішення, ніж вже знайдене.

Компоненти алгоритму:

а) дерево рішень:

1) кожен вузол дерева відповідає певній підмножині вершин, які були безпосередньо активовані.

2) кожна гілка представляє вибір додаткової вершини для безпосередньої активації.

б) функція оцінки (Lower Bound):

1) оцінює мінімальну кількість додаткових вершин, які ще потрібно активувати з поточного стану, щоб досягти цілі.

2) допомагає визначити, чи варто продовжувати розглядати поточну гілку.

в) верхня межа (Upper Bound):

1) найкраще знайдене рішення на даний момент (мінімальна кількість вершин для активації).

2) використовується для відсікання гілок, які не можуть покращити поточне найкраще рішення.

г) відсікання гілок:

1) якщо оцінка для поточного вузла перевищує верхню межу, ця гілка не розглядається далі.

Покроковий опис алгоритму:

1. ініціалізація: створюємо кореневий вузол дерева з початковою множиною активованих вершин $A=S$ та порожньою множиною безпосередньо активованих вершин $T=\emptyset$, та встановлюємо верхню межу U на нескінченність.

2. рекурсивний перебір:

2.1. вибір вузла для розширення: вибір наступного вузла, яким буде розширено наше поточне рішення, за принципом пошуку вглиб;

2.2. перевірка умови завершення: якщо цільова вершина X активована в поточному стані, оновлюємо верхню межу U та зберігаємо рішення, якщо воно краще за попереднє. Потім повертаємось до попереднього вузла;

2.3. обчислення оцінки: обчислюємо кількість активованих вершин у множині поточного вузла. Якщо ця оцінка вища за верхню межу U, відсікаємо цю гілку;

2.4. генерація нащадків: розглядаємо всі можливі вершини $v \in V \setminus A$, які можна безпосередньо активувати. Для кожної такої вершини створюємо новий вузол з множинами $A' = A \cup \{v\}$ та $T' = T \cup \{v\}$. Після чого, пропагуємо активацію від нової вершини та оновлюємо A'.

6.5.6 Тестування алгоритму

Для алгоритму було створене тестове спрощене середовище, яке містило основні функціональні елементи, що важливі для роботи алгоритму. Відповідно деталей уроків, чи інших навчальних компонент не було, основною задачею є тестування алгоритму визначення множини завдань, що вимагає від нас лише інформацію про вершини (картки), зв'язки між ними та значення активації і впливу кожної з вершин.

Основним тестом виступив граф зображений на рисунку 6.1.

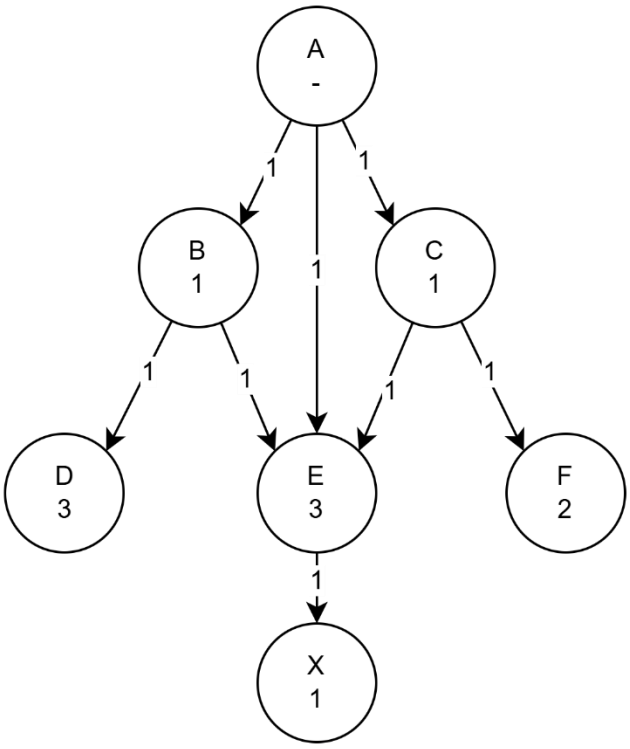

Рисунок 6.1 — Граф умовного навчального курсу

На цьому рисунку зображено граф, де кожна вершина має назву та величину активації, а зв'язки відображають вплив вершини на суміжні з нею. Для заданого графу вершина A вже активована і є нашою стартовою множиною.

Розглянемо декілька прикладів. Для початку оберемо цільовою вершиною – E. Ця вершина має величину активації – 3. На неї мають вплив суміжні вершини A, B і C. Усі разом ці вершини створюють достатній вплив для активації вершини E. Але за умовою задачі наша стартова множина активованих вершин складається

лише з вершини A. Аби вершини B та C також здійснили свій вплив їм потрібно бути активованими. Їх величини активації рівні 1 і на них здійснює вплив лише вершина A, у якої величина цього впливу також рівна 1, чого достатньо для активації. Множина вершин потрібних до активації вершини E наведена на рисунку 6.2.

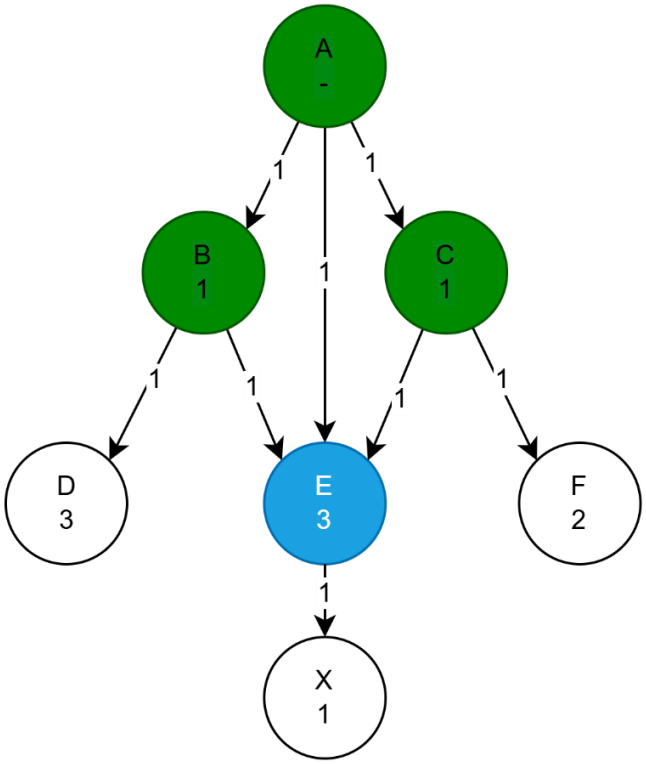

Рисунок 6.2 — Мінімальна множина активації вершини E

Розглянемо цей же грав але змінимо цільову вершину на F. Явно видно, що вона не доступна для активації через недостатній вплив суміжних вершин. Що означає що розроблений метод має повернути порожню множину.

Видозмінимо цей граф створивши опціональний шлях до X через F, та зробимо F доступним до активації. Це дозволить перевірити коректність створеного алгоритму та підтвердити що він здатний знаходити оптимальні множини. Новий граф зображено на рисунку 6.3.

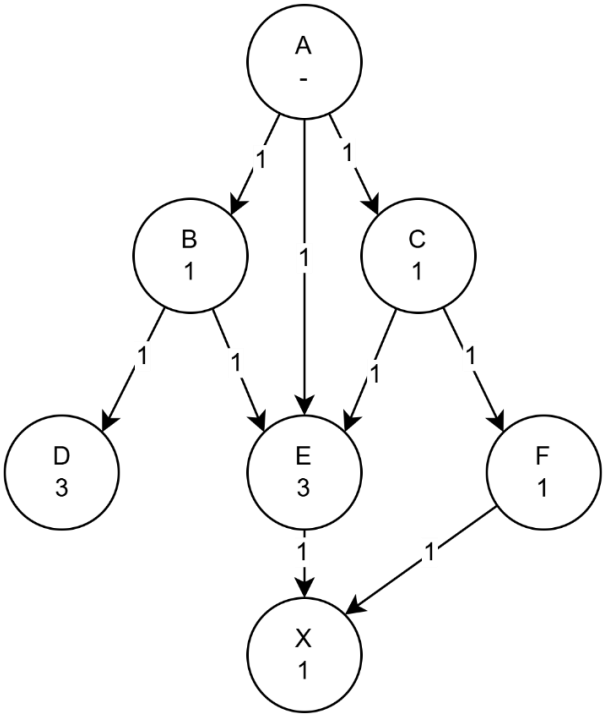

Рисунок 6.3 — Модифікований граф

Очевидно що для такого графу можливість активації вершини X варіативна. Її активація можлива як через вершину E, так і через вершину F. Варіанти активації зображено на рисунках 6.4 та 6.5 відповідно.

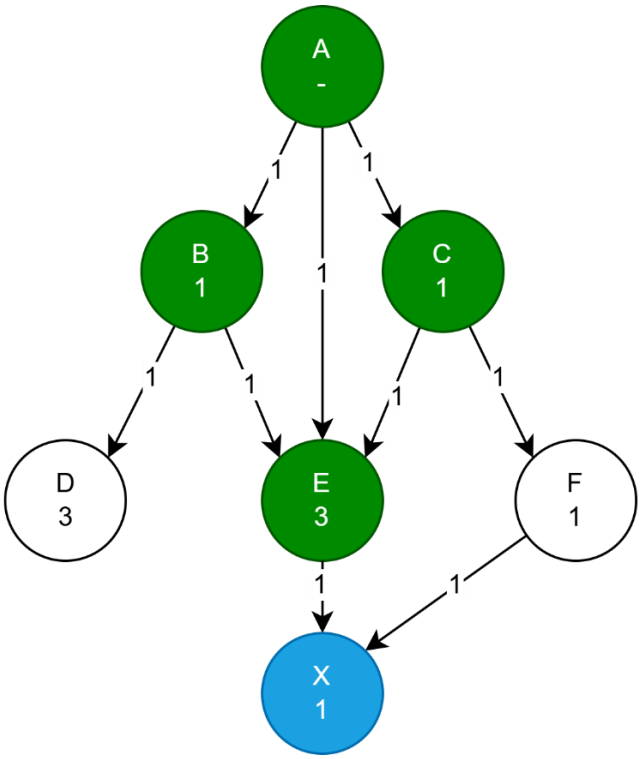

Рисунок 6.4 — Множина активації через вершину E

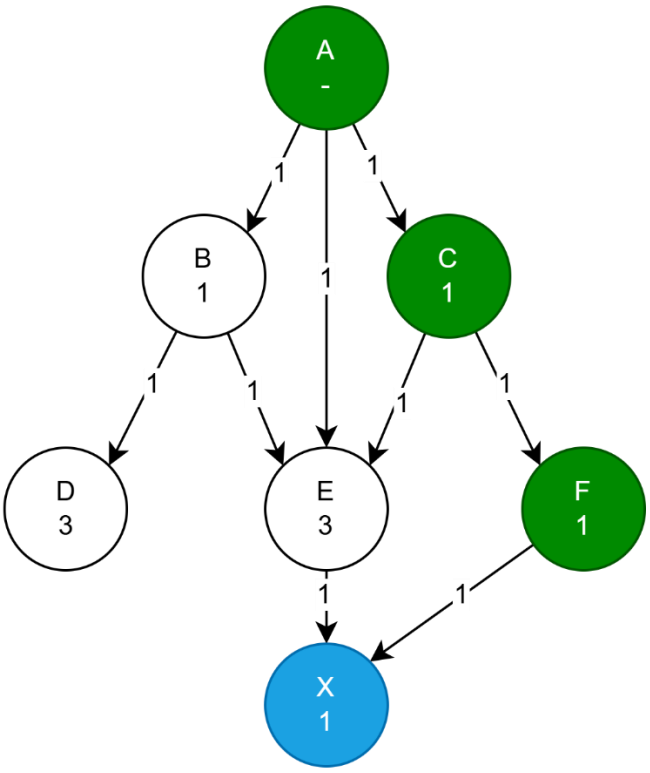

Рисунок 6.5 — Множина активації через вершину F

Очевидно що варіант множини через вершину F є кращим, оскільки містить в собі меншу кількість вершин.

Наведемо ці тестові варіанти разом з результатами роботи програми розробленої на мові Python у таблиці 6.2.

Таблиця 6.2 — Тестування алгоритму

ID	Цільова вершина	Множина	Фактичний результат	Час виконання (секунди)
1	E	{A, B, C}	A-B-C	0.001
2	D	{∅}	None	0.0005
3	X	{A, C, F}	A-C-F	0.009

Висновки до розділу 6

У цьому розділі розроблено конструктор курсів, який є ключовим компонентом освітньої платформи. Описано підхід до розробки графічного інтерфейсу, що полягає у мінімалістичності та розділені на робочий простір та панель керування, що забезпечує викладачам інтуїтивно зрозумілий та ефективний інструмент для створення та управління навчальними курсами.

Наведено можливості конструктора щодо створення та редагування навчальних карток, управління зв'язками між уроками, додавання інтерактивних елементів, таких як завдання різного формату та досягнення. Описано можливість налаштування зв'язків між уроками за допомогою вагових коефіцієнтів, що дозволяє викладачам моделювати послідовність та варіативність навчального процесу. Це забезпечує гнучкість у створенні як простих, так і складних курсів, адаптованих до потреб студентів.

Розглянуто дизайн конструктора, натхненний таким графічним інструментом як Figma, що дозволяє використання як традиційних пристроїв вводу (клавіатури, миші), так і сенсорних панелей. Це розширює можливості користувачів і робить платформу доступною для роботи з використанням різних пристроїв керування, що покращує загальний користувацький досвід.

Описано інтерактивність конструктора, яка забезпечується функціями виділення, масштабування, переміщення різних елементів, підтримкою жестів та контекстних вікон. Це сприяє плавній навігації та дозволяє швидко вносити зміни в структуру або вміст курсу, роблячи процес створення максимально простим і ефективним.

Додатково розроблено алгоритм планування навчального процесу для студента. Наведено суть алгоритму, яка полягає у підсвічуванні певних уроків, проходження яких дозволить відкрити обраний студентом урок. Для реалізації цього рішення обрано метод «гілок та меж», як підхід, що гарантує знаходження найкоротшого шляху до бажаного уроку без виконання повного перебору в більшості випадків. Це дозволяє отримати оптимальний результат за відносно

прийнятний час, оскільки курси не мають надзвичайно складних структур, а студенти зазвичай використовують цей механізм на проміжних етапах навчання, де розміри графа зменшені завдяки їхньому прогресу.

Таким чином, у розділі розроблено та описано конструктор курсів, який поєднує універсальність, гнучкість та інтуїтивність, надаючи викладачам усі необхідні інструменти для створення навчальних програм. А студентам забезпечує можливість взаємодії з курсами у тому ж робочому просторі з обмеженням на редагування.

7 ТЕСТУВАННЯ СИСТЕМИ

У даному розділі представлено процес тестування системи, детальний опис тест-кейсів, порядок їх реалізації для перевірки відповідності програмного продукту функціональним вимогам.

7.1 Мета випробувань

Метою випробувань є перевірка відповідності розроблених функцій, комплексу поставлених задач наведених у функціональних вимогах. Ці вимоги передбачають перевірку коректності реалізації функцій системи, її стабільності та здатності ефективно виконувати необхідні завдання в різних умовах та функціонувати відповідно до визначених специфікацій.

7.2 Загальні положення

Тестування програмного забезпечення – це процес, спрямований на оцінку та перевірку якості програмного продукту. Визначає наскільки продукт відповідає вказаним вимогам та виконує передбачені функції. Функціональність системи перевіряється у ситуаціях, створених штучно для імітації реальних процесів роботи системи. Для цього створюються спеціальні набори тестів, тест-кейсів, для перевірки тих чи інших вимог [24].

Кожний тест складається з наступних компонентів:

– цілі тестування: вказує на функціональні особливості програми, які перевіряються тестуванням;

– вхідні дані: чітко визначені дані, які вводяться у систему для виконання тестування;

– передумови: умови, які мають бути виконані перед початком тестування для забезпечення коректної оцінки (наприклад, наявність певних даних у базі або авторизований користувач);

– етапи виконання: послідовність операцій, які потрібно провести для реалізації тесту;

– очікуваний результат: результат, який передбачається від системи після завершення тестування відповідно до наданих вхідних даних і кроків виконання;

– фактичний результат: результат, отриманий після виконання тесту, цей результат порівнюється з очікуваним для встановлення оцінки відповідності;

– статус: статус тесту (пройдено або не пройдено).

7.3 Модульне тестування

Модульне тестування полягає у перевірці окремих методів або класів системи в ізоляції від інших компонентів. Це дозволяє виявити та виправити помилки на ранніх стадіях розробки.

7.3.1 Використані інструменти

– Spring Boot Test: Інтеграція Spring Boot з тестовим середовищем, що дозволяє легко тестувати компоненти, які використовують Spring Context;

– Postman: Потужний інструмент для тестування RESTful API. Використовувався для перевірки коректності роботи API ендпойнтів, аналізу відповідей сервера та автоматизації тестів за допомогою колекцій.

7.3.2 Spring Boot Test

На основі програмного коду бекенду були створені тестові випадки для основних методів сервісних класів.

Для прикладу наведемо тести для класу CourseService:

7.3.2.1 Тест 1: Перевірка створення порожнього курсу

Код тесту 1 поданий на рисунку 7.1.

```
@SpringBootTest
public class CourseServiceTest {

    @Autowired
    private CourseService courseService;

    @MockBean
    private CourseRepository courseRepository;

    @Test
    public void testSaveEmptyCourse() {
        SaveCourseDto courseDto = new SaveCourseDto();
        courseDto.setTitle("Test Course");

        Course course = new Course();
        course.setCourseId("1");
        course.setTitle("Test Course");
        course.setDescription("No description");
        course.setCreationDate(new Date());

        when(courseRepository.save(any(Course.class))).thenReturn(course);

        Course savedCourse = courseService.saveEmptyCourse(courseDto,
"test@example.com");

        assertNotNull(savedCourse);
        assertEquals("Test Course", savedCourse.getTitle());
        assertEquals("No description", savedCourse.getDescription());
    }
}
```

Рисунок 7.1 — Код тесту 1

Пояснення до коду:

– метод testSaveEmptyCourse перевіряє, чи метод saveEmptyCourse коректно створює новий курс з переданими даними;

– використовується анотація @MockBean для мокування залежності CourseRepository;

– метод when(courseRepository…) налаштовує поведінку мок-об'єкта;

– виконується перевірка результату за допомогою assertNotNull та assertEquals.

7.3.2.2 Тест 2: Перевірка пошуку курсу за ідентифікатором

Код тесту 2 поданий на рисунку 7.2.

```
@SpringBootTest
public class CourseServiceTest {

    @Autowired
    private CourseService courseService;

    @MockBean
    private CourseRepository courseRepository;

    @Test
    public void testFindCourse() {
        String courseId = "1";
        String email = "test@example.com";

        Course course = new Course();
        course.setCourseId(courseId);
        course.setTitle("Test Course");
        course.setAuthorEmail(email);


when(courseRepository.findById(courseId)).thenReturn(Optional.of(course));

        CourseDto courseDto = courseService.findCourse(courseId, email);

        assertNotNull(courseDto);
        assertEquals("Test Course", courseDto.getTitle());
    }
}
```

Рисунок 7.2 — Код тесту 2

Пояснення до коду:

– метод testFindCourse перевіряє, чи метод findCourse коректно знаходить курс за ідентифікатором та повертає відповідний DTO;

– тест ізолює CourseService від реальної бази даних, використовується анотація @MockBean, яка створює підроблений (mocked) екземпляр CourseRepository;

– метод courseRepository.findById(courseId) мокується, щоб повертати Optional із підготовленим об’єктом Course;

– за допомогою методу assertNotNull перевіряється, що CourseDto не є null, тобто метод findCourse повернув валідний результат;

– використовується метод assertEquals для перевірки, що назва курсу (title) у поверненому об’єкті CourseDto відповідає очікуваній назві ("Test Course").

7.3.2.3 Тест 3: Перевірка видалення курсу

Код тесту 3 наведений на рисунку 7.3.

```
@SpringBootTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
public class CourseServiceIntegrationTest {

    @Autowired
    private CourseService courseService;
    @Autowired
    private CourseRepository courseRepository;
    @Autowired
    private LessonRepository lessonRepository;
    @Autowired
    private StorageService storageService;

    @Test
    @Transactional
    public void testRemoveCourseAndLessons() {
        String courseId = "1";
        String email = "test@example.com";

        // Create and save course and lessons in the database
        Course course = new Course();
        course.setCourseId(courseId);
        course.setAuthorEmail(email);

        Lesson lesson1 = new Lesson();
        lesson1.setLessonId("lesson1");
        lesson1.setCourse(course);

        Lesson lesson2 = new Lesson();
        lesson2.setLessonId("lesson2");
        lesson2.setCourse(course);

        course.setLessons(Arrays.asList(lesson1, lesson2));

        courseRepository.save(course);

        courseService.remove(courseId, email);

        assertFalse(courseRepository.findById(courseId).isPresent());
        assertTrue(lessonRepository.findByCourse(course).isEmpty());
    }
}
```

Рисунок 7.3 — Код тесту 3

Пояснення до коду:

– метод testRemoveCourseAndLessons перевіряє, чи метод remove у CourseService коректно видаляє курс разом із пов’язаними уроками. Для цього тестується інтеграційна поведінка компонентів, що включає роботу з базою даних;

– створюється екземпляр курсу Course із заданим courseId і authorEmail;

– до курсу додаються два уроки Lesson (lesson1 і lesson2), які зв’язуються із курсом за допомогою методу setCourse;

– дані зберігаються у базі через courseRepository;

– метод courseService.remove викликається для видалення курсу за його courseId та email;

– за допомогою courseRepository.findById(courseId) перевіряється, чи курс було видалено з бази даних. Очікується, що Optional буде порожнім;

– за допомогою lessonRepository.findByCourse(course) перевіряється, чи всі уроки, пов’язані з курсом, також були видалені. Очікується, що список поверне порожній результат;

– анотація @Transactional: Забезпечує виконання тесту в межах транзакції, що гарантує, що всі зміни, зроблені під час тесту, будуть скасовані після завершення виконання;

– @AutoConfigureTestDatabase: Забезпечує використання реальної бази даних замість вбудованої, що дозволяє перевірити поведінку в умовах, максимально наближених до продакшн-середовища;

– цей тест не використовує mock-об’єкти для CourseRepository чи LessonRepository, а взаємодіє з реальною базою даних через репозиторії. Це дозволяє перевірити повну інтеграцію між рівнями сервісів і бази даних.

7.3.3 Оцінка Відсотка Покриття Коду Тестами

Для оцінки покриття коду тестами використовувався інструмент JaCoCo. Результати показали наступне покриття для бекенд-компонентів:

– покриття класів: 85% , не були покриті тестами класи контролери (так як вони перевірялись вручну через Postman);

– покриття методів: 80%, не були покриті допоміжні методи, що працюють з перетворенням та екстракцією даних. Більшість з цих методів є згенерованими за рахунок анотацій фреймворку та їх тестування можна вважати не доцільним та занадто затратним за часом, так як згенеровані методи мають дуже чітку поведінку;

– покриття рядків коду: 75%, враховуючи попередні показники можна вважати результат у 75% відмінним.

7.4 Результати випробувань

Наведемо результати тестування вимог, встановлених до системи, за допомогою тест-кейсів [25]. Відповідні дані буде подано у таблицях 7.1 – 7.8.

Таблиця 7.1 — Тестування функції авторизації користувача

Ціль	Перевірка авторизації користувача в систему .
Передумова	– дані для авторизації користувача (викладача): електронна пошта та пароль були попередньо створені при реєстрації та записані у базу даних KeyCloak; – користувач знаходиться на сторінці входу до платформи та натиснув кнопку для авторизації.
Вхідні дані	Дані аутентифікації.
Кроки виконання	– у поле «Електронна пошта» ввести логін користувача, який відповідає формату електронної адреси, наприклад «kolyatest@test.com»; – у поле «Пароль» ввести пароль користувача, який відповідає електронній пошті, «secret1111»; – натиснути кнопку «Login».

Ціль	Перевірка авторизації користувача в систему .
Очікуваний результат	Користувач успішно авторизований в системі і отримав доступ до даних що належать відповідному профілю.
Фактичний результат	Користувач авторизувався та потрапив на головну сторінку панелі курсів, де відображається загальних список створених ним курсів (якщо вони існують).
Статус	Пройдено.

Таблиця 7.2 — Тестування функції перегляду наявних курсів користувача

Ціль	Перевірка платформи на здатність надавати користувачу доступ до перегляду його курсів (для викладача створених курсів, для студента - підписаних).
Передумова	– користувач авторизований у систему; – користувач вже має курси у власній теці, у випадку їх відсутності система запропонує користувачу створити новий курс, або підписатись на існуючий, в залежності від ролі користувача.
Вхідні дані	Запит на отримання списку курсів.
Кроки виконання	Натиснути на панелі керування на вкладку «Home Page»;
Очікуваний результат	Список курсів, закріплених за користувачем (створених викладачем, або підписані студентом) відображений на екрані.
Фактичний результат	Список курсів відображається.
Статус	Пройдено.

Таблиця 7.3 — Тестування функції відкриття курсу студентом

Ціль	Перевірка інструментів відображення, взаємодії з наявними курсами та візуалізації його структури.
Передумова	– студент авторизований у системі; – студент вже попередньо записаний на курс та має до нього доступ.
Вхідні дані	Інформація бажана до перегляду студентом (ідентифікатор або назва курсу)
Кроки виконання	Для перегляду детальної інформації курсу: – натиснути на панелі керування на вкладку «HomePage»; – обрати плитку курсу з списку курсів і натиснути на неї; – переглянути курс з його деталями та структурою.
Очікуваний результат	Студент отримав доступ та може бачити курс та завдання.
Фактичний результат	Студент отримав доступ до курсу та бачить поточний стан та усю структуру курсу.
Статус	Пройдено.

Таблиця 7.4 — Тестування функції створення курсу

Ціль	Перевірка функціоналу створення нового курсу викладачем на платформі.
Передумова	– викладач авторизований у систему;
Вхідні дані	Дані для створення нового курсу (назва, уроки, опціонально інші дані)
Кроки виконання	– натиснути на панелі керування на вкладку «Course Constructor»; – ввести назву курсу; – зберегти курс одразу з двома шаблонними уроками, або створити власні;

Ціль	Перевірка функціоналу створення нового курсу викладачем на платформі.
	– якщо викладач бажає створити власний урок він може натиснути «+» на існуючому уроці для автоматичного створення дочірнього, або натиснути ПКМ у пустому просторі поля конструктора та вибрати опцію «Create new node»; – за бажанням викладач може додати якусь інформацію у створені уроки; – якщо ні то уроки заповняться по шаблону для базового збереження; – натиснути кнопку «Save Course».
Очікуваний результат	Курс створено успішно без помилок у процесі вводу, дані відразу зберігаються та відображається у системі на сторінці «Home Page» у списку курсів.
Фактичний результат	Система успішно зберегла створений у конструкторі курс. Уся введена інформація була збережена, уроки що не були заповнені також були збережені як «шаблонні» для фіксації структури проєкту. Викладач бачить підтвердження створення курсу на відповідній сторінці з списком усіх створених курсів.
Статус	Пройдено.

Таблиця 7.5 — Тестування функції редагування курсу викладачем

Ціль	Перевірка функціоналу редагування існуючого курсу викладачем.

Передумова	– користувач авторизований у системі я к викладач; – викладач перейшов до сторінки зі списком курсу
Вхідні дані	Дані для зміни якогось елементу курсу (назва, дані для нового уроку, або для зміни існуючого).
Кроки виконання	Для редагування будь-якого елементу курсу: – натиснути на панелі керування на вкладку «Home Page»; – обрати курс, який підлягає редагуванню, із списку проєктів і натиснути на ньому кнопку «Edit»; – внести зміни (змінити назву або структуру курсу, модифікувати якийсь урок) – натиснути кнопку «Save Course», або «Save Lesson» у випадку зміни даних уроку (обидві кнопки викличуть перевірку на зміни та збережуть усі зміни які стосуються курсу); – перевірити відображення оновленої інформації у конструкторі.
Очікуваний результат	Система повідомляє про успішне збереження даних. Користувач бачить що структура курсу змінилась.
Фактичний результат	Система повідомила про успішне збереження даних. Користувач бачить що структура курсу змінилась. При повторному вході у конструктор з поточним курсом зміни не зникли.
Статус	Пройдено.

Таблиця 7.6 — Тестування функції запису студента на курс

Ціль	Перевірка функції запису студенту на обраний ним курс.
Передумова	– студент авторизований у системі;

	– у системі існують готові опубліковані курси, на які студент може піписатись.
Вхідні дані	
Кроки виконання	– натиснути на панелі керування на вкладку «Courses»; – обрати плитку курсу з списку курсів і натиснути на неї; – поглянути на деталі курсу та натиснути «Enroll».
Очікуваний результат	Студент отримав доступ до курсу та може бачити його у своїй теці на домашній сторінці.
Фактичний результат	Студент підписався на курс, та бачить його у власній теці.
Статус	Пройдено.

Таблиця 7.7 — Тестування функції публікації викладачем навчального курсу

Ціль	Перевірка функціоналу публікування курсу викладачем для доступу студентам.
Передумова	– викладач авторизований у системі; – у викладача є створений курс з статусом «не опубліковано»
Вхідні дані	Створений курс з статусом «не опубліковано»
Кроки виконання	– натиснути на панелі керування на вкладку «HomePage»; – обрати курс з списку курсів і натиснути на кнопку «Publish»; – переглянути статус курсу.
Очікуваний результат	Викладач оновив статус курсу, у списку курсів він має бачити мітку - «опубліковано» навпроти відповідного курсу.

Ціль	Перевірка функціоналу публікування курсу викладачем для доступу студентам.
Фактичний результат	Викладач оновив статус курсу, у списку курсів він має мітку - «опубліковано»
Статус	Пройдено.

Таблиця 7.8 — Тестування створення досягнення викладачем для завдання

Ціль	Перевірка інструментів створення досягнень викладачем для наявного курсу.
Передумова	– викладач авторизований у системі; – у викладача є створений курс або він у процесі створення;
Вхідні дані	Дані для створення досягнення.
Кроки виконання	– натиснути на панелі керування на вкладку «HomePage»; – обрати плитку курсу з списку курсів і натиснути на неї; – переглянути курс з його деталями та структурою; – натиснути кнопку «Achievements»; – у новому вікні натиснути «Create Achievement» – заповнити дані про досягнення, визначити об’єкт досягнення та правило здобуття;
Очікуваний результат	Викладач створив досягнення яке може побачити у панелі досягнень;
Фактичний результат	Викладач впевнився у створені досягнення через наявність його у списку досягнень курсу в панелі Achievements.

Ціль	Перевірка інструментів створення досягнень викладачем для наявного курсу.
Статус	Пройдено.

Таблиця 7.9 — Тестування адміністративних можливостей системи

Ціль	Перевірка доступу до даних користувачів адміністраторам, з можливість управління цими даними у випадку необхідності.
Передумова	– у платформи існують системні адміністратори; – вони зареєстровані як адміністратори у системі безпеки платформи через додаток KeyCloak;
Вхідні дані	Дані які треба змінити або переглянути про користувачів.
Кроки виконання	– адміністратор має адресу хостингу додатку KeyCloak, по ньому він має перейти та авторизуватись; – після чого він потрапить у realm (фрагмент запущеного додатку, це обумовлено тим що ми можемо використовувати цей застосунок одночасно для декількох сервісів чи систем) нашої платформи; – перед ним відкриється великий набір інструментів, йому потрібно обрати «Users» на панелі керування; – там йому відкриється список усіх користувачів, він може відкрити будь-якого з них та внести зміни до його даних, або ж, що дуже зручно і корисно, визначити виконання дії при вході користувача в систему: – «Підтвердити електронну пошту» надсилає користувачеві електронного листа для підтвердження адреси електронної пошти; – «Оновити профіль» вимагає від користувача ввести нову особисту інформацію;

Ціль	Перевірка доступу до даних користувачів адміністраторам, з можливість управління цими даними у випадку необхідності.
	– «Оновити пароль» вимагає від користувача ввести новий пароль; – «Налаштувати OTP» вимагає налаштування мобільного генератора паролів; – аналогічно адміністратор може перейти до вкладки «Sessions» і так само керувати активними сесіями користувачів, наприклад припинимо сесію користувача User1 (видаляючи її); – після виконаних операції, адміністратор має натиснути кнопку «Save» та зберегти зміни.
Очікуваний результат	Адміністратор після виключення сесії користувача User1. Може переконатись що його сесія відсутня у таблиці сесій.
Фактичний результат	Адміністратор переконався що сесія користувача User1 завершена.
Статус	Пройдено.

7.5 Матриця трасувань

Traceability-матриця, або матриця відповідності вимог наведена у таблиці 7.9 забезпечує відстеження відповідності між вимогами та виконаними тестами, що гарантує повне покриття функціональності системи. Оскільки система на даний момент має лише одну версію матриця подана як таблиця ключ-значення.

Таблиця 7.10 — Матриця відповідності вимог

Ідентифікатор вимоги	Назва вимоги	Статус задоволення вимоги
FR-1	Управління користувачами	Пройдено

Ідентифікатор вимоги	Назва вимоги	Статус задоволення вимоги
FR-1.1	Реєстрація користувачів	Пройдено
FR-1.2	Аутентифікація користувачів	Пройдено
FR-1.3	Ролі та права доступу користувачів	Пройдено
FR-1.4	Редагування профілю користувача	Пройдено
FR-2	Управління курсами	Пройдено
FR-2.1	Створення курсів	Пройдено
FR-2.2	Редагування курсів	Пройдено
FR-2.3	Видалення курсів	Пройдено
FR-2.4	Додавання навчальних матеріалів	Пройдено
FR-2.5	Конструювання адаптивних курсів	Пройдено
FR-3	Навчальний процес	Пройдено
FR-3.1	Проходження курсів	Пройдено
FR-3.2	Тестування та перевірка знань	Пройдено
FR-3.3	Система прогресу	Пройдено
FR-4	Адміністративні функції	Пройдено
FR-4.1	Моніторинг активності користувачів	Пройдено
FR-4.2	Управління ролями та правами доступу	Пройдено
FR-5	Керування контентом	Пройдено

Ідентифікатор вимоги	Назва вимоги	Статус задоволення вимоги
FR-5.1	Додавання контенту	Пройдено
FR-5.2	Редагування контенту	Пройдено
FR-5.3	Видалення контенту	Пройдено

Висновки до розділу 7

У даному розділі було наведено опис процесу тестування функціоналу та модулів освітньої платформи. В рамках модульного тестування було застосовано Spring Boot Test та Postman.

За допомогою Spring Boot Test ми змогли оцінити роботу модулів, що відповідають за збереження та обробку даних. При чому застосовувалось мокування даних та функцій для ізоляції результатів тестування від робочих даних. Додатково був використаний Postman для тестування RESTful API ендпойнтів, що оброблюються контролерами, для перевірки їх коректного реагування на дані, що передавалися у запитах.

Модульні тести дозволили перевірити коректність роботи основних елементів бекенду: сервісів, контролерів та репозиторіїв. Покриття коду тестами склало: 85% від загальної кількості класів, 80% від загальної кількості методів та 75% від загальної кількості програмного коду. Це можна вважати відмінним результатом враховуючи, що приблизно 15% коду є конфігурацією для роботи фреймворку і не підлягають тестуванню так як не впливають на бізнес-логіку продукту.

Для перевірки відповідності функціональним вимогам, встановленим у розділі 3, було розроблено ряд тест-кейсів, які допомогли систематично оцінити роботу системи в різних сценаріях використання. Також була створена матриця відповідності вимогам (Traceability Matrix), яка забезпечує відстеження

відповідності між функціональними вимогами та тест-кейсами. Це дозволило переконатися, що всі вимоги були повністю покриті тестами та опрацьовані.

Результати випробувань підтвердили, що освітня платформа ефективно виконує всі функціональні можливості, які від неї очікувалися. Додатково, після проведення набору тест-кейсів, вони були встановлені у статус "Пройдено", що забезпечує надійність та стабільність системи, її відповідність потребам користувачів.

8 СТАРТАП ПРОЄКТ

8.1 Опис ідеї проєкту

Таблиця 8.1 — Опис ідеї стартап-проекту

<table>
<tr><th>Зміст ідеї</th><th>Напрямки застосування</th><th>Вигоди для користувача</th></tr>
<tr><td rowspan="2">Розробка інтуїтивно зрозумілого конструктора курсів, що дозволяє викладачам створювати та керувати навчальним контентом без необхідності програмування.
Інтеграція універсальних технік гейміфікації, що дозволить їх використовувати у будь-якому навчальному курсі
Розробка платформи під конструктор курсів та визначені техніки гейміфікації для якісною взаємодії користувачів з визначеними елементами.</td><td>Приватні заклади освіти</td><td>Економія часу та ресурсів викладача;
Широкий функціонал, який дозволяє варіативно спроектувати курс.
Адаптувати курси під домашні завдання які легше та цікавіше виконувати учням;</td></tr>
<tr><td>Онлайн-школи,
Самоосвіта,
Корпоративні тренінги</td><td>Спрощення процесу створення курсів викладачами;
Підвищена зацікавленість студентів у навчальному процесі.
Покращений процес навчання за рахунок системи нагород та можливості вибору стратегії навчання студентом.</td></tr>
</table>

Таблиця 8.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Мій проект	Конкурент1 Coursera	Конкурент2 Udemy	Конкурент3 Duolingo			
1.	Інтуїтивний конструктор курсів з графічним інтерфейсом	Реалізований зручний конструктор курсів з підтримкою гейміфікації та гнучкості у створенні довільних структур навчання	Складний інструментарій, який застосовується виключно за констультацією працівників платформи та з їх допомогою. Не має опціональності та розвинутих технік гейміфікації	Розширенийфункціонал, можливість створювати навчальний курс без допомоги з платформи, відсутність гейміфікації з боку навчального курсу.	Вузька доступність виключно команди розробникуів. Залежність від програмування. Вузьконаправленість створених курсів.			✓
2.	Інтеграція гейміфікаційних елементів	Підтримка графічного представлення у вигляді маршрутів, опціональність завдань, інтерактивність з можливістю отримання винагород у вигляді бейджів за різноманітні завдання або дії	Відсутня, якщо не вважати сертифікати як нагороди за досягненняу навчанні.	Часткова підтримка: платформа надає власний набір досягнень а нагород у вигляді бейджів за них, що спонукає до відвідування і навчання.	Повна підтримка різного виду гейміфікацій, від серії відвідувань, до досягнень за швидкість та якість виконнання завдань, інтерактивні марафони, тощо.			✓
3.	Можливість кастомізації без програмування	Незалежність від програмування.	Відносна незалежність від програмування.	Відносна незалежність від програмування.	Залежність від програмування.			✓

4.	Адаптивний дизайн для різних пристроїв	Система працює у всіх доступних для комп'ютерів та лептопів форматах. Викладацька частина - конструктор не може бути використаний через телефонні присторії. Студентська складова може працювати на будь-яких пристороях.	Система працює у всіх доступних для комп'ютерів та лептопів форматах. Студентська частина доступна на всіх персональних присторях: планшет, смартфон, комп'ютер,лептоп.	Система працює у всіх доступних для комп'ютерів та лептопів форматах. Студентська частина доступна на всіх персональних присторях: планшет, смартфон, комп'ютер,лептоп.	Система працює у всіх доступних для комп'ютерів та лептопів форматах. Студентська частина доступна на всіх персональних присторях: планшет, смартфон, комп'ютер,лептоп.		✓	
5.	Підтримка мультимедійного контенту	Висока	Висока	Висока	Висока		✓	

Пояснення:

а) сильні сторони (S):

1) інтуїтивний та простий у використанні графічний конструктор;

2) можливість кастомізації без необхідності програмування;

б) нейтральні сторони (N):

1) адаптивний дизайн та підтримка мультимедіа на рівні з конкурентами;

2) підтримка різного виду мультимедійного контенту.

8.2 Технологічний аудит ідеї проєкту

Таблиця 8.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розробка освітньої платформи	Backend: Spring Boot (Java), RESTful API Frontend: React.js, HTML5, CSS3 База даних: MySQL Хмарні сервіси: AWS Безпека: KeyCloak	Наявні, широко використовується	Доступна, є спеціалісти з досвідом роботи
2	Інтеграція модуля досягнень	Backend: Spring Boot (Java), RESTful API Frontend: React.js, HTML5, CSS3 База даних: MySQL	Наявні, широко використовується	Доступна, є спеціалісти з досвідом роботи
3	Розробка графічного конструктору курсів з підтримкою елементів гейміфікації	Backend: Spring Boot (Java), RESTful API Frontend: React.js, ReactFlow	Наявні бібліотеки які виконують потрібні функції, потребується розробки власного рішення в майбутньому	Доступна для команди технологія. У подальшому планується перехід на власне рішення.

Обрана технологія реалізації ідеї проєкту:

– використання Spring Boot для бекенду дасть нам широкі можливості у реалізації якісного RESTfullAPI, з забезпеченням високої швидкодії та надійності, які Spring закріпив за собою як найбільш популярний фреймворк великих Enterprise проєктів;

– React.js для фронтенду забезпечує швидку та ефективну розробку. Додатково для легкої інтеграції усіх ідей графічного інтерфейсу для конструктора курсів було використано бібліотеку ReactFlow, вона проста використанні, але доволі обмежена у функціоналі. Для реалізації ідей її цілком достатньо, але для покращення надійності та швидкодії у подальшому планується перехід на власне рішення на базі TypScript;

– база даних MySQL гарантує надійне збереження даних і легку інтеграцію з фреймворком Spring. Надійнсть та безпеку даних забезпечує загально доступний застосунок з відкритою кодовою базою – KeyCloak, його інтеграція дозволяє швидко та надійно реалізувати аутентифікацію, реєстрацію, контроль сесій та доступ до даних;

– інструменти тестування (JUnit, SpringTest) дозволяють забезпечити якість продукту.

Висновок щодо можливості технологічної реалізації проекту: так, проєкт може бути технологічно реалізований за допомогою доступних технологій, які є наявними на ринку та доступні команді розробників. Щоправда, у подальшому вбачається потреба в заміні бібліотек що застосовуються у графічній складовій конструктору курсів на власне рішення. Це викликано тим що ReactFlow, надає зручний інструментарій для створення графоподібних структур, але він обмежений у своєму функціоналі по створенню інших елементів, які могли б знадобитись при розвитку проєкту. Тому в подальшому буде ця бібліотека буде замінена власним рішенням на основі TypeScript та React. Такий стек заснований на графічному дизайнері Figma, який надає широкий спектр можливостей при використанні базових технологій.

8.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 8.4 — Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	10-15 основних платформ
2	Загальний капіталізація, $	Близько $185,2 млрд
3	Динаміка ринку (якісна оцінка)	Зростає завдяки популярності дистанційного навчання
4	Наявність обмежень для входу	Висока конкуренція на глобальному ринку з боку мегакорпорацій, необхідність інновацій, адаптацій до локальних ринків
5	Специфічні вимоги до стандартизації та сертифікації	Відповідність стандартам безпеки даних, захисту персональної інформації
6	Середня норма рентабельності в галузі, %	~15%

Глобальний ринок онлайн-освіти можна вважати привабливим, адже демонструє стійке зростання. Згідно з прогнозами Statista, у 2024 році обсяг ринку досягне $185,2 млрд, з очікуваним середньорічним темпом зростання (CAGR) 8,62%, що призведе до обсягу $257,7 млрд до 2028 року [26].

Таблиця 8.5 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Необхідність створення онлайн-курсів без складнощів	Викладачі навчальних закладів Онлайн-репетитори Корпоративні тренери	Відсутність технічних навичок Потреба в швидкому та зручному розгортанні курсів	Інтуїтивний інтерфейс Можливість кастомізації Надійність
2	Мотивація студентів до навчання	Особи які шукають нові методи для вивчення потрібної їм інформації	Бажання інтерактивності Потреба у заохочувальному процесі	Інтерактивні завдання, цікавий процес вивчення матеріалу, мобільний доступ до сервісу

Таблиця 8.6 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкуренція	Ризик втрати частки ринку, через високу конкуренцію	Розробка унікальних функцій, фокус на зручний UX/UI
2	Нестабільність економічного середовища	Можливе зниження попиту через економічні труднощі	Оптимізація витрат, розробка локальної цінової політики
3	Правові обмеження	Нові регуляції щодо захисту даних	Відповідність стандартам, юридичні консультації

Таблиця 8.7 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання попиту на онлайн-освіту	Розширення ринку, більше потенційних клієнтів	Активний маркетинг, розширення функціоналу
2	Партнерства з освітніми закладами	Можливість швидкого масштабування на навчальні заклади, розширення сфери послуг	Створення партнерських програм
3	Інтеграція з новими популярними технологіями	Додавання AI-функцій, для персоналізації навчання	Інвестування в інновації, співпраця з великими компаніями розробниками великих LLM систем

Таблиця 8.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Тип конкуренції	Монополістична конкуренція	Необхідність виділятися через унікальні пропозиції
За рівнем конкурентної боротьби	Глобальний ринок	Можливість виходу на міжнародні ринки

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
За галузевою ознакою	Внутрішньогалузева конкуренція	Фокус на специфічних потребах освітнього сектору
Конкуренція за видами товарів	Товарно-видова	Потреба в диференціації продукту
За характером конкурентних переваг	Нецінова конкуренція	Інвестування в якість, інновації та сервіс
За інтенсивністю	Марочна конкуренція	Побудова сильного бренду та лояльності клієнтів

Таблиця 8.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Опис
Прямі конкуренти	Coursera, Udemy, KhanAcademy, Duolingo
Потенційні конкуренти	Нові стартапи в сфері EdTech, технологічні гіганти, які можуть увійти в ринок. Moodle, у випадку виходу на ринок навчальних закладів.
Постачальники	Постачальники хмарних сервісів, технологічних рішень
Клієнти	Самоосвіта, освітні заклади, корпоративні клієнти, індивідуальні викладачі.
Товари-замінники	Традиційне офлайн-навчання, інші платформи з фокусом на різні технології.

Висновки до таблиці 3.9:

– інтенсивність конкуренції: Висока, але є можливості для входу через інновації;

– бар'єри входження: Вимоги до технологій та інвестицій, але вони подоланні;

– сила постачальників: Низька, багато альтернативних технологічних рішень;

– сила споживачів: Середня, клієнти шукають кращі рішення за оптимальною ціною;

– загроза замінників: Середня, але тренд на онлайн-навчання посилюється.

Таблиця 8.10 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Інтуїтивний графічний інтерфейс	Споживачі цінують простоту використання без потреби в навчанні
2	Гнучкість конструктора курсів	Можливість адаптування різноманітних матеріалів у формат курсів, незалежно від їх тематики, дає змогу залучити більше користувачів.
3	Впровадження елементів гейміфікації	Мотивація студентів через гейміфікаційні елементи, такі як бали, нагороди та досягнення, покращує навчальний процес.
4	Інтеграція повного циклу навчання	Забезпечення всіх етапів від створення курсу до використання студентами в освітніх цілях

Таблиця 8.11 — Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим проєктом						
			–3	–2	–1	0	+1	+2	+3
1	Інтуїтивно зрозумілий інтерфейс	14			✓				
2	Гнучкість конструктора курсів	20	✓						
3	Впровадження елементів гейміфікації	16		✓					
4	Мобільна доступність платформ	12					✓		
5	Безпека та конфіденційність даних	18				✓			
6	Інтеграція повного циклу навчання	16				✓			

Таблиця 8.12 — SWOT-аналіз

Сильні сторони:	Слабкі сторони:
Інтуїтивно зрозумілий інтерфейс; Гнучкість у створенні курсів; Зручність та простота у використанні; Широка доступність користувача.	Обмежена кількість варіацій шаблонів для курсів доступних на старті; Впізнаваність бренду відсутня, або слабка на момент старту; Висока конкуренція в освітньому секторі.

Можливості:	Загрози:
Розширення платформи для різних типів навчання; Впровадження нових технологій зі штучним інтелектом;.	Зниження інтересу до онлайн-освіти через насиченість ринку; Можлива поява потужних конкурентів.

Таблиця 8.13 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Запуск пілотного проекту як освітньої онлайн платформи або партнерської програми для закладу	Висока	9 місяців
2	Масштабування на ринок навчальних закладів, партнерські програми	Середня	1 рік
3	Глобальний запуск платформи	Середня	1.5 роки

Для нас найбільш актуальним є спроба запуску пілотного проекту з партнерськими, скоріше за все, приватними освітніми закладами, яким би могли запропонувати інтегрувати систему у процес домашніх завдань, надаючи учням можливість проходити домашні завдання через платформу, закріплюючи матеріал та отримуючи різні винагороди, які викладачі самі зможуть налаштувати. Це дозволить отримати зворотний зв'язок, вдосконалити продукт та закріпитися на ринку.

8.4 Розроблення ринкової стратегії проекту

8.4.1 Визначення стратегії охоплення ринку

Таблиця 8.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи	Готовність сприйняти продукт	Орієнтовний попит	Інтенсивність конкуренції	Простота входу
1	Індивідуальні викладачі, тренери та репетитори	Висока	Високий	Висока	Висока
2	Приватні освітні установи	Висока	Середній	Висока	Середня
3	Вищі та середні навчальні заклади	Середня	Низький	Низька	Середня
4	Особи, які шукають альтернативні методи навчання	Висока	Високий	Висока	Високий
4	Корпоративні установи	Середня	Високий	Висока	Середня

Охоплення ринку буде частковим і здійсненим за стратегією концентрованого маркетингу. Фокус буде зосереджено на трьох сегментах із найбільшою готовністю сприйняти продукт, а саме:

– індивідуальні викладачі: сегмент із високою готовністю до впровадження нових технологій, зацікавлені у нових продуктах, які здатні надати кращі умови для розміщення та розповсюдження їх навчальних матеріалів. Перевагою стане

зручний конструктор курсів, та подібний до гри підхід до навчання. Це зацікавить цільову аудиторію та спонукатиме їх до користування нашою платформою;

– приватні освітні установи: цей сегмент демонструє високу готовність до сприйняття інноваційних освітніх рішень, особливо якщо вони сприяють покращенню якості навчального процесу та дозволяють виділитися серед конкурентів. Приватні освітні установи зацікавлені у платформах, які пропонують інтеграцію з існуючими системами управління навчанням та можливість створення адаптивних програм (можливо переведенням у інтерактивний курс частини програми та певного виду робіт, наприклад, домашніх завдань);

– особи, які шукають альтернативні методи навчання: група, що активно цікавиться інноваційними підходами, не сприймають, або негативно ставляться до строгих академічних методів, надають перевагу гнучкими формами навчання. Цей сегмент демонструє високу готовність до впровадження нових рішень, оскільки студенти прагнуть отримати доступ до навчальних матеріалів, що відповідають їхнім індивідуальним потребам і надають можливість вивчати цей матеріал у не звичний спосіб та з певною опціональністю, яка, щоправда, залежить у більшій мірі від викладача. Застосування ігрових елементів, мобільність і зручний інтерфейс стануть ключовими перевагами платформи для цієї групи.

8.4.2 Визначення базової стратегії розвитку

Таблиця 8.15 — Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Обрана альтернатива розвитку	Стратегія охоплення ринку	Ключові конкурентні позиції	Базова стратегія розвитку
1	Створення освітньої платформи з елементами гейміфікації.	Концентрований маркетинг, що зосереджений на індивідуальних викладачах, студентах, та приватних установах, яким така платформа могла б надати цікаві їм послуги	Зручний конструктор курсів з інтуїтивним інтерфейсом та з можливістю використання геміфікації; Інтерактивність при навчанні та ігрові механіки, що передбачають дослідження та винагороди за успіхи	Стратегія диференціації: запропонувати унікальний досвід навчання за допомогою елементів гейміфікації та надання простого та функціонального інструментарію інструментів для викладачів.

Таблиця 8.16 — Визначення стратегії конкурентної поведінки

№ п/п	Питання	Відповідь
1	Чи є проект «першопрохідцем» на ринку?	Ні, але пропонує унікальні рішення
2	Чи буде компанія шукати нових споживачів або забирати існуючих у конкурентів?	Обидва варіанти
3	Чи буде компанія копіювати основні характеристики товару конкурента?	Ні, фокус на унікальних перевагах конструктору курсів та інтеграцій його з платформою
4	Стратегія конкурентної поведінки	Стратегія лідеру – наступу, через набір унікальних рішень та характеристик, можливість їх інтеграції у велику кількість цільових груп, доцільною стратегією буде лідерська-наступальна, що означає масштабування продукту, адаптація та застосування нових технологічних рішень (розширення теки гейміфікації, впровадження штучного інтелекту), за рахунок яких буде розширюватись наша цільова аудиторія

Таблиця 8.17 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентні позиції	Вибір асоціацій
1	Зручність використання для користувачів різного віку та рівня підготовки.	Стратегія диференціації	Інтуїтивний інтерфейс, кастомізація, інновації	Простота використання Гнучкість Інтуїтивність
2	Інтерактивність, зручність для самонавчання	Стратегія спеціалізації	Ігровий підхід до навчання, винагорода за успіхи	Захопливість Ефективність Дослідження
3	Інструменти для роботи з курсами та організація навчального простору в середині курсів	Стратегія диференціації	Зручний графічний конструктор курсів, підтримка мультимедіа та елементів гейміфікації.	Креативність Гнучкість Персоналізації

8.5 Розроблення маркетингової програми стартап-проекту

8.5.1 Формування маркетингової концепції товару

Таблиця 8.18 — Формування маркетингової концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Просте та багатофункціональне створення онлайн-курсів	Економія часу, відсутність потреби в програмуванні, при можливості створення курсу з широким набором можливостей.	Інтуїтивний конструктор курсів, з функціоналом для створення нагород, зручного визначення правил зарахування нагород, визначення тестів, завдань, лекцій, відео.
2	Потреба у платформі, яка реалізує усі можливості курсів створених спеціальним конструктором	Можливість легко публікувати створені курси та підписуватись на них студентам і використовувати усі можливості, які визначив викладач.	Застосунок з розпіденням за ролями, який ефективно керує даними які створені у конструкторі курсів, для відображення та взаємодії з клієнтами

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
3	Постійна підтримка та інтеграція	Постійне покращення функціоналу, введення нових технік гейміфікації. Подальше інтегрування штучного помічника-консультанта який підніме процес навчання на новий рівень.	Активне ком'юніті команди розробників, яке буде підтримувати проєкт як на рівні комунікації з спільнотою, так і на технічному рівні.

8.5.2 Розробка трирівневої маркетингової моделі товару

Таблиця 8.19 — Опис трьох рівнів моделі товару

<table>
<tr><th>Рівні товару</th><th colspan="3">Сутність та складові</th></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Платформа для онлайн-навчання надає наступні можливості:
– зручний графічний конструктор курсів, який дозволить швидко створити курс з нелінійною структурою та великою кількістю додаткових можливостей по побудові складних навчальних структур, модулів, створенню нагород та цікавих завдань для їх досягнення;
– інтерактивна платформа для студентів з можливістю проходження курсів у зручному для учнів форматі, колекціонування нагород за дослідництво та подолання складностей.</td></tr>
<tr><td rowspan="8">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Інтуїтивно зрозумілий інтерфейс для створення курсів і навчальних матеріалів.</td><td>М</td><td>Тх</td></tr>
<tr><td>. Мобільний додаток для доступу до навчальних матеріалів.</td><td>Нм</td><td>Вр</td></tr>
<tr><td>Висока швидкість доступу до контенту.</td><td>М</td><td>Тх</td></tr>
<tr><td>Інтерактивність курсу через гейміфікацію.</td><td>Нм</td><td>Тл</td></tr>
<tr><td colspan="3">Якість: відповідає сучасним стандартам програмного забезпечення. Має карту тестувань.</td></tr>
<tr><td colspan="3">Пакування: доступний через сайт платформи, що підтримує кросплатформу.</td></tr>
<tr><td colspan="3">Марка: LearnLegueLab</td></tr>
</table>

Рівні товару	Сутність та складові
III. Товар із підкріпленням	До продажу: Доступна сама платформа та конструктор, продаються курси, які створюються зареєстрованими викладачами. Прибуток відсотково розподіляється.
	Після продажу: Підтримка користувачів, регулярні оновлення платформи, нові функції, технічна підтримка.

Захист від копіювання буде здійснений за рахунок декларування авторських прав на ідею гейміфікованого графічного конструктора курсів та створення торгової марки за обраною назвою LearnLegueLab.

8.5.3 Визначення меж встановлення ціни

Таблиця 8.20 — Визначення цінових меж

№ п/п	Категорія	Рівень цін на товари-замінники (Приватні курси)	Рівень цін на товари-аналоги (Онлайн-платформи)	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Місячна Підписка на сервіс		60 USD в місяць	400 USD в місяць і вище. Якщо коропоративний доступ, то оцінюється дохідність компанії вище 8000 USD на місяць	30 USD в місяць
2	Купівля курсів	500 USD. за курс	150 USD. за курс		120 USD за курс
3	Корпора-тивний доступ		300 USD в місяць за користувача		100 USD в місяць за користувача

8.5.4 Визначення оптимальної системи збуту

Таблиця 8.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки	Функції збуту постачальника	Глибина каналу збуту	Оптимальна система збуту
1	Онлайн пошук потенційними клієнтами.	Пряма взаємодія.	Нульовий рівень (пряма продажа)	Власна система збуту через онлайн-платформу
2	Рекомендації партнерів, освітніх закладів	Співпраця з партнерами, адаптація продукту до їхніх потреб	Локальний рівень (партнерські програми)	Партнерські програми та інтеграції з освітніми установами
3	Участь у галузевих заходах (виставки, конференції)	Презентація продукту, демонстрації	Локальний рівень (партнерські програми)	Участь у виставках, конференціях та співпраця з організаторами
4	Рекомендації через професійні спільноти та форуми	Онлайн-презентації, вебінари	Глобальний рівень (партнерські програми світового масштабу)	Використання вебінарів та онлайн-презентацій для залучення клієнтів

№ п/п	Специфіка закупівельної поведінки	Функції збуту постачальника	Глибина каналу збуту	Оптимальна система збуту
5	Активний маркетинг у соціальних мережах та SEO	Реклама, контент-маркетинг	Нульовий рівень (пряма продажа)	Використання соціальних мереж, блогів та SEO для залучення трафіку

8.5.5 Концепція маркетингових комунікацій

Таблиця 8.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки клієнтів	Канали комунікацій	Ключові позиції для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Активне використання інтернету та соцмереж	Соціальні мережі, освітні форуми, вебінари	Простота, гнучкість, сучасність	Привернути увагу, показати переваги продукту	Наш слоган: «Play Course, Learns Fast, Get new skills»
2	Пошук ефективних освітніх рішень.	Email-розсилки, SEO, контент-маркетинг	Інтуїтивний інтерфейс, кастомізація, кросплатформеність	Підкреслити унікальні функції та переваги платформи	«Новий досвід у навчанні - грайте та вчіться з LearnLegueLab»

№ п/п	Специфіка поведінки клієнтів	Канали комунікацій	Ключові позиції для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
3	Орієнтація на професійний розвиток та навчання	Онлайн-реклама, партнерські програми	Інновації, інтеграція	Підвищити обізнаність про бренд та залучити нових клієнтів	«LearnLegueLab – досліджуй, навчайся – набувай нових навичків»

Висновки до розділу 8

Проведений аналіз показав, що ринок онлайн-освіти є привабливим для запуску нашого стартап-проекту. Існує зростаючий попит на інтуїтивні, гнучкі та кросплатформенні платформи для створення та управління навчальними курсами. Наша ідея має сильні конкурентні переваги, такі як інтуїтивний графічний інтерфейс, висока гнучкість та кастомізація, інтеграція повного циклу навчання, а також підтримка кросплатформеності, що забезпечує доступність на різних пристроях.

Технологічний аудит підтвердив можливість реалізації проекту за допомогою доступних та відомих технологій, таких як Spring Boot для бекенду та React.js для фронтенду. Ці технології забезпечують стабільність, продуктивність та масштабованість платформи. Аналіз ринкового середовища виявив як можливості, такі як зростання онлайн-освіти та партнерства з освітніми закладами, так і загрози, включаючи високу конкуренцію та швидкі технологічні зміни. Запропоновані стратегії дозволяють ефективно врахувати ці фактори та забезпечити успішне впровадження проекту на ринок.

Обрана стратегія концентрованого маркетингу з фокусом на групи приватних викладачів та студентів які тяжіють до онлайн-навчання, а також на групу

приватних шкіл є граною можливістю для швидкого виходу на ринок та створення аудиторії. Розроблена маркетингова програма спрямована на ефективне позиціонування продукту, залучення клієнтів через канали, якими вони активно користуються, та підвищення обізнаності про бренд через різні комунікаційні інструменти.

Подальша імплементація проекту є доцільною, оскільки він має потенціал стати конкурентоспроможним гравцем на ринку онлайн-освіти, задовольняючи потреби споживачів та пропонуючи унікальні переваги. Високий рівень покриття коду тестами, успішне проходження тест-кейсів та використання матриці трасування свідчать про якість розробленого програмного забезпечення та його відповідність встановленим вимогам. Це є запорукою успішного впровадження та подальшого розвитку нашої освітньої платформи.

ВИСНОВКИ

Дана дисертація присвячена проєктуванню та розробці універсальної освітньої платформи з імплементацією гейміфікованих технік. Розроблена платформа спрямована на спрощення та підвищення ефективності онлайн-навчального процесу, забезпечуючи ефективне створення та публікацію курсів, а також впровадження гейміфікацій, які підвищують мотивацію та продуктивність студентів.

Основною метою розробки було створення прототипу освітньої платформи, яка б задовольняла визначені функціональні вимоги, оптимізацію та вдосконалення ключових етапів навчального процесу, таких як створення курсів, управління уроками, а також імплементування універсальних та найбільш вдалих практик гейміфікації.

У рамках проектування було здійснено аналіз предметної області, визначено процеси діяльності акторів системи: викладачів та студентів, окреслено ключові функції платформи, які сприяють освітньому процесу. Це дало змогу сформулювати цілі й задачі, які спрямовані на дослідження можливостей гейміфікації та створення робочого прототипу платформи з інтегрованим у нього конструктором, що забезпечує повний та інтуїтивний інструментарій створення навчальних курсів, та застосуванням визначених технік гейміфікації, які мають покращити досвід усіх учасників платформи, як викладачів, так і студентів.

Детальний аналіз найбільших освітніх платформ, таких як Khan Academy, Duolingo, Coursera, Udemy та edX, дозволив виявити різноманітні підходи до оптимізації навчального процесу та впровадження гейміфікації. Було визначено, що незважаючи на подібність у наданні навчальних матеріалів через курси з тестовими та письмовими завданнями, а також через відео, аудіо та текстові формати, платформи суттєво різняться у відкритості до викладачів та підходах до конструювання курсів. Наприклад, Khan Academy та Duolingo активно інтегрують гейміфікаційні елементи, які значно підвищують мотивацію користувачів через нагороди, рівні та інтерактивні завдання. Водночас, академічно орієнтовані

сервіси, такі як Coursera та edX, зосереджуються на офіційній сертифікації та високоякісному контенті, що накладає певні обмеження на інструкторів у створенні курсів. Udemy займає проміжне становище, пропонуючи широкий вибір практично орієнтованих курсів без значного акценту на гейміфікації, проте відрізняється доступністю та гнучкістю, що приваблює студентів із різними потребами.

На основі проведеного аналізу було визначено основні критерії для побудови ефективної освітньої платформи: забезпечення доступності для викладачів, що дозволить незалежним інструкторам легко створювати курси без зайвих бар'єрів, інтеграція гейміфікаційних елементів для підвищення мотивації користувачів, а також наявність зручного конструктора, який покращить загальний досвід викладачів та позитивно вплине на якість курсів. Інтеграція системи нагород, таких як значки, рівні та прогрес-бари, а також можливість створення курсів довільної тематики з гнучким форматом навчання, стане основою для підвищення зацікавленості як студентів, так і викладачів, забезпечуючи захоплюючий, ефективний і зручний навчальний процес для всіх учасників платформи.

Технологічний стек, обраний для реалізації платформи, включав Java з використанням Spring Boot для бекенду та React.js для фронтенду. Ці технології були обрані за їхню надійність, масштабованість та підтримку великою спільнотою розробників. Використання Spring Security у поєднанні з Keycloak забезпечило високий рівень безпеки та ефективне управління доступом користувачів. Контейнеризація додатків за допомогою Docker забезпечила портативність та ізоляцію середовищ розробки, тестування та розгортання, що сприяло стабільності та відтворюваності системи. Вибір реляційної СУБД MySQL був обґрунтований її високою продуктивністю, простотою використання та можливістю масштабування, що відповідає потребам платформи у збереженні структурованих даних та підтримці транзакцій.

Архітектура інформаційної системи була реалізована за принципами MVC, що сприяло розподілу обов’язків між компонентами та спрощенню процесу розробки та підтримки системи. Детальні діаграми класів, компонентів та послідовностей дій користувачів дозволили глибше зрозуміти внутрішню логіку

системи та ефективно моделювати її функціональні можливості. Конструктор курсів, розроблений у рамках платформи, забезпечує викладачам інтуїтивно зрозумілий та ефективний інструмент для створення та управління навчальними курсами. Мінімалістичний графічний інтерфейс, розділений на робочий простір та панель керування, дозволяє користувачам зосередитися на створенні якісного навчального контенту. Інтерактивні елементи, такі як завдання різного формату та досягнення, а також можливість налаштування зв'язків між уроками за допомогою вагових коефіцієнтів, забезпечують гнучкість та адаптивність навчального процесу. Аналогічним чином студенти взаємодіють з курсами через модифіковану графічну складову конструктора, яка дозволяє виконувати завдання, отримувати досягнення та переглядати структуру курсу.

Математичне забезпечення системи, зокрема використання алгоритму оптимізації на основі методу «гілок та меж», дозволило ефективно планувати навчальний процес студентам, мінімізуючи час виконання завдань та адаптуючись до змінних умов роботи користувачів. Це сприяло підвищенню ефективності управління проектами та забезпеченню своєчасної реалізації навчальних програм.

Процес тестування системи, що включав модульне тестування за допомогою Spring Boot Test та тестування RESTful API за допомогою Postman, дозволив ретельно перевірити функціональність окремих компонентів платформи. Використання Spring Boot Test забезпечило можливість ізольованого тестування сервісів, контролерів та репозиторіїв, що сприяло виявленню та усуненню потенційних помилок на ранніх стадіях розробки. Postman дозволив ефективно перевірити коректність роботи ендпойнтів, забезпечуючи відповідність відповідей сервера очікуваним результатам при різних сценаріях використання. Розроблені тест-кейси включали підготовку тестового середовища, визначення передумов, вхідних даних, послідовності кроків виконання тестів, а також очікуваних та фактичних результатів. Результати тестування підтвердили високий рівень якості коду, а покриття тестами склало 85% для класів, 80% для методів та 75% загального обсягу коду, що свідчить про високий рівень перевірки основних компонентів системи та їх взаємодії.

Проведений аналіз ринку онлайн-освіти підтвердив високу актуальність розробленої платформи, оскільки існує зростаючий попит на інтуїтивні, гнучкі та кросплатформенні рішення для створення та управління навчальними курсами. Вибір стратегії концентрованого маркетингу з фокусом на групи приватних викладачів, студентів, які тяжіють до онлайн-навчання, та приватних шкіл, дозволяє швидко виходити на ринок та створювати аудиторію. Розроблена маркетингова програма спрямована на ефективне позиціонування продукту, залучення клієнтів через активні канали комунікації та підвищення обізнаності про бренд через різні інструменти просування.

Загалом, мета дослідження була досягнута. Завдяки ретельному аналізу аналогів було визначено вдалі практики гейміфікації. Проєктування прототипу системи було виконано з урахуванням усіх функціональних вимог, що дало успішно впровадити елементи гейміфікації та графову систему структури курсу до конструктора. В свою чергу конструктор став опорою платформи, яка створена для взаємодії викладачів з конструктором, а студентів з створеними курсам, що також означає інтеграцію гейміфікацій для взаємодії студентів з усіма можливостями, які викладач може створити у конструкторі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Khan Academy — About [Електронний ресурс]. URL: https://www.khanacademy.org/about

2. Khan Academy — What Types of Content Can I Assign to My Students? [Електронний ресурс]. URL: https://support.khanacademy.org/hc/en-us/articles/18564282990861-What-types-of-content-can-I-assign-to-my-students

3. Khan Academy — Badges [Електронний ресурс]. URL: https://www.khanacademy.org/badges

4. Duolingo — About [Електронний ресурс]. URL: https://uk.duolingo.com/info

5. Business Model Analyst — Duolingo Business Model [Електронний ресурс]. URL: https://businessmodelanalyst.com/duolingo-business-model/?srsltid=AfmBOopLCXEuxQur09LbA0yyqwjBipRNKKMKsMgRLimMCFWp2cHEeZYP

6. Coursera — About Us [Електронний ресурс]. URL: https://about.coursera.org/

7. Coursera — Coursera Plus [Електронний ресурс]. URL: https://www.coursera.org/courseraplus

8. Udemy — About Us [Електронний ресурс]. URL: https://about.udemy.com

9. InfoStride — Udemy Business and Revenue Model [Електронний ресурс]. URL: https://infostride.com/udemy-business-model

10. edX — About Us [Електронний ресурс]. URL: https://www.edx.org/about-us

11. SOLID: The First 5 Principles of Object Oriented Design [Електронний ресурс]. URL: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-fiveprinciples-of-object-oriented-design

12. React Team — React documentation hub [Електронний ресурс]. URL: https://reactjs.org/

13. Spring Framework Documentation, електронна документація. URL: https://docs.spring.io/spring-framework/reference/overview.html

14. Project Lombok — електронна документація. URL: https://projectlombok.org

15. MySQL Documentation, електронна документація. URL: https://dev.mysql.com/doc

16. PostgreSQL — Офіційна документаці, електронна документація. URL: https://www.postgresql.org/docs/current

17. Keycloak Documentation, електронний документація. URL: https://www.keycloak.org/documentation

18. Docker Documentation, електронна документація. URL: https://docs.docker.com/reference/

19. Xu, A. System Design Interview: An Insider's Guide. 2nd ed., 2020. 322 с.

20. Zourmpakis A.I., Kalogiannakis M., Papadakis S. Adaptive Gamification in Science Education: An Analysis of the Impact of Implementation and Adapted Game Elements on Students' Motivation. Computers. URL: https://www.researchgate.net/publication/371868959_Adaptive_Gamification_in_Science_Education_An_Analysis_of_the_Impact_of_Implementation_and_Adapted_Game_Elements_on_Students'_Motivation

21. Landers R.N., Auer E.M., Collmus A.B., Armstrong M.B. Gamification Science, Its History and Future: Definitions and a Research Agenda. URL: https://www.researchgate.net/publication/325297221_Gamification_Science_Its_History_and_Future_Definitions_and_a_Research_Agenda

22. IBM. — What is Data Modeling? Електронний ресурс. URL: https://www.ibm.com/topics/data-modeling.

23. BigCommerce — What is an API. Електронний ресурс. URL: https://www.bigcommerce.com/blog/what-is-an-api/#what-is-an-api.

24. IBM. — What is Software Testing. Електронний ресурс. URL: https://www.ibm.com/topics/software-testing.

25. Guru99 — How to Write Test Cases: Sample Template with Examples. Електронний посібник. URL: https://www.guru99.com/test-case.html.

26. Statista — Глобальний ринок онлайн-освіти. Електронний ресурс. URL: https://www.statista.com/outlook/dmo/eservices/online-education/worldwid