Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:
протокол засідання кафедри
№ 19 від 05 червня 2025 р.
В.о. завідувача кафедри ММАД
_____ Струков В.М.
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему «Розробка веб-додатку для вивчення мов програмування»

Захищено на засіданні ЕК № _____
протокол № _____ від _____._____.2025 р.
Оцінка _____ / _____
Голова ЕК
_____ Фесенко Г.В.
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КС-42
Спеціальності:
122 Комп'ютерні науки
(шифр і назва спеціальності підготовки)
Боровик Олександра Олександрівна
(прізвище, ім'я та по батькові, підпис)
Керівник phd, доцент
Лисицький К.Є.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент к.т.н., доцент
Нарєжній О.П.
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра математичного моделювання та аналізу даних
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ММАД

_____ Володимир СТРУКОВ
підпис ім'я, прізвище

"03" лютого 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Боровик Олександри Олександрівни
(прізвище, ім'я, по батькові студента)

1. Тема роботи: «Розробка веб-додатку для вивчення мов програмування»

керівник роботи Лисицький Костянтин Євгенійович, phd, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від " 16 " квітня 2025 року № 4101-5/962

2. Строк подання студентом роботи 2 червня 2025 року

3. Перелік питань, які потрібно розробити:

ознайомитись з особливостями сучасних веб-додатків для онлайн-навчання, зокрема платформ, орієнтованих на вивчення мов програмування; провести огляд існуючих методів організації навчального контенту (курси, модулі, уроки) та реалізації інтерактивних елементів, таких як тести та практичні завдання на написання коду; проаналізувати алгоритми реалізації системи відстеження прогресу користувача; створити модель структури бази даних для забезпечення ефективної роботи користувацьких профілів та курсів; розробити архітектуру веб-додатку з розділенням на компоненти, враховуючи принципи модульності та масштабованості; провести програмну реалізацію та експериментальне дослідження функціональності веб-додатку.

4. План роботи

№ з/п	Назви етапів роботи
1	Формулювання мети та повного переліку завдань для виконання кваліфікаційної роботи (КР).
2	Пошук та аналіз інформаційних джерел за темою КР.
3	Визначення особливостей веб-додатків для онлайн-навчання мов програмування.
4	Розробка моделей структури бази даних та навчального контенту.
5	Проєктування архітектури веб-додатку з урахуванням принципів модульності та масштабованості.
6	Розробка дизайну користувацького інтерфейсу з урахуванням адаптивності та принципів UX/UI.
7	Програмна реалізація функціональних модулів веб-додатку: реєстрація, профіль, курси, уроки, прогрес.
8	Тестування та експериментальне дослідження роботи веб-додатку.
9	Підсумок отриманих результатів та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.

5. Дата видачі завдання 03.02.2025 року

Студент _____ О. О. Боровик
(підпис) (ініціали, прізвище)

Керівник роботи _____ К. Є. Лисицький
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Обсяг кваліфікаційної роботи становить 61 сторінку, містить 3 розділи, 11 рисунків, 1 додаток та 22 джерел.

У роботі представлено процес проєктування та розробки веб-додатку для вивчення мов програмування. Метою роботи є створення інтерактивної онлайн-платформи для організації індивідуального навчального процесу з можливістю проходження курсів, тестування, виконання завдань та відстеження особистого прогресу.

Проаналізовано особливості сучасних веб-додатків для онлайн-навчання та існуючі методи організації навчального контенту. У процесі розробки було використано технології Spring Boot, Spring Security, Spring Data JPA, Thymeleaf, Bootstrap, HTML/CSS та JavaScript. Архітектура додатку реалізована з використанням шаблону MVC. Для розробки використовувалося середовище IntelliJ IDEA, для роботи з базами даних – MySQL.

У результаті розроблено функціональний прототип веб-додатку для вивчення програмування, що поєднує в собі модульність, безпечну автентифікацію, адаптивність та можливість подальшого масштабування. Забезпечено гнучку організацію навчального контенту та індивідуалізацію освітнього процесу. Новизна роботи полягає у комплексному застосуванні зазначених сучасних Java-технологій для створення освітнього веб-додатку.

Створений прототип може бути розгорнутий та використаний для організації онлайн-навчання програмуванню, а також слугувати основою для подальшого розширення функціоналу, додавання нових навчальних курсів та інтеграції додаткових інтерактивних елементів.

Ключові слова: ВЕБ-ДОДАТОК, ПРОГРАМУВАННЯ, ОНЛАЙН-НАВЧАННЯ, SPRING BOOT, БАЗА ДАНИХ, КОРИСТУВАЧ, ПЕРСОНАЛІЗАЦІЯ, MVC, АВТЕНТИФІКАЦІЯ.

ABSTRACT

The qualification work comprises 61 pages, 3 chapters, 11 figures, 1 appendix, and 22 references.

This paper presents the process of designing and developing a web application for learning programming languages. The objective of the work is to create an interactive online platform for organizing an individual learning process with the ability to take courses, test, complete tasks, and track personal progress.

The features of modern web applications for online learning and existing methods of organizing educational content are analyzed. In the development process, Spring Boot, Spring Security, Spring Data JPA, Thymeleaf, Bootstrap, HTML/CSS, and JavaScript technologies were used. The application architecture is implemented using the MVC template. The IntelliJ IDEA environment was used for development, and MySQL was used to work with databases.

As a result, a functional prototype of a web application for learning programming has been developed that combines modularity, secure authentication, adaptability, and the possibility of further scaling. It provides flexible organization of educational content and individualization of the educational process. The novelty of the work lies in the integrated application of these modern Java technologies to create an educational web application.

The developed prototype can be deployed and used to organize online programming education. It can also serve as a foundation for further functional expansion, adding new training courses, and integrating additional interactive elements.

Keywords: WEB APPLICATION, PROGRAMMING, ONLINE LEARNING, SPRING BOOT, DATABASE, USER, PERSONALIZATION, MVC, AUTHENTICATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 6

ВСТУП 7

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 10

1.1 Огляд сучасних підходів та методик онлайн-вивчення мов програмування 10

1.2 Аналіз функціональних можливостей та архітектурних рішень існуючих веб-платформ для вивчення програмування 15

1.3 Порівняльний аналіз переваг та недоліків розглянутих аналогів 19

1.4 Обгрунтування доцільності розробки веб-додатку 23

2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ 24

2.1 Вимоги до проєктованого веб-додатку 24

2.1.1 Функціональні вимоги 24

2.1.2 Нефункціональні вимоги 25

2.2 Вибір технологічного стеку для розробки 26

2.2.1 Обґрунтування вибору архітектури 27

2.2.2 Обґрунтування вибору стеку технологій 28

2.3 Проєктування структури бази даних 29

2.4 Опис реалізації основних модулів системи 33

2.4.1 Реєстрація та авторизація користувачів 33

2.4.2 Управління курсами та навчальним контентом 35

2.4.3 Відстеження прогресу навчання 36

2.4.4 Профіль користувача 37

2.5 Проєктування UI/UX 38

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБ-ДОДАТКУ 41
3.1 Досягнення поставленої мети та вирішення завдань..... 41
3.2 Візуалізація реалізованого інтерфейсу..... 42
3.3 Опис та оцінка отриманих результатів..... 48
3.4 Області застосування та значущість роботи..... 49
ВИСНОВКИ..... 52
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 54
ДОДАТОК А ЛІСТИНГИ ПРОГРАМНОГО КОДУ 56

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних.

ІТ – інформаційні технології.

МН – машинне навчання.

МООК – масові відкриті онлайн-курси.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ШІ – штучний інтелект.

API – Application Programming Interface, інтерфейс прикладного програмування.

CRUD – Create, Read, Update, Delete – створення, читання, оновлення, видалення.

CSS – Cascading Style Sheets, каскадні таблиці стилів.

DTO – Data Transfer Object, об'єкт передавання даних.

ER-діаграма – Entity-Relationship Diagram, діаграма сутність-зв’язок.

HTML – HyperText Markup Language, мова гіпертекстової розмітки.

HTTP – HyperText Transfer Protocol, протокол передавання гіпертексту.

IDE – Integrated Development Environment, інтегроване середовище розробки.

JPA – Java Persistence API, інтерфейс збереження даних у Java.

MVC – Model-View-Controller, модель-представлення-контролер, архітектурний шаблон.

NoSQL – Not Only SQL, нереляційна база даних.

ORM – Object-Relational Mapping, об’єктно-реляційне відображення.

SQL – Structured Query Language, мова структурованих запитів.

UI – User Interface, інтерфейс користувача.

UX – User Experience, досвід користувача.

XSS – Cross-Site Scripting, міжсайтове скриптове впровадження.

ВСТУП

Інформаційні технології (ІТ) стали невід'ємною частиною сучасного світу, а програмування – ключовим умінням, що відкриває великі можливості як для професійної діяльності, так і для особистого розвитку. Швидкий розвиток цифрових сервісів, автоматизація виробничих і бізнес-процесів, поява нових напрямків – таких як штучний інтелект (ШІ), машинне навчання (МН), розробка мобільних застосунків – зумовлюють високий попит на фахівців із навичками програмування.

Цей зростаючий попит стимулює активний розвиток освітніх технологій, зокрема онлайн-платформ для вивчення мов програмування. Сучасний ринок освітніх послуг пропонує широкий спектр веб-додатків та інтерактивних курсів, що дозволяють ефективно здобувати знання та практичні навички у зручному форматі, незалежно від рівня початкової підготовки. Такі платформи, як Codecademy, Mate academy, GoIT, SoloLearn, Mimo та інші, пропонують навчальні матеріали, тести, практичні завдання і доступ до спільнот для обговорення. Вони відіграють важливу роль у популяризації програмування та полегшують вхід у професію.

Проте, незважаючи на велику кількість існуючих рішень, багато з них мають певні обмеження: складну навігацію, перевантажений інтерфейс, обмеження у безкоштовній версії або недостатньо гнучку систему мотивації. Крім того, такі сервіси часто орієнтовані на широку аудиторію і не дозволяють адаптувати навчальний процес до індивідуальних потреб користувачів.

У зв'язку з цим залишається актуальним завдання створення нових або вдосконалення існуючих веб-додатків для навчання програмуванню. Такі інструменти мають враховувати сучасні освітні методики, пропонувати інтуїтивно зрозумілий інтерфейс, структуровану подачу матеріалу, інтерактивні завдання та систему керування користувацьким профілем.

Актуальність даної роботи полягає у необхідності створення зручного та функціонального веб-додатку для вивчення мов програмування, який би

відповідав сучасним вимогам і був корисним інструментом для студентів, початківців у сфері IT та всіх охочих самостійно опановувати програмування.

Об’єктом дослідження є процес інтерактивного вивчення мов програмування з використанням сучасних веб-технологій.

Предметом дослідження є методи, моделі та технології розробки багатофункціонального веб-додатку для вивчення мов програмування, зокрема використання фреймворку Spring Boot, системи безпеки Spring Security, технології доступу до даних Java Persistence API (JPA) та шаблонізатора Thymeleaf.

Метою даної кваліфікаційної роботи є розробка веб-додатку для інтерактивного вивчення мов програмування з можливістю реєстрації користувачів, відстеження прогресу та виконання завдань.

Для досягнення поставленої мети було визначено такі завдання:

1. ознайомитись з особливостями сучасних веб-додатків для онлайн-навчання, зокрема платформ, орієнтованих на вивчення мов програмування;
2. проаналізувати існуючі методи організації навчального контенту та реалізації інтерактивних елементів, таких як теоретичні блоки, тести та практичні завдання на написання коду;
3. проаналізувати підходи до реалізації системи відстеження прогресу користувача;
4. створити модель структури бази даних (БД) для забезпечення ефективної роботи користувацьких профілів, навчальних курсів, розділів, уроків та завдань;
5. розробити архітектуру веб-додатку з розділенням на компоненти відповідно до шаблону модель-представлення-контролер (MVC), враховуючи принципи модульності та масштабованості;
6. спроєктувати дизайн інтерфейсу користувача, враховуючи адаптивність та основні принципи користувацького досвіду та

інтерфейсу (UX/UI) для забезпечення зручної взаємодії користувача з додатком;

7. здійснити програмну реалізацію функціональних модулів веб-додатку, включаючи: систему реєстрації та авторизації, управління профілем, відображення списку курсів, навігацію по розділах та уроках, а також відображення сторінок завдань;
8. забезпечити взаємодію з реляційною базою даних MySQL за допомогою JPA для зберігання та обробки даних;
9. провести тестування функціональних можливостей розробленого веб-додатку шляхом перевірки основних сценаріїв використання.

Наукова новизна роботи полягає у комплексному застосуванні сучасних Java-технологій, для створення освітнього веб-додатку, що поєднує в собі модульність, безпечну автентифікацію, адаптивність та підтримку розширення. Веб-додаток орієнтований на гнучку організацію навчального контенту та індивідуалізацію освітнього процесу.

Практичне значення отриманих результатів полягає у створенні прототипу веб-додатку, який може бути розгорнутий та використаний для організації онлайн-навчання програмуванню, а також слугувати основою для подальшого розширення функціоналу та додавання нових навчальних курсів. Додаток демонструє ефективне застосування обраного стеку технологій для вирішення задач у сфері освітніх веб-платформ.

Веб-додаток реалізовано з використанням Spring Boot (фреймворк для створення Java-застосунків), Thymeleaf (шаблонізатор для HTML), MySQL (система управління базами даних), що забезпечує його розширюваність та підтримку.

Таким чином, дана кваліфікаційна робота має на меті створення зручного інструменту для вивчення програмування, заснованого на сучасних підходах до розробки програмного забезпечення (ПЗ).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд сучасних підходів та методик онлайн-вивчення мов програмування

З розвитком ІТ та процесів глобалізації, дистанційне навчання, зокрема онлайн-освіта, стрімко набуває все більшої популярності. Вивчення мов програмування не є винятком, адже ця сфера потребує постійного оновлення знань та практичних навичок. Сучасні підходи та методики онлайн-вивчення мов програмування еволюціонували від простих текстових посібників до інтерактивних платформ, що поєднують різні дидактичні інструменти. Динамічність ІТ-індустрії вимагає від освітніх ресурсів гнучкості, актуальності та здатності швидко адаптуватися до нових технологій та потреб ринку праці.

Початкові етапи онлайн-навчання часто обмежувалися публікацією статичних навчальних матеріалів. Це могли бути завантажувані файли з конспектами лекцій, відсканові підручники або ж просто відеозаписи аудиторних лекцій без будь-якої можливості активної взаємодії учня з матеріалом. Однак з розвитком веб-технологій та педагогічних досліджень, онлайн-курси з програмування значно ускладнилися та вдосконалилися. Ключовим етапом у цьому розвитку стала поява масових відкритих онлайн-курсів (МООК) [1]. Окрім традиційних навчальних матеріалів, таких як відеолекції, текстові ресурси та домашні завдання, МООК запропонували інтерактивні форуми для користувачів. Це сприяло формування спільноти студентів, викладачів та асистентів, створюючи простір для обговорень, взаємодопомоги та обміну знаннями. Такий підхід зробив МООК однією з найновіших форм дистанційного навчання у світовій освітній сфері. Платформи, такі як Coursera, edX, Prometheus або Udacity, запропонували структуровані навчальні програми, які демократизували доступ до якісної освіти від провідних університетів та компаній по всьому світу [2]. Проте, варто зазначити, що багато ранніх МООК-платформ все ще зберігали пасивну

модель споживання контенту. Справжній прорив у вивченні програмування стався з появою інтерактивних навчальних середовищ, які зробили акцент на активній участі учня. Це дозволило змістити фокус з «вивчення про програмування» на «навчання програмуванню» через безпосередню практику, забезпечуючи динамічний процес розв'язання проблем та отримання миттєвого результату.

Одним з найбільш ефективних та широко застосовуваних підходів є інтерактивне навчання (Interactive Learning). Ця методика базується на активній взаємодії користувача з навчальним матеріалом. Замість пасивного сприйняття інформації, студенти виконують практичні завдання, пишуть код безпосередньо в браузері та отримують миттєвий зворотний зв’язок. Це дозволяє швидко виявляти та виправляти помилки, що значно прискорює процес засвоєння матеріалу.

Інтерактивність забезпечується за допомогою кількох ключових інструментів:

1) Вбудованих середовищ розробки (IDE) – ці інструменти імітують реальне робоче середовище програміста, надаючи доступ до редактора коду, компілятора/інтерпретатора, та інструментів відладки.
2) Систем автоматичної перевірки коду (Automated Code Review Systems). Вони аналізують написаний учнем код на відповідність вимогам завдання, ефективність, коректність виводу та наявність помилок. Це може бути реалізовано через набори тестів (юніт-тести, інтеграційні тести), аналіз синтаксису чи навіть статичний аналіз коду.
3) Симуляцій та віртуальних лабораторій: для демонстрації роботи складних систем або мереж.
4) Платформ для командної роботи (Collaborative Platforms), що дозволяють кільком учням працювати над одним проєктом, імітуючи умови реальної розробки. Цей формат заохочує розвиток

критичного мислення, аналітичних навичок та самостійності, роблячи навчальний процес більш ефективним та привабливим для сучасної аудиторії [3].

Ще одним поширеним підходом, що значно підвищує залученість користувачів, є гейміфікація (Gamification). Вона полягає у використанні ігрових елементів (таких як бали за виконані завдання, віртуальні значки за досягнення, рівні складності, турнірні таблиці) у неігровому контексті для підвищення мотивації та зацікавленості користувачів. Гейміфікація в освіті передбачає застосування цих ігрових механік, а також систем нарахування балів і винагород, аби зробити навчальний процес більш захоплюючим і динамічним. Психологічною основою гейміфікації є вроджене прагнення людини до досягнень, визнання та конкуренції. Однією з ключових переваг гейміфікації є здатність підвищувати як внутрішню, так і зовнішню мотивацію**.** Внутрішня мотивація походить від радості та задоволення, які людина відчуває під час виконання самої діяльності. Внутрішня мотивація робить навчальну діяльність більш веселою та цікавою, перетворюючи складні завдання на захопливі виклики. З іншого боку, зовнішня мотивація керується зовнішніми винагородами, такими як бали, значки, місця у рейтингах. Ці ігрові елементи заохочують учнів досягати конкретних цілей і бачити свій прогрес. Задовольняючи обидва типи мотивації, гейміфікація забезпечує всебічний підхід до заохочення залучення учнів і сприяння ефективному навчанню [4]. Дослідження ж показують, що поєднання гейміфікації з інтерактивним навчанням значно покращує практичне застосування знань, розвиває критичне мислення та підвищує самостійність студентів [3]. Однак, варто зазначити, що надмірне захоплення зовнішніми винагородами може відволікати від змісту навчання та знижувати рівень внутрішньої зацікавленості до самого предмету.

Проєктне навчання (Project-Based Learning) є ще однією ефективною методикою, що активно використовується в онлайн-освіті програмування. Замість вивчення окремих концепцій, студентам пропонується працювати над реальними або максимально наближеними до реальних проєктів. Це дозволяє

застосувати отримані знання на практиці, розвинути навички розв'язання проблем, командної роботи та критичного мислення. Окрім того, проєктне навчання допомагає студентам опановувати сучасні інструменти та технології, які використовуються в галузі. Часто такі проєкти супроводжуються наставництвом (менторською підтримкою) або взаємоперевіркою, що не лише покращує якість виконаних завдань, але й розвиває навички конструктивної критики та професійного обговорення. Додатково, цей метод сприяє формуванню портфоліо студента, що є важливим етапом для подальшого працевлаштування або участі в стартапах [5].

Методика «навчання малими порціями» або мікронавчання (Microlearning) передбачає подачу матеріалу невеликими, легко засвоюваними модулями. Кожен модуль фокусується на одній конкретній темі, тривалість якої може коливатися від кількох секунд до 10-15 хвилин. Такий підхід дозволяє уникнути перевантаження інформацією, сприяє кращому засвоєнню знань та забезпечує гнучкість у навчальному процесі. Психологічною основою ефективності мікронавчання є принципи короткострокової пам'яті та концентрації уваги. Дослідження показують, що людський мозок ефективніше обробляє та запам'ятовує інформацію, яка подається короткими, цілеспрямованими блоками. Мікронавчання вважається особливо ефективним для зайнятих людей, які мають обмежену кількість часу на навчання. І, крім того, воно сприяє підвищенню продуктивності та концентрації, оскільки дозволяє зосередитися на одному завданні за раз, мінімізуючи відволікання та забезпечуючи відчуття завершеності після кожного невеликого кроку [6].

Зростає також популярність підходу адаптивного навчання (Adaptive Learning), що є однією з найпрогресивніших форм персоналізації освітнього процесу та відображає сучасні тенденції у розвитку онлайн-освіти. Завдяки використанню алгоритмів ШІ та МН, навчальна платформа може аналізувати прогрес користувача, виявляти прогалини у знаннях та адаптувати навчальний план, пропонуючи персоналізовані завдання та матеріали. Це дозволяє оптимізувати процес навчання для кожної людини, зосереджуючись на

слабких місцях та закріплюючи вже засвоєні знання. Крім того, такий підхід сприяє підвищенню мотивації користувачів, оскільки вони отримують індивідуально підібраний навчальний контент, що відповідає їхнім потребам та рівню знань. Це створює відчуття більшого контролю над власним навчанням, що, у свою чергу, підвищує їхню впевненість у власних силах та наполегливість у виконанні навіть складних програмних завдань. У результаті адаптивне навчання робить освітній процес більш ефективним та гнучким, допомагаючи кожному студенту досягти максимальних результатів у вивченні програмування [7].

Крім того, важливою частиною онлайн-навчання є відео-лекції, які дозволяють візуалізувати процес написання коду, демонстрацію функціональності програм та пояснення складних концепцій. Відеоматеріали часто доповнюються текстовими матеріалами, що містять детальні пояснення, приклади коду, посилання на додаткові джерела інформації та різні вправи для самостійної роботи. Таке поєднання візуальної та текстової інформації дозволяє охопити різні стилі сприйняття інформації. Це забезпечує більш глибоке та всебічне засвоєння матеріалу, дозволяючи кожному студенту обрати найбільш комфортний та ефективний спосіб навчання. Особливо цінним таке комплексне представлення є для засвоєння практичних навичок, де розуміння теорії підкріплюється демонстрацією її застосування та подальшою практикою.

Підсумовуючи, сучасні підходи до онлайн-вивчення мов програмування є комплексними та багатогранними. Вони поєднують інтерактивні елементи, гейміфікацію, проєктне та мікронавчання, а також адаптивні механізми для створення ефективного та захоплюючого освітнього досвіду. Комбінація цих підходів дозволяє створювати ефективні освітні продукти, що враховують різні стилі навчання та рівні підготовки користувачів та забезпечують високий рівень залученості, мотивації та успішності у вивченні програмування.

1.2 Аналіз функціональних можливостей та архітектурних рішень існуючих веб-платформ для вивчення програмування

Як було розглянуто у попередньому підрозділі, онлайн-навчання мов програмування базується на комплексі підходів, таких як інтерактивне навчання, гейміфікація, проєктне навчання, мікронавчання та адаптивні технології. У цьому підрозділі проаналізуємо, як ці методики реалізовані у функціоналі популярних веб-платформ, а також розглянемо типові архітектурні рішення, що забезпечують їхню роботу. Метою є виявлення успішних практик та потенційних недоліків, що стануть основою для проєктування власного веб-додатку.

Сучасні освітні платформи активно впроваджують різноманітні методики для підтримки навчального процесу, що відображає їхнє прагнення до створення максимально ефективного та привабливого освітнього середовища.

Наприклад, Codecademy є яскравим прикладом платформи, що ефективно використовує інтерактивне навчання. Вона пропонує широкий вибір курсів з різних мов програмування (Python, JavaScript, Java, SQL, C++ тощо) та технологій. Ключова особливість полягає в тому, що користувачі вивчають матеріал, виконуючи практичні завдання безпосередньо у вбудованому редакторі коду з миттєвим зворотним зв'язком [8]. Це дозволяє студентам одразу ж застосовувати отримані знання, бачити результат своїх дій та оперативно виправляти помилки. Codecademy також пропонує чітко структуровані «кар’єрні шляхи», що включають елементи проєктного навчання на завершальних етапах, де студенти створюють портфолійні проєкти. Для підтримки мотивації використовують елементи гейміфікації, такі як щоденні цілі (так звані «streaks» – серії безперервних днів навчання) та значки за проходження курсів [9].

Платформи, такі як SoloLearn та Mimo, акцентують увагу на мікронавчанні, подаючи матеріал короткими, легко засвоюваними уроками. Такий формат ідеально підходить для мобільних пристроїв та навчання «на

ходу», що робить ці платформи доступними для широкої аудиторії з обмеженим часом [10, 11]. Обидві платформи активно використовують гейміфікацію: користувачі отримують бали за успішне виконання завдань, підвищують рівні, беруть участь у змаганнях на лідербордах і використовують віртуальні валюти. Інтерактивне навчання реалізоване через невеликі практичні завдання та тести після кожного блоку теорії. SoloLearn, крім того, виділяється потужним соціальним компонентом, де користувачі можуть ділитися своїм кодом, обговорювати завдання та ставити питання, формуючи активну навчальну спільноту.

Унікальним прикладом повністю безкоштовної платформи є freeCodeCamp, що фокусується на проєктному навчанні та підготовці до кар'єри веб-розробника. Вона пропонує інтегровані «сертифікації» з різних областей, які складаються з великою кількістю інтерактивних уроків та серій, реальних проєктів [12]. Студенти повинні створити проєкти самостійно, використовуючи надані інструменти та ресурси, а потім вони перевіряються системою або спільнотою. freeCodeCamp також активно використовує інтерактивне навчання у своєму вбудованому редакторі коду з миттєвим зворотним зв'язком для малих завдань, а також має надзвичайно велику та активну онлайн-спільноту на форумах та Discord-серверах, де користувачі можуть отримувати допомогу, обговорювати проблеми та ділитися досвідом. Відсутність традиційної гейміфікації компенсується фокусом на здобутті реальних навичок та сертифікатів, що мають вагу на ринку праці.

Українські платформи, такі як Mate academy та GoIT, часто фокусуються на проєктному навчанні та підготовці до працевлаштування. Їхні програми зазвичай триваліші та інтенсивніші, з акцентом на розробці комплексних проєктів, наближених до реальних комерційних завдань [13, 14]. Значну увагу приділено менторській підтримці, де студенти отримують індивідуальні консультації від досвідчених розробників, а також заохочують командну роботу над проєктами. Практичні домашні завдання перевіряються як вручну менторами, так і за допомогою автоматизованих систем, що забезпечує

глибокий аналіз виконаної роботи. У свою чергу, LeetCode та HackerRank – це платформи, що спеціалізуються на практиці розв’язання алгоритмічних задач та підготовці до технічних співбесід. Вони максимально використовують інтерактивне навчання через велику базу завдань з автоматичною перевіркою рішень різними мовами програмування, а елементи гейміфікації також відіграють значну роль у мотивації користувачів [15, 16].

Таким чином, онлайн-освітні платформи пропонують широкий спектр функцій для підтримки навчального процесу. Серед ключових можливостей, які є спільними для більшості розглянутих рішень, варто виділити:

1) Управління профілем користувача – дозволяє студентам відстежувати свій прогрес у навчанні, зберігати персональні налаштування, керувати підписками та отримувати сертифікати після завершення курсів.
2) Каталог курсів – зручна система навігації дозволяє користувачам швидко знаходити потрібні курси, фільтруючи їх за мовою програмування, рівнем складності або тематикою.
3) Система коментування та обговорень – функціонал для спілкування, де студенти мають змогу ставити запитання, ділитися власним досвідом, отримувати допомогу від спільноти та викладачів. Це підтримує колективне навчання та взаємодію.
4) Підтримка різних мов програмування – платформи забезпечують доступ до широкого спектра технологій для вивчення, що дозволяє обирати напрямок, який відповідає цілям навчання.
5) Системи відстеження прогресу – візуалізація пройденого матеріалу, виконаних завдань, отриманих балів, що допомагає користувачам бачити свій прогрес та мотивує до подальшого навчання.

Отже, для ефективної реалізації цих освітніх підходів необхідне відповідне технічне забезпечення. Архітектура онлайн-освітніх платформ повинна бути добре продуманою, щоб гарантувати стабільність,

масштабованість та інтерактивність навчального процесу. Хоча конкретні технічні рішення можуть бути комерційною таємницею, на основі аналізу відкритих джерел та загальних тенденцій у веб-розробці можна виділити кілька ключових аспектів.

Сучасні освітні платформи традиційно використовують клієнт-серверну архітектуру, яка є базовою для більшості веб-додатків. У цій моделі фронтенд (клієнтська частина) відповідає за інтерактивний користувацький інтерфейс та взаємодію з користувачем. Зазвичай реалізований на JavaScript-фреймворках, таких як React, Angular, Vue.js, що дозволяють створювати динамічні та інтерактивні користувацькі інтерфейси. Бекенд (серверна частина) обробляє бізнес-логіку додатку, взаємодію з базою даних, автентифікацію користувачів, управління курсами та завданнями, а також інтеграції з зовнішніми сервісами. Для розробки бекенду часто використовують фреймворки на базі Node.js, Python (Django, Flask), Java (Spring Boot), які забезпечують надійність та масштабованість [17].

Важливими аспектами є гнучкість та масштабованість архітектури, особливо для підтримки інтерактивних середовищ (наприклад, виконання коду користувача), систем автоматичної перевірки завдань та потенційного впровадження адаптивного навчання на базі ШІ. У таких випадках часто доцільним є використання мікросервісної архітектури, де окремі компоненти (наприклад, сервіс автентифікації, сервіс курсів, сервіс виконання коду, сервіс прогресу користувача) розробляються та масштабуються незалежно [18]. Це підвищує відмовостійкість системи, спрощує розробку та обслуговування великих проєктів, а також дозволяє використовувати різні технології для різних мікросервісів.

Щодо БД, у подібних системах зазвичай використовуються як:

1) Реляційні (SQL) бази даних для зберігання високоструктурованих даних, таких як профілі користувачів, інформація про курси, завдання, результати тестів та метадані. Вони забезпечують цілісність даних та складні запити.

2) Нереляційні (NoSQL) бази даних для зберігання менш структурованих або динамічних даних, таких як прогрес користувача (логі активності, частково виконані завдання), кеш, сесії або контент уроків, що може бути легко масштабованим.

Для забезпечення виконання коду різними мовами, інтеграції з системами тестування або аналітики часто використовуються внутрішні та зовнішні API [19]. Ці інтерфейси дозволяють різним компонентам системи взаємодіяти між собою, а також дозволяють стороннім розробникам або сервісам інтегруватися з платформою.

Окрім цього, для забезпечення високої доступності, балансування навантаження та кешування контенту, сучасні платформи активно використовують хмарні сервіси, такі як Amazon Web Services, Google Cloud Platform або Microsoft Azure. Ці платформи надають широкий спектр послуг, включаючи обчислювальні потужності, сховища даних, мережеві сервіси та інструменти для моніторингу, що дозволяє швидко масштабувати інфраструктуру відповідно до зростаючого навантаження та географічно розподіляти сервіси для мінімізації затримок.

Таким чином, сучасні освітні платформи комбінують низку педагогічних підходів, технічних інструментів та архітектурних рішень для забезпечення ефективного навчання. При проєктуванні власного веб-додатку доцільно орієнтуватися на перевірені практики, що поєднують адаптивність, інтерактивність та можливість масштабування.

1.3 Порівняльний аналіз переваг та недоліків розглянутих аналогів

Аналіз функціональних можливостей та освітніх підходів, реалізованих на популярних веб-платформах для вивчення програмування, дозволяє виділити низку спільних переваг, а також специфічних недоліків та обмежень, властивих різним типам ресурсів. Проведений порівняльний аналіз є важливою передумовою для обґрунтування доцільності розробки нового веб-додатку, який міг би запропонувати ефективне рішення.

Існуючі онлайн-платформи для навчання програмуванню пропонують значні переваги, які зробили їх такими популярними та ефективними для мільйонів користувачів:

1) Високий рівень інтерактивності та практичної спрямованості. Платформи, такі як Codecademy, LeetCode, SoloLearn, freeCodeCamp та Mimo, ефективно реалізують інтерактивне навчання завдяки вбудованим редакторам коду, автоматичній перевірці завдань та швидкому зворотному зв’язку. Це сприяє глибшому зануренню в матеріал, та дозволяє експериментувати з кодом у режимі реального часу без потреби в налаштуванні локального середовища. Це особливо важливо для початківців.
2) Гейміфікація та мікронавчання для підвищення мотивації. Особливо мобільно-орієнтовані платформи, як SoloLearn та Mimo, активно застосовують елементи гейміфікації та мікронавчання. Такий підхід сприяє підвищенню мотивації, особливо на початкових етапах навчання, оскільки створює відчуття прогресу, досягнення та змагання. Та, у свою чергу, дозволяє займатися у зручний час навіть при обмежених часових ресурсах.
3) Структурований контент та кар’єрні траєкторії. Багато платформ, зокрема Codecademy, Coursera, freeCodeCamp, а також українські Mate academy та GoIT, пропонують добре структуровані навчальні програми та курси, що ведуть користувача від основ до більш складних тем та проєктного навчання. Це забезпечує послідовність та повноту знань, а чіткі навчальні плани допомагають користувачам орієнтуватися в матеріалі та бачити довгострокові цілі навчання та планувати професійний розвиток.
4) Фокус на практичних проєктах та підготовці до ринку праці. Платформи типу буткемпів (Mate academy, GoIT) роблять акцент на проєктному навчанні, роботі над реальними або наближеними до реальних завданнями. Додатковою перевагою є менторство та

цілеспрямована підготовка до співбесід. Також варто відзначити freeCodeCamp, яка надає сертифікації, що вимагають виконання великих проєктів, що мають значну цінність для формування портфоліо. Це забезпечує випускникам практичний досвід та портфоліо, що підвищує шанси на швидке та успішне працевлаштування.

5) Доступність великої кількості навчальних матеріалів та спільнот. Більшість платформ надають доступ до широкого спектру курсів з різних мов програмування та технологій, що дозволяє користувачам обирати відповідний їм напрямок. А активні спільноти дозволяють обмінюватися досвідом, ставити запитання та отримувати підтримку, створюючи сприятливе середовище для спільного навчання.

Незважаючи на значні переваги, сучасні онлайн-платформи для вивчення програмування мають низку обмежень, які можуть знижувати загальну ефективність навчання для певних категорій користувачів:

1) Поверхневість знань через надмірне спрощення. Занадто сильний акцент на мікронавчанні або надмірно керовані завдання в інтерактивних середовищах можуть призвести до формування поверхневих знань без глибокого розуміння фундаментальних концепцій. Користувачі можуть навчитися виконувати конкретні шаблони завдання, але мати труднощі з адаптацією до нових, нетипових проблем. Крім того, відсутність необхідності працювати з локальним середовищем також знижує підготовленість до реальних умов роботи.
2) Обмежена ефективність гейміфікації для довгострокової мотивації. Хоча гейміфікація може бути ефективною на початкових етапах, її вплив на довгострокову внутрішню мотивацію може бути обмеженим. Надмірний фокус на зовнішніх винагородах може відвертати увагу від самого процесу навчання

та цінності знань, перетворюючи навчання на «гонитву за очками», а не за глибоким розумінням предмета.

3) Фінансові бар’єри та вимоги до самодисципліни. Повний доступ до якісних курсів, особливо тих, що пропонують менторство та кар’єрного супроводу, часто є платним і може бути досить дорогим. Це створює фінансовий бар’єр для багатьох потенційних студентів. Крім того, ефективне онлайн-навчання вимагає високого рівня самоорганізації та самодисципліни, що може бути проблемою для деяких користувачів, особливо за відсутності зовнішнього контролю, чітких дедлайнів або достатньої мотивації.
4) Обмеженість проєктного навчання на масових платформах. На багатьох масових платформах проєктне навчання може мати обмежений характер, зводячись до виконання шаблонних проєктів без достатньої індивідуалізації та глибокого занурення у специфіку реальної розробки. Це не завжди дозволяє студентам повністю розкрити творчий потенціал, отримати досвід розробки власних унікальних ідей.
5) Відсутність або слабка реалізація адаптивного навчання. Незважаючи на потенціал, справді ефективні системи адаптивного навчання, які б динамічно підлаштовували контент та складність під індивідуальні потреби кожного учня, на більшості платформ реалізовані обмежено або знаходяться на початковому етапі розвитку.
6) Технічні обмеження вбудованих середовищ. Вбудовані IDE, хоч і зручні, проте можуть мати обмеження щодо функціоналу, підтримуваних бібліотек або версій мов програмування. Це не завжди відповідає реальним робочим інструментам розробника, де потрібна гнучкість, доступ до різних фреймворків та можливість налаштування оточення. Це може ускладнити перехід від навчання на платформі до реальної професійної діяльності.

1.4 Обгрунтування доцільності розробки веб-додатку

Проведений порівняльний аналіз у розділі 1.3 свідчить про відсутність ідеальної платформи, яка б задовольняла потреби всіх категорій користувачів та ефективно поєднувала б усі переваги сучасних освітніх методик без жодних недоліків. Більшість розглянутих платформ роблять акцент на певних аспектах: одні – на інтерактивності та гейміфікації для початківців, інші – на глибокому зануренні та проєктному підході для тих, хто прагне кардинально змінити професію.

Виявлені недоліки, такі як поверхневість знань, обмеження гейміфікації, висока вартість деяких рішень та недостатня персоналізація, вказують на наявність простору для вдосконалення. Таким чином, існує потреба у створенні веб-платформи, яка б усувала зазначені недоліки, поєднуючи найкращі практики сучасних освітніх підходів. Пропонована розробка буде спрямована на створення такого рішення, що надасть користувачам доступ до структурованого та глибокого теоретичного матеріалу з ефективними інтерактивними практичними завданнями, механізмів, що підтримують внутрішню мотивацію, а також можливістю роботи над невеликими, але значущими проєктами, що імітують реальні умови розробки. Особлива увага планується приділити доступності базового функціоналу без суттєвих фінансових бар'єрів, що дозволить охопити більшу частину людей. Це підкреслює актуальність та доцільність проведення даного дослідження та реалізації проєкту в межах даної роботи.

2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ

2.1 Вимоги до проєктованого веб-додатку

На основі аналізу сучасних підходів до онлайн-навчання, оцінки функціональних можливостей існуючих платформ, а також виявлених переваг і недоліків їх аналогів, формулюються вимоги до проєктованого веб-додатку. Вони спрямовані на створення ефективного освітнього інструменту, що поєднує структуроване навчання, інтерактивну взаємодію та зручне управління користувацьким досвідом. Чітке визначення функціональних та нефункціональних характеристик є критично важливим для забезпечення ефективності системи. Ці вимоги слугуватимуть основою для всіх наступних етапів проєктування, реалізації та тестування системи, забезпечуючи відповідність кінцевого продукту поставленій меті.

2.1.1 Функціональні вимоги

Функціональні вимоги визначають конкретні дії та функції, які повинен надавати система на всіх етапах взаємодії з користувачем, починаючи від реєстрації і завершуючи оцінкою прогресу у навчанні [20]. Нижче наведено основні функціональні вимоги до веб-додатку для вивчення мов програмування, згруповані за логічними модулями.

1) Управління користувачами та автентифікація
    a) Система повинна надавати можливість реєстрації нових користувачів із зазначенням таких даних, як ім’я, прізвище, адреса електронної пошти та пароль.
    b) Система повинна забезпечувати безпечну автентифікацію зареєстрованих користувачів за допомогою логіна (email) та пароля.
    c) Користувач повинен мати доступ до особистого профілю з можливістю перегляду та редагування особистих даних (ім’я, прізвище, пароль).
    d) Система повинна відображати прогрес користувача шляхом підрахунку завершених курсів та уроків.

2) Управління навчальним контентом та курсами
    a) Система повинна забезпечувати відображення списку доступних навчальних курсів.
    b) Користувач повинен мати можливість переглядати детальну інформацію про обраний курс: назву, опис та його структуру.
    c) Навчальний контент повинен мати ієрархічну структуру. Кожен курс поділений на розділи, які в свою чергу складаються з уроків.
    d) Уроки повинні містити текстову теорію, завдання для виконання, а також окремі розділи, присвячені виконанню завдань з написання коду.
3) Процес навчання
    a) Користувач повинен мати можливість переходити між розділами та уроками в межах курсу.
    b) Система повинна зберігати прогрес проходження курсів та уроку користувачем.
    c) Система повинна надавати зворотний зв'язок за результатами виконання завдань.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики ПЗ, що не описують її функціональність безпосередньо, але суттєво впливають на ефективність, зручність та надійність системи. Вони охоплюють загальні властивості веб-додатку, які регламентують як саме повинна функціонувати система, а не які функції вона виконує.

На відміну від функціональних вимог, що описують конкретні дії та задачі, нефункціональні вимоги зосереджені на таких аспектах, як продуктивність, масштабованість, безпека, зручність користування та відповідність стандартам якості. Їх дотримання гарантує стабільну роботу системи за різних умов експлуатації [20].

1) Зручність використання

a) Інтерфейс користувача повинен бути інтуїтивно зрозумілим, простим у використанні та візуально привабливим.
b) Навігація по сайту повинна бути логічною та послідовною.
c) Система повинна надавати зрозумілі повідомлення про помилки та інформативні підказки користувачеві.

2) Безпека
   a) Система повинна забезпечувати конфіденційність та цілісність даних користувачів, зокрема паролів (використання хешування).
   b) Автентифікація та управління сесіями користувачів повинні бути реалізовані з використанням надійних механізмів.
   c) Система повинна бути захищена від основних вразливостей (наприклад, XSS, SQL ін'єкції) відповідно до стандартних практик безпечної розробки.
3) Технологічні аспекти
   a) Система повинна бути розроблена з використанням сучасного, надійного та добре підтримуваного стеку технологій, що забезпечує реалізацію визначених функціональних та нефункціональних вимог.

Таким чином, сформульовані вимоги слугуватимуть основою для проєктування архітектури та розробки функціоналу веб-додатку, що є предметом даної кваліфікаційної роботи. Вони відображають прагнення створити збалансований освітній інструмент, що враховує як сучасні тенденції в онлайн-навчанні, так і потреби користувачів.

2.2 Вибір технологічного стеку для розробки

Вибір архітектури та технологічного стеку для проєктування веб-додатку є критично важливим етапом, що безпосередньо впливає на його функціональність, продуктивність, масштабованість, безпеку та зручність подальшої підтримки. На основі сформульованих вимог (у розділі 2.1) та з урахуванням сучасних підходів до розробки веб-додатків, для проєктованої системи було обрано багатошарову (багаторівневу) архітектуру у поєднанні з

технологічним стеком на базі Spring Boot. Цей вибір забезпечує високу надійність, гнучкість та відповідність сучасним стандартам розробки.

2.2.1 Обґрунтування вибору архітектури

Проєктований веб-додаток реалізовано з використанням багатошарової архітектури, яка передбачає логічне та фізичне розділення системи на окремі рівні або шари. Кожен з цих шарів виконує свою специфічну, чітко визначену роль, що значно спрощує розробку, тестування та налагодження. Такий підхід мінімізує взаємний вплив змін у різних частинах системи, підвищує гнучкість та полегшує підтримку та розширення функціоналу. Крім того, модульність архітектури сприяє спрощенню тестування, дозволяючи проводити інтеграційне та юніт-тестування окремих компонентів, а також створює передумови для подальшого масштабування системи в майбутньому [21].

Основні шари проєктованого веб-додатку мають наступні відповідальності. Шар представлення, або веб-шар, представлений контролерами, відповідає за обробку вхідних HTTP-запитів, їх маршрутизацію та формування відповіді, повертаючи динамічні HTML-сторінки через шаблонізатор Thymeleaf. Цей шар організовано відповідно до архітектурного патерну MVC, де контролери керують потоком запитів, взаємодіючи з моделями для отримання даних та обираючи відповідні представлення для відображення. Це є точкою входу для взаємодії користувача з системою. Сервісний шар інкапсулює бізнес-логіку додатку, обробляючи запити від контролерів та координуючи операції з даними. Він є центральним компонентом, що забезпечує виконання всіх функціональних вимог системи. Для взаємодії з базою даних слугує шар доступу до даних, реалізований через репозиторії. Цей рівень забезпечує абстракцію для операцій зі зберігання та отримання даних, приховуючи деталі роботи з системою управління базами даних (СУБД) та взаємодіючи з нею через Spring Data JPA. Фундаментом для цих шарів є шар сутностей, що містить моделі даних, які відображають структуру таблиць бази даних та об'єкти бізнес-домену. Для ефективної передачі даних між шарами використовуються об'єкти передачі даних (DTO),

що дозволяє відокремити внутрішню структуру даних від їх представлення на клієнтській стороні, підвищуючи безпеку та гнучкість системи.

2.2.2 Обґрунтування вибору стеку технологій

Вибір технологій для реалізації проєкту базувався на принципах надійності, популярності, наявності великої спільноти розробників, широких можливостей для інтеграції, а також відповідності вимогам до продуктивності та безпеки системи.

Для розробки фронтенду, тобто клієнтської частини веб-додатку, використовуються стандартні веб-технології: HTML5, CSS3 та JavaScript. Вони є основою для структури, стилізації та забезпечення інтерактивності користувацького інтерфейсу. Для генерації динамічних HTML-сторінок на стороні сервера обрано Thymeleaf як шаблонізатор. Його інтеграція зі Spring Framework, а також можливість працювати з HTML-файлами як зі звичайними шаблонами, значно спрощує розробку. Візуальна привабливість та адаптивність інтерфейсу на різних пристроях забезпечується за допомогою фреймворку Bootstrap.

Бекенд, або серверна частина системи, розробляється на мові програмування Java, що відома своєю стабільністю, продуктивністю, широким спектром бібліотек та великою спільнотою. Це забезпечує високу надійність та легкість масштабування додатку. В якості основного фреймворку використано Spring Boot, який значно прискорює процес розробки та розгортання Java-додатків завдяки автоматичній конфігурації та вбудованим серверам. Управління проєктом та його залежностями здійснюється за допомогою системи автоматизації збирання Maven, що забезпечує послідовну та стандартизовану збірку проєкту. Для реалізації механізмів автентифікації, авторизації та управління доступом на основі ролей обрано Spring Security. Це потужне та гнучке рішення дозволяє забезпечити високий рівень безпеки даних користувачів, включаючи хешування паролів за допомогою PasswordEncoder, та захист від поширених веб-вразливостей, що було визначено нефункціональними вимогами. Робота з реляційними базами даних

значно спрощується завдяки Spring Data JPA, яка надає об’єктно-реляційне відображення (ORM) та мінімізує необхідність писати багато шаблонного коду для операцій CRUD.

В якості СУБД було обрано MySQL. Цей вибір обумовлений високою продуктивністю, надійністю та широким поширенням MySQL, що робить її однією з найпопулярніших реляційних СУБД для веб-додатків. MySQL ефективно підтримує структуровані дані, забезпечує цілісність інформації та підходить для зберігання профілів користувачів, інформації про курси, уроки, завдання та прогрес навчання, що є критично важливим для даного проєкту.

Загалом, обрана багатошарова архітектура, заснована на патерні MVC, у поєднанні зі стеком технологій на базі Java та Spring Boot (Spring Security, Spring Data JPA, MySQL, Thymeleaf) забезпечує міцний фундамент для розробки функціонального, безпечного, масштабованого та зручного у використанні веб-додатку для вивчення мов програмування, що повністю відповідає сформульованим вимогам та цілям даної кваліфікаційної роботи.

2.3 Проєктування структури бази даних

Проєктування ефективної та логічно організованої структури бази даних є фундаментальним етапом у розробці будь-якого програмного забезпечення, що працює з даними. Це безпосередньо впливає на продуктивність, цілісність даних, зручність розробки та подальшої підтримки системи. На основі сформульованих функціональних вимог (у розділі 2.1) та обґрунтованого вибору реляційної СУБД MySQL (у розділі 2.2), було розроблено структуру бази даних, яка забезпечить надійне зберігання інформації про користувачів, навчальні курси, їхній вміст та прогрес навчання.

База даних спроєктована з використанням реляційної моделі, що дозволяє ефективно керувати структурованими даними та підтримувати їхню цілісність за допомогою зв’язків між таблицями. Кожна сутність системи має унікальний первинний ключ, а зв’язки між таблицями реалізовано за допомогою зовнішніх ключів, що забезпечує узгодженість даних.

Для зберігання та організації навчального процесу в системі визначено наступні ключові сутності, що відповідають реалізованій моделі даних:

1) Користувачі (Users) – зберігає інформацію про зареєстрованих користувачів платформи.
    a) *id* – унікальний ідентифікатор користувача, що генерується автоматично.
    b) *email* – електронна пошта користувача, використовується для входу, є унікальною.
    c) *password_hash* – хешований пароль користувача для безпечної автентифікації.
    d) *name* – ім’я користувача.
    e) *surname* – прізвище користувача.
    f) *created_at* – дата та час створення облікового запису користувача.
    g) *role* – роль користувача в системі (*'USER'* або *'ADMIN'*).
2) Курси (Courses) – представляє навчальні курси, доступні на платформі.
    a) *id* – унікальний ідентифікатор курсу.
    b) *title* – назва курсу.
    c) *description* – детальний опис курсу.
    d) *cover_image* – шлях до зображення обкладинки курсу.
    e) *created_at* – дата та час створення курсу.
3) Розділи (Sections) – представляє розділи курсу; кожен розділ належить певному курсу, забезпечуючи ієрархічну структуру.
    a) *id* – унікальний ідентифікатор розділу.
    b) *title* – назва розділу.
    c) *position* – порядковий номер розділу в межах курсу для забезпечення правильної послідовності відображення.
    d) *course_id* – посилання на курс, до якого належить розділ.

4) Уроки (Lessons) – описує уроки, що входять до складу розділів.
    a) *id* – унікальний ідентифікатор уроку.
    b) *title* – назва уроку.
    c) *position* – порядковий номер уроку в межах розділу.
    d) *section_id* – посилання на розділ, до якого належить урок.
5) Сторінки уроків (Lesson_pages) – кожен урок складається з однієї або декількох сторінок, які містять безпосередній навчальний контент. Тип вмісту сторінки визначається перерахуванням PageType (THEORY, QUIZ, CODE_TASK), що забезпечує строгу типізацію та обмежує можливі значення, підвищуючи цілісність даних.
    a) *id* – унікальний ідентифікатор сторінки.
    b) *content* – основний вміст сторінки (текст, код, питання вікторини).
    c) *page_type* – тип вмісту сторінки (наприклад, *'THEORY'*, *'QUIZ'*, *'CODE_TASK'*).
    d) *position* – порядковий номер сторінки в межах уроку.
    e) *lesson_id* – посилання на урок, до якого належить сторінка.
6) Прогрес користувача (User_progress) – відстежує прогрес кожного користувача в проходженні уроків.
    a) *id* – унікальний ідентифікатор запису прогресу.
    b) *user_id* – посилання на користувача.
    c) *lesson_id* – посилання на урок, прогрес якого відстежується.
    d) *completed* – ознака завершення уроку користувачем.
    e) *last_page* – номер останньої сторінки уроку, яку користувач переглянув. Це дозволяє продовжити навчання з місця зупинки.
    f) *updated_at* – дата та час останнього оновлення прогресу по уроку.

Між описаними сутностями встановлено наступні реляційні зв'язки, що забезпечують цілісність та логічну послідовність даних, відображаючи структуру навчального процесу:

1) Один-до-багатьох (One-to-Many):
    a) Courses – Sections: Один курс може містити багато розділів. Кожен розділ належить тільки одному курсу (course_id у таблиці Sections є зовнішнім ключем до Courses).
    b) Sections – Lessons: Один розділ може містити багато уроків. Тоді як кожен урок належить тільки одному розділу (section_id у таблиці Lessons є зовнішнім ключем до Sections).
    c) Lessons – Lesson_pages: Один урок може складатися з багатьох сторінок, а кожна сторінка належить тільки одному уроку (lesson_id у таблиці Lesson_pages є зовнішнім ключем до Lessons).
    d) Users – UserProgress: Один користувач може мати багато записів про прогрес, що відстужують кожен урок, який він почав або завершив.
    e) Lessons – UserProgress: Один урок може мати багато записів про прогрес, оскільки багато користувачів можуть його проходити.
2) Багато-до-багатьох (Many-to-Many):
    a. Взаємозв’язок між Users та Lessons (через який відстежується прогрес) реалізовано через проміжну таблицю UserProgress.

Для візуального представлення описаної структури бази даних та взаємозв’язків між сутностями розроблена ER-діаграма (Entity-Relationship Diagram). Вона графічно ілюструє таблиці, їхні атрибути, первинні та зовнішні ключі, а також типи зв’язків між ними, забезпечуючи чітке розуміння моделі даних системи. ER-діаграма проєкту бази даних наведена на рисунку 2.1.

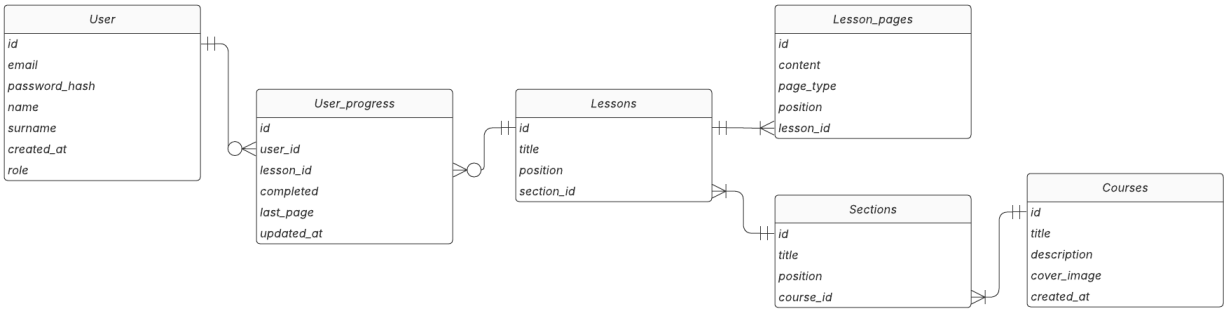

Рисунок 2.1 – ER-діаграма моделі даних веб-додатку

Проєктована структура бази даних є гнучкою та розширюваною, що дозволяє легко додавати нові типи контенту, курси, а також реалізовувати додаткові функціональні можливості у майбутньому. Вона оптимізована для ефективного зберігання навчальних матеріалів та швидкого доступу до даних користувачів і їхнього прогресу.

2.4 Опис реалізації основних модулів системи

Реалізація веб-додатку базується на багатошаровій архітектурі та обраному стеку технологій, детально описаних у попередніх підрозділах. У цьому підрозділі буде представлено опис реалізації основних функціональних модулів системи, що відображають її ключові можливості та задовольняють сформульовані функціональні вимоги (у розділі 2.1). Кожен модуль розроблено з урахуванням принципів модульності, чіткого розділення відповідальності та зручності підтримки.

2.4.1 Реєстрація та авторизація користувачів

Одним із основних функціональних компонентів системи є механізм автентифікації та авторизації користувачів, що надає доступ до персоналізованих можливостей платформи, зокрема перегляд профілю, відстеження прогресу навчання та участь у навчальних курсах. Реалізація цього механізму здійснюється за допомогою фреймворку Spring Security [22], який інтегрується з Spring Boot та дозволяє реалізувати сучасні підходи до безпеки веб-додатків.

Реєстрація нових користувачів реалізована за допомогою динамічних HTML-форм на базі шаблонізатора *Thymeleaf*. Користувач вводить такі обов’язкові дані: ім’я, прізвище, електронну адресу (яка виконує роль логіну), пароль, а також підтвердження паролю.

Для передачі даних від форми до контролера використовується DTO *RegisterDto*. Цей об'єкт містить анотації валідації (*@NotEmpty*, *@Email*, *@Size*), які автоматично перевіряють введені користувачем дані на відповідність заданим критеріям, таких як коректний формат електронної пошти або мінімальна довжина пароля. Крім того, на рівні контролера реалізована додаткова логіка перевірки на співпадіння пароля та його підтвердження, а також перевірка на унікальність електронної пошти в базі даних.

Перед збереженням у базі даних введеним користувачем пароль хешується за допомогою алгоритму *BCrypt*, що забезпечує захист від атак. Після успішної реєстрації користувачу автоматично надається роль *USER*, яка зберігається у відповідному полі сутності користувача.

Структура DTO-об’єкта *RegisterDto* для реєстрації та логіка створення нового користувача наведено в Лістинг А.1 та Лістинг А.2 (Додатку А).

Авторизація користувачів виконується через стандартну форму входу Spring Security, яка обробляє введену електронну пошту та пароль. У разі введення коректних облікових даних (електронна пошта та пароль), користувач отримує доступ до захищених ресурсів, а саме до особистого кабінету, перегляду курсів, відстеження прогресу тощо. За успішного входу користувач перенаправляється на сторінку */dashboard*.

Конфігурацію безпеки, включаючи дозволи для окремих маршрутів, сторінку входу, параметри автентифікації та виходу з системи, а також налаштування шифрування паролів, представлено в Лістингу А.3.

Після успішної авторизації користувача зберігається сесія, за допомогою якої визначається його автентифікований статус. Крім того, завдяки гнучкій конфігурації ролей користувачів, можна ефективно управляти правами

доступу, що особливо важливо для реалізації адміністративного функціоналу та розмежування рівнів взаємодії у системі.

2.4.2 Управління курсами та навчальним контентом

Управління курсами та навчальним контентом є центральною функціональністю системи. Кожен курс складається з одного або кількох розділів, що, у свою чергу, містять уроки. Структура даних і взаємозв’язки між сутностями були детально розглянуті в розділі 2.3. Такий підхід дозволяє логічно структурувати навчальні матеріали та забезпечити поступове ускладнення подачу інформації відповідно до принципів адаптивного навчання.

Для реалізації доступу до навчального контенту розроблено низку контролерів та сервісів, які відповідають за взаємодію з БД та їх динамічне відображення на веб-сторінках за допомогою шаблонізатора Thymeleaf.

Контролер *CourseController* обробляє запити, пов'язані з переглядом курсів. При переході на сторінку */courses*, він отримує список усіх доступних курсів з *CourseRepository*, передає його до моделі та відповідно відображає їх на сторінці. У разі переходу на сторінку конкретного курсу (*/courses/{id}*), контролер завантажує повну інформацію про вибраний курс та всі пов’язані з ним розділи. Якщо ж користувач автентифікований, то система також підраховує кількість пройдених уроків та загальну кількість уроків у цьому курсі, що дозволяє відобразити його індивідуальний прогрес. Фрагмент коду методу *viewCourse*, що реалізує дану логіку, наведено в Лістинг А.4

Центральним компонентом для навігації по уроках та їхньому контенту є *LessonController*. Його метод *viewLessonPage* обробляє запити у форматі */lessons/{lessonId}/pages/{pageIndex},* забезпечуючи відображення конкретної сторінки уроку. Для правильного порядку подачі контенту сторінки сортуються за полем *position*. Далі контролер визначає тип поточної сторінки: теоретичний матеріал, тест або практичне завдання. Залежно від типу, до моделі додаються відповідні DTO-об'єкти, такі як *QuizQuestionDTO* або *CodeTaskDTO,* що використовуються для генерації певних елементів

інтерфейсу користувача. Фрагмент реалізації методу *viewLessonPage* наведено в Лістингу А.5.

2.4.3 Відстеження прогресу навчання

Даний модуль є ключовим для підтримки мотивації та персоналізації навчання, відображаючи досягнення користувача. Прогрес фіксується на рівні кожного курсу та розраховується як відношення кількості завершених уроків до загальної кількості уроків у курсі.

Інформація про завершення уроків зберігається у відповідній таблиці БД UserProgress, що пов'язує користувача з ідентифікаторами пройдених уроків. Після завершення уроку (наприклад, після перегляду всіх сторінок або виконання завдання) ця інформація автоматично фіксується у системі. У *CourseController* реалізовано логіку, яка при завантаженні сторінки окремого курсу обчислює:

a) загальну кількість уроків у курсі;
b) кількість уроків, які даний користувач уже завершив;
c) передача кількості завершених та загальної кількості уроків до шаблону для візуалізації у вигляді прогрес-бару.

На головній сторінці (/dashboard) виводиться для користувача загальний огляд прогресу по кожному з курсів, до яких він має доступ. Завдяки цьому користувач може швидко оцінити свій статус навчання та обрати курс для продовження. Прогрес же слугує основою для формування рекомендацій, зокрема для пріоритетного відображення незавершених курсів. У HTML-шаблоні прогрес виводиться у вигляді відсоткового значення та числового співвідношення між пройденими та загальною кількістю уроків, як показано у Лістингу А.6.

Під час реалізації модуля відстеження прогресу навчання виникла необхідність зберігати та оновлювати інформацію про проходження кожного окремого уроку для кожного користувача. Одним із викликів було забезпечення точності фіксації завершеності уроку з урахуванням багатосторінкової структури навчальних матеріалів.

Для вирішення цієї задачі було реалізовано сервіс *UserProgressService*, який відповідає за обробку прогресу користувача по уроках. Зокрема, він зберігає останню переглянуту сторінку уроку, а також визначає, чи вважається урок завершеним (наприклад, коли користувач переглянув усі сторінки). Таким чином, прогрес оновлюється тільки у разі досягнення нового прогресу, що дозволяє уникнути зайвих записів у базу даних та зменшити навантаження. Таке рішення дало змогу забезпечити масштабованість системи, спростити обробку даних у контролерах і покращити загальну продуктивність при роботі з навчальними курсами.

2.4.4 Профіль користувача

Модуль користувацького профілю надає можливість користувачам переглядати та змінювати персональні дані, а також керувати безпекою облікового запису. Через вебінтерфейс користувач може оновити ім’я та прізвище, а також змінити пароль у відповідному розділі налаштувань.

Сторінка профілю доступна лише після автентифікації та реалізована у вигляді окремого шаблону */profile*. Отримання даних користувача здійснюється через механізм Spring Security (Principal) з подальшим пошуком об'єкта *User* у базі даних за електронною адресою.

Оновлення основної інформації реалізується через метод POST, який перевіряє валідність введених даних та зберігає зміни до БД через репозиторій. Для покращення зручності користувача, після збереження змін відбувається перенаправлення з повідомленням про успішне оновлення. Фрагмент коду, що відповідає за отримання та оновлення профілю користувача, представлено у Лістингу А.7.

Зміна пароля відбувається на окремій сторінці */profile/security*. Для підвищення безпеки реалізовано перевірку поточного пароля (через *PasswordEncoder.matches*) та підтвердження нового пароля. У разі успішної перевірки новий пароль хешується та зберігається. Логіка зміни пароля детально ілюструється у Лістингу А.8.

Одним із викликів при реалізації модуля було забезпечення безпечної та зручної зміни пароля. Важливо було уникнути помилок при перевірці поточного пароля та не допустити збереження нового пароля без підтвердження. Для цього було створено DTO-об’єкт *PasswordChangeDto*, який дозволяє зручно передавати та застосовувати анотації валідації. Також було використано *BCryptPasswordEncoder* із Spring Security для надійного хешування паролю.

2.5 Проєктування UI/UX

Проєктування ефективного та інтуїтивно зрозумілого користувацького інтерфейсу та забезпечення позитивного досвіду користувача є дійсно важливим для успіху будь-якого освітнього веб-додатку. Метою проєктування UI/UX для даного веб-додатку для вивчення мов програмування було створення інтуїтивно зрозумілого, зручного у використанні та візуально привабливого середовища, яке б сприяло ефективному навчанню та підтримувало мотивацію користувачів. На початковому етапі проєктування було проведено детальний аналіз існуючих освітніх платформ для вивчення програмування. Цей аналіз дозволив виявити їхні сильні сторони у плані UI/UX, а також ідентифікувати типові проблеми, з якими стикаються користувачі. Особливу увагу було приділено інтерактивним елементам та візуалізації прогресу. Основними обмеженнями, що вплинули на проєктування, були необхідність забезпечення кросплатформності та адаптивності, а також обмеження часу та ресурсів для розробки, що вимагало оптимізації та вибору перевірених рішень.

При розробці дизайну інтерфейсу було взято до уваги ключові принципи. Передусім, це простота та інтуїтивність, завдяки чому користувачі, навіть з мінімальним досвідом, могли легко орієнтуватися та знаходити потрібну інформацію. Щоб реалізувати цей підхід, використовувалась методика поступового ускладнення, що починалася з базових ескізів інтерфейсу з мінімальним рівнем деталізації. Послідовність та узгодженість елементів керування, навігації, типографіки та колірної схеми підтримується

на всіх сторінках додатку, створюючи відчуття стабільності. Цей принцип реалізовувався через використання єдиної дизайн-системи, заснованої на бібліотеці Bootstrap. Мінімалізм та фокус на контенті дозволили уникнути перевантаження інтерфейсу зайвими елементами, зосередивши увагу користувача на навчальному матеріалі та виконанні завдань. Це досягнуто використанням темного фону для основного контенту та мінімального використання декоративних елементів. Дизайн повністю орієнтований на користувача, враховуючи потреби аудиторії, що вивчає програмування, з різним рівнем підготовки. Система надає чіткий зворотний зв'язок за допомогою стандартизованих повідомлень та візуальних індикаторів валідації форм. Ключовим технічним рішенням для забезпечення адаптивності інтерфейсу, стало використання CSS-фреймворку Bootstrap. Це дозволило забезпечити коректне відображення додатку на різних пристроях, від широкоформатних моніторів до екранів мобільних телефонів.

Проєктування інтерфейсу охоплювало розробку структури та компонування основних сторінок системи. Задача декомпозиції інтерфейсу була вирішена шляхом виділення загальних компонентів, таких як хедер (з логотипом, навігаційним меню до курсів і профілю) та футеру. Головна сторінка надає автентифікованому користувачеві швидкий огляд доступних курсів у вигляді карток з назвою та з візуальними індикаторами його прогресу. Сторінка окремого курсу містить детальну інформацію про обраний курс, його опис та список розділів з уроками, включаючи індивідуальний прогрес користувача. Сторінка самого уроку є центральним елементом для відображення навчального контенту, який може бути теоретичним матеріалом, тестовими завданнями або практичними на написання коду з полем для введення та навігаційними кнопками. Сторінка профілю користувача дозволяє переглядати та редагувати особисті дані та налаштування безпеки, включаючи окрему сторінку для зміни паролю. Сторінки ж реєстрації та входу забезпечують процес створення облікового запису та аутентифікації за допомогою мінімалістичних форм з чітко

визначеними полями та повідомленнями про помилки валідації. Візуалізацію спроєктованої структури ключових сторінок буде представлено у Розділі 3.

Для забезпечення зручності переміщення по веб-додатку було спроєктовано логічну навігаційну модель. Верхнє навігаційне меню надає швидкий доступ до ключових розділів і реалізоване у вигляді «липкого» елемента, що залишається на екрані під час прокручування сторінки для забезпечення постійної доступності. Також передбачені інтуїтивно зрозумілі логічні переходи між сторінками: від списку курсів до деталей курсу, від курсу до уроків, а також між сторінками в рамках одного уроку.

Візуальний стиль обрано стриманий та чистий. Колірна гама базується на темно-синіх та фіолетових відтінках для фону, що забезпечує комфортне візуальне сприйняття та зменшує навантаження на очі при тривалій роботі. Акцентні кольори, такі як яскраво-синій, фіолетовий, зелений (для успіху) та червоний (для помилок), використовуються для виділення важливих елементів керування, індикаторів прогресу та підсвічування синтаксису коду, забезпечуючи при цьому чіткий контраст та мінімізуючи відволікаючі фактори. Шрифти підбиралися з урахуванням забезпечення високої читабельності та комфорту для очей при тривалій роботі.

Для створення позитивного досвіду користувача також було враховано кілька ключових аспектів. Чітке відображення прогресу проходження курсів та уроків за допомогою прогрес-барів та візуальних позначок виконаних завдань мотивує користувачів. Легкість виконання завдань (реєстрації, входу, вибору курсу, навігації по уроках, оновлення профілю) спроєктована так, щоб вимагати мінімум зусиль. Система надає інформативний зворотний зв’язок за допомогою своєчасних та зрозумілих повідомлень. Таким чином, продумане проєктування користувацького інтерфейсу та досвіду користувача спрямоване на створення ефективного, зручного та приємного у використанні навчального середовища, що є невід’ємною частиною успішного освітнього веб-додатку.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБ-ДОДАТКУ

3.1 Досягнення поставленої мети та вирішення завдань

У результаті виконаної кваліфікаційної роботи було досягнуто основної мети – розроблено веб-додаток для вивчення мов програмування з можливістю реєстрації користувачів, відстеження прогресу та виконання завдань. Реалізовано функціональність, яка дозволяє користувачам проходити навчальні курси, зберігати особисті дані та переглядати власний прогрес, що підтверджує досягнення поставленої мети.

Зокрема для досягнення поставленої мети було вирішено всі поставлені завдання:

1) Проведено ознайомлення з особливостями сучасних освітніми веб-додатками, платформ, орієнтованих на вивчення мов програмування. Це дозволило зрозуміти поточні тенденції та виявити підходи до організації навчання онлайн.
2) Проаналізовано існуючі методи організації навчального контенту та реалізації інтерактивних елементів, таких як теоретичні блоки, тести та практичні завдання на написання коду. Результати аналізу були враховані при проєктуванні структури уроків та їх вмісту.
3) Проаналізовано підходи до реалізації системи відстеження прогресу користувача. Цей аналіз став основою для розробки функціоналу, що дозволяє користувачам моніторити свій навчальний прогрес.
4) Створено модель структури БД для забезпечення ефективної роботи користувацьких профілів, навчальних курсів, розділів, уроків та завдань. Це гарантувало надійне зберігання та управління всіма даними системи.
5) Розроблено архітектуру веб-додатку з розділенням на компоненти відповідно до шаблону MVC, що забезпечило структурованість,

зручність підтримки та можливість подальшого розширення функціональності.

6) Спроєктовано інтерфейс користувача з урахуванням принципів UX/UI. Інтерфейс є інтуїтивно зрозумілим, адаптивним для різних пристроїв і підтримує основні сценарії взаємодії користувача з додатком.
7) Здійснено програмну реалізацію функціональних модулів веб-додатку, включаючи: систему реєстрації та авторизації, управління профілем, відображення списку курсів, навігацію по розділах та уроках, а також відображення сторінок завдань з різними типами контенту.
8) Забезпечено взаємодію з реляційною базою даних MySQL за допомогою JPA для зберігання та обробки даних, що гарантувало цілісність та ефективний доступ до інформації.
9) Проведено тестування функціональних можливостей розробленого веб-додатку шляхом перевірки основних сценаріїв використання, що підтвердило коректність його роботи та відповідність визначеним вимогам.

Таким чином, усі поставлені завдання були успішно виконані, що дозволило створити повноцінний та функціональний веб-додаток для інтерактивного вивчення мов програмування.

3.2 Візуалізація реалізованого інтерфейсу

Для забезпечення зручної та інтуїтивно зрозумілої взаємодії користувача з веб-додатком було реалізовано сучасний і адаптивний інтерфейс, побудований з урахуванням принципів UX/UI-дизайну. Графічна частина була створена з використанням шаблонізатора Thymeleaf у поєднанні з HTML, CSS, а також стилізаційними фреймворками для підтримки адаптивності.

Першою сторінкою, з якою взаємодіє користувач, є головна сторінка, що доступна без авторизації. Вона містить короткий опис функціоналу додатку, пояснення його мети, а також запрошення до реєстрації чи входу. Таким

чином, новий користувач одразу отримує уявлення про можливості платформи та принципи її використання.

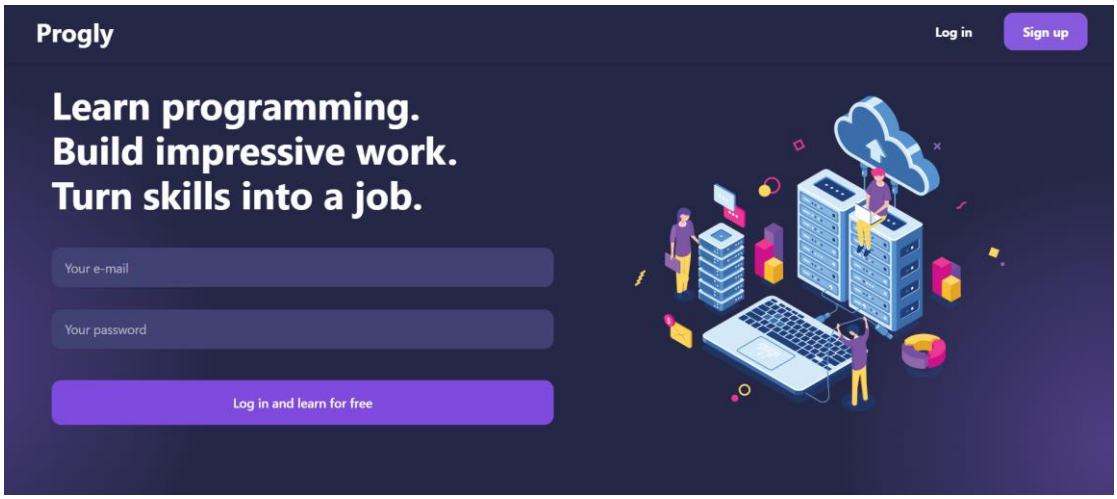

Рисунок 3.1 – Головна сторінка веб-додатку до входу

Для подальшої взаємодії з системою передбачені сторінки реєстрації та входу (рисунок 3.2 та 3.3 відповідно). Вони містять мінімалістичні форми з чітко визначеними полями для введення електронної пошти, пароля та інших необхідних даних, а також забезпечують візуальний зворотний зв'язок у разі помилок валідації, що є важливим для зручності та безпеки користувача.

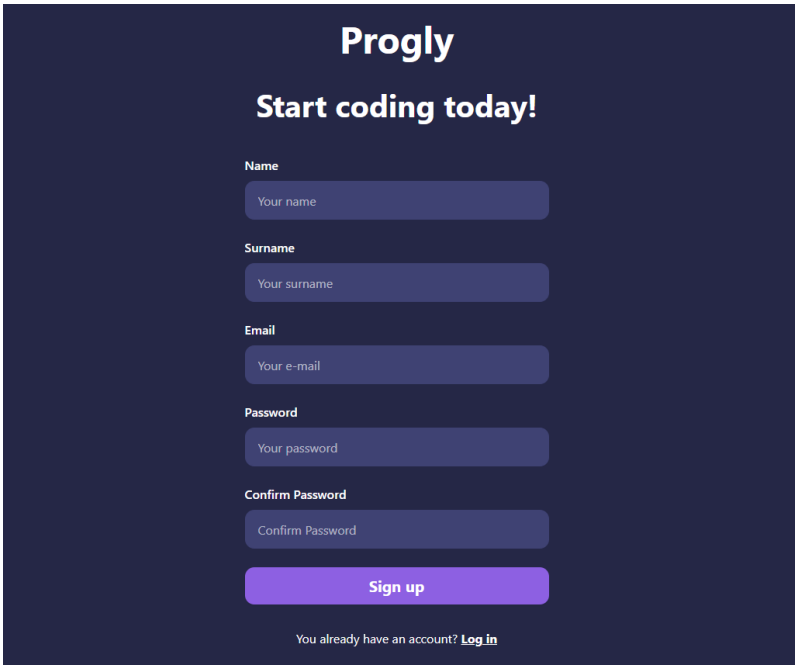

Рисунок 3.2 – Сторінка реєстрації нового користувача

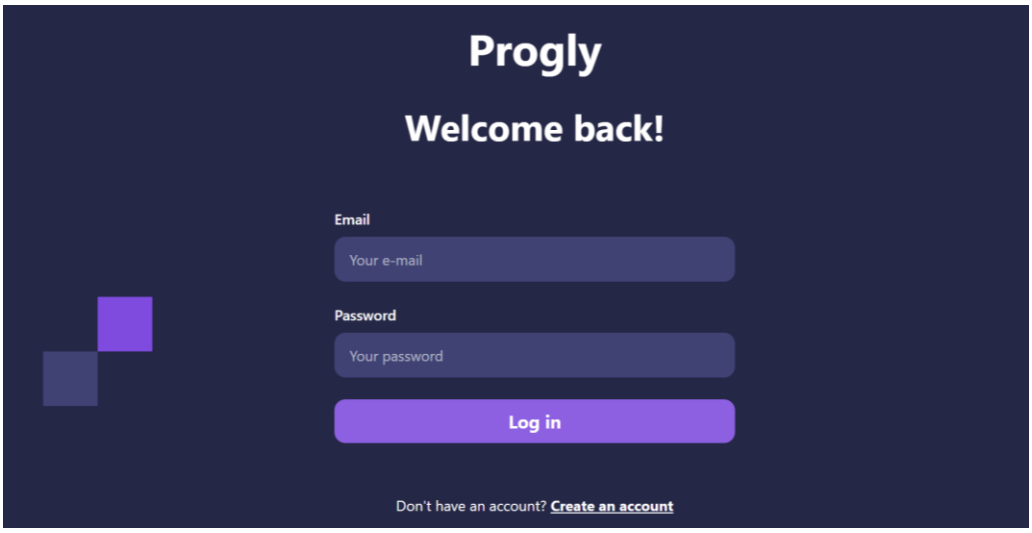

Рисунок 3.3 – Сторінка входу до системи

Після успішної автентифікації користувач потрапляє на персоналізовану головну сторінку (рисунок 3.4), яка єосновною точкою до навчальних матеріалів. Тут відображається огляд доступних навчальних курсів, кожен з яких представлений у вигляді інформативної картки. Картки курсів містять назву, короткий опис та візуальний індикатор прогресу проходження курсу користувачем. Це дозволяє швидко оцінити свій статус навчання та обрати курс для продовження.

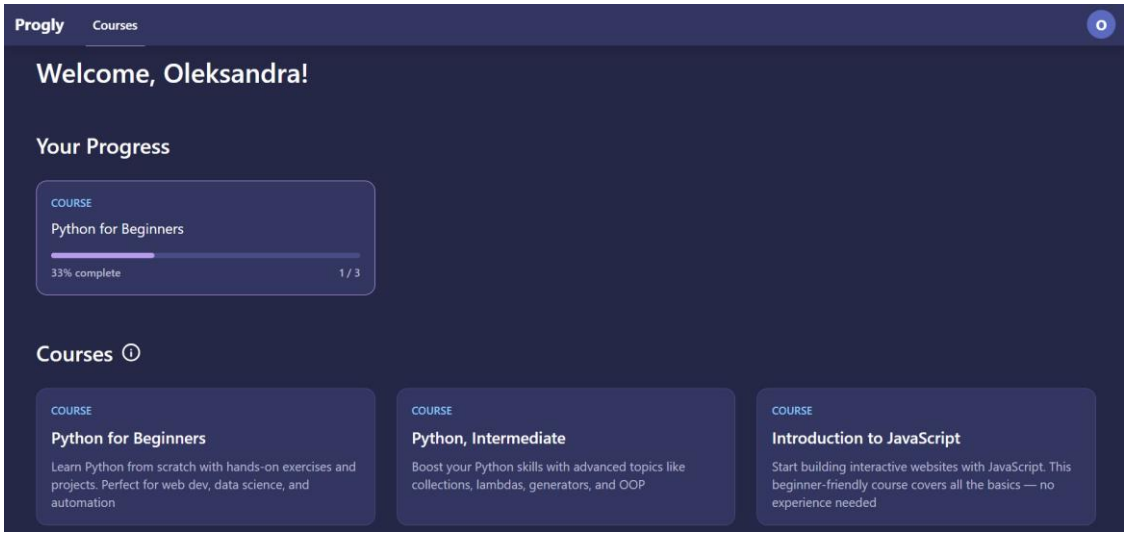

Рисунок 3.4 – Головна сторінка після входу

При виборі певного курсу користувач переходить на його деталізовану сторінку (рисунок 3.5). Вона містить розширений опис курсу та структурований перелік усіх розділів та уроків, що входять до його складу. А також також відображається їхній індивідуальний прогрес біля назви даного курсу.

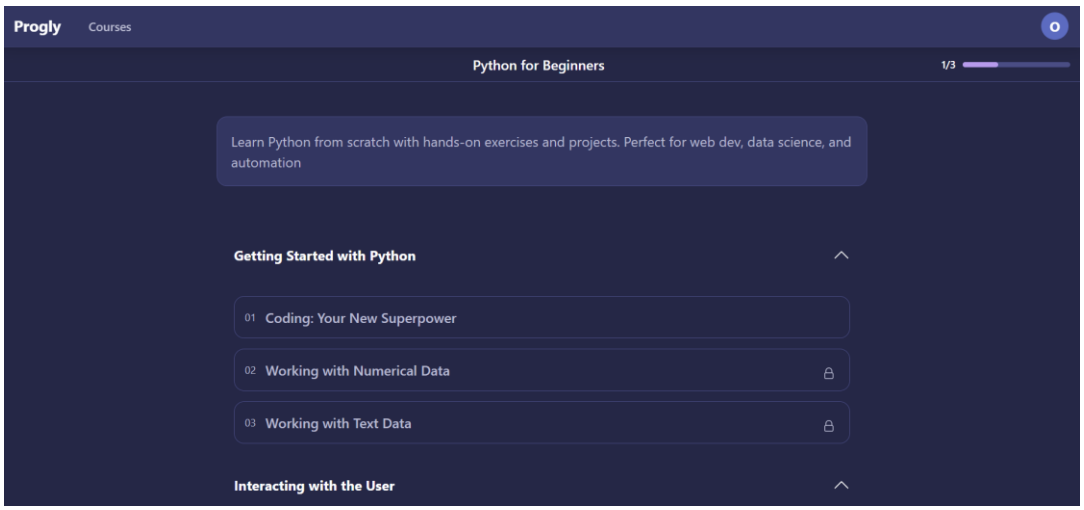

Рисунок 3.5 – Деталізована сторінка окремого курсу

Модуль уроків, що є центральним елементом подачі навчального контенту, підтримує різноманітні формати, забезпечуючи інтерактивне та гнучке навчання. Теоретичні сторінки уроків (рисунок 3.6) містять структурований текстовий матеріал, який можна легко переглядати. Для перевірки знань передбачені сторінки уроків з тестами (рисунок 3.7), де користувачі можуть відповідати на запитання. Окрім того, сторінки із завданнями на написання коду (рисунок 3.8) надають інтерактивне середовище для практики, де користувач може писати код безпосередньо у вбудованому редакторі та надсилати його для перевірки.

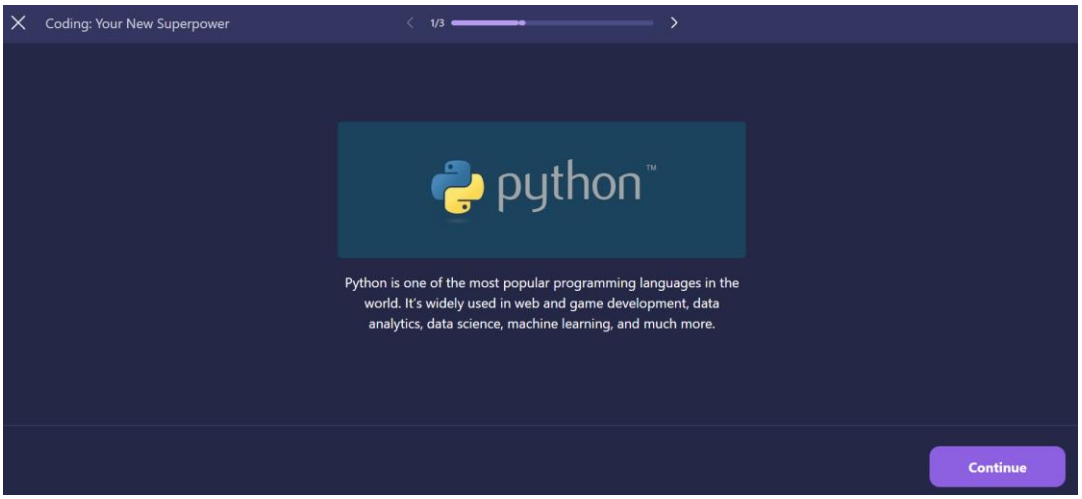

Рисунок 3.6 – Сторінка уроку з теоретичним матеріалом

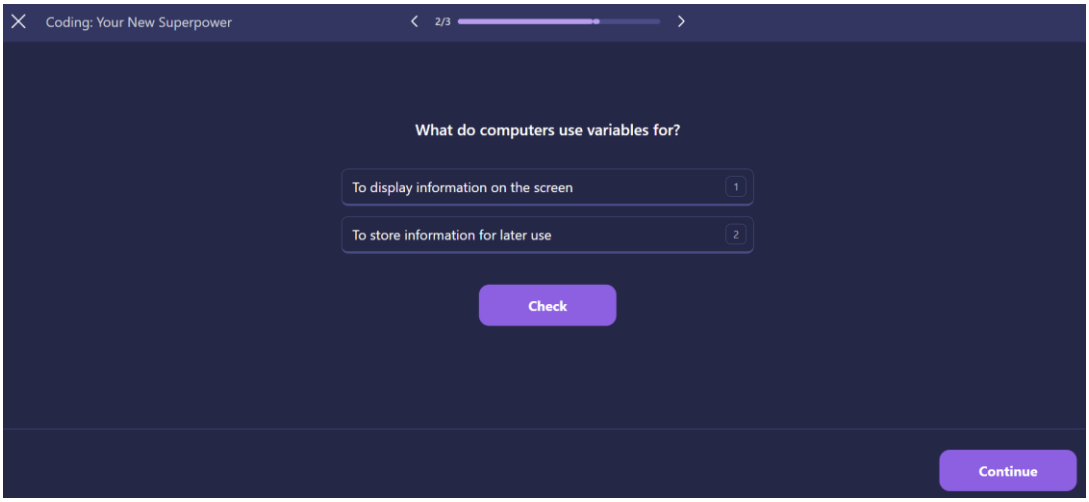

Рисунок 3.7 – Сторінка уроку з тестовим завданням

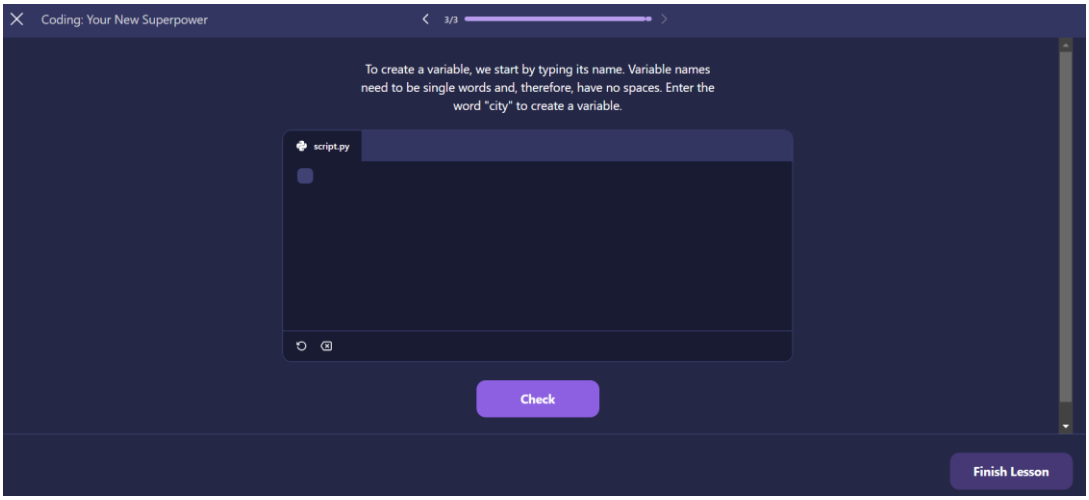

Рисунок 3.8 – Сторінка уроку з завданням на написання коду

Щодо модулю профілю, то він надає користувачам можливість керувати своїми особистими даними. Інтерфейс профілю організований у вигляді бічного меню, яке дає змогу швидко перемикатися між підрозділами: «Account Details», «Security», «Messages», «Courses» та «Sign Out». Це забезпечує швидкий доступ до налаштувань без необхідності покидати особисту сторінку. Сторінка «Account Details» (рисунок 3.9) дозволяє переглядати та редагувати основну інформацію, таку як ім'я та прізвище. Зміни ж зберігаються після натискання кнопки «Save changes». А підрозділ «Security» (рисунок 3.10) забезпечує безпечну процедуру оновлення пароля з обов'язковою перевіркою поточного пароля та підтвердженням нового, підкреслюючи увагу до безпеки користувацьких даних.

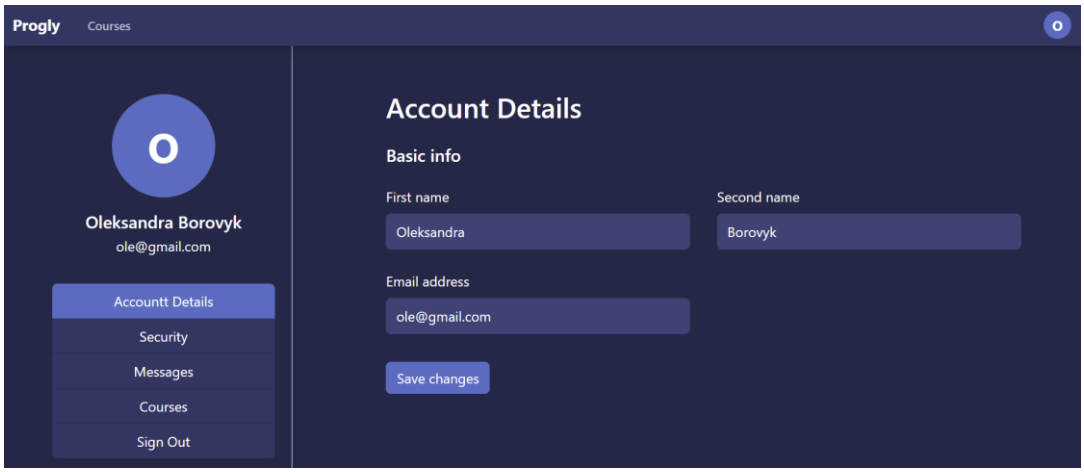

Рисунок 3.9 – Сторінка профілю користувача «Account Details»

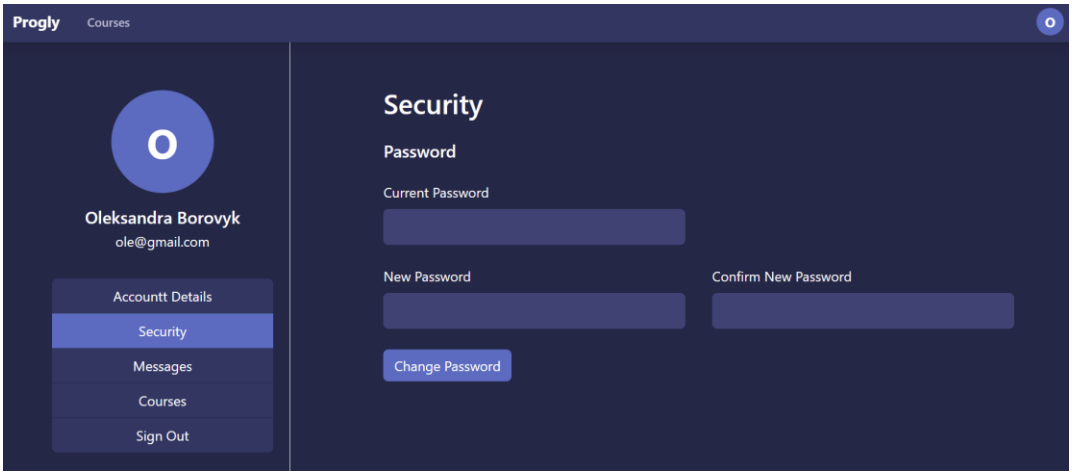

Рисунок 3.10 – Сторінка профілю користувача «Security»

Представлені візуальні матеріали детально відображають основні функціональні можливості розробленого веб-додатку та демонструють його адаптивний та інтуїтивно зрозумілий інтерфейс.

3.3 Опис та оцінка отриманих результатів

Розроблений веб-додаток є повноцінним функціональним рішенням, що відповідає поставленим вимогам та забезпечує ефективний навчальний процес. Оцінка отриманих результатів базується на аналізі реалізованих можливостей, їх відповідності нефункціональним вимогам, а також порівнянні з існуючими світовими тенденціями у сфері онлайн-освіти.

Усі ключові функціональні вимоги, визначені у розділі 2.1.1, були успішно реалізовані. Система забезпечує надійну реєстрацію та автентифікацію користувачів, включаючи безпечне хешування паролів та валідацію вхідних даних, що є критично важливим для конфіденційності та цілісності інформації. Завдяки реалізованому модулю управління навчальним контентом, користувачі можуть ефективно працювати з курсами, розділами та уроками, що містять теоретичні матеріали, тести та практичні завдання з написання коду. Ця ієрархічна модель даних забезпечує гнучкість у створенні та організації навчальних матеріалів, тоді як відстеження прогресу навчання відображає завершені уроки та курси, дозволяючи користувачу продовжити навчання з місця зупинки, що значно підвищує мотивацію та персоналізацію навчального процесу. Додатково, зручне управління профілем користувача надає можливість оновлювати особисті дані та безпечно змінювати пароль.

З точки зору нефункціональних вимог, веб-додаток демонструє високу зручність використання. Інтерфейс є інтуїтивно зрозумілим, візуально привабливим та адаптивним завдяки використанню Bootstrap, а логічна навігація та чіткий зворотний зв'язок сприяють комфортному користувацькому досвіду. Належний рівень безпеки гарантується інтеграцією Spring Security та використанням *BCrypt* для хешування паролів, що захищає систему від більшості поширених веб-вразливостей, таких як Cross Site Scripting та SQL-ін’єкції. А обраний технологічний стек забезпечує

стабільність, надійність та високу продуктивність системи, що підтверджено ручною перевіркою основних сценаріїв використання та тестування функціональності.

Розроблений веб-додаток гармонійно вписується у сучасні світові тенденції онлайн-освіти. Реалізація інтерактивних завдань з написання коду та тестів безпосередньо у браузері відповідає запиту на активне залучення користувача, відрізняючи його від пасивного перегляду відео. Наявність детального відстеження прогресу та можливість продовжити навчання з місця зупинки відповідають сучасним тенденціям адаптивного та персоналізованого навчання. Модульність та гнучкість контенту дозволяють легко додавати нові навчальні матеріали, що є перевагою над статичними курсами.

Незважаючи на успішну реалізацію основних функціональних та нефункціональних вимог, у процесі розробки були виявлені певні обмеження та напрямки для подальшого вдосконалення. Наприклад, на поточному етапі відсутня повноцінна адміністративна панель для зручного створення, редагування та управління курсами, уроками та користувачами через інтерфейс. Хоча інтерактивні завдання реалізовані, функціонал інтерактивної перевірки коду є базовим, підтримуючи обмежену кількість мов програмування для виконання та перевірки. Також наразі відсутні вбудовані комунікаційні інструменти між користувачами (форуми, чати), що могло б значно покращити досвід навчання та обмін знаннями. Крім того, наявна базова система рекомендацій є обмеженою та може бути розширена до повноцінного механізму персоналізованих рекомендацій на основі аналізу успішності або вподобань користувача. Ці моменти вказують на потенційні напрямки для подальшого розвитку та вдосконалення веб-додатку, перетворюючи їх на можливості для розширення його функціональних можливостей.

3.4 Області застосування та значущість роботи

Розроблений веб-додаток для вивчення мов програмування має широкі області застосування та значну господарську, наукову та соціальну

значущість. Він може стати ефективним інструментом для широкого кола користувачів та сприятиме підвищенню доступності якісної ІТ-освіти.

Насамперед, веб-додаток орієнтований на індивідуальне самостійне навчання. Він надає гнучке та структуроване середовище для тих, хто прагне вивчити основи програмування або поглибити свої знання у конкретній мові. Завдяки цьому він є ідеальним інструментом для:

1) Новачків у сфері ІТ, які бажають здобути базові навички для старту в професії.
2) Студентів технічних спеціальностей як додатковий ресурс для закріплення матеріалу та практичного відпрацювання навичок.
3) Людей, що змінюють професію, яким потрібен доступний спосіб освоїти нові навички.
4) Вчителів або викладачів для використання в якості інтерактивного матеріалу на уроках або як доповнення до основної програми навчання.
5) Корпоративних тренінгів для швидкого навчання співробітників основам програмування.

Крім зазначених сфер застосування, виконана кваліфікаційна робота має також багатогранну значущість, що охоплює господарський, науковий та соціальний аспекти.

Господарська значущість проєкту полягає у його потенціалі для створення економічно ефективного освітнього інструменту. Зменшення витрат на навчання завдяки доступності онлайн-платформи може зробити ІТ-освіту більш масовою та привабливою. У майбутньому, за умови розвитку функціоналу, додаток може бути монетизований наприклад через моделі підписки, що забезпечить його сталий розвиток. Він може стати альтернативою або доповненням до дорогих комерційних курсів, пропонуючи якісний контент за меншу вартість або навіть безкоштовно для базових модулів.

Наукова значущість роботи проявляється у дослідженні та застосуванні сучасних підходів до організації інтерактивного навчального контенту та відстеження прогресу. Проєкт демонструє ефективну інтеграцію технологій бекенду та фронтенду для створення цілісної освітньої платформи. Розробка та реалізація системи відстеження прогресу з урахуванням багатосторінкової структури уроків, а також інтеграція механізмів для виконання та перевірки коду, є внеском у розуміння архітектурних рішень для подібних систем. Результати цієї роботи можуть слугувати основою для подальших досліджень у галузі адаптивного навчання, автоматизованої оцінки знань та персоналізації навчального досвіду в онлайн-середовищі.

Соціальна значущість веб-додатку полягає у його здатності підвищити доступність та якість IT-освіти для широкого кола користувачів. У сучасному світі, де цифрові навички стають критично важливими, надання інтуїтивно зрозумілого та ефективного інструменту для вивчення програмування дозволить опанувати програмування людям незалежно від їхнього місця проживання, рівня підготовки чи фінансових можливостей. Це також допоможе збільшити кількість людей, які розуміються на цифрових технологіях, адже навчання кодування через такі інтерактивні платформи робить суспільство більш обізнаним та технологічно підготовленим. Надання якісних знань та практичних навичок сприяє підвищенню конкурентоспроможності фахівців на ринку праці. Крім того, збільшення кількості людей, які вміють програмувати, створює сприятливе середовище для підтримки інновацій та генерації нових ідей.

Таким чином, розроблений веб-додаток є не лише функціональним програмним продуктом, а й має значний потенціал для позитивного впливу на освітню та професійну сфери.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено повноцінний веб-додаток для вивчення мов програмування, що відповідає сучасним вимогам до онлайн-освітніх платформ. Застосунок орієнтований на інтерактивне навчання та забезпечує користувачам комфортний доступ до структурованих освітніх матеріалів у зручному веб-середовищі. Під час розробки було використано сучасні технології, зокрема Spring Boot, Spring Security, Spring Data JPA, Thymeleaf та Bootstrap, що дозволило створити функціональний, зручний та безпечний застосунок. Серед основних реалізованих функцій – реєстрація та автентифікація користувачів, перегляд курсів і уроків, проходження теоретичних і практичних матеріалів, виконання тестів і завдань, а також відстеження прогресу користувача. Структура курсів побудована ієрархічно, що дає змогу ефективно організовувати навчальний контент та масштабувати платформу в майбутньому.

Особливо важливим аспектом реалізації стала персоналізація процесу навчання. Застосунок дозволяє користувачам продовжити навчання з того місця, на якому було зупинене, а також надає візуальний супровід прогресу за допомогою прогрес-барів. Інтерфейс було розроблено з урахуванням принципів UX/UI дизайну, що робить платформу інтуїтивно зрозумілою, привабливою та зручною для новачків.

Додаток демонструє високі нефункціональні характеристики, серед яких варто відзначити стабільність роботи, безпечне зберігання даних, хешування паролів та валідацію введеної інформації. Завдяки модульній архітектурі реалізоване рішення є гнучким і придатним до подальшого вдосконалення та розширення функціональності.

У ході розробки було виявлено перспективні напрями розвитку платформи. Зокрема, доцільним є впровадження адміністративної панелі для керування контентом та користувачами, розширення функціоналу виконання коду з підтримкою декількох мов програмування, додавання системи

коментарів і комунікації між користувачами, а також реалізація рекомендаційного механізму на основі інтересів та прогресу користувача. У перспективі розглядається створення мобільного додатку, інтеграція аналітичних засобів для викладачів та оптимізація платформи для ефективної роботи з великою кількістю одночасних користувачів.

Таким чином, поставлену мету було досягнуто, всі основні завдання реалізовано в повному обсязі, а результати розробки можуть бути покладені в основу для створення повноцінної освітньої платформи, здатної забезпечити ефективне та персоналізоване дистанційне навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Масові відкриті онлайн-курси. URL: https://surl.li/hximla. Дата звернення: 05.05.2025.
2. Пеганов В. Найпопулярніші масові відкриті онлайн-курси. Їх структура, переваги та недоліки. 2024. URL: https://naurok.com.ua/naypopulyarnishi-masovi-vidkriti-onlayn-kursi-h-struktura-perevagi-ta-nedoliki-416516.html. Дата звернення: 05.05.2025.
3. Chad F. Gamification and Interactive Learning for Enhanced Knowledge Application.URL:https://www.researchgate.net/publication/389174873_Gamification_and_Interactive_Learning_for_Enhanced_Knowledge_Application. Дата звернення: 05.05.2025.
4. Serhat Kurt. Gamification, What It Is, How It Works, Examples. URL: https://educationaltechnology.net/gamification-what-it-is-how-it-works-examples/. Дата звернення: 05.05.2025.
5. What is PBL? URL: https://www.pblworks.org/what-is-pbl. Дата звернення: 05.05.2025.
6. Microlearning: Your Most Productive Tool to Learn New Skills. URL: https://www.neuefische.de/en/community/bootcamps-and-community/microlearning-your-most-productive-tool-to-learn-new-skills. Дата звернення: 05.05.2025.
7. Serhat Kurt. Adaptive Learning: What is It, What are its Benefits and How Does it Work? URL: https://educationaltechnology.net/adaptive-learning-what-is-it-what-are-its-benefits-and-how-does-it-work/. Дата звернення: 08.05.2025.
8. Codecademy. URL: https://www.codecademy.com/. Дата звернення: 08.05.2025.
9. Codecademy’s Interactive Courses: A Comprehensive Review of the Popular Coding Platform. URL: https://algocademy.com/blog/codecademys-interactive-courses-a-comprehensive-review-of-the-popular-coding-platform/. Дата звернення: 08.05.2025.

10. Mimo. URL: https://mimo.org/. Дата звернення: 08.05.2025.
11. Sololearn. URL: https://www.sololearn.com/en/. Дата звернення: 08.05.2025.
12. Freecodecamp. URL: https://www.freecodecamp.org/. Дата звернення: 09.05.2025.
13. Mate academy. URL: https://mate.academy/. Дата звернення: 09.05.2025.
14. GoIT. URL: https://goit.global/ua/. Дата звернення: 09.05.2025.
15. LeetCode. URL: https://leetcode.com/. Дата звернення: 09.05.2025.
16. HackerRank. URL: https://www.hackerrank.com/. Дата звернення: 09.05.2025.
17. Cherukuri B. R. Building Scalable Web Applications: Best Practices for Backend Architecture.URL:https://www.researchgate.net/publication/384663189_Building_Scalable_Web_Applications_Best_Practices_for_Backend_Architecture. Дата звернення: 12.05.2025.
18. Microservices vs Monolithic Architecture: A Comparison to Guide Your 2025 Strategy. URL: https://kitrum.com/blog/microservices-vs-monolithic-architecture/. Дата звернення: 12.05.2025.
19. Henderson L. The Role of APIs in Modern Web and Mobile Applications. URL: https://dev.to/dr_lachlanhenderson_4dfa/the-role-of-apis-in-modern-web-and-mobile-applications-2coa. Дата звернення: 12.05.2025.
20. Функціональні та нефункціональні вимоги (з прикладами). URL: https://surl.li/begdix. Дата звернення: 15.05.2025.
21. Про архітектуру додатків. URL: https://foxminded.ua/arkhitektura-zastosunku/. Дата звернення: 15.05.2025.
22. Securing a Web Application. URL: https://spring.io/guides/gs/securing-web. Дата звернення: 16.05.2025.

ДОДАТОК А
ЛІСТИНГИ ПРОГРАМНОГО КОДУ

Лістинг А.1 – Структура DTO для реєстрації користувача

```
@Getter
@Setter
public class RegisterDto {
    @NotEmpty(message = "Email must not be empty")
    @Email(message = "Please enter a valid email address")
    private String email;

    @Size(min=6, message = "Minimum Password length is 6
characters")
    private String password;

    @NotEmpty(message = "Please confirm your password")
    private String passwordConfirm;

    @NotEmpty(message = "Name must not be empty")
    private String name;

    @NotEmpty(message = "Surname must not be empty")
    private String surname;
}
```

Лістинг А.2 – Фрагмент контролера UserController для обробки реєстрації

```
@PostMapping("/register")
    public String register (
            Model model,
            @Valid @ModelAttribute RegisterDto registerDto,
            BindingResult result){

if(!registerDto.getPassword().equals(registerDto.getPasswordConf
irm())){
            result.addError(
                    new FieldError("registerDto",
"passwordConfirm", "Password and confirm password do not
match"));
        }

        User appUser = repo.findByEmail(registerDto.getEmail());
        if(appUser!=null){
            result.addError(
                    new FieldError("registerDto", "email",
                            "Email address is already used"));
        }
```

```
        // Якщо є помилки валідації, повернути форму реєстрації
        if(result.hasErrors()){
            return "register";
        }

        try{
            var bCryptEncoder = new BCryptPasswordEncoder();


            // Створення нового об'єкта користувача
            User newUser = new User();
            newUser.setName(registerDto.getName());
            newUser.setSurname(registerDto.getSurname());
            newUser.setEmail(registerDto.getEmail());
            newUser.setRole("client");
            newUser.setCreatedAt(new Date());

newUser.setPasswordHash(bCryptEncoder.encode(registerDto.getPass
word()));
            repo.save(newUser); // Збереження користувача

            model.addAttribute("registerDto", new
RegisterDto());
            model.addAttribute("success", true);
        } catch (Exception e) {
            result.addError(
                    new FieldError("registerDto", "name",
                            e.getMessage()));
        }

        return "redirect:/login?registered";
    }
```

Лістинг А.3 – Конфігурація безпеки Spring Security

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        return http
                .authorizeHttpRequests( auth -> auth
                        .requestMatchers("/").permitAll()
                        .requestMatchers("/images/**",
"/css/**").permitAll()
                       .requestMatchers("/register").permitAll()
                        .requestMatchers("/login").permitAll()
                        .requestMatchers("/logout").permitAll()
                        .anyRequest().authenticated()
                )
```

```
                .formLogin(form -> form
                        .loginPage("/login")
                        .usernameParameter("email")
                        .passwordParameter("password")
                        .defaultSuccessUrl("/dashboard", true)
                )
                .logout(config -> config.logoutSuccessUrl("/"))
                .build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(); // Використання
BCrypt алгоритму для хешування паролів
    }
}
```

Лістинг А.4 – Фрагмент контролера CourseController для перегляду курсу

```
@GetMapping("/courses/{id}")
public String viewCourse(@PathVariable Long id, Model model,
Principal principal) {
    Course course = courseRepository.findById(id)
            .orElseThrow(() -> new
IllegalArgumentException("Invalid course Id: " + id));
    List<Section> sections =
sectionService.getSectionsByCourseId(id);
    model.addAttribute("course", course);
    model.addAttribute("sections", sections);

    if (principal != null) {
        User user =
userService.getByEmailOrThrow(principal.getName());

        // Збираємо всі уроки курсу
        List<Lesson> allLessons = sections.stream()
                .flatMap(section ->
section.getLessons().stream())
                .toList();

        // Підрахунок пройдених уроків
        long completedCount = allLessons.stream()
                .filter(lesson ->
userProgressService.isLessonCompleted(user, lesson))
                .count();

        model.addAttribute("completedLessons", completedCount);
        model.addAttribute("totalLessons", allLessons.size());
    }
    return "course_view";
}
```

Лістинг А.5 – Фрагмент LessonController для відображення уроків

```
@GetMapping("/lessons/{lessonId}/pages/{pageIndex}")
    public String viewLessonPage(@PathVariable Long lessonId,
                                 @PathVariable int pageIndex,
                                 Model model, Principal
principal) {
        Lesson lesson = lessonService.getLessonById(lessonId);
        List<LessonPage> pages = lesson.getPages()
                .stream()

.sorted(Comparator.comparingInt(LessonPage::getPosition))
                .toList();

        if (pageIndex < 0 || pageIndex >= pages.size()) {
            return "redirect:/lessons/" + lessonId + "/pages/0";
        }

        LessonPage currentPage = pages.get(pageIndex);

        if (principal != null) {
            User user =
userService.getByEmailOrThrow(principal.getName());
            userProgressService.updateProgress(user, lesson,
pageIndex, pages.size());
        }

        model.addAttribute("lesson", lesson);
        model.addAttribute("page", currentPage);
        model.addAttribute("currentIndex", pageIndex);
        model.addAttribute("totalPages", pages.size());

        if (currentPage.getPageType() == PageType.QUIZ) {
            QuizQuestionDTO quiz =
QuizQuestionDTO.fromContent(currentPage.getContent());
            model.addAttribute("quiz", quiz);
        }

        if (currentPage.getPageType() == PageType.CODE_TASK) {
            CodeTaskDTO task =
CodeTaskDTO.fromContent(currentPage.getContent());
            model.addAttribute("codeTask", task);
        }

        return "page_view";
    }
```

Лістинг А.6 – Фрагмент HTML-шаблону для виводу прогресу

```
<div class="flex items-center justify-between font-mimopro font-
medium text-xs text-product2-content-secondary">
    <h4
```

```
th:text="${#numbers.formatDecimal(progress.completedLessons *
100.0 / progress.totalLessons, 0, 'POINT', 0, 'COMMA') + '%
complete'}">0% complete</h4>
    <h4>
        <span th:text="${progress.completedLessons}">0</span> /
        <span th:text="${progress.totalLessons}">0</span>
    </h4>
</div>
```

Лістинг А.7 – Фрагмент коду методу updateProfile в ProfileController

```
@PostMapping("/profile")
public String updateProfile(@Valid @ModelAttribute("user") User
updatedUser, BindingResult bindingResult, Principal principal,
Model model, RedirectAttributes redirectAttributes) {

    if (bindingResult.hasErrors()) {
        return "profile";
    }
    User user = userRepository.findByEmail(principal.getName());
    user.setName(updatedUser.getName());
    user.setSurname(updatedUser.getSurname());
    userRepository.save(user);

    redirectAttributes.addFlashAttribute("success", "Changes
saved successfully!");
    return "redirect:/profile";
}
```

Лістинг А.8 – Фрагмент коду методу changePassword в ProfileController

```
@PostMapping("/profile/security")
public String changePassword(@Valid
@ModelAttribute("passwordChangeDto") PasswordChangeDto form,
                             BindingResult bindingResult,
                             Principal principal,
                             Model model,
                             RedirectAttributes
redirectAttributes) {

    String email = principal.getName();
    User user = userRepository.findByEmail(email);


    // Перевірка відповідності поточного пароля
    if (!passwordEncoder.matches(form.getCurrentPassword(),
user.getPasswordHash())) {
        bindingResult.rejectValue("currentPassword", null,
"Incorrect current password");
    }
```

```
    // Перевірка відповідності паролів
    if
(!form.getNewPassword().equals(form.getConfirmPassword())) {
        bindingResult.rejectValue("confirmPassword", null,
"Passwords do not match");
    }

    if (bindingResult.hasErrors()) {
        model.addAttribute("user", user);
        return "profile_security";
    }

user.setPasswordHash(passwordEncoder.encode(form.getNewPassword(
)));
    userRepository.save(user);

    redirectAttributes.addFlashAttribute("success", "Password
changed successfully.");
    return "redirect:/profile/security";
}
```