Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Веб-платформа для покращення навичок програмування через надання завдань із кодування
(тема)

Виконав:

студент 4 курсу, групи ПЗПІ-20-5

Цісаренко О. І.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Побіженко І.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Цісаренко Олександру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-платформа для покращення навичок програмування через надання завдань із кодування
Затверджена наказом по університету від 20.05.2024р. № 471 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи Розробити програмний застосунок із серверною частиною для вирішення проблеми недостатнього розвитку навичок програмування серед розробників. Розробити проект за допомогою .NET 6 (LTS), Angular, EF Core, HealthChecks, Polly, Dapper, Grpc, Docker, RabbitMQ, MS SQL Server.
4. Перелік питань, що потрібно опрацювати в роботі Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	10.04.2024	*виконано*
2	Створення специфікації ПЗ	12.04.2024	*виконано*
3	Проектування ПЗ	15.03.2024	*виконано*
4	Розробка ПЗ	26.05.2024	*виконано*
5	Тестування ПЗ	28.05.2024	*виконано*
6	Оформлення пояснювальної записки	29.05.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Оцінка роботи рецензентом, отримання відзиву від керівника кваліфікаційної роботи, попередній захист роботи та проходження нормо-контролю	08.06.2024 - 15.06.2024	*виконано*
9	Здача роботи у електронний архів, допуск роботи до захисту завідувачем кафедри	15.06.2024	*виконано*
10	Захист кваліфікаційної роботи	18.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Цісаренко О.І.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Побіженко І.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 61 стор., 30 рис., 7 джерел.

ЗАДАЧІ, ПЛАТФОРМА, НАВИЧКИ, АВТОМАТИЗАЦІЯ, ПЕРЕВІРКА, БАЗИ ДАНИХ, ПЛАТФОРМА ASP.NET, MVC, LINQ, АДМІНІСТРУВАННЯ, СТАТИСТИКА, МОДЕЛІ, C#, ANGULAR, EF CORE.

Об’єктом даної розробки є реалізація навчальної системи з програмування із клієнтською та серверною частиною з архітектурою використання мікро-сервісів.

Мета розробки – створити програмну систему із серверною частиною для покращення навичок програмування через надання завдань з кодування.

Метод рішення – середовище розробки Visual Studio, Visual Studio Code, MSSQL-Studio, мови програмування C# та TypeScript.

У результаті розробки було створено програмну систему із серверною частиною для проходження різного рівня задач задля покращення навичок програмування, запису дій кожного користувача щодо його прогресу, менеджмент задач та їх рівня складності.

TASKS, PLATFORM, SKILLS, AUTOMATION, VERIFICATION, DATABASES, ASP.NET PLATFORM, MVC, LINQ, ADMINISTRATION, STATISTICS, MODELS, C#, ANGULAR, EF CORE.

The object of this development is the implementation of a programming training system with a client and server part using microservices architecture.

The purpose of the development is to create a software system with a server part to improve programming skills by providing coding tasks.

The solution method is Visual Studio development environment, Visual Studio Code, MSSQL-Studio, C# and TypeScript programming languages.

As a result of the development, a software system with a server part was created to complete different levels of tasks to improve programming skills, record the actions of each user regarding his progress, manage tasks and their level of complexity.

Я, Цісаренко Олександр Ігорович, студент гр. ПЗПІ-20-5, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Веб-платформа для покращення навичок програмування через надання завдань з кодування», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

PM – Project Manager

LTS – Long Term Support

VS Code – Visual Studio Code

HTTP – Hypertext Transfer Protocol

MVD – Model-View-Delegate

SDK – Serial Development Kit

TCP – Transmission Control Protocol

WPF – Windows Presentation Foundation

XML – Extensible Markup Language

MS SQL – Microsoft Structured Query Language

ВСТУП

В сучасному світі, де технології швидко розвиваються, важливість володіння навичками програмування стає необхідністю. Програмування вже давно перестало бути виключною сферою інженерії, стаючи важливим інструментом у багатьох професіях, від маркетингу до наукових досліджень. Однак, навчитися програмувати може бути викликом для багатьох, особливо для початківців.

У цьому контексті виникає потреба в ефективних засобах для навчання програмуванню, які б допомагали студентам і професіоналам розвивати свої навички у зручний та захоплюючий спосіб. Самостійне вирішення завдань з програмування, аналогічно до платформи CodeWars чи LeetCode, є одним із ефективних методів вдосконалення вмінь у цій області.

Ця робота присвячена розробці веб-платформи, спрямованої на полегшення процесу вивчення та покращення навичок програмування шляхом надання користувачам різноманітних завдань з кодування. Основна мета полягає в створенні інтерактивного середовища, що сприятиме розвитку програмістської культури та підвищенню рівня компетентності у програмуванні серед широкого кола користувачів, перевіряючи надані рішення до завдань на правильність та оцінюючи загальний розвиток програміста у напрямку їх вирішення.

Проект включатиме вивчення сучасних тенденцій у галузі навчання програмуванню, а також технічну реалізацію програмної платформи. Робота має на меті створення інструменту, що не лише надає користувачам можливість вдосконалювати свої навички, але й забезпечить їхню мотивацію та зацікавленість у процесі навчання.

Очікується, що результати цього дослідження сприятимуть створенню ефективної веб-платформи, яка стане важливим інструментом у навчанні програмуванню та розвитку ІТ-спільноти на багатьох рівнях освіти в Україні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

У контексті розробки веб-платформи для покращення навичок програмування через надання завдань з кодування важливим аспектом є розуміння культури самоосвіти в ІТ-середовищі. Існуючі практики та підходи до самонавчання у програмуванні визначаються не лише самими завданнями, але й способами їх виконання та наданням зворотного зв'язку.

Основні аспекти культури самонавчання у програмуванні:

– вибір навчальних ресурсів: успішне навчання програмуванню починається з вибору відповідних матеріалів та ресурсів. Це може бути відкриті онлайн-курси, підручники, відео-уроки, або спільноти програмістів, де можна обмінюватися досвідом та отримувати поради;

– систематичність та дисципліна: ефективне навчання вимагає систематичного підходу та дисципліни. Постійна практика, навіть якщо це невеликі кроки щодня, допомагає закріпити знання та розвивати навички;

– зворотний зв'язок та взаємодія: важливим елементом культури самонавчання є можливість отримання зворотного зв'язку та взаємодії з іншими учасниками. Це може бути у вигляді обговорення завдань, взаємної допомоги та взаємного навчання у спільнотах програмістів;

– ведення pet-проекту: задля закріплення своїх навичок початківець повинен писати власний pet-проект, який буде вдосконалюватись протягом досить довгого часу, таким чином створюючи своє портфоліо задля показу на співбесіді на вакансію.

Розробка веб-платформи для покращення навичок програмування повинна враховувати ці аспекти, створюючи зручне та стимулююче середовище для самонавчання програмістів будь-якого рівня. Платформа повинна надавати доступ до різноманітних навчальних ресурсів, сприяти систематичному навчанню та створювати можливості для взаємодії та обміну досвідом між користувачами.

Для веб-платформи, спрямованої на покращення навичок програмування через надання завдань з кодування, останній етап також має вирішальне значення.

Завершення розробки проекту включає ряд ключових дій, що забезпечують успішну реалізацію проекту та його подальший розвиток.

Отже, завершальний етап розробки проекту включає такі ключові аспекти:

– завершення робіт та документування проекту: у цьому етапі виконуються останні роботи з розробки та дописується увесь код проекту. Крім того, створюється документація, яка описує функціональні можливості, архітектуру та інші важливі аспекти проекту;

– оцінка виконання поставлених завдань та аналіз результатів: проводиться оцінка виконання завдань, визначається відповідність результатів поставленим цілям та вимогам. Проводиться аналіз отриманих результатів для виявлення сильних та слабких сторін проекту та можливостей для подальшого вдосконалення;

– передача знань та документації зацікавленим особам: забезпечується передача знань та документації користувачам, адміністраторам та іншим зацікавленим особам. Це може включати проведення навчальних семінарів, створення довідкової документації та інструкцій з використання платформи.

1.2 Аналіз конкурентів

Аналізуючи ринок платформ для покращення навичок програмування, можна виділити кілька основних продуктів, які вже мають певну репутацію та користувацьку базу. Серед них – Project Euler, CodeWars, LeetCode, та HackerRank. Кожна з цих платформ має свої переваги та недоліки, які визначають їхню популярність серед програмістів з різним досвідом та інтересами.

Project Euler - це платформа, спрямована на розв'язання математичних задач за допомогою програмування. Вона пропонує складні математичні завдання, які вимагають креативного підходу та ефективних алгоритмів (див. рис. 1.2.1).

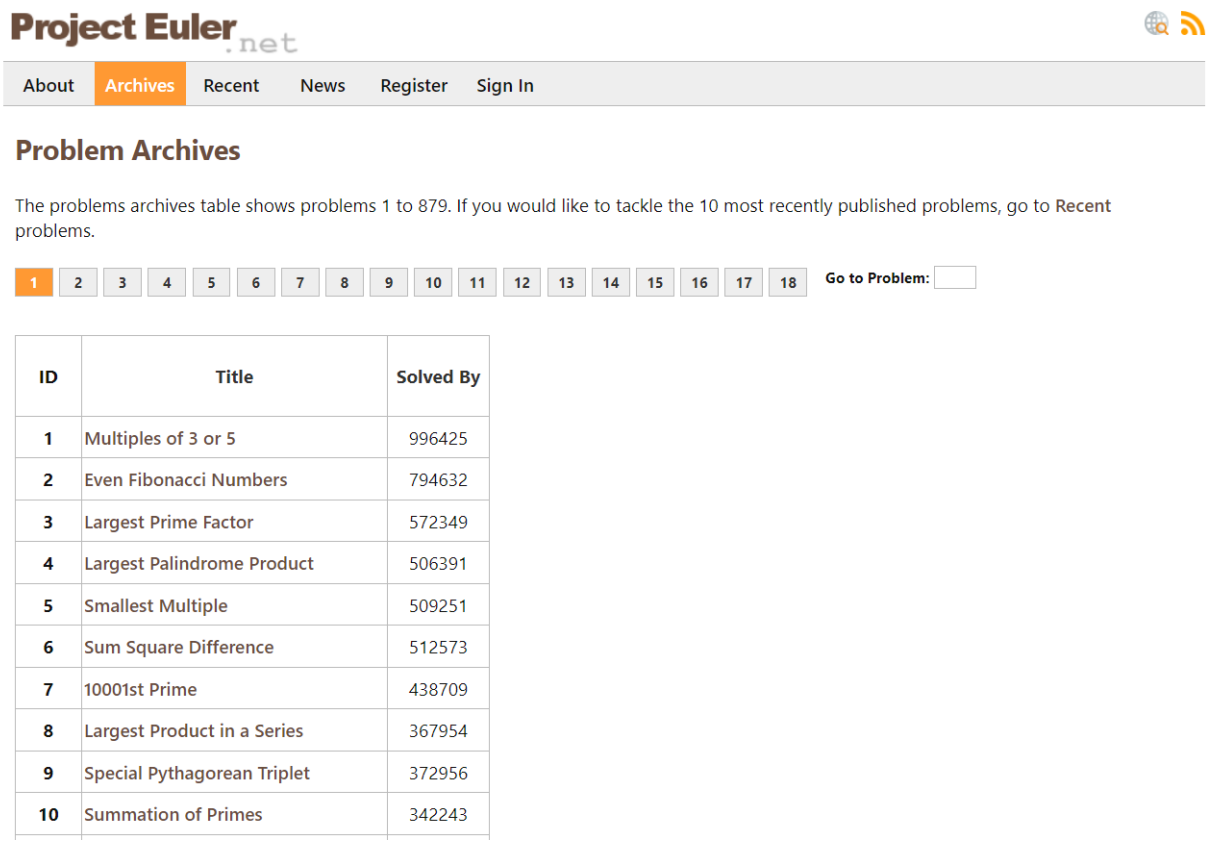


Рисунок 1.2.1 – Інтерфейс платформи Project Euler

CodeWars - це інтерактивна платформа, яка надає набір завдань з програмування та алгоритмів у вигляді "кат" (challenges). Користувачі можуть вирішувати ці завдання, використовуючи будь-яку мову програмування, та отримувати рівень "кю" (kyu) за кожне вирішене завдання (див. рис. 1.2.2). Тут є таблиця учасників за весь день, за увесь чат, наявний чат для спілкування на сервері такої платформи як Discord, а також є можливість оцінювати код інших користувачів платформи на наявність проблем з рефакторингу та задля оптимізації рішень.

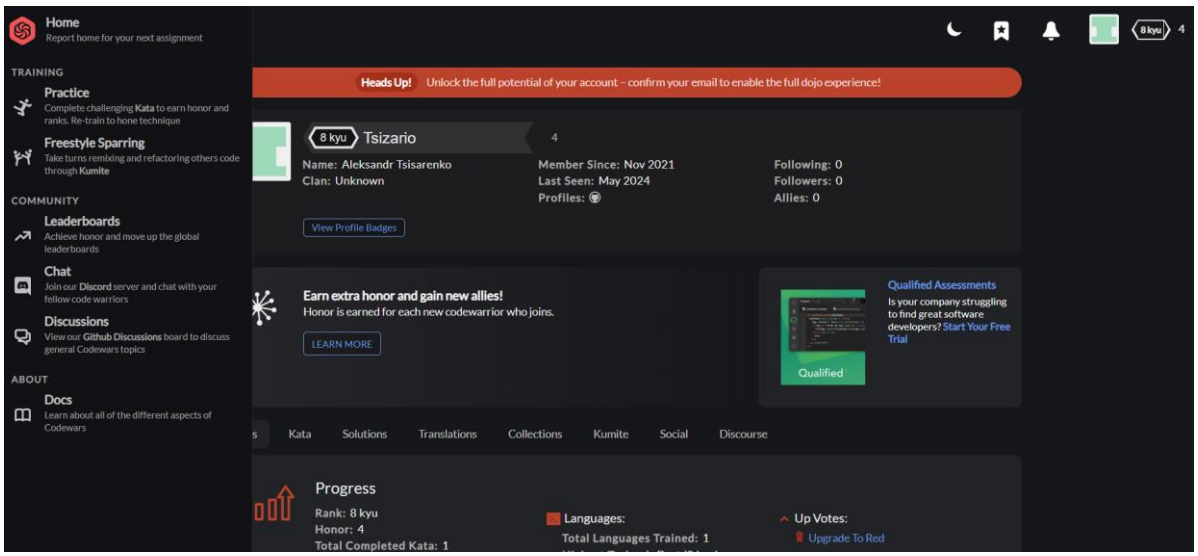


Рисунок 1.2.2 – Інтерфейс платформи CodeWars

LeetCode - це платформа, спеціалізована на підготовці до технічних співбесід та вирішенні задач з програмування. Вона містить велику колекцію завдань з алгоритмів, структур даних та інших тем, які часто зустрічаються на співбесідах у технологічних компаніях (див. рис. 1.2.3). Платформа дозволяє записувати активність аккаунту, які задачі було вирішено за рівнями складності, загальне місце у рейтингу платформи. Також періодично проводяться змагання з програмування, приз з яких буде підписка на преміум-аккаунт із доданими можливостями щодо більш прискореної перевірки завдання, пріоритет від підтримки платформи та інше.

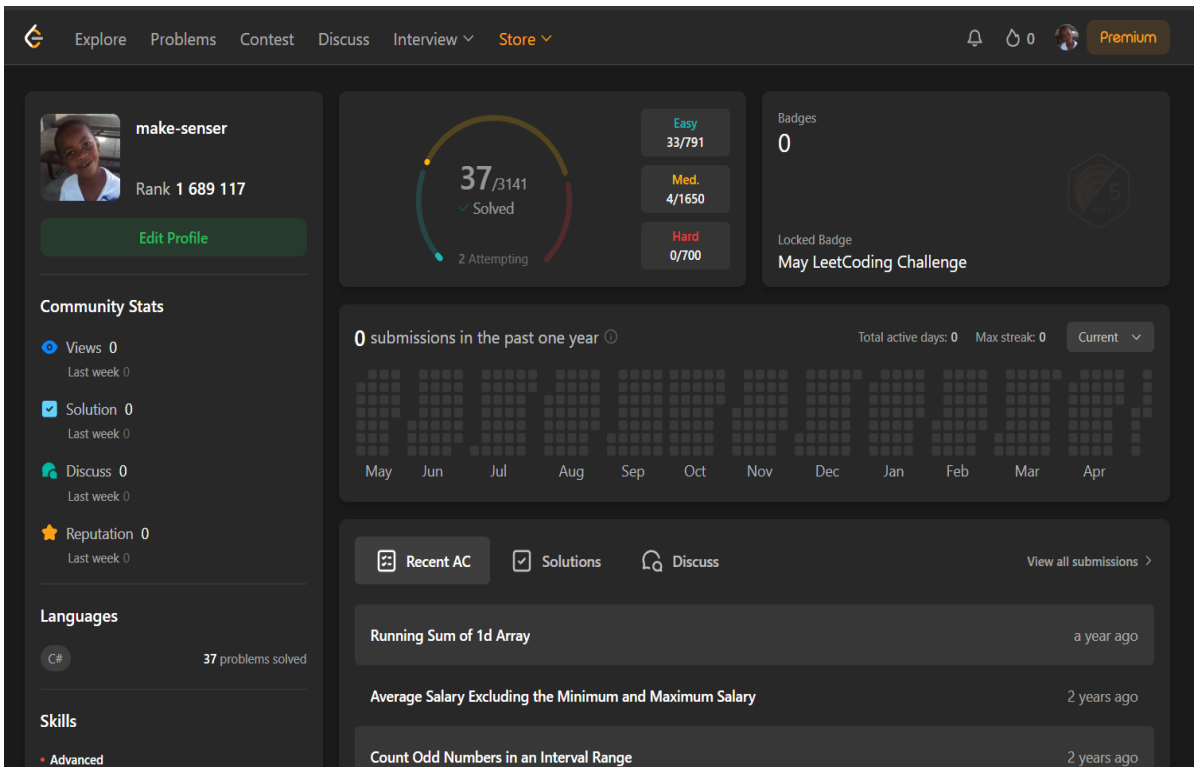


Рисунок 1.2.3 – Інтерфейс платформи LeetCode

Зараз розглянемо плюси та мінуси порівняно із нашим проектом, виходячи із розглянутого функціоналу цих платформ.

Переваги Project Euler:

– зосередженість на математичних задачах: Project Euler спеціалізується на задачах з математики, що дає користувачам можливість розвивати свої навички в цій області через програмування. Вирішення математичних задач може поліпшити абстрактне мислення та логічні вміння;

– велика колекція завдань: сайт містить велику кількість різноманітних математичних завдань, починаючи від простих до дуже складних. Це дає користувачам можливість поступово підвищувати рівень складності і розвивати свої навички поступово;

– сприяння креативності та ефективних алгоритмів: задачі часто вимагають креативного підходу та розробки ефективних алгоритмів для їх вирішення. Це дозволяє користувачам розвивати свої аналітичні та алгоритмічні навички.

Недоліки:

– обмежена спеціалізація: оскільки Project Euler фокусується виключно на математичних задачах, він може бути менш привабливим для тих, хто шукає різноманітніші завдання з програмування;

– не підходить для початківців: деякі задачі на платформі можуть бути дуже складними для початківців у програмуванні або математиці. Це може викликати відчуття розчарування та знизити мотивацію користувачів;

– відсутність інтерактивності: платформа в основному пропонує текстові описи задач та очікує від користувачів написання власного коду для їх вирішення. Деяким користувачам може бракувати інтерактивності, яку надають інші платформи з програмуванням.

Переваги CodeWars:

– широкий вибір завдань: платформа пропонує великий і різноманітний набір завдань з програмування та алгоритмів. Вони варіюються за складністю і тематикою, що дозволяє користувачам з різним рівнем досвіду знаходити відповідні завдання для вирішення;

– можливість використання будь-якої мови програмування;

– система рейтингу "кю": рейтинг "кю" (kyu), який визначає рівень складності завдань та досвід користувача. Вирішуючи завдання, користувачі отримують "кю", що дозволяє вимірювати їхній прогрес та зростання;

– спільнота та обмін досвідом: платформа має активну спільноту користувачів, яка обговорює завдання, надає поради та допомагає один одному у вирішенні складних проблем. Це створює можливість для співпраці та обміну досвідом між програмістами.

– відсутність структурованих курсів: проект не надає структурованих курсів або програм навчання. Він складається з окремих завдань, які користувачі вибирають самостійно. Деяким користувачам може бракувати структурованого підходу до навчання;

– вимога до самостійності: вирішення завдань вимагає від користувачів самостійного розв'язання проблеми та написання відповідного коду;

– можливість "застрягнути" на складних завданнях: оскільки платформа має широкий вибір завдань різної складності, користувачі можуть "застрягнути" на деяких складних завданнях без належного підходу до розв'язання.

Переваги LeetCode:

– проходження завдань у вигляді блоків за темою, яка цікавить користувача ресурсу;

– підготовка до технічних співбесід: LeetCode часто використовується для підготовки до технічних співбесід та інтерв'ю в IT-компаніях. Вирішення завдань на цій платформі допомагає користувачам покращити свої навички програмування та алгоритмічного мислення;

– активна спільнота: платформа має велику та активну спільноту користувачів, яка обговорює завдання, надає поради та допомагає один одному будь-якою мовою програмування;

– змагання від розробників продукту: можливість прийняти участь у змаганнях від LeetCode команди та отримати можливість безкоштовно отримати преміум-акаунт.

Недоліки:

– велика різноманітність завдань на алгоритми: це сайт, що спрямований переважно на алгоритмічні завдання, а не на практичне застосування програмування у реальних проектах;

– відсутність персоналізованого навчання: не надає персоналізованих курсів або програм навчання;

– деякий функціонал можливий лише із придбанням преміум-підписки.

Під час розробки вимог до нашої веб-платформи ми ретельно вивчили функціонал і особливості інших схожих проектів. Це важливо, оскільки намагалися виявити, що робить ці платформи привабливими для користувачів та які

можливості можна покращити або впровадити у нашій системі для забезпечення конкурентної переваги.

Наприклад, ми звернули увагу на потребу підвищення гнучкості у налаштуваннях для користувачів, щоб кожен міг налаштувати платформу під свої індивідуальні потреби та вподобання. Крім того, ми розглядали можливості покращення аналітичних засобів, щоб надати користувачам більше інформації про їхні успіхи та прогрес у вирішенні завдань. І, звичайно, ми працювали над зниженням вартості для кінцевого користувача, щоб зробити нашу платформу більш доступною та привабливою для широкого кола людей, які хочуть покращити свої програмістські навички.

1.3 Виявлення та вирішення проблем

Перед проектуванням вимог до нашої веб-платформи для покращення навичок програмування ми детально проаналізували основні проблеми, з якими стикаються студенти, початківці та досвідчені програмісти. Це важливо, оскільки намагаємося визначити, які аспекти можна вдосконалити або впровадити у нашу систему для забезпечення її ефективності та привабливості для користувачів.

Отже, ми звернули увагу на проблему неефективного навчання програмуванню через відсутність структурованих завдань та засобів моніторингу прогресу. Також було розглянуто можливості поліпшення механізмів контролю та оцінки навичок програмування для різних рівнів користувачів. Крім того, ми зосередились на створенні спільноти користувачів, де можна обмінюватися досвідом та порадами щодо вирішення складних завдань.

Ці проблеми стосуються багатьох людей, що прагнуть покращити навички програмування незалежно від рівня досвіду. Тому наша платформа покликана надати рішення для цих проблем шляхом створення структурованого та ефективного середовища для навчання та практики програмування.

1.4 Постановка задачі

Метою даної кваліфікаційної роботи є розробка програмної системи для тренування навичок програмування програмістами з різними рівнями навичок. Система покликана автоматизувати моніторинг процесу вирішення задач, аналізу цих рішень, підвищити навички програмування та культивувати самостійний розвиток програмістів.

Для досягнення цієї мети необхідно розв'язати наступні завдання:

– провести дослідження існуючих платформ для навчання програмування, аналіз потреб цільової аудиторії та визначення вимог до системи;

– розробити технічні та функціональні вимоги до системи, враховуючи потреби користувачів;

– створити архітектурну схему, визначення ключових модулів та компонентів системи;

– спроектувати бази даних для зберігання інформації про користувачів, завдання та їхній прогрес;

– розробити серверну і клієнтську частини системи: архітектура серверних сервісів та розробка користувацького інтерфейсу;

– виконати тестування розробленої системи: тестування різних модулів програми для забезпечення стабільності та надійності системи;

– розгорнути систему на хостингу: забезпечення доступності системи для кінцевих користувачів та додати інтеграцію з іншими хмарними сервісами.

Вирішення цих завдань дозволить створити зручну та інноваційну платформу для навчання програмування, яка допоможе користувачам підвищити їхні навички та зростати як професіонали.

3 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

3.1 Функціональні вимоги

Під час розробки було реалізовано весь функціонал клієнтської частини, перевірку взаємодії клієнтської із серверною частиною, створено інтерфейси для передачі, збереження та читання даних. Клієнтська частина не має зберігати будь-які дані користувача у браузері; відображення даних має відбуватися виключно через обробку запиту серверною частиною та передачу відповіді гостю або авторизованому користувачеві.

Основні вимоги щодо взаємодії між клієнтською та серверною частинами включають наступне:

– клієнтська частина не має передавати ідентифікатор користувача, якщо він не авторизований;

– клієнтська частина повинна автоматично обмежувати доступ для гостя до функціоналу, який потребує автентифікації;

– клієнтська частина має надсилати запит на оновлення даних користувача до серверної частини при оновленні сторінки браузера.

В системі передбачити можливість підтримки 3 груп користувачів: Гість, Користувач, Адміністратор.

Для детального розгляду майбутнього функціоналу системи необхідно чітко сформувати вимоги функціональні та не функціональні для охоплення всього спектру завдань задля надійного проекту та персоналу.

Функціональні вимоги:

– можливість додавання нових завдань для користувачів з можливістю визначення їхньої складності та тематики;

– реєстрація користувачів та створення особистих профілів з можливістю збереження історії вирішених завдань та досягнень;

– зберігання основної інформації про користувача, такої як логін, улюблена мова програмування, скільки задач вирішено, кількість днів, які провів користувач на платформі;

– можливість системи автоматично оцінювати та перевіряти рішення користувачів на відповідність завданню;

– адміністративний доступ: функціонал для адміністраторів системи для керування користувачами, модерування завдань та контролю за безпекою;

– інтуїтивно зрозумілий користувацький інтерфейс з легкою навігацією;

– підтримка інтерфейсу з багатьма мовами;

– захист даних за допомогою шифрування;

– реалізація політик доступу та ролей для користувачів.

Нефункціональні вимоги:

– забезпечити конфіденційність та цілісність даних користувачів та завдань;

– гість не повинен мати можливість створювати та редагувати інформацію у системі;

– провести оптимізацію швидкодії системи для мінімізації часу відгуку та завантаження сторінок, тобто максимальний час відповіді системи на запит користувача не повинен перевищувати 5 секунд;

– оновлення компонентів клієнтської частини без перезавантаження сторінки;

– система повинна ефективно працювати з ростом кількості користувачів та завдань;

– забезпечити сумісність системи з різними браузерами та пристроями;

– зробити додавання нового функціоналу у безперебійному режимі роботи платформи;

– надати можливість перекладу інтерфейсу системи на різні мови для зручності користувачів із різних країн.

Наявність цих вимог допоможе забезпечити розробку надійної, ефективної, сучасної та зручної для користувачів веб-платформи для навчання програмування шляхом вирішення завдань.

3.2 Припущення та залежності

Припущення – це фактори, які вважаються істинними або реалізованими для цілей планування проекту, але які мають потенціал нести ризики, якщо вони не відповідають дійсності. Розглянемо ключові припущення:

– припускається, що користувачі будуть мати базові навички з програмування, використання комп'ютерів та веб-додатків для ефективного користування платформою;

– припускається, що початківці та досвідчені програмісти будуть активно використовуватимуть систему після її впровадження і забезпечать достатньо інформації для ефективного моніторингу та управління;

– припускається, що платформа буде потребувати регулярного технічного обслуговування та оновлень для забезпечення її надійності та актуальності;

– припускається, що забезпечення безпеки даних та відповідність правилам конфіденційності буде критичними аспектами під час впровадження та експлуатації платформи;

– припускається, що користувачі будуть відкриті для використання веб-платформи для поліпшення своїх навичок програмування.

Залежності – це зовнішні або внутрішні фактори, на які проект спирається для свого успішного завершення. Розглянемо залежності платформи:

– вибір технологічного стеку для розробки, включаючи серверну платформу, бази даних та фреймворки для клієнтської частини, буде визначальним фактором для успішності проекту;

– постійний та надійний доступ до Інтернету буде необхідним для забезпечення функціонування веб-платформи та доступу користувачів до неї;

– інтеграція з існуючими обліковими системами, такими як Google Account, Microsoft Account та інше може стати важливою для оптимізації робочих процесів користувачів.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проектування UML

Під час проектування системи у якості архітектурного рішення було обрано клієнт-серверну архітектуру, яка є одним із архітектурних шаблонів програмного забезпечення та панівною концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Завдяки тому, що компоненти системи з такою архітектурою можуть виконуватись на різних вузлах, виконуючи серверні та клієнтські функції. Така архітектура дозволяє підвищити безпеку, надійність та продуктивність.

Для розроблюваної системи була зроблена діаграма розгортання (див. рис. 3.1.1).

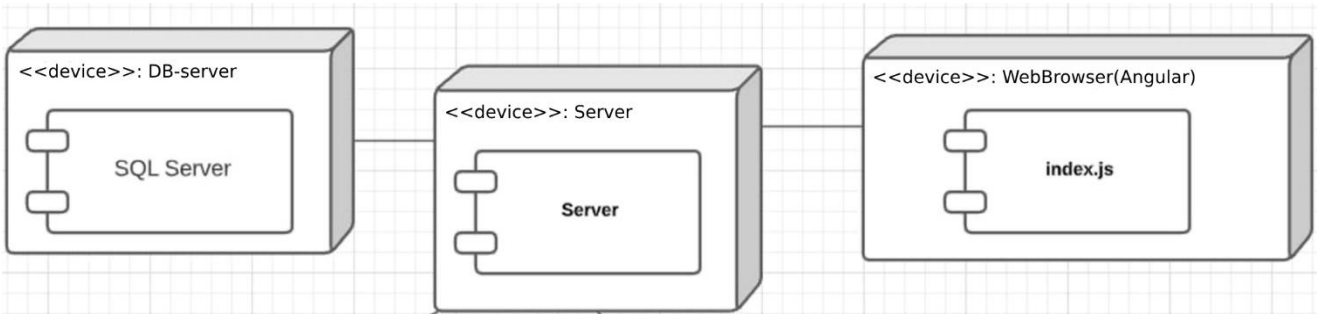


Рисунок 3.1.1 – UML діаграма розгортання проекту «Веб-платформа для покращення навичок програмування через надання завдань з кодування»

Уся система складається з 3 компонентів: сервер, веб-додаток, база даних. На сервері знаходиться back-end програмного продукту та виконується більша частина бізнес-логіки. Веб-додаток дозволяє користувачам мати доступ до системи, взаємодіяти з проектом через зрозумілий для людини інтерфейс. База даних зберігає основну інформацію про користувачів, завдання з кодування, рішення задач та спроби їх виконання зареєстрованими користувачами.

У нашій веб-орієнтованій програмній системі для відстеження активності працівників на підприємстві задіяно три основні типи користувачів, кожен з яких має унікальний набір ролей та можливостей (див. рис. 3.1.2), а саме:

– гість;

– користувач;

– адміністратор.

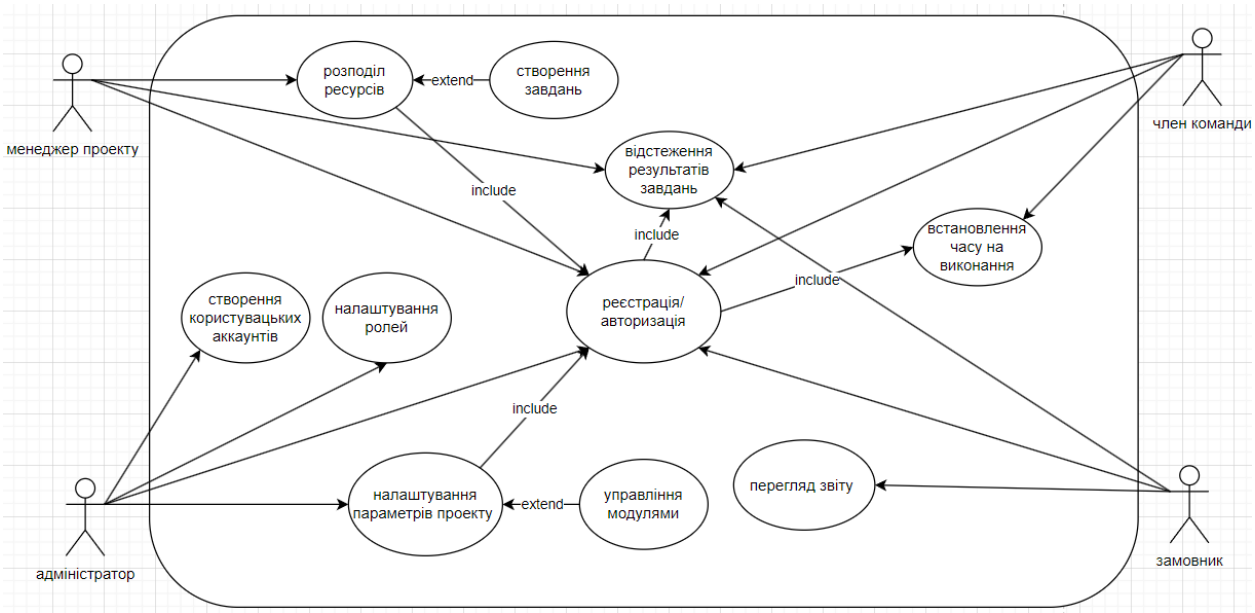


Рисунок 3.1.2 – Use Case діаграма

Далі розберемо більш детально можливості користувачів системи:

– перегляд головної сторінки та можливість авторизуватися (гість);

– створення та управління користувачами (адміністратор):

– створення задач та вирішення для них (адміністратор);

– створення користувацьких акаунтів (користувач);

– конфігурація системи (адміністратор):

– налаштування параметрів системи (користувач);

– вирішення задач (користувач);

– спроба перевірки виконання задачі на правильність (користувач);

– перегляд спроб на вирішення завдання (користувач);

– відстеження прогресу завдань (користувач):

– запис часу, витраченого на завдання (користувач);

– співпраця та комунікація (користувач, адміністратор):

– використання форумів та системи коментарів для обговорення рішення задач (користувач, адміністратор).

Ці ролі та обов'язки користувачів формують основу взаємодії з системою і визначають, яким чином кожен тип користувачів використовує можливості платформи для досягнення своїх цілей.

3.2 Проектування архітектури серверної частини

Наша програмна серверна система ґрунтується на технологіях .NET 7 (LTS) [1], Entity Framework Core (EF Core), базах даних Microsoft SQL Server (MSSQL) [2], фреймворк управління ролями ASP.NET Identity Management, бібліотека логування Serilog, gRPC, посередник повідомлень RabbitMQ та PostrgreSQL.

.NET 7 (LTS) є стабільною версією платформи .NET, яка гарантує тривалий період підтримки від Microsoft. Вона надає широкі можливості для створення додатків для різних платформ із застосуванням спільної бази коду та покращує продуктивність розробки.

Entity Framework Core (EF Core) є модулем доступу до даних для .NET, який дозволяє зручно працювати з базами даних, використовуючи об'єктно-орієнтований підхід. Він спрощує взаємодію з базою даних, абстрагуючи розробників від деталей SQL запитів та забезпечуючи більшу концентрацію на бізнес-логікі додатків.

Microsoft SQL Server (MSSQL) є потужною та надійною системою управління реляційними базами даних, яка широко використовується для зберігання, обробки та аналізу даних у корпоративних додатках. Вона забезпечує високу продуктивність та безпеку обробки даних.

ASP.NET Identity - це система аутентифікації та авторизації для платформи ASP.NET, яка надає потужні засоби для управління користувачами, їхніми ролями та правами доступу в веб-додатках. Вона дозволяє реалізувати функціонал аутентифікації через різні методи, такі як пароль, соціальні мережі або зовнішні постачальники, такі як Azure Active Directory. Крім того, ASP.NET Identity

забезпечує можливість керувати ролями користувачів, визначати їхні права доступу та реалізовувати інші функції управління безпекою.

Serilog - це потужна бібліотека для реєстрації подій (логування) в додатках .NET. Вона надає гнучкий та розширюваний механізм для збору, фільтрації та виведення журнальних повідомлень, що дозволяє розробникам ефективно відстежувати та аналізувати роботу своїх програм. Serilog підтримує різноманітні набори виведення, включаючи консоль, файли журналу, бази даних та інші джерела, а також має можливості для налаштування форматування повідомлень і рівнів ведення логів.

gRPC (Google Remote Procedure Call) - це відкритий механізм віддаленого виклику процедур, розроблений Google. Він базується на протоколі HTTP/2, що дозволяє швидке, ефективне та надійне взаємодіяти між різними компонентами системи. gRPC використовує простий, але потужний механізм серіалізації Protocol Buffers для обміну даними між клієнтом і сервером, що робить його особливо ефективним для великих та складних додатків.

RabbitMQ - це система повідомлень, що використовується для сприяння асинхронної комунікації між різними складовими системи (у нашому випадку – між Docker-контейнерами). Вона базується на протоколі AMQP (Advanced Message Queuing Protocol) і дозволяє ефективно обмінюватися повідомленнями між різними компонентами програмного забезпечення, що працюють в розподіленому середовищі. RabbitMQ підтримує багато різних сценаріїв використання, таких як розподілена обробка, реалізація мікросервісної архітектури, розробка схеми взаємодії між додатками та інші.

Зв'язок між цими технологіями наступний:

– використовуючи .NET 7 для створення додатків, проект може запускатися на різних операційних системах, таких як Windows, Linux та macOS. Entity Framework Core (EF Core) виступає як проміжний шар між додатком .NET і базою даних, спрощуючи управління даними. Завдяки EF Core легко створювати, читати, оновлювати та видаляти дані, використовуючи об'єктні моделі .NET;

– конфігурація з'єднання EF Core з MSSQL визначається у файлі конфігурації додатку .NET або програмним кодом. У цій конфігурації вказується рядок підключення до MSSQL, що містить необхідні параметри для доступу до конкретної бази даних;

– Entity Framework Core також дозволяє визначати моделі даних у коді .NET, які автоматично з’єднуються на відповідні таблиці та поля в MSSQL. Після визначення моделей, EF Core генерує SQL запити, які виконуються в базі даних, керуючи створенням, оновленням, видаленням та запитами даних;

– оптимізація та продуктивність в EF Core полягає в можливості використання розширених функцій MSSQL, таких як транзакції, індекси, згортання та розгортання запитів, що допомагає оптимізувати виконання запитів і підвищує продуктивність додатків. Щодо безпеки, використання EF Core дозволяє реалізувати засоби безпеки на рівні додатку, такі як перевірка вхідних даних і управління доступом на рівні рядків. Це доповнює можливості безпеки, доступні в MSSQL, такі як аутентифікація, авторизація та шифрування даних (див. рис. 3.2.1);

– використовуючи Docker, можливо ефективно розподіляти ресурси за допомогою контейнеризації окремо розроблених компонентів, що представлені у проекті, а також гарантується дуже зручний запуск всієї системи «в один клік».

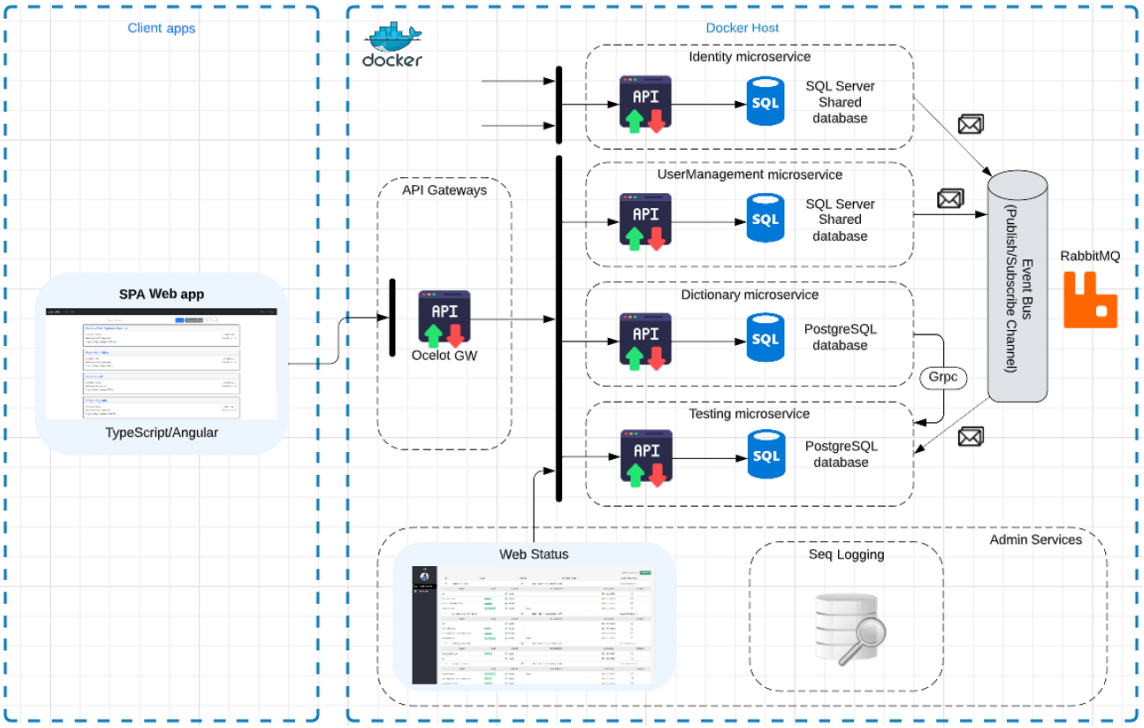


Рисунок 3.2.1 – Архітектура проекту

Виходячи з вищевказаного, програмна система розроблена з використанням кросплатформенної архітектури, яка дозволяє запускати додатки як на сервері, так і на стороні клієнта. Це досягається завдяки використанню служб .NET 7, які можуть працювати в контейнерах Linux або Windows, залежно від хосту Docker.

Архітектура системи реалізує парадигму з декількома автономними мікросервісами, кожен з яких володіє власними даними та базами даних. Кожен мікросервіс реалізує різні підходи, такі як CRUD або DDD [7], з використанням HTTP як протоколу зв'язку між клієнтськими програмами та мікросервісами.

Для забезпечення асинхронного зв'язку між мікросервісами та розповсюдження оновлень даних використовується шина подій на основі інтеграційних подій та посередника легких повідомлень RabbitMQ.

Завдяки використанню чистої архітектурі (англ. *Clean Architecture*) доменний і прикладний рівні знаходяться в центрі дизайну, відомого як ядро системи (див. рис.3.2.2). Бізнес-логіка розміщується в цих двох рівнях, хоча вони містять різні види бізнес-логіки. Вони розглядаються як деталі, і бізнес-рівні не повинні залежати від рівнів презентації та інфраструктури. Замість того, щоб бізнес-логіка

залежала від доступу до даних або інших питань інфраструктури, ця залежність інвертується: деталі інфраструктури та реалізації залежать від прикладного рівня.

Ця функціональність досягається шляхом визначення абстракцій або інтерфейсів на прикладному рівні, які потім реалізуються типами, визначеними на інфраструктурному рівні. Поширеним способом візуалізації цієї архітектури є використання серії концентричних кіл, подібних до цибулевої архітектури.

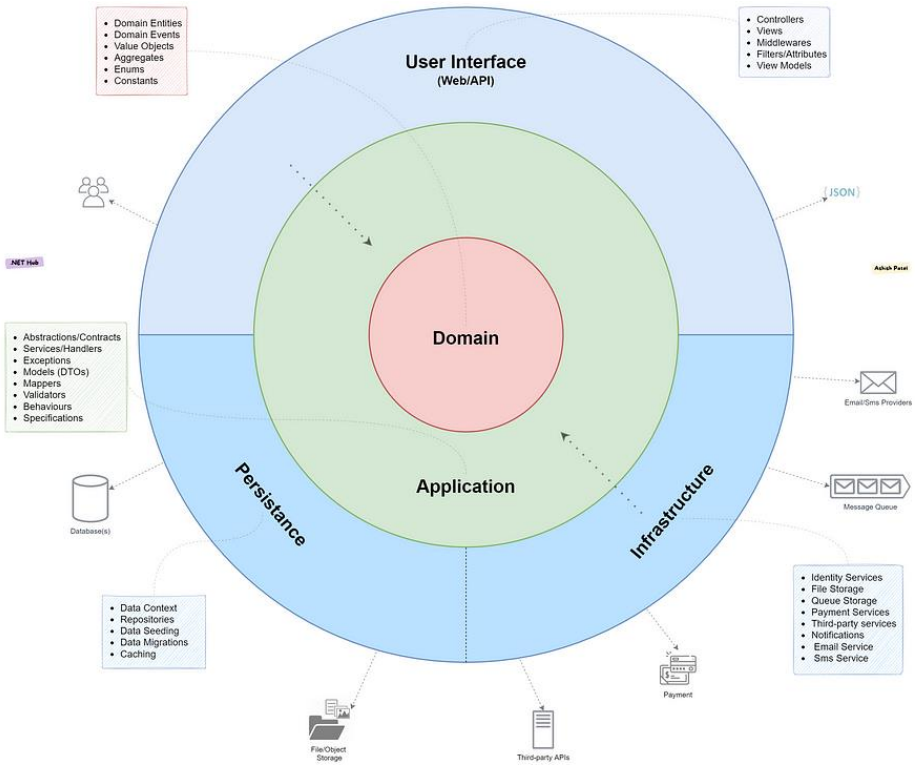


Рисунок 3.2.2 – Clean-architecture

3.3 Користувацький інтерфейс

Front-end програми реалізований з використанням фреймворку Angular [5] та базового стилю відображення, наданого bootstrap. Це дозволяє забезпечити сучасний та зручний інтерфейс для користувачів.

Для реалізації аутентифікації та авторизації в застосунку використовується OIDC Client, що забезпечує зручну роботу з ідентифікацією користувачів.

Користувачі можуть взаємодіяти з системою через інтуїтивний та дружній інтерфейс, який дозволяє виконувати потрібні дії швидко та ефективно. Він

підтримує всі необхідні функції для зручного використання програми та надає доступ до всіх функціональних можливостей, що надаються серверною частиною системи.

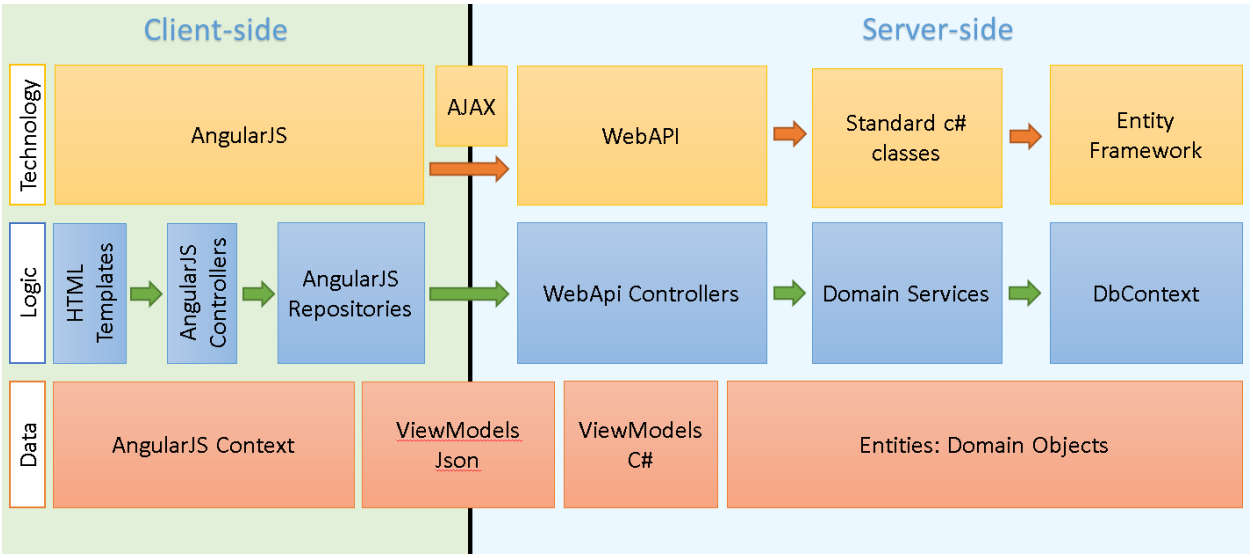


Рисунок 3.3 – Архітектура додатку на Angular

Виходячи з рисунку 3.3, ми маємо наступну архітектуру, що:

– перший ряд - це зв'язок між різними використовуваними технологіями;

– середній ряд представляє логіку нашого додатку, а останній - контейнери даних, що використовуються на цих класах.

Використовуючи попередню діаграму, легко побачити, які дані використовуються на кожному рівні.

Контролер WebAPI надсилає та отримує дані у форматі JSON. Об'єкти JSON автоматично транслюються в моделі представлення C#. WebAPI-контролери також можуть зіставляти моделі представлення з об'єктами домену, щоб використовувати служби домену. WebAPI-контролери також можуть запитувати доменні об'єкти до доменних служб і переводити їх у ViewModels c# перед відправкою назад на клієнтську сторону у форматі JSON.

Контейнери та шари даних:

– контекст AngularJS: контекст заповнюється Angular репозиторієм. До нього прив'язуються HTML-шаблони;

– ViewModels JSON: дані передаються від клієнта до сервера і навпаки у форматі JSON. За замовчуванням WebAPI-контролер автоматично надсилає дані в цьому форматі. Відправлені та отримані дані у форматі JSON автоматично зіставляються з очікуваним об'єктом;

– ViewModels C#: дані, які WebAPI-контролер надсилає та отримує як параметр. Перетворення між JSON та C# моделями представлення виконується автоматично контролером WebAPI. Цей клас є підмножиною наших об'єктів домену сутностей плюс деякі обчислювані властивості, які можуть знадобитися нашим представленням. Він може бути використаний, наприклад, для того, щоб приховати деякі дані на стороні сервера на стороні клієнта.

– сутності: сутності Entity Framework або специфічні для домену класи. Всі дані, що відносяться до домену. Ці класи повністю незалежні від подання. Сервіси домену або логіка додатку повинні використовувати ці класи. Відображення від і до сутностей до моделей представлення C# може бути виконано автоматично або за допомогою AutoMapper.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Реалізація серверної частини

На рисунку 4.1 наведено ітераційну структуру файлової системи нашого застосунку де можна побачити усі шари нашої архітектури.

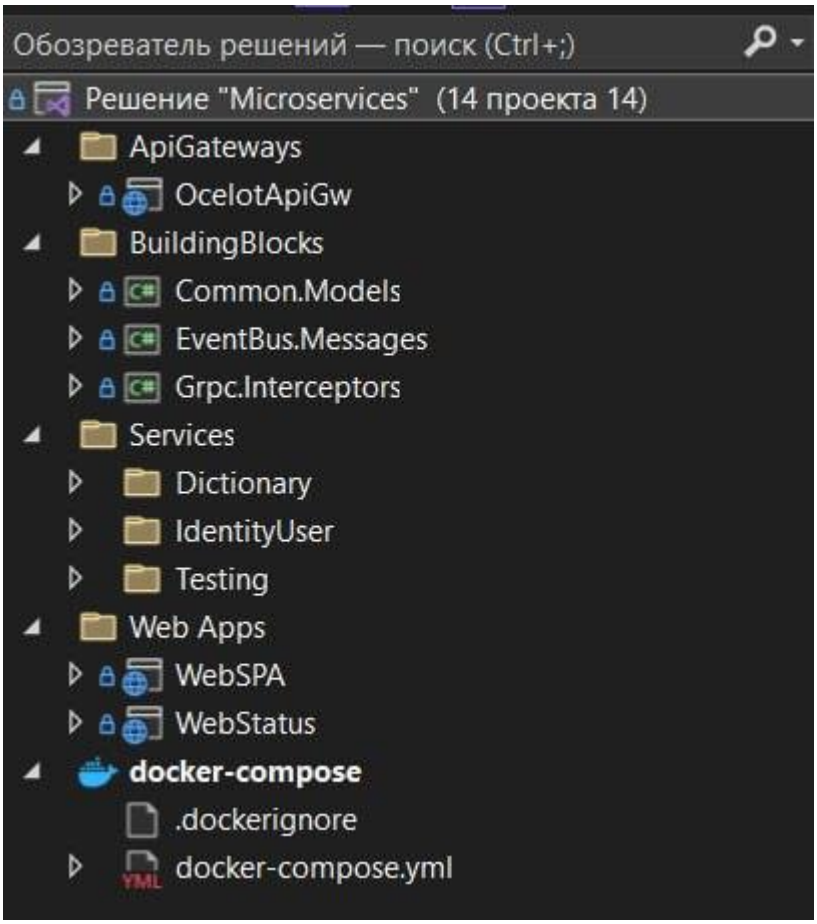


Рисунок 4.1 – Файлова система

Нижче розберемо більш детально цю структуру:

– ApiGateways: цей модуль містить конфігураційні налаштування додатку для хостингу на сервері. Файл «ocelot.json» описує, як вхідні HTTP-запити повинні бути маршрутизовані до вихідних служб на основі шляху та методу HTTP. Також він містить деякі налаштування для аутентифікації на деяких маршрутах;

– BuildingBlocks: містить моделі даних DTO [3] та інтерфейси, які використовуються для обміну даними між компонентами фронтенду та бізнес-

логікою бекенду. Моделі можуть включати класи або інтерфейси, які описують структуру даних, моделі результатів та обробку запитів;

– Services: містить сервіси для комунікації між сервером та базою даних. Містить основну логіку функціонування програми, класи, що представляють доменні моделі додатку [4], а також модуль для заповнення даними для тестування функціонування програми;

– WebApps: містить компонент WebStatus, що відповідає за сбір інформації щодо інших модулів програми, їх активності та статус виконання роботи, а також компонент WebSPA, що є клієнтським модулем з користувацьким інтерфейсом та логікою клієнта, призначений для взаємодії з користувачем та виконання основної функціональності програми;

– Docker-compose файли: налаштовують середовище для запуску, розгортання та керування додатками у вигляді Docker-контейнерів і дозволяють легко налаштовувати різні параметри та середовища для різних етапів розробки та випуску програмного забезпечення.

4.2 Проектування бази даних

Проектуючи базу даних, було створено наступну ER-діаграму проекту (див. рис. 4.2).

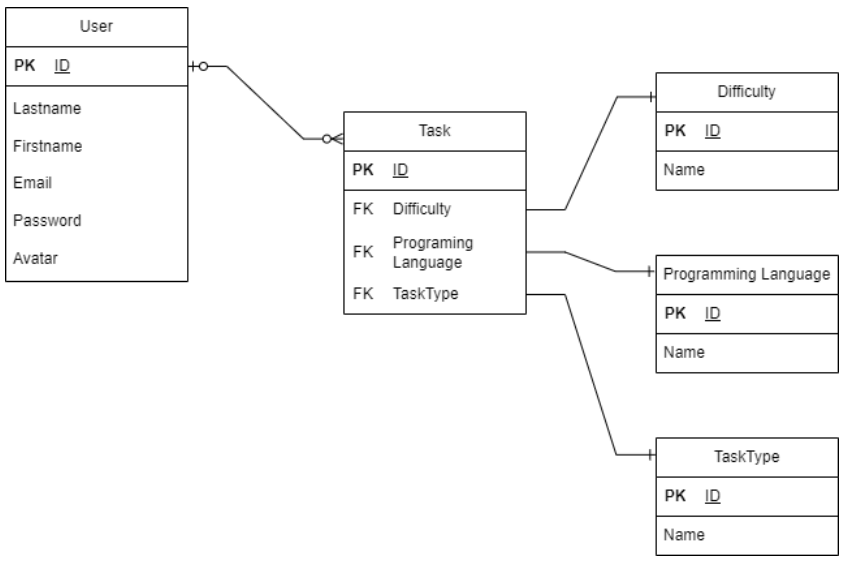


Рисунок 4.2 – ER-діаграма CodeTester

Основні об'єкти бази даних нашої системи та їхні зв'язки:

Таблиця User (Користувачі):

– UserID (PK): унікальний ідентифікатор користувача;

– Lastname: прізвище користувача;

– Firstname: ім’я користувача;

– Email: електронна адреса користувача;

– Password: пароль профілю користувача;

– Avatar: аватар профілю користувача.

Таблиця Task (Завдання):

– TaskID (PK): унікальний ідентифікатор завдання;

– UserID (FK): ідентифікатор користувача, якому призначене завдання;

– DifficultyID (FK): ідентифікатор рівня складності завдання;

– ProgrammingLanguageID (FK): ідентифікатор мови програмування, що використовується для завдання;

– TaskTypeID (FK): ідентифікатор типу завдання.

Таблиця Difficulty (Складність):

– DifficultyID (PK): унікальний ідентифікатор рівня складності;

– DifficultyLevel: рівень складності.

Таблиця ProgrammingLanguage (Мова програмування):

– ProgrammingLanguageID (PK): унікальний ідентифікатор мови програмування;

– Name: назва мови програмування.

Таблиця TaskType (Тип завдання):

– TaskTypeID (PK): унікальний ідентифікатор типу завдання;

– TypeName: назва типу завдання.

Взаємозв'язки між таблицями:

– Користувач може мати багато завдань, але кожне завдання призначене одному користувачу (зв'язок один-до-багатьох між User і Task).

– Кожне завдання має один рівень складності, але рівень складності може бути у багатьох завдань (зв'язок один-до-багатьох між Difficulty і Task).

– Кожне завдання використовує одну мову програмування, але мова програмування може бути у багатьох завданнях (зв'язок один-до-багатьох між ProgrammingLanguage і Task).

– Кожне завдання має один тип, але тип може бути у багатьох завдань (зв'язок один-до-багатьох між TaskType і Task).

4.3 Реалізація UI/UX

CodeTester дозволяє відображати наявні задачі для вирішення, взаємодіяти з ними (вирішувати, перевіряти правильність рішення), а також переглядати усі та вдалі спроби вирішення користувачем платформи.

Також ми можемо сортувати задачі за наявним типом важкості, мова програмування для вирішення, тип задачі та прапорцем «вирішена/не вирішена».

Основні особливості Angular-додатку:

Розширені можливості налаштування: CodeTester надає спектр можливостей налаштування, які дозволяють змінювати вигляд аватару, введення нового рішення, статистика аккаунта за вирішеними задачами та інше.

Підтримка інтерактивності: CodeTester дозволяє користувачам взаємодіяти з платформою, а саме додавати, редагувати та видаляти задачі прямо на сторінці. Крім того, ви можете налаштовувати якою мовою програмування вирішувати цю задачу, кількість спроб тощо.

Далі на рисунках 4.3.1, 4.3.2, 4.3.3, 4.3.4 наведено приклади елементів інтерфейсу.

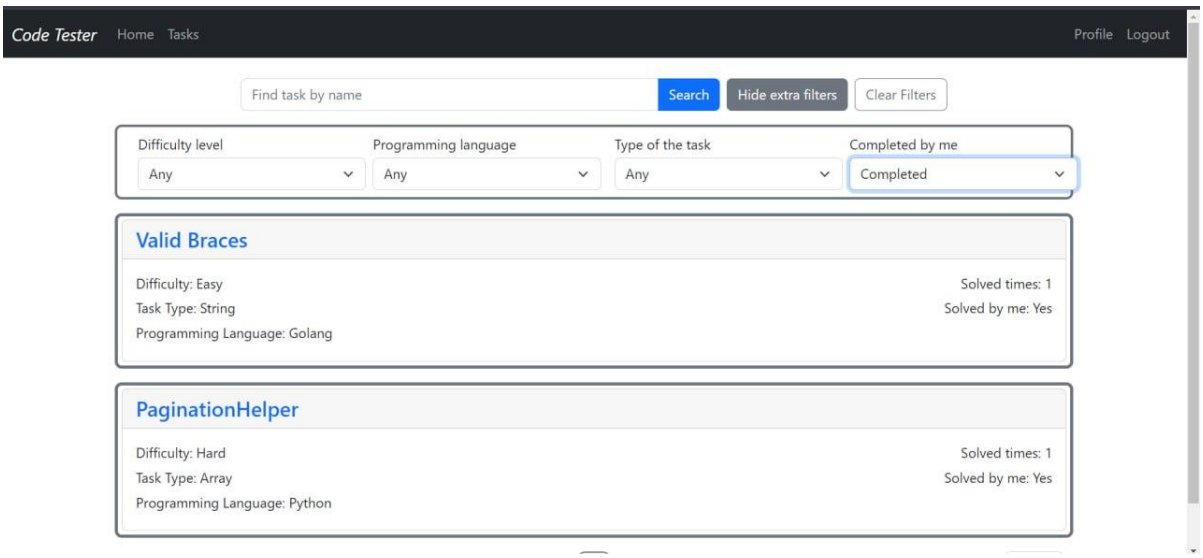


Рисунок 4.3.1 – Сторінка із завданнями платформи CodeTester

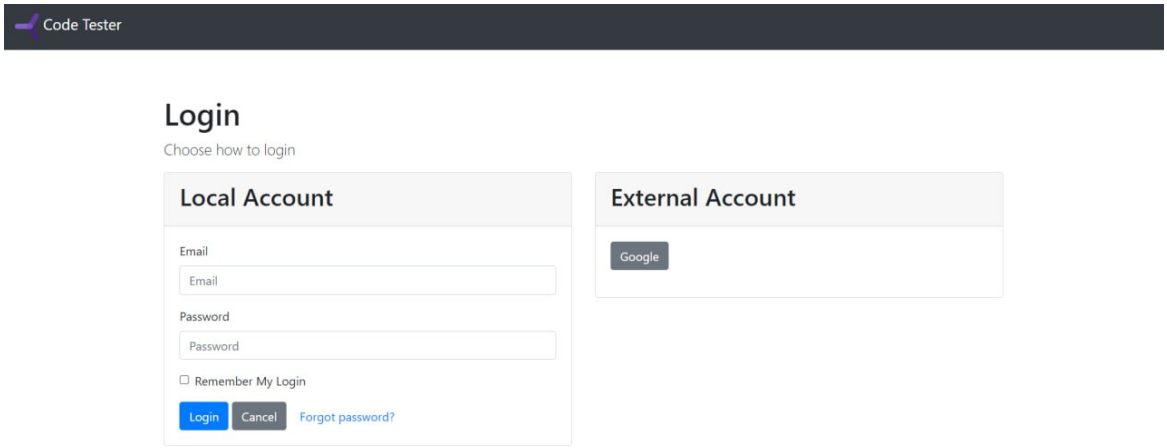


Рисунок 4.3.2 - Приклад сторінки із логіном на сайт

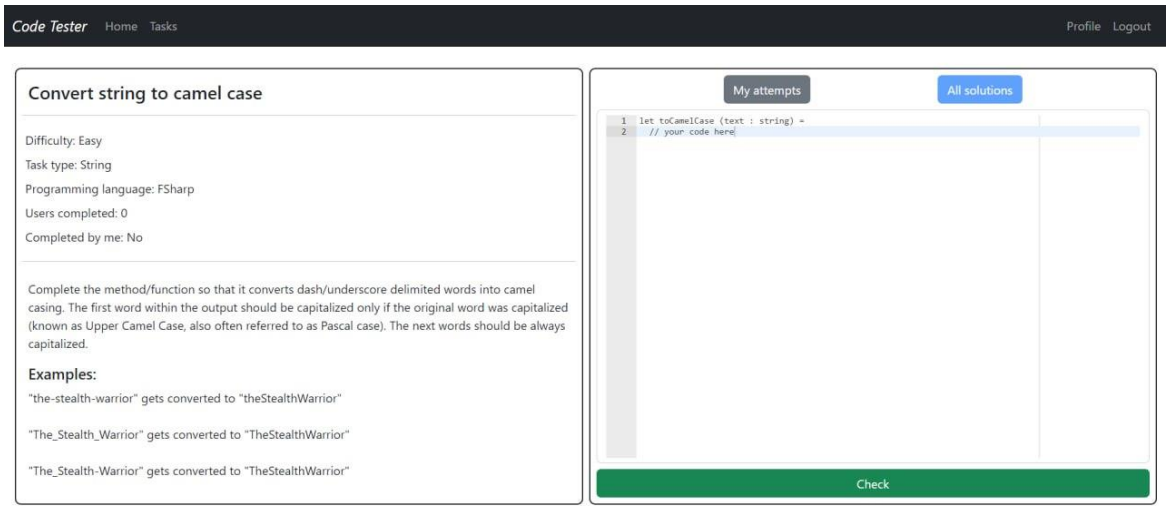


Рисунок 4.3.3 – Сторінка надання спроби для вирішення задачі

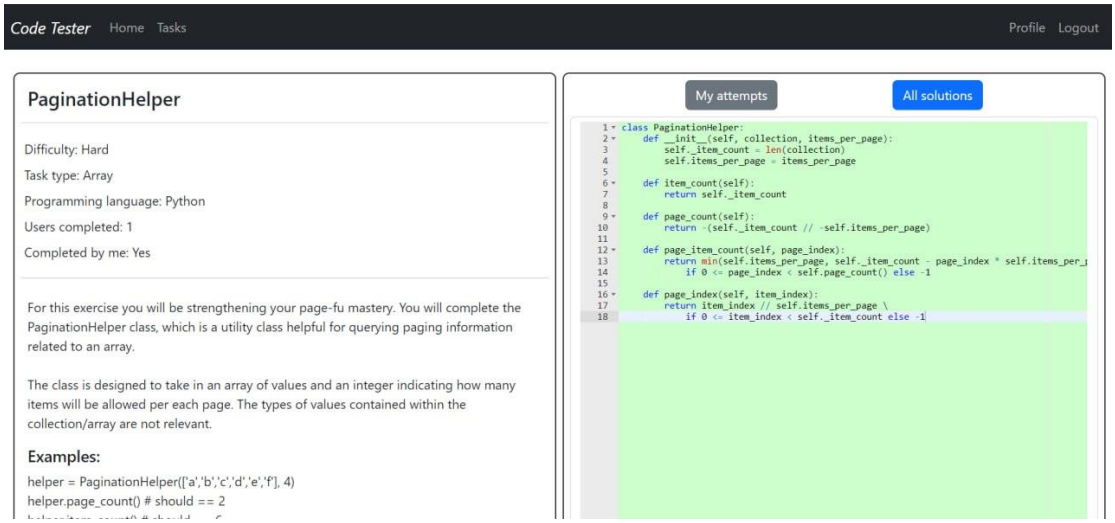


Рисунок 4.3.4 – Вдала спроба вирішення задачі

Отже, платформа дозволяє відображати наявні задачі для вирішення, взаємодіяти з ними (вирішувати, перевіряти правильність рішення), а також переглядати усі та вдалі спроби вирішення користувачем платформи.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому проекті використано інтегрований тестовий інструментарій у вигляді окремого мікросервісу для тестування роботи нашого застосунку на серверній частині.

Клієнтська частина тестується власноруч завдяки перебору всіх можливих сценаріїв роботи застосунку та перевірки відповідного інтерфейсу на кшталт заміру часу відклику. Процес та результати тестування представлені у вигляді тестових випадків.

Усі тестові випадки включають декілька складових. По-перше, визначається мета тестування. Наприклад, це може бути перевірка функціоналу додавання завдань з кодування, їх редагування, наявність прапорця «Вирішена» після того, як користувач надав правильне рішення до задачі тією мовою програмування, що зазначена в задачі. По-друге, описується початковий стан та умови проведення тестування. Наприклад, вищевказані дії можливі лише тоді, коли користувач зробив вхід у свій профіль (це може бути як Google-аккаунт чи власноруч зареєстрований аккаунт у системі за допомогою поштової скриньки.

Наступним етапом є виконання послідовних дій, які тестувальник повинен виконати для проведення тесту. У цьому випадку набір дій може бути таким: додавання, редагування та видалення завдань з кодування (у ролі адміністратора), надання рішення для завдань та її перевірка, отримання прапорця «Виконано» при вдалій спробі виконання завдання з кодування (у ролі зареєстрованого користувача), додавання, редагування та видалення аватару у профілі користувача (у ролі зареєстрованого користувача або адміністратора), реєстрація чи авторизація користувача на головній сторінці платформи, використання функції «Забув пароль?» для надсилання листа на поштову скриньку, якщо попередньо користувач був зареєстрований у системі, задля відновлення паролю до аккаунту (у ролі гостя). Відповідні дії із заздалегідь дотриманими умовами тестування повинні мати у висновку статус «Позитивний», інакше – система повідомляє про помилки користувача за допомогою спеціальних випадаючих вікон у правому верхньому куті екрану.

Окремим пунктом для тестування є перевірка обмеженості на коректність введення даних. Метою такого тесту є перевірка система на валідність введених даних користувачем системи під час вирішення завдання, реєстрації аккаунту за допомогою пошти, перевірка пароля на кшталт рівня складності. Тестувальник вводить пошту з некоректним доменом поштової скриньки або вводить пароль, що є дуже простим: наприклад, «123456». Очікуваний результат – система повинна повідомити користувача про помилку за допомогою спеціальних повідомлень біля відповідних вікон вводу. Статус такого тестового випадку буде «Позитивний».

Ці тестові випадки допомагають перевірити різні аспекти роботи програмної системи та переконатися в її коректності та надійності перед релізом у production. Ручне тестування дозволяє детально перевірити кожен елемент системи, виявити можливі проблеми та забезпечити високу якість програмного забезпечення.

ВИСНОВОК

Мета цієї роботи полягала в створенні програмної системи для відстеження активності користувачів на платформі CodeTester. Основні завдання включали розробку зручного інтерфейсу для користувачів на будь-якому рівні вмінь у програмуванні, а також створення надійної бази даних для зберігання та обробки важливої інформації.

Під час розробки було реалізовано користувацький інтерфейс з використанням фреймворку Angular, що забезпечив швидку та ефективну взаємодію з користувачами. Серверна частина системи, побудована на .NET Core, забезпечила стабільне з'єднання з базою даних Microsoft SQL Server та ефективне управління даними.

Використання Entity Framework Core дозволило забезпечити гнучкий mapping моделей даних на базу даних, що сприяло швидкому оновленню даних. Система управління завданнями була інтегрована з інтерфейсом користувача, що дозволило адміністраторам ефективно керувати користувачами та завданнями.

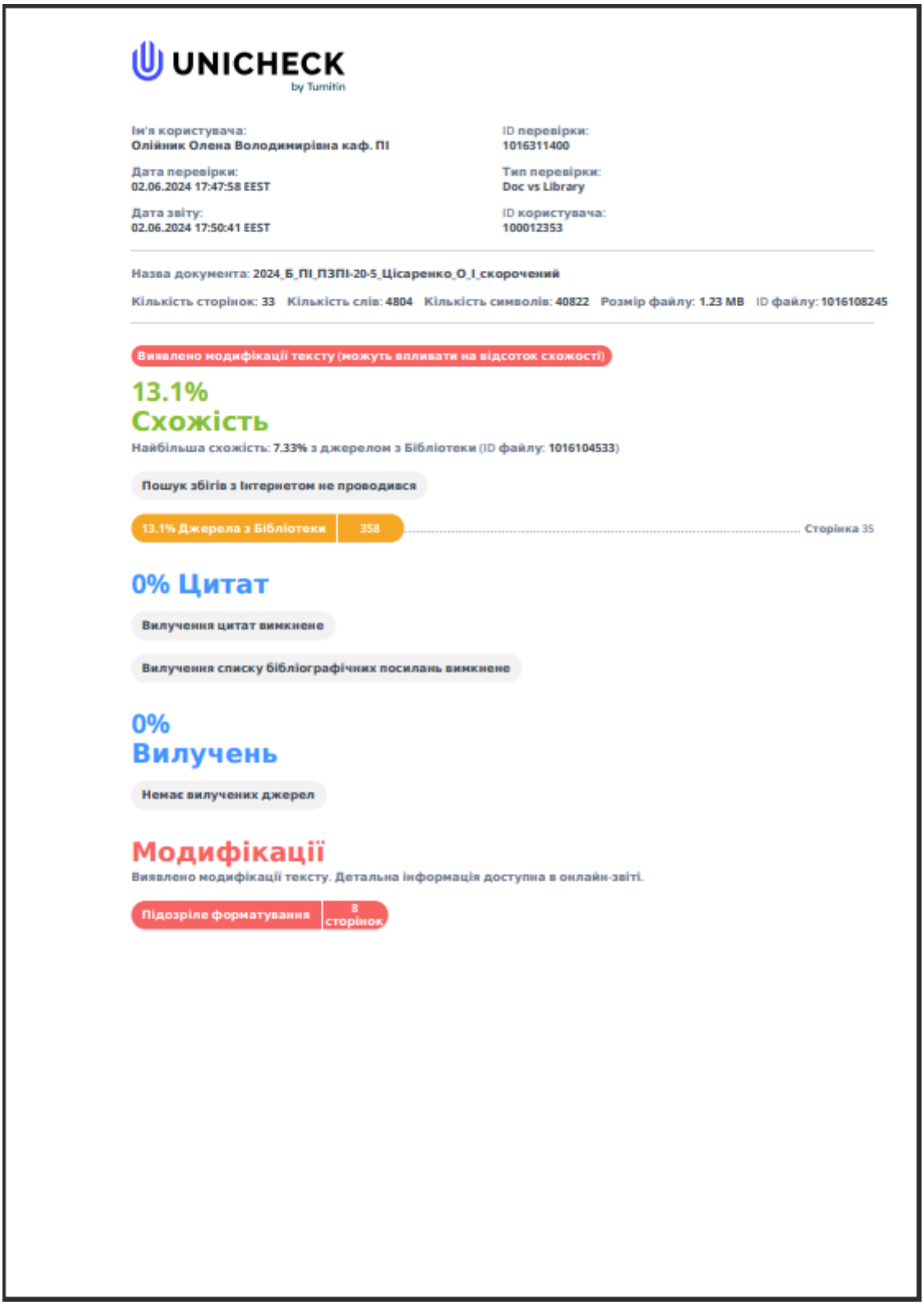
Майбутнє розвиток платформи передбачає розширення функціональності для аналізу продуктивності користувачів, покращення звітності та інтерфейсу користувача для забезпечення зручності роботи. Також планується інтеграція з іншими системами для розширення можливостей авторизації на сервіс (наприклад, інші поштові сервіси та Github).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мартін Р., Мартін М. Принципи, патерни та методики гнучкої розробки мовою C#. Київ. 1-ше видання, Ліра, 2011. 757 с.
2. SQL Stored Procedures. URL: https://www.w3schools.com/sql/sql_stored_procedures.asp (дата звернення: 04.05.2023).
3. Model validation in ASP.NET Core MVC and Razor Pages. URL: https://docs.microsoft.com/enus/aspnet/core/mvc/models/validation?view=aspnetcore-3.1 (дата звернення: 05.05.2023).
4. Create Data Transfer Objects (DTOs). URL: https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-apiwith-entity-framework/part-5 (дата звернення: 04.05.2024).
5. Angular. URL: https://angular.io/tutorial/first-app (дата звернення: 04.04.2024)
6. Angular Material. URL: https://material.angular.io/ (дата звернення: 08.05.2024).
7. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Book by Robert Cecil Martin. 2017 с.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016311400

Дата перевірки:
02.06.2024 17:47:58 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
02.06.2024 17:50:41 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-5_Цісаренко_О_І_скорочений

Кількість сторінок: 33 Кількість слів: 4804 Кількість символів: 40822 Розмір файлу: 1.23 MB ID файлу: 1016108245

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.1%
Схожість

Найбільша схожість: 7.33% з джерелом з Бібліотеки (ID файлу: 1016104533)

Пошук збігів з Інтернетом не проводився

13.1% Джерела з Бібліотеки 358 Сторінка 35

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 8 сторінок

ДОДАТОК Б

Слайди презентації

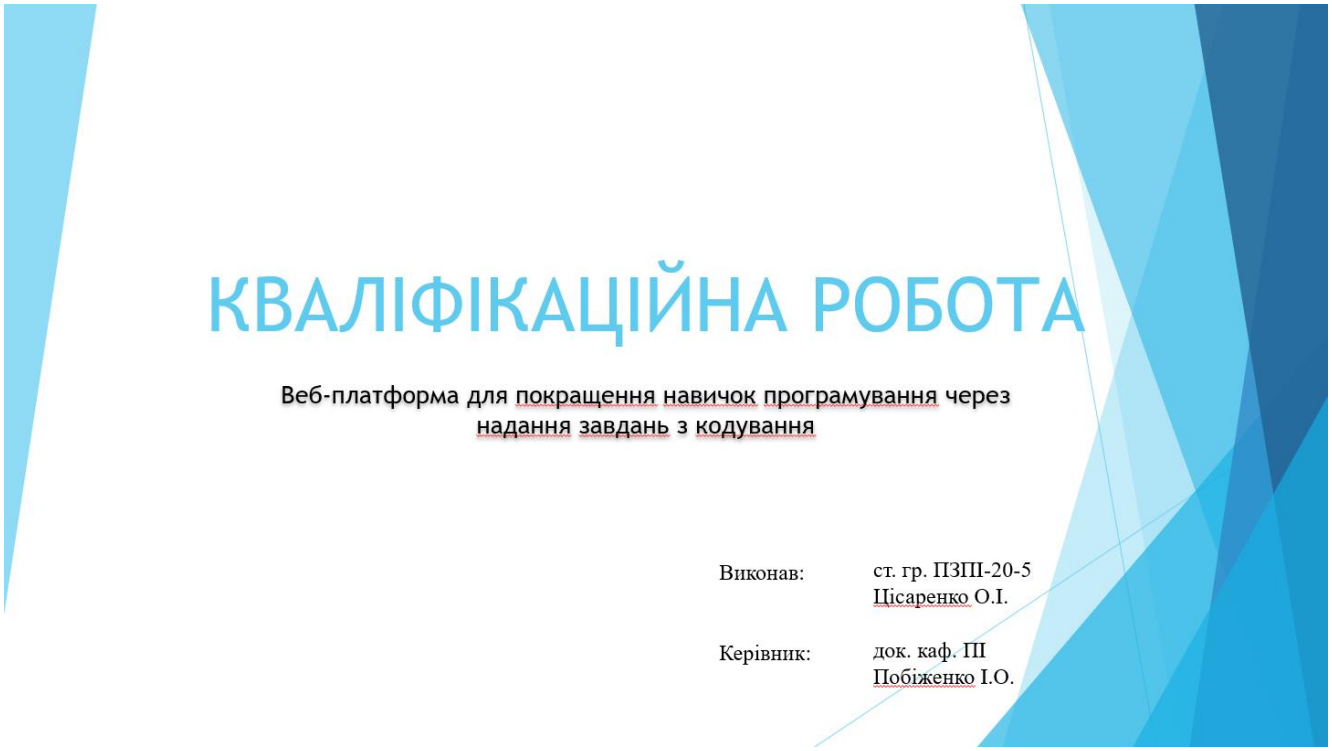


Рисунок Б.1 – Слайд 1 (рисунок виконаний самостійно)

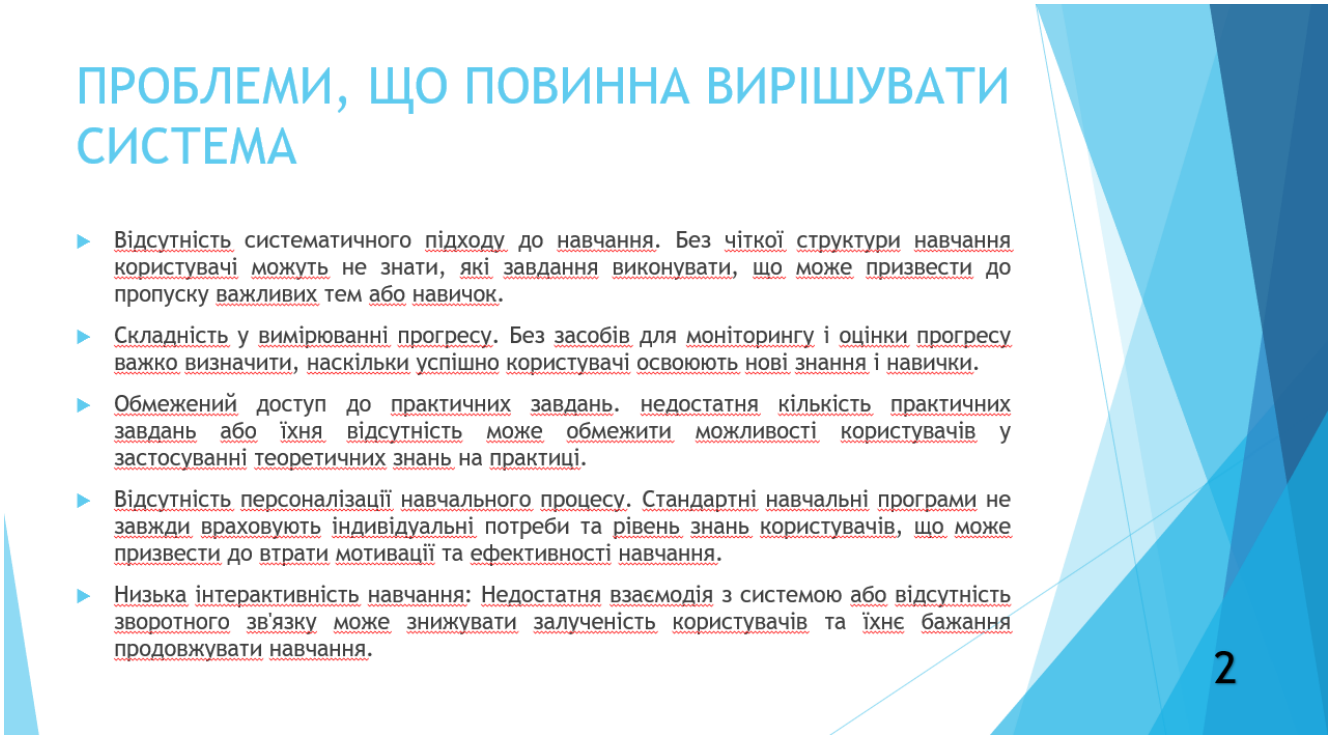


Рисунок Б.2 – Слайд 2 (рисунок виконаний самостійно)

Рисунок Б.3 – Слайд 3 (рисунок виконаний самостійно)

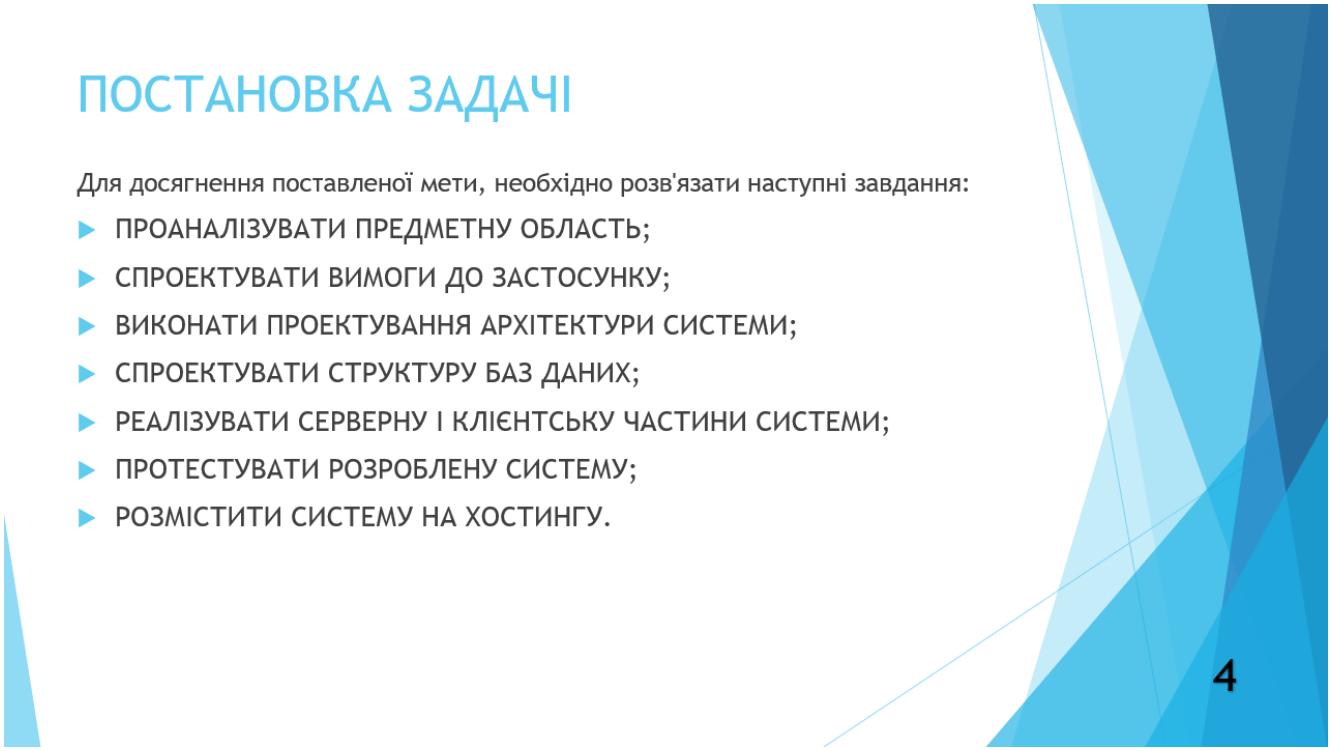
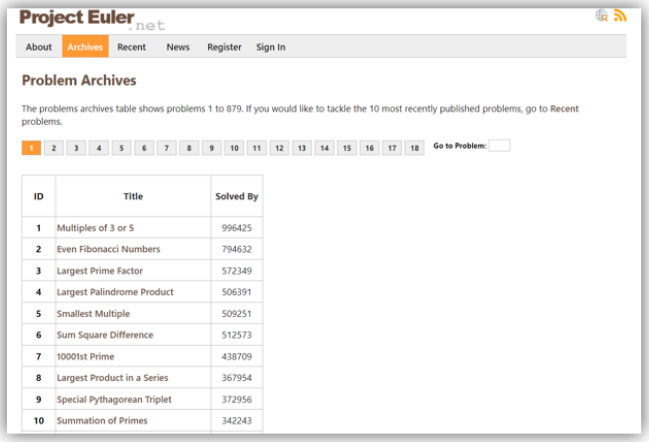


Рисунок Б.4 – Слайд 4 (рисунок виконаний самостійно)

АНАЛОГИ



- *Project Euler* - це платформа, спрямована на розв'язання математичних задач за допомогою програмування. Вона пропонує складні математичні завдання, які вимагають креативного підходу та ефективних алгоритмів.

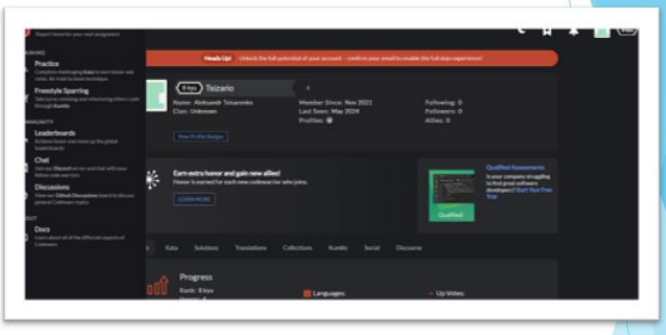
Переваги: Зосередженість на математичних задачах, велика колекція завдань, розвиток креативності та ефективних алгоритмів.
Недоліки: Обмежена спеціалізація, складність для початківців, відсутність інтерактивності.

5

Рисунок Б.5 – Слайд 5 (рисунок виконаний самостійно)

АНАЛОГИ

- CodeWars - це інтерактивна платформа, яка надає набір завдань з програмування та алгоритмів у вигляді "кат" (challenges). Користувачі можуть вирішувати ці завдання, використовуючи будь-яку мову програмування, та отримувати рівень "кю" (kyu) за кожне вирішене завдання



Переваги: Широкий вибір завдань, підтримка різних мов програмування, система рейтингу "кю", активна спільнота для обміну досвідом.

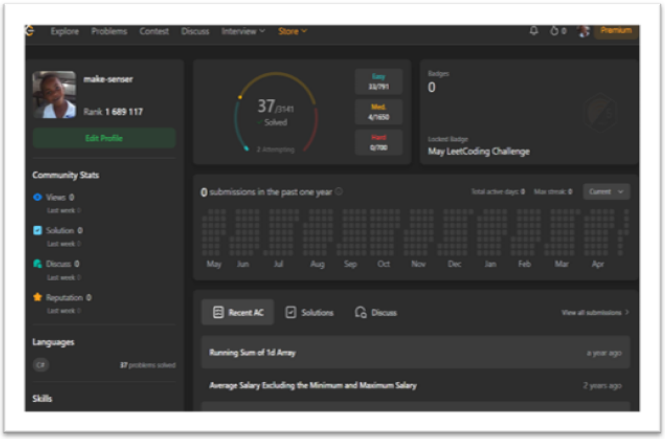
Недоліки: Відсутність структурованих курсів, вимога до самостійності, можливість "застрягнути" на складних завданнях.

6

Рисунок Б.6 – Слайд 6 (рисунок виконаний самостійно)

АНАЛОГИ

- LeetCode - це платформа, спеціалізована на підготовці до технічних співбесід та вирішенні задач з програмування. Вона містить велику колекцію завдань з алгоритмів, структур даних та інших тем, які часто зустрічаються на співбесідах у технологічних компаніях. Платформа дозволяє записувати активність аккаунту, які задачі було вирішено за рівнями складності, загальне місце у рейтингу платформи. Також періодично проводяться змагання з програмування, приз з яких буде підписка на преміум-аккаунт із доданими можливостями щодо більш прискореної перевірки завдання, пріоритет від підтримки платформи та інше.



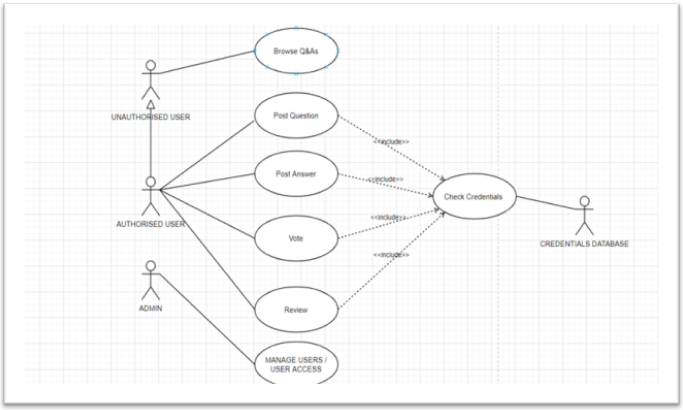
Переваги LeetCode:
Тематичні блоки завдань, підготовка до технічних співбесід, активна спільнота користувачів, змагання з можливістю отримати преміум-акаунт.

Недоліки:
Основний фокус на алгоритми, відсутність персоналізованого навчання, деякий функціонал доступний лише з преміум-підпискою.

7

Рисунок Б.7 – Слайд 7 (рисунок виконаний самостійно)

USE-CASE ДІАГРАММА



У НАШІЙ ВЕБ-ПЛАТФОРМІ ДЛЯ ПОКРАЩЕННЯ НАВИЧОК ПРОГРАМУВАННЯ ЗАДІЯНО ТРИ ОСНОВНІ ТИПИ КОРИСТУВАЧІВ, КОЖЕН З ЯКИХ МАЄ УНІКАЛЬНИЙ НАБІР РОЛЕЙ ТА МОЖЛИВОСТЕЙ:

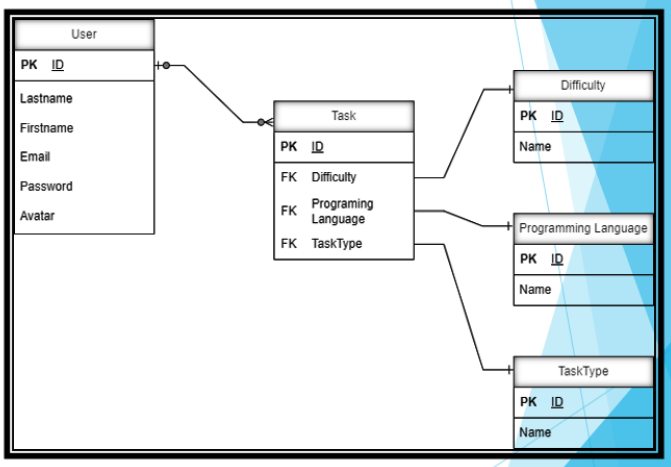
- ГІСТЬ
- КОРИСТУВАЧ
- АДМІНІСТРАТОР

8

Рисунок Б.8 – Слайд 8 (рисунок виконаний самостійно)

СХЕМА БД

Для даного проекту, що спрямований на управління даними клієнтів, було вирішено використовувати реляційну базу даних Microsoft SQL Server. Вона відома своєю надійністю, безпекою та високою продуктивністю, що робить її ідеальним вибором для корпоративних застосунків, які потребують ефективного управління великими обсягами даних і складною бізнес-логікою.



9

Рисунок Б.9 – Слайд 9 (рисунок виконаний самостійно)

ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

Фронт-енд частина:
- TypeScript
- Angular
- Bootstrap
- HTML 5/CSS 3
- OIDC Client

Бек-енд частина:
- C#
- ASP.NET
- ASP.NET Identity
- Entity Framework Core
- IdentityServer
- Serilog
- MSSQL
- HealthChecks
- MassTransit
- Polly
- gRPC

10

Рисунок Б.10 – Слайд 10 (рисунок виконаний самостійно)

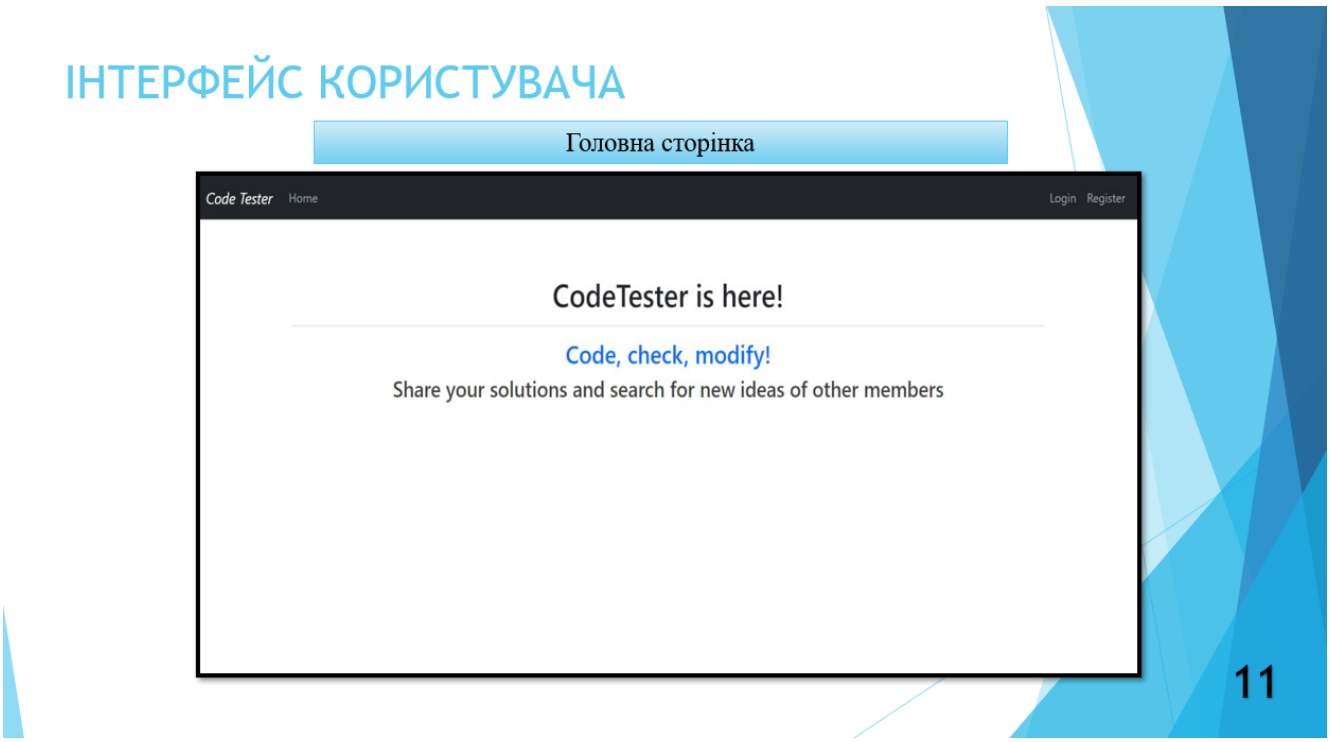


Рисунок Б.11 – Слайд 11 (рисунок виконаний самостійно)

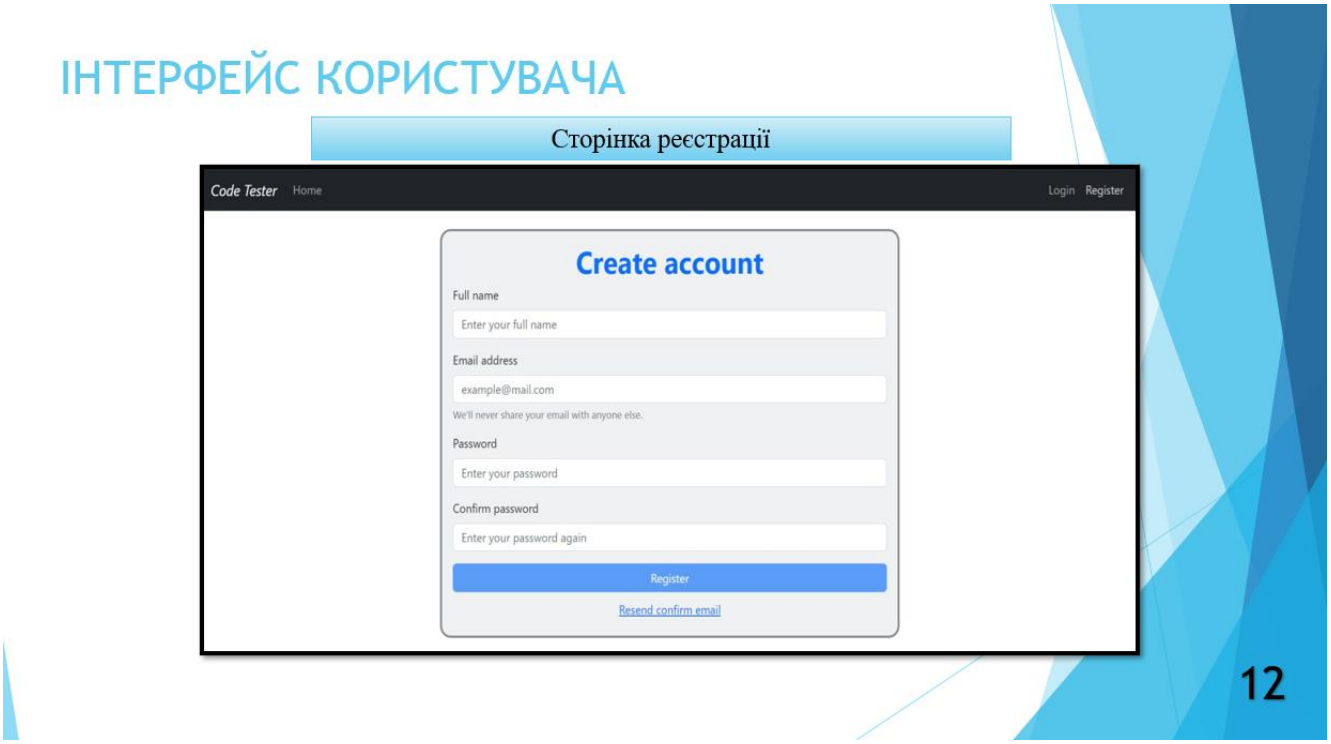


Рисунок Б.12 – Слайд 12 (рисунок виконаний самостійно)

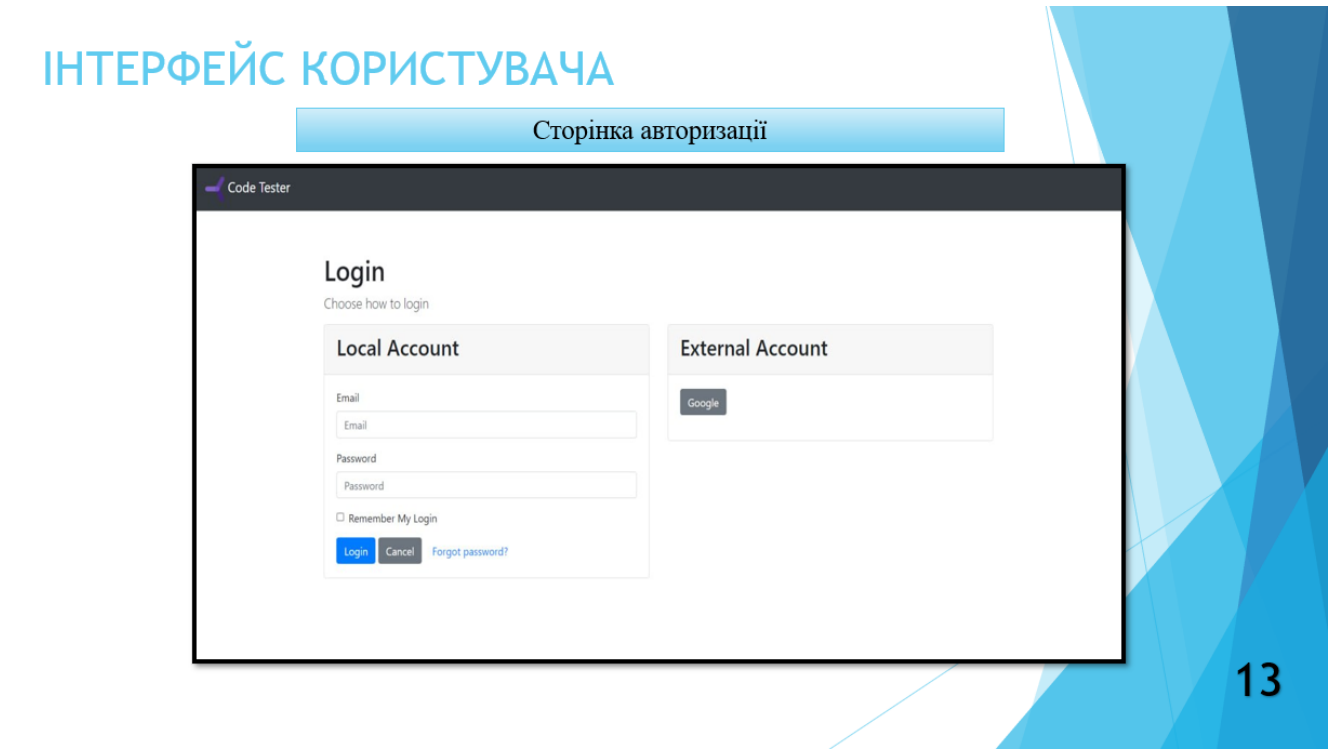


Рисунок Б.13 – Слайд 13 (рисунок виконаний самостійно)

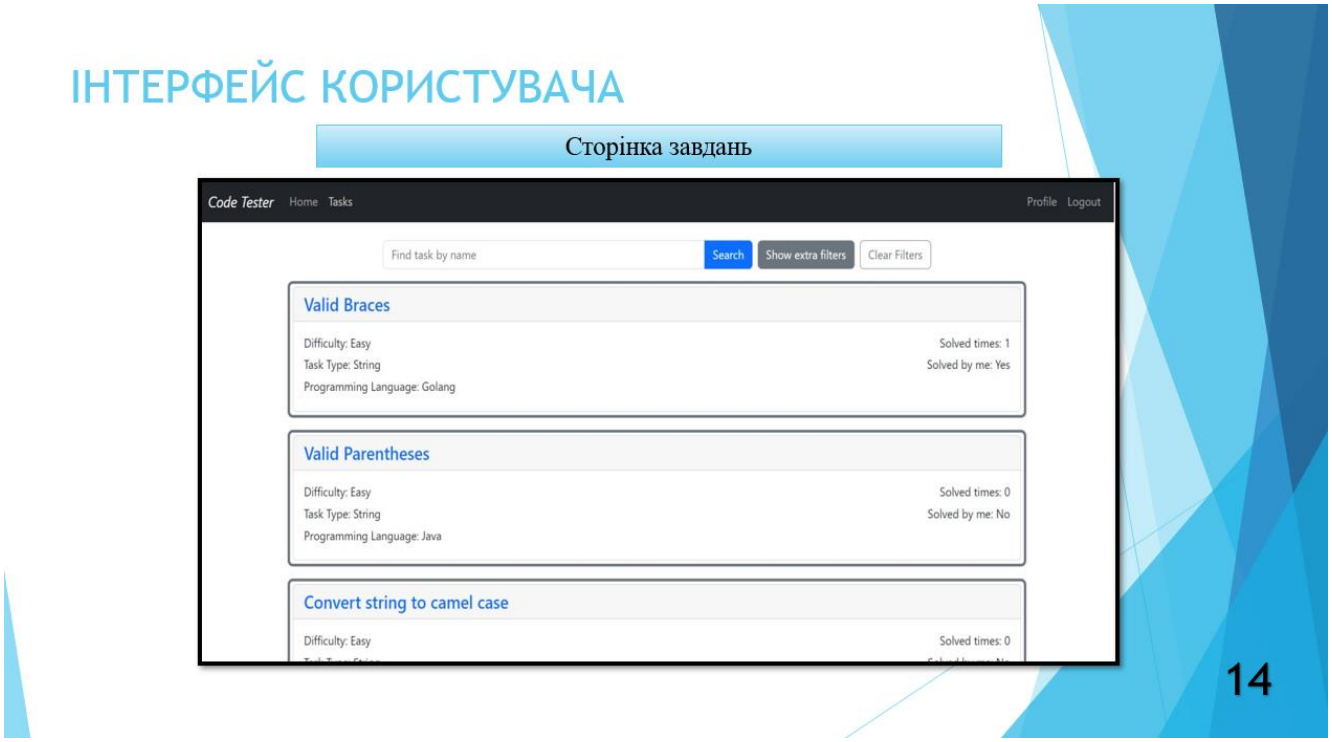
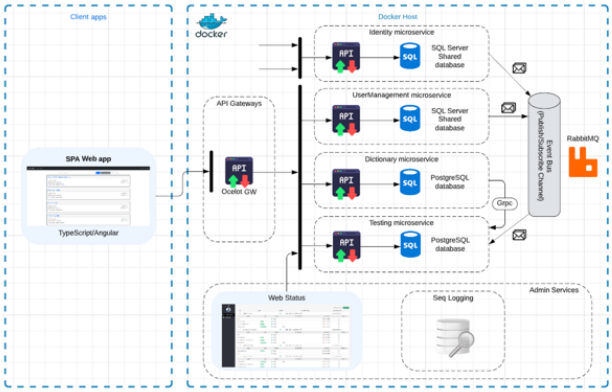


Рисунок Б.14 – Слайд 14 (рисунок виконаний самостійно)

АРХІТЕКТУРА



- Архітектура системи реалізує парадигму з декількома автономними мікросервісами, кожен з яких володіє власними даними та базами даних. Кожен мікросервіс реалізує різні підходи з використанням HTTP як протоколу зв'язку між клієнтськими програмами та мікросервісами.
- Для забезпечення асинхронного зв'язку між мікросервісами та розповсюдження оновлень даних використовується шина подій на основі інтеграційних подій та посередника легких повідомлень RabbitMQ.
- Завдяки використанню чистої архітектурі (англ. Clean Architecture) доменний і прикладний рівні знаходяться в центрі дизайну, відомого як ядро системи. Бізнес-логіка розміщується в цих двох рівнях, хоча вони містять різні види бізнес-логіки. Вони розглядаються як деталі, і бізнес-рівні не повинні залежати від рівнів презентації та інфраструктури. Замість того, щоб бізнес-логіка залежала від доступу до даних або інших питань інфраструктури, ця залежність інвертується: деталі інфраструктури та реалізації залежать від прикладного рівня.

15

Рисунок Б.15 – Слайд 15 (рисунок виконаний самостійно)

ВИСНОВКИ

- Проведено аналіз предметної галузі
- Проведено UML моделювання
- Розроблено схему бази даних
- Обрані засоби реалізації програмної системи
- Програмно реалізовано веб-систему для менеджменту робочого процесу із проектом та менеджменту персоналу на підприємстві
- Протестовано веб-система для менеджменту робочого процесу із проектом та менеджменту персоналу на підприємстві

16

Рисунок Б.16 – Слайд 16 (рисунок виконаний самостійно)

ДОДАТОК В

Специфікація вимог до програмного продукту

Специфікація ПЗ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
Веб-платформа для покращення навичок програмування через надання завдань з кодування

Студент гр. ПЗПІ-20-5 _____ Цісаренко О.І.

Харків
2024 р.

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

Програмний продукт є веб-орієнтованою програмною системою для покращення навичок програмування через надання завдань з кодування. Він надає можливості ефективного управління процесом навчання, включаючи створення та управління користувачами, налаштування ролей та прав доступу, конфігурацію системи, управління модулями та плагінами, а також забезпечення комунікації між учасниками. Програмний продукт підтримує взаємодію з користувачами, включаючи перегляд результатів і прогресу, надання зворотного зв'язку та інші функції, що сприяють покращенню співпраці. Менеджери можуть ефективно керувати користувачами, створювати акаунти, налаштовувати права доступу і параметри системи. Користувачі отримують доступ до інструментів для співпраці та комунікації, включаючи форуми, вікі та систему коментарів, що полегшує командну роботу і забезпечує швидке вирішення питань. Back-end частина реалізована на платформі .NET 6 (LTS) з використанням EF Core, HealthChecks, Polly, Dapper та Grpc, front-end – на Angular, база даних використовує MS SQL Server.

1.2 Мета

Метою розробки цього документа є створення веб-платформи із завдань з кодування CodeTester, яка надаватиме можливість користувачам покращувати навички програмування шляхом вирішення алгоритмічних завдань будь-якою мовою програмування для користувача.

1.3 Межі

Продукт розрахований на використання будь-яким користувачем світу будь-де. Має обмеження по кількості одночасних користувачів, розмірів бази даних і інтеграції з іншими системами. Продукт буде розроблений на основі веб-технологій і не включає розробку мобільних додатків.

Межі продукту також включають обмеження по функціоналу, зосереджуючись на ключових аспектах керування завданнями, такими як перевірка на правильність рішення, подання рішення у чергу, конфігурація системи, автоматизація рутинних завдань, і забезпечення комунікації між учасниками проекту. Не передбачається інтеграція з широким спектром зовнішніх систем, окрім необхідних для виконання авторизації через улюблений сервіс користувачів.

1.4 Посилання

Ця веб-орієнтована програмна система призначена для спрощення та автоматизації процесу покращення навичок програмування шляхом надання завдань із кодування. Впровадження даного продукту дозволить знизити кількість ручної роботи при організації навчального процесу, оптимізувати комунікацію між усіма учасниками, а також покращити загальну ефективність розвитку програмістських навичок.

Система аутентифікації та авторизації забезпечить безпечний доступ та контроль над даними, що є критично важливим для забезпечення конфіденційності інформації. Це дозволить користувачам та адміністраторам працювати в безпечному середовищі, де дані захищені від несанкціонованого доступу.

Ця система допоможе організаціям та індивідуальним користувачам раціоналізувати процеси навчання програмуванню, що дозволить більш ефективно використовувати ресурси та досягати поставлених цілей. Використання сучасних технологій, таких як .NET 6 (LTS), Angular, EF Core, HealthChecks, Polly, Dapper, Grpc, та MS SQL Server забезпечить високу надійність та продуктивність системи, роблячи процес навчання зручним та ефективним.

1.5 Означення та абревіатура

– .NET – програмна платформа від Microsoft для розробки додатків;

– Angular – фреймворк для JavaScript/TypeScript для створення користувацьких інтерфейсів;

– API – програмний інтерфейс додатків (Application Programming Interface);

– HTTPS – протокол захищеного перенесення гіпертексту (HyperText Transfer Protocol Secure);

– UI/UX – інтерфейс користувача / досвід користувача (User Interface/User Experience);

– SQL – мова структурованих запитів (Structured Query Language);

– EF Core (Entity Framework Core) – об'єктно-реляційний маппер (ORM) для .NET, який дозволяє розробникам працювати з базами даних, використовуючи .NET об'єкти;

– HealthChecks – бібліотека для перевірки стану здоров'я сервісів та додатків, що допомагає в моніторингу та забезпеченні надійності роботи систем;

– Polly – бібліотека для обробки винятків та політик надійності, таких як повторні спроби, обмеження часу виконання, розімкнуті схеми та інші;

– Dapper – легковажний ORM для .NET, який забезпечує високу продуктивність при роботі з базами даних завдяки використанню сирих SQL запитів;

– Docker – платформа для контейнеризації, яка дозволяє розробникам створювати, розгортати та керувати додатками в контейнерах;

– RabbitMQ – брокер повідомлень з відкритим вихідним кодом, який підтримує різні протоколи обміну повідомленнями, такі як AMQP. Він дозволяє додаткам спілкуватися між собою асинхронно, забезпечуючи надійний обмін даними, керування чергами та обробку повідомлень у розподілених системах;

2 ЗАГАЛЬНИЙ ОПИС

2.1 Песпективи продукту

Веб-платформа для покращення навичок програмування через надання завдань із кодування має на меті стати провідним інструментом для навчання та вдосконалення навичок програмування. Очікується, що платформа буде постійно оновлюватися та розширюватися, інтегруючись з новими технологіями та інструментами для підвищення ефективності навчання.

Майбутні версії продукту можуть включати підтримку технологій штучного інтелекту (AI) та машинного навчання (ML) для автоматизації аналізу коду, надання персоналізованих рекомендацій та адаптивного навчання, що допоможе користувачам швидше освоювати нові концепції та покращувати свої навички.

Перспективи продукту також включають можливість створення мобільних додатків для зручнішого доступу користувачів до завдань та навчальних матеріалів з будь-якого місця та в будь-який час. Це забезпечить підвищення мобільності та гнучкості у навчальному процесі.

Розширення функціоналу може включати інтеграцію з іншими навчальними платформами та сервісами, такими як системи управління навчанням (LMS), онлайн-курси, відеоуроки та інші інструменти, що використовуються в освітньому середовищі. Це дозволить створити єдину екосистему для всебічного навчання програмуванню.

Покращення користувацького досвіду (UX) буде здійснюватися через впровадження персоналізованих сповіщень та рекомендацій, що допоможе користувачам більш ефективно використовувати систему та досягати своїх навчальних цілей. Система також буде адаптуватися до зворотного зв'язку від користувачів, щоб постійно вдосконалюватися та відповідати їхнім потребам.

Загалом, перспективи розвитку цієї веб-платформи спрямовані на створення більш інноваційного, інтегрованого та ефективного інструменту для навчання програмуванню, що сприятиме підвищенню продуктивності навчання та задоволеності користувачів.

2.2 Функції продукту

Продукт включає наступні функції:

– Управління користувачами: створення та управління користувачами (адміністратор, користувач); створення користувацьких акаунтів, налаштування ролей та прав доступу;

– Комунікація та співпраця: інтегровані форуми, чати та системи коментарів для забезпечення ефективної комунікації та співпраці між користувачами, викладачами та студентами;

– Управління завданнями: створення, перегляд, оновлення та закриття завдань; встановлення термінів виконання та оцінювання завдань користувачами платформи;

– Конфігурація системи: налаштування параметрів системи для забезпечення її оптимальної роботи; управління різними аспектами платформи через єдиний інтерфейс адміністрування;

– Інтеграція з зовнішніми сервісами: можливість інтеграції з іншими навчальними платформами, сервісами для перевірки коду або зберіганням даних, щоб забезпечити комплексний підхід до покращення навчання програмування;

2.3 Характеристики користувачів

Можливості адміністратора:

- Створення та управління користувачами (адміністратор);
- Створення користувацьких акаунтів;
- Налаштування ролей та прав доступу;
- Конфігурація системи (адміністратор);
- Налаштування параметрів системи;
- Управління завданнями та користувачами.

Можливості користувача:

- Вирішення завдань;
- Обговорення завдань з іншими користувачами у коментарях;
- Надання зворотного зв'язку для завдань.

Можливості гостя:

– Перегляд головної сторінки;

– Реєстрація чи авторизація на платформі за власноруч зареєстрованої пошти чи Google-пошти.

2.4 Загальні обмеження

Максимальна кількість одночасних користувачів: Продукт може мати обмеження на максимальну кількість користувачів, які можуть одночасно використовувати систему. Це може вимагати оптимізації та масштабування для забезпечення стабільної роботи під час пікових навантажень.

Обсяг даних в базі даних SQL Server: продукт може мати обмеження на обсяг даних, що можуть бути збережені в базі даних SQL Server. Враховується необхідність збереження історичних даних для аналітики та звітності.

Пропускна здатність мережі: продукт може мати обмеження на пропускну здатність мережі, що вимагає оптимізації та планування для забезпечення ефективного обміну даними між користувачами та сервером.

Обмеження по інтеграції з зовнішніми системами: система може бути обмежена в можливостях інтеграції з деякими зовнішніми сервісами через технічні або безпекові причини. Це може вплинути на широкий спектр можливостей інтеграції з іншими системами.

2.5 Припущення та залежності

Доступ до сучасних веб-браузерів і стабільного інтернет-з'єднання: Припускається, що користувачі мають доступ до сучасних веб-браузерів і стабільного інтернет-з'єднання для використання веб-сервісу.

Базові навички користувачів з комп'ютером та Інтернетом: припускається, що користувачі мають базові навички роботи з комп'ютером та Інтернетом для безпроблемного користування веб-сервісом.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішнього інтерфейсу

3.1.1 Інтерфейс користувача

Інтерфейс користувача має бути розроблений на основі Angular, забезпечуючи зручний та інтуїтивно зрозумілий досвід користування. Він повинен включати компоненти для створення запитів, інформаційні панелі для перегляду статусу запитів, функціонал для здійснення онлайн-платежів та налаштування для керування профілем користувача. Інтерфейс повинен бути адаптивним, підтримувати різні розміри екранів та забезпечувати однаковий рівень зручності як на настільних комп'ютерах, так і на мобільних пристроях.

Користувачам повинні бути доступні інтуїтивно зрозумілі меню та навігаційні елементи, що полегшують пошук потрібної інформації та виконання основних завдань. Інтерфейс також повинен включати функції доступності для користувачів з обмеженими можливостями, забезпечуючи доступність для широкого кола користувачів.

3.1.2 Апаратний інтерфейс

Апаратний інтерфейс веб-сервісу не вимагає спеціалізованого обладнання і може оптимально функціонувати на хмарних платформах, таких як, наприклад, Azure. Система повинна бути оптимізована для роботи з .NET API на серверному рівні та Angular на клієнтському рівні. Забезпечення підтримки резервного копіювання та відновлення даних дозволить зберігати цілісність даних у випадку апаратних збоїв або інших технічних проблем.

3.1.3 Програмний інтерфейс

Програмний інтерфейс включає RESTful API для взаємодії між front-end та back-end, а також точки інтеграції для взаємодії з зовнішніми сервісами. Це API підтримує основні операції CRUD (створення, читання, оновлення, видалення) для управління даними і забезпечує безпечну аутентифікацію та авторизацію

користувачів. Його документація є чіткою та зрозумілою, щоб розробники могли легко інтегруватися з системою та розширювати її функціонал. Інтерфейси розроблені з урахуванням принципів безпеки для захисту даних від несанкціонованого доступу та забезпечення шифрування конфіденційної інформації.

3.1.4 Комунікаційний протокол

Система використовує протокол HTTPS для забезпечення безпеки даних під час їх передачі між компонентами. Front-end та back-end взаємодіють через REST API, що забезпечує стандартизований та ефективний обмін інформацією. HTTPS гарантує захист даних від несанкціонованого доступу та маніпуляцій, забезпечуючи конфіденційність та цілісність даних. Комунікаційний протокол оптимізований для максимальної швидкості передачі даних та мінімальної затримки, щоб система швидко реагувала на запити користувачів. Додатково, протокол включає механізми автентифікації та авторизації для контролю доступу до ресурсів системи.

3.1.5 Обмеження пам'яті

Продукт має бути оптимізованим для роботи в середовищі з обмеженими ресурсами, включаючи обмеження по оперативній пам'яті і дисковому просторі. Система повинна ефективно використовувати доступні ресурси, забезпечуючи високу продуктивність навіть при обмеженій кількості оперативної пам’яті та дискового простору. Необхідно забезпечити механізми для моніторингу та управління ресурсами, що дозволить вчасно виявляти та усувати проблеми, пов'язані з недостатністю ресурсів. Оптимізація коду та архітектури системи дозволить зменшити навантаження на пам’ять та забезпечить стабільну роботу навіть при значних навантаженнях.

3.1.6 Операції

Основні функції включають обробку запитів на обслуговування, генерацію сповіщень, а також резервне копіювання та відновлення даних. Усі ці операції мають бути автоматизовані і виконуватися через CI/CD пайплайн з використанням GitHub Actions. Вони повинні працювати швидко та ефективно, щоб забезпечити високу продуктивність системи та задоволення користувачів. Система повинна бути готова масштабуватися для обробки збільшеного обсягу запитів та даних без втрати продуктивності. Крім того, всі операції повинні бути документовані та мати можливість відстеження та моніторингу для виявлення та вирішення проблем.

3.2 Атрибути програмного продукту

3.2.1 Надійність

Програмний продукт має гарантувати високу надійність шляхом застосування механізмів автоматичного резервування та дублювання даних. Система використовує альтернативні засоби для автоматизованого резервування та відновлення даних, що дозволяє запобігати втратам інформації. Надійність системи забезпечується завдяки стійкій архітектурі, яка включає резервне копіювання даних, автоматичне відновлення після збоїв та системний моніторинг. Для підвищення надійності система розгорнута в Azure зонах, що гарантує відмінну доступність та стійкість до випадкових збоїв у різних географічних областях. Крім того, система підтримує тестування на стресові та навантажувальні умови, щоб виявити можливі слабкі місця та забезпечити їх усунення перед розгортанням у реальному середовищі.

3.2.2 Доступність

Продукт повинен бути доступний для користувачів цілодобово без значних перерв у роботі. Використання технологій хмарного хостингу гарантує надійну доступність та швидке відновлення послуг у випадку виникнення проблем. Це досягається завдяки автоматичному масштабуванню і розподілу навантаження між серверами. У системі передбачено також резервне копіювання та відновлення, щоб

оперативно відновити роботу після можливих збоїв або інцидентів. Крім цього, ключовим аспектом є надійні мережеві з'єднання та резервування критичних елементів системи для забезпечення безперебійної роботи.

3.2.3 Безпека

Забезпечення безпеки є ключовим аспектом нашого продукту, і ми вкладаємо значні зусилля для захисту як даних, що зберігаються, так і тих, що передаються. Використання платформи Azure дозволяє нам забезпечити високий рівень безпеки та захисту нашої інфраструктури. Ми використовуємо шифрування для забезпечення конфіденційності даних, а протокол HTTPS гарантує захищену передачу інформації між компонентами системи. Для аутентифікації та авторизації користувачів ми застосовуємо сучасні методи захисту, такі як OAuth. Ми також активно захищаємося від різних видів атак, таких як SQL-ін'єкції, XSS та CSRF, і регулярно проводимо аудити безпеки та тестування на проникнення. Всі дані користувачів захищені згідно з вимогами GDPR або іншими відповідними нормативними актами.

3.2.4 Супроводжуваність

Для забезпечення легкості супроводу система повинна мати чітко структурований код і детально описану документацію. Використання технологій .NET та React спрощує процес супроводу та оновлення компонентів. Ефективне супроводження досягається завдяки модульній архітектурі, що дозволяє легко вносити зміни та додавати новий функціонал без впливу на інші частини системи. Документація має включати опис всіх компонентів, API та інструкції з розгортання та супроводу системи.

3.2.5 Переносимість

Продукт розробляється з урахуванням легкості переносу між різними середовищами, завдяки його гнучкій архітектурі та конфігурації. Ми уникаємо прив'язки до конкретних технологій або інфраструктури, щоб забезпечити

можливість легко переміщувати систему між різними середовищами, включаючи локальні, хмарні та інші. Крім того, ми забезпечуємо можливість швидкого та безпроблемного розгортання системи на різних платформах та середовищах без необхідності великої переробки або адаптації.

3.2.6 Продуктивність

Система має бути швидкореагуючою та високопродуктивною, щоб задовольняти потреби користувачів. Це досягається завдяки оптимізації коду та використанню ефективних алгоритмів, які дозволяють знизити затримки і забезпечити оперативне виконання завдань. Ми також систематично моніторимо продуктивність і проводимо регулярне навантажувальне тестування для виявлення та усунення можливих вузьких місць. Використання масштабованих архітектурних рішень допомагає нам забезпечувати стабільну продуктивність навіть при значному збільшенні навантаження на систему.

3.3 Вимоги бази даних

Для ефективного зберігання та швидкого доступу до великих обсягів інформації розглянуто використання MS SQL Server. Він має вбудовані механізми оптимізації запитів та можливості резервного копіювання та відновлення, що дозволяє нам забезпечити високу продуктивність та надійність. Наші основні вимоги до бази даних включають підтримку складних запитів та транзакцій, що забезпечить швидке виконання операцій та збереження цілісності інформації. Для забезпечення безпеки даних ми плануємо використовувати механізми шифрування та контролю доступу на основі ролей (RBAC). Крім того, ми плануємо використовувати механізми реплікації для забезпечення високої доступності та можливості відновлення даних у випадку збоїв.