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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису"
УДК_004.4_____

«До захисту допущено»
В.о. зав.кафедри
_____ Олександр КОВАЛЬ
“___”_____2022 р.

Магістерська дисертація

За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення
на тему: Інформаційна система моніторингу виконання тестових завдань

Виконав: студент 2 курсу, групи ТВ-12мп
Гнатишин Михайло Степанович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц.,к.ф.-м.н. Свинчук О.В._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _професор кафедри ЦТЕ, проф., д.т.н. Отрох С.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
в.о. зав. кафедри
_____ Олександр КОВАЛЬ
(підпис)
«____» _____ 2022 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Гнатишину Михайлу Степановичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Інформаційна система моніторингу виконання тестових завдань
Науковий керівник доц.,к.ф.-м.н. Свинчук О.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "07" листопада 2022 року №4067-с
2. Строк подання студентом дисертації 05 грудня 2022 року
3. Вихідні дані до роботи мова програмування TypeScript, платформа Node.js, система керування базами даних PostgreSQL
4. Перелік питань, які потрібно розробити Аналіз існуючих систем для моніторингу виконання тестових завдань, аналіз сучасних методів для виконання користувацького програмного коду, обґрунтований вибір технологій для імплементації інформаційної системи, проведення моделювання програмного забезпечення, розробка архітектури системи та розробка плану впровадження програмного забезпечення, написання програмного коду та тестування.
5. Орієнтований перелік ілюстративного матеріалу «Актуальність роботи», «Мета, обєкт та предмет дослідження», «Постановка задачі», «Існуючі аналоги», «Архітектура системи», «Засоби розробки», «Робота з програмою», «Висновки», «Публікації».
6. Орієнтований перелік публікацій
1. Гнатишин М.С, Жмуркевич В.І., Свинчук О.В. Інформаційна система. тестування студентів. XV Міжнародна науково-практична конференція: Інформаційні технології і автоматизація – 2022 (м. Одеса, 20-21 жовтня 2022 р). С.110.
7. Дата видачі завдання «01» жовтня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.10.2021	виконано
2	Дослідження предметної області	01.10.2021 - 01.11.2021	виконано
3	Постановка вимог до проєктування системи	01.11.2021 - 15.11.2021	виконано
4	Дослідження існуючих рішень	15.11.2021 - 01.12.2021	виконано
5	Підготовка публікацій	01.12.2021 - 09.12.2021	виконано
6	Розробка програмного продукту	05.03.2022 - 07.10.2022	виконано
7	Тестування	07.10.2022 - 11.11.2022	виконано
8	Захист програмного продукту	17.10-2022 – 21.10.2022	виконано
9	Підготовка магістерської дисертації	22.10.2022 – 20.11.2022	виконано
10	Передзахист	23.11.2022	виконано
11	Захист	21.12.2022	виконано

Студент _____ Гнатишин М.С.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Свинчук О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми. Удосконалення наявних інструментів для самостійного контролю знань студентами, які навчаються в закладах вищої освіти, а також потреба у впровадженні сучасних інформаційних систем в освітній процес є важливим завданням сьогодення. Всі наявні більше спеціалізуються на широкій аудиторії студентів, для яких достатньо звичайних тестових завдань. Тому є необхідність у впровадженні мультифункціонального процесу тестування знань, який би включав різні види тестування, можливість обмеження часу виконання, а також підтримував би завдання, які вимагають написання програмного коду.

Мета роботи. Метою даного дослідження є створення інформаційної системи моніторингу виконання тестових завдань для самоконтролю та оцінки якості знань.

Завдання наукового дослідження:

- проаналізувати існуючі програмні системи моніторингу виконання тестових завдань в освітньому процесі;
- розробити перелік вимог до системи, провести моделювання та проектування програмного забезпечення;
- розробити інформаційну систему, що включає наступні підзадачі:
  - створення системи побудови та відображення статистики проходження тестів для студентів;
  - створення системи проходження тестів;
  - створення системи побудови та відображення статистики проходження тестів для викладачів;
  - створення підсистеми запуску та перевірки завдань, що вимагають написання програмного коду;
- виконати тестування системи.

Об'єкт дослідження. Об’єктом дослідження є моніторинг виконання тестових завдань.

Предмет дослідження. Предметом дослідження є інформаційна система моніторингу виконання тестових завдань.

Методи дослідження. Алгоритм виконання програмного коду на різних мовах програмування для перевірки завдань, що вимагають написання програмного коду.

Практичне значення одержаних результатів. Реалізована інформаційна система може використовуватися в закладах вищої освіти для покращення навчального процесу та підвищення рівня знань студентів шляхом самостійної перевірки засвоєння пройденого в університеті матеріалу. За допомогою інформаційної системи моніторингу виконання тестових завдань планується вносити позитивні зміни до навчальних матеріалів викладачів. Студенти ж отримують чудовий інструмент для самоконтролю та перевірки знань.

Особистий внесок магістранта. Розроблено систему моніторингу виконання тестових завдань, вдосконалено існуючі аналоги шляхом реалізації завдань, які вимагають написання програмного коду, та алгоритму перевірки таких завдань за допомогою компіляції та виконання користувацьких програм.

Апробація результатів дисертації. Основні положення роботи доповідались і обговорювались на XV Міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2022», м. Одеса 20-21 жовтня 2022 р.

Публікації. Результати досліджень магістерської дисертації представлено в 1 публікації.

Структура та обсяг дисертації. Робота складається із вступу, шести розділів, висновку, списку використаних джерел із 19 найменувань, 2 додатків. У дисертації наявні містить 36 рисунків та 19 таблиць. Обсяг роботи — 99 сторінок, з них обсяг списку використаних джерел — 2 сторінки.

Ключові слова: тестування, моніторинг знань, веб-застосунок, навчальний процес, анонімне тестування, автоматизація.

ABSTRACT

Actuality of theme. Improvement of existing tools for independent control of knowledge by students studying in institutions of higher education, as well as the need to introduce modern information systems into the educational process. The need for a multifunctional knowledge testing process, which would include various types of testing, the possibility of limiting execution time, and would also support such types of tasks that require writing software code. Currently, similar systems do not exist, since all existing ones are more specialized for a wide audience of students, for whom ordinary test tasks are enough.

The purpose of the study. The purpose of this study is to create an information system for monitoring the performance of test tasks for self-control and assessment of the quality of knowledge.

Tasks of scientific research:

- analyze the existing software systems for monitoring the performance of test tasks in the educational process;
- develop a list of requirements for the system, conduct modeling and software design;
- develop an information system that includes the following subtasks:
  - creation of a system for building and displaying test passing statistics for students;
  - creating a system of passing tests;
  - creation of a system for building and displaying test passing statistics for teachers;
  - creating a subsystem for launching and checking tasks that require writing software code;
- perform system testing.

Object of study. The object of the study is the monitoring of the performance of test tasks.

Subject of study. The subject of the study is the information system for monitoring the performance of test tasks.

Research methods. Algorithm for executing software code in different programming languages for testing tasks that require writing software code.

The practical significance of the obtained results. The implemented information system can be used in institutions of higher education to improve the educational process and increase the level of students' knowledge by independently checking the assimilation of the material studied at the university.

With the help of the information system for monitoring the performance of test tasks, it is planned to make positive changes to the teaching materials of teachers.Students, in turn, receive an excellent tool for self-control and knowledge testing.

Personal contribution of the master's student. A system for monitoring the execution of test tasks was developed, existing analogs were improved by implementing tasks that require writing software code, and an algorithm for checking such tasks using the compilation and execution of custom programs.

Approbation of the results of the dissertation. The main provisions of the work were reported and discussed at the XV International Scientific and Practical Conference "Information Technologies and Automation - 2022", Odesa, October 20-21, 2022.

Publications. The results of the research of the master's thesis are presented in 1 publication.

The structure and scope of the dissertation. The work consists of an introduction, seven chapters, a conclusion, a list of used sources from 19 items, and 2 appendices. The dissertation contains 36 figures and 19 tables. The volume of the work is 99 pages, of which the volume of the list of used sources is 2 pages.

Keywords: testing, knowledge monitoring, web application, learning process, anonymous testing, automation.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 11

ВСТУП 12

1 ПОСТАНОВКА ЗАДАЧІ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ 13

1.1 Створення системи побудови та відображення статистики проходження тестів для студентів 13

1.2 Створення системи проходження тестів 14

1.3 Створення системи побудови та відображення статистики проходження тестів для викладача 14

1.4 Створення підсистеми запуску та перевірки завдань, що вимагають написання програмного коду 15

1.5 Потенційні користувачі системи 16

Висновки до розділу 1 16

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ 17

2.1 Аналіз існуючих аналогів 17

2.1.1 Moodle 17

2.1.2 Leetcode 19

2.1.3 TestGorilla 20

2.2 Проблема професійної орієнтації студентів 22

2.3 Виконання користувацького програмного коду в хмарі 22

Висновки до розділу 2 24

3 ЗАСОБИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ 25

3.1 Засоби розробки веб-застосунку 25
3.1.1 Мова програмування JavaScript 25
3.1.2 Односторінковий веб-застосунок 27
3.1.3 Мова запитів GraphQL 29
3.1.4 Платформа Node.js 31
3.1.5 Веб-фреймоворк NestJS для Node.js 32
3.2 Засоби збереження даних 34
3.2.1 Бази даних у веб-застосунках 34
3.2.2 PostgreSQL 35
3.3 Засоби розробки та розгортання 36
3.3.1 Середовище розробки програмного забезпечення IntelliJ IDEA 36
3.3.2 Система контролю версій git 37
Висновки до розділу 3 39
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ 40
4.1 Архітектура веб-застосунку 40
4.2 Реалізація клієнтської частини 41
4.3 Реалізація серверної частини 43
4.3.1 Модуль виконання завдань що вимагають написання програмного коду 44
4.3.2 Модуль статистики 45
4.3.3 Модуль виконання тестових завдань 47
4.4 База даних 49
4.4.1 Таблиця спроби проходження тесту 50

4.4.2 Таблиця запитання під час проходження тесту 51
4.4.3 Таблиця відповідей студентів 52
Висновки до розділу 4 53
5 РОБОТА КОРИСТУВАЧА З ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ 54
5.1 Інструкція студента 54
5.1.1 Пошук курсів та тестів 54
5.1.2 Початок проходження курсів та тестів 56
5.1.3 Виконання завдання в тесті 57
5.1.4 Перегляд результатів 60
5.1.5 Історія проходження курсів та тестів 62
5.2 Інструкція викладача 63
5.2.1 Перегляд загальної статистики 63
5.2.2 Перегляд статистики окремого тесту 64
Висновки до розділу 5 65
6 СТАРТАП ПРОЄКТ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ 66
6.1 Опис ідеї проекту 66
6.2 Технологічний аудит ідеї проєкту 68
6.3 Аналіз ринкових можливостей запуску стартап проєкту 69
6.4 Розробка ринкової стратегії проєкту 76
6.5 Розроблення маркетингової програми 79
ВИСНОВКИ 83
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 84

ДОДАТОК А 86
ДОДАТОК Б..... 90
ДОДАТОК В 100

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

API – Application Program Interface, програмний інтерфейс застосунку

БД – база даних

ІТ – інформаційні технології

ІС – інформаційна система

UI – User interface, користувацький інтерфейс

ЗВО – заклад вищої освіти

SQL – Structured Query Language

GraphQL – Graph Query Language, графова мова запитів

SOAP – Simple Object Access Protocol, протокол обміну структурованими повідомленнями.

REST – Representational State Transfer, підхід до архітектури мережевих протоколів

HTTP – HyperText Transfer Protocol, протокол передачі гіпертекстових документів

ВСТУП

Перевірка власних знань за допомогою тестування є чудовим способом закріпленням нового матеріалу. При цьому зазвичай потрібно декілька раз пройти один і той самий тест, щоб остаточно засвоїти щойно пройдену інформацію. Від якості системи тестування також залежить ефективність її використання в освітньому процесі. Наприклад, якщо тестування – це тільки одноманітні запитання з відкритою відповіддю, де можна просто вгадати правильний варіант, то розв’язання задачі з написанням коду буде ефективним завданням для перевірки знань програміста.

Важливим аспектом є також спеціалізація студентів, оскільки в деяких галузях виникає потреба в специфічних видах тестування. Наприклад, для студентів в галузі інженерії програмного забезпечення типовим завданням є написання невеликої кількості програмного коду, яка б розв’язувала певну задачу, наприклад, додавала б два числа.

Тому й виникає потреба в мультифункціональному процесі тестування знань, який би включав різні види тестування, можливість обмеження часу виконання, а також підтримував би такі види завдань, які вимагають написання програмного коду. Наразі аналогічних систем не існує, оскільки всі наявні більше спеціалізуються на широкій аудиторії студентів, для яких достатньо звичайних тестових завдань.

Дана система моніторингу виконання тестових завдань призначена для студентів та викладачів університету. Основний функціонал системи призначений для студента, для якого відкриті різні можливості, починаючи з пошуку курсів, закінчуючи виконанням завдань, перевіркою правильності відповідей та складання результатів тестування.

Система спеціалізується в галузі інформаційних технологій, оскільки містить можливість виконання задач, що вимагають написання програмного коду, але може бути використана і для інших галузей.

1 ПОСТАНОВКА ЗАДАЧІ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ

Метою даного проєкту є створення інформаційної системи моніторингу виконання тестових завдань для підвищення якості засвоєння навчальних матеріалів студентами. Основною аудиторією такої системи є викладачі та студенти закладів вищої освіти. При цьому в майбутньому до користування даною інформаційною системою можна буде залучати й роботодавців.

1.1 Створення системи побудови та відображення статистики проходження тестів для студентів

Одним з найважливіших призначень системи є допомога студентам з оцінюванням рівня їхніх знань з тієї чи іншої дисципліни в певній галузі. Для кращого розуміння свого прогресу в навчанні критично необхідною є можливість ретроспективного аналізу успішності, тобто перегляду попередніх результатів та порівняння їх з поточними для визначення прогалин в знаннях.

При цьому необхідно забезпечити анонімність студентів таким чином, щоб неможливо було ідентифікувати, який студент пройшов той чи інший тест. Також корисним доповненням до системи буде надання студентам можливості перегляду тестів, які вони пройшли в минулому, дозволу переглядати статистику та історію проходження тестів, включаючи дату й час проходження.

1.2 Створення системи проходження тестів

Основою для системи тестування студентів є, власне кажучи, саме тестування, тобто система, що дозволяє виконувати завдання, перевіряти правильність виконання, не допускати вгадування під час проходження тестів.

Дана система передбачає генерацію для кожного студента унікального (в межах множини можливих унікальних комбінацій запитань) набору запитань, визначеного викладачем, відображення запитань студенту та функціонал для введення відповіді.

При цьому для запитання з варіантами відповіді студенту необхідно тільки натиснути на правильний на його думку варіант. Для запитання з розгорнутою відповіддю йому потрібно ввести текст своєї відповіді. А для запитання, що вимагає написання програмного коду, йому потрібно вибрати мову програмування серед передбачених викладачем, ввести програмний код, перевірити його виконання на тестовому наборі даних.

Після проходження тестів система генерує звіт, де вказано, які запитання були виконані правильно, а які ні. Також відображається загальний відсоток правильності виконання тесту в цілому.

1.3 Створення системи побудови та відображення статистики проходження тестів для викладача

Однією з найважливіших підсистем даної системи є підсистема моніторингу виконання користувачами тестів та завдань. Для цього створений розділ статистики для викладача, який містить загальну статистику, а саме:

- кількість користувачів, які проходили тести даного викладача протягом останніх шести місяців;

- середній бал (у відсотковому співвідношенні) серед усіх тестів викладача протягом останніх шести місяців;
- найпопулярніші тести.

Також буде доступна детальна статистика по кожному тесту, який викладач створив раніше, а саме:

- статистика виконання тесту;
- статистика проходження тесту;
- оцінка задоволенням після проходження тесту.

Вся інформація, яка отримана з системи, відображається в графічному форматі за допомогою графіків та таблиць для максимально ефективного сприйняття та аналізу інформації, що допоможе викладачам краще будувати навчальні плани та розуміти потреби студентів.

1.4 Створення підсистеми запуску та перевірки завдань, що вимагають написання програмного коду

Завдання, які вимагають написання програмного коду, передбачають виконання практичних завдань в сфері інженерії програмного забезпечення. При цьому студент може обрати мову програмування серед дозволених викладачем, якою бажає написати програмний код. Для зручності написання коду поле для вводу повинне підтримувати базову підсвітку синтаксису обраної мови програмування.

Студенту доступні тестові набори даних для перевірки правильності роботи програми. При цьому, після виконання даного завдання, код, написаний студентом, виконується на всьому наборі вхідних та вихідних даних, які були внесені викладачем.

Такі завдання можуть мати обмеження по часу виконання програми, щоб не допустити використання явно не ефективних алгоритмів (наприклад, використання повного перебору).

1.5 Потенційні користувачі системи

Дана інформаційна система може використовуватися студентами з метою перевірки знань з певної теми/дисципліни. Важливою особливістю додатку є відсутність ідентифікації студентів, тобто даним програмним продуктом зможе скористатись будь-який зацікавлений користувач.

Другою групою потенційних користувачів є викладачі закладів вищої освіти, в яких є можливість викладати свої навчальні матеріали у вигляді тестів на сторінці. Для викладачів дуже цінним буде отримання статистики проходження опублікованих тестів, оскільки це допоможе покращити навчальні матеріали, зробити їх більш актуальними й такими, які відповідають поточним потребам студентів.

Висновки до розділу 1

У цьому розділі виконано аналіз головних складових частин інформаційної системи моніторингу виконання тестових завдань, сформульовано вимоги та опис функціоналу кожної з підсистем.

Розглянуто потенційних користувачів інформаційної системи, їхню взаємодію з системою, а також між собою. Наведено приклади розширення аудиторії новими видами користувачів в майбутньому за умови розширення функціоналу самої інформаційної системи.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

У рамках аналізу предметної області було досліджено потреби студентів та викладачів. Також було досліджено існуючі аналоги. Наразі не існує прямих аналогів запропонованій інформаційній системі, оскільки ті, що схожі по функціоналу, не мають всього необхідного та можуть лише частково задовольнити потреби, які може вирішити запропонована система.

2.1 Аналіз існуючих аналогів

Програмні продукти, які виконують функцію організації тестування, не є новим винаходом, оскільки на комерційному та відкритому ринках представлена величезна кількість різноманітних мобільних та веб-застосунків, які є більш узагальненими або призначеними для якоїсь конкретної галузі.

2.1.1 Moodle

Дане програмне забезпечення не можна назвати звичайним програмним продуктом, оскільки це ціла екосистема інструментів для побудови дуже спеціалізованої, оформленої для власних потреб, веб-платформи для дистанційного навчання та керування навчальним процесом [1]. Оскільки Moodle поширюється за відкритою ліцензією, ним може скористатись будь-який заклад вищої освіти, коледж чи школа. За допомогою інструментів Moodle можна організувати електронний університет з поділом на групи, розкладом, графіком занять, журналом оцінювання, викладеними матеріалами та курсами, тестами та практичними завданнями.

Завдячуючи таким позитивним сторонам цього продукту, він активно застосовується в університетах. Кожна кафедра чи факультет може мати свою унікальну систему дистанційного навчання на базі Moodle [2]. При цьому в кожної з таких систем буде свій план використання та свої вимоги. Наприклад, деяким кафедрам зручно організовувати тестування за допомогою Moodle, комусь зручно викладати там матеріали і виставляти оцінки, щоб студенти могли їх бачити, а для когось це просто зручна організація навчального процесу. У цілому, Moodle має зрозумілий та зручний інтерфейс (рис. 2.1).

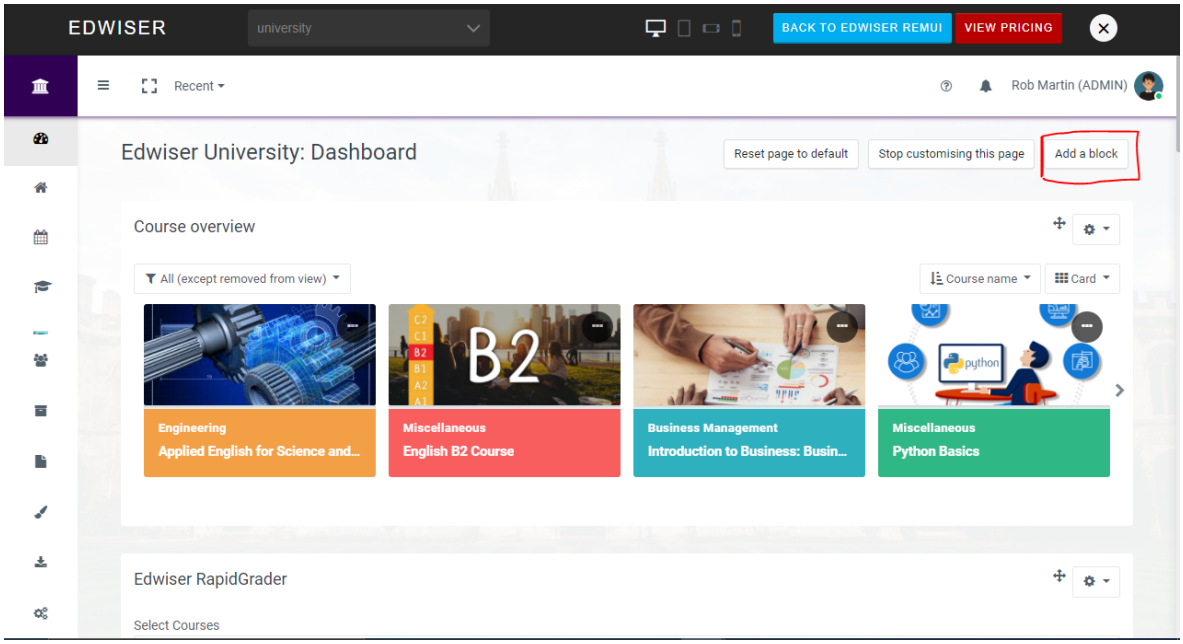


Рисунок 2.1 – Користувацький інтерфейс Moodle

У даному випадку Moodle розглядається саме як програмне забезпечення для тестування студентів і в нього є ряд недоліків. Оскільки це більш узагальнений програмний продукт, то він орієнтований на більшість клієнтської бази, а відповідно таких особливостей як можливість виконання завдань, що вимагають написання програмного коду, тут немає.

Тому можна зробити висновок, що запропонована інформаційна система для моніторингу виконання тестових завдань має кілька переваг над Moodle, а саме: відсутність авторизації для студентів, зручна статистика для студентів та

викладачів, а також можливість виконання завдань, що вимагають написання програмного коду.

2.1.2 Leetcode

Даний програмний продукт, на відміну від попереднього, є представником частково комерційного ринку, оскільки був розроблений для того, щоб приносити прибуток, хоч і за використання додаткових функцій, а не основного функціоналу. Leetcode [3] здобув свою популярність на фоні актуалізації теми виконання завдань, що вимагають написання програмного коду на співбесідах в компанії, які розшукували для себе інженерів-програмістів. Оскільки звичною практикою стало виконання таких завдань на інтерв'ю, то до цього потрібно було готуватись. У пригоді для цього ставав Leetcode, який пропонує дуже великий вибір задач, розділених за темами та складністю. Крім цього, даний програмний продукт підтримує велику кількість мов програмування й під кожну мову програмування для кожної задачі тут різні обмеження по часу виконання та ресурсів комп'ютера [4].

Іншим важливим аспектом Leetcode є подання виконання тестів як анонімного змагання між користувачами, оскільки після успішного виконання тієї чи іншої задачі можна дізнатись, наскільки програміст виконав його краще чи гірше порівняно з іншими користувачами, які виконували також це завдання. Наприклад, «ваше рішення працює швидше, ніж у 90% користувачів». Таким чином, з'являється бажання спробувати ще раз, але вже краще виконати це завдання, щоб результат був кращим за попередній.

Leetcode має простий та зрозумілий інтерфейс, а головною його функціональною особливістю є редактор програмного коду, який підтримує дуже велику кількість мові програмування (рис. 2.2).

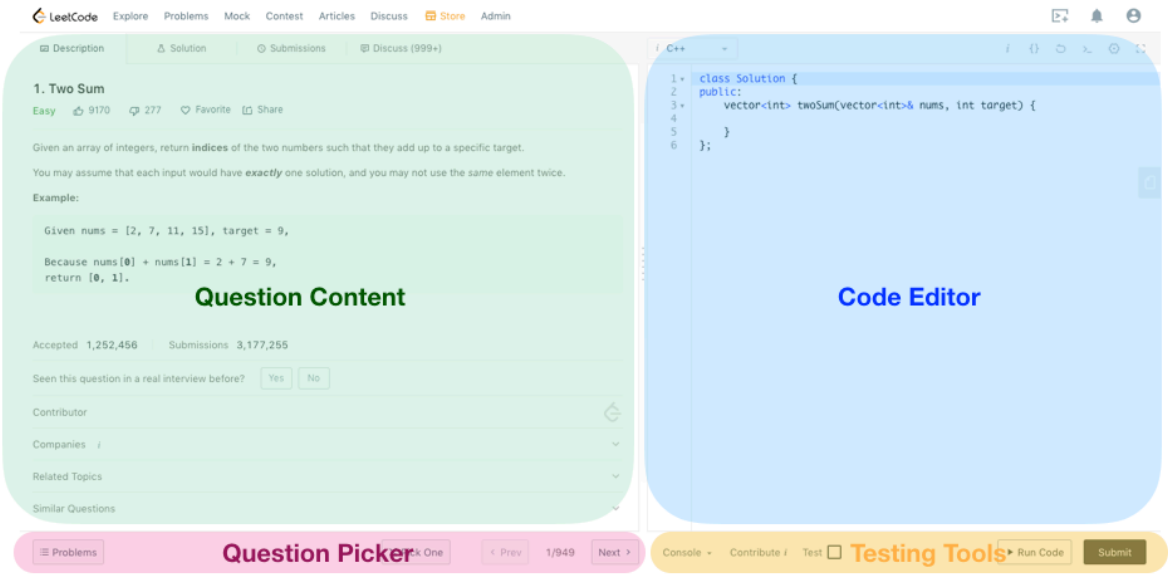


Рисунок 2.2 – Користувацький інтерфейс LeetCode

Проте в LeetCode є один великий мінус, а саме: завдання тут створює сама компанія, яка розробляє продукт і в користувачів немає можливості викладати авторські завдання. Крім того, в даному програмному продукті є тільки завдання, що вимагають написання програмного коду, в той час як в запропонованій системі моніторингу виконання тестових завдань є ще запитання з відкритою відповіддю та запитання з переліком варіантів відповідей

2.1.3 TestGorilla

Цей програмний продукт є найбільш сучасним серед розглянутих аналогів, тому в нього є багато переваг над конкретними. Його основна місія – це спрощення процесу проведення інтерв'ю в компанії на посади в сфері інженерії програмного забезпечення. TestGorilla є повністю комерційним і потребує оплати за використання [5].

З точки зору організації тестування, він є доволі схожий до інформаційної системи моніторингу виконання тестових завдань, оскільки теж передбачає

створення тестів одними користувачами для того, щоб їх проходили інші користувачі. При цьому даний сервіс підтримує запитання з набором правильних відповідей, відкриті запитання та запитання, що вимагають написання програмного коду. Завдячуючи своїй сучасності, даний програмний продукт має прогресивний користувацький інтерфейс, що забезпечує комфортну навігацію та використання системи користувачами (рис. 2.3).

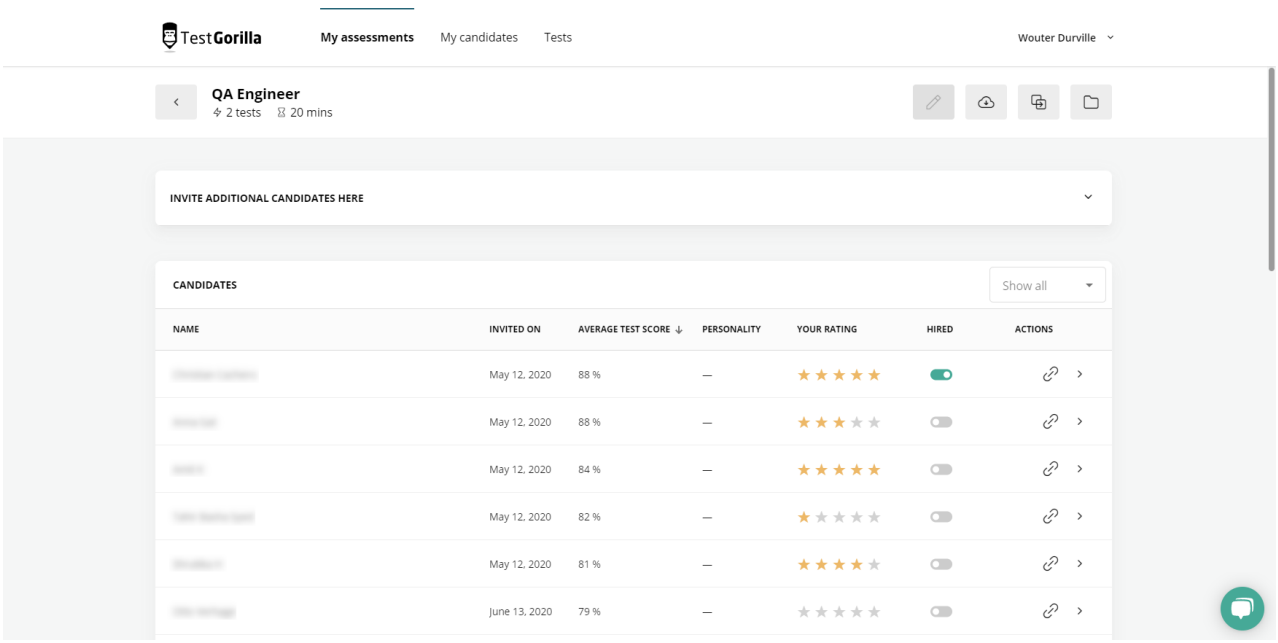


Рисунок 2.3 – Користувацький інтерфейс TestGorilla

Розглянутий програмний продукт є чудовим аналогом інформаційної системи моніторингу виконання тестових завдань, проте він має кілька недоліків. Першим й основним недоліком є відсутність анонімності, оскільки передбачається, що користувачу, який створив тест, відомо хто і з яким результатом його пройшов. Іншим недоліком є більш концептуальний підхід до реалізації програмного забезпечення, який базується на комерційних засадах, в той час як інформаційна система моніторингу виконання тестових завдань розробляється як внутрішня система для університету.

2.2 Проблема професійної орієнтації студентів

Вища освіта в галузі інформаційних технологій є без перебільшення унікальною на даний момент. Тут, на відміну від інших галузей освіти, студенти можуть починати працювати за своєю професією вже навіть з першого курсу бакалаврату. Це стає можливим завдяки гнучкості самого навчального процесу, яке більше націлене на виконання певних завдань і демонстрацію результатів, ніж на вивчення великої кількості теоретичної інформації та проведення часу на семінарах.

У середньому, студенти, які навчаються на спеціальностях, пов’язаних з програмуванням, починають працювати з третього чи з четвертого курсу. Це зумовлено тим, що самі компанії зацікавлені найняти до себе спеціалістів вже на ранньому етапі, щоб після закінчення університету студент був справжнім фахівцем, готовим до виконання реальних задач.

Проте не всі самостійно можуть знайти свою першу роботу ще під час навчання. Інколи цьому допомагає співпраця університету з приватними компаніями, інколи ті чи інші курси професійного спрямування, а інколи викладачі, які зацікавлені в працевлаштуванні студентів.

Наразі не існує спеціалізованої інформаційної системи, яка б допомогла студентам з оцінкою власного рівня знань в тій чи іншій галузі програмування. Дана система допомогла б краще підготуватись до стажування в компаніях та й загалом підтягнути свій рівень знань за допомогою виконання різних практичних задач.

2.3 Виконання користувацького програмного коду в хмарі

Спеціалісти в сфері інженерії програмного забезпечення вже давно звикли до такої процедури як запуск програмного коду на локальному комп’ютері. Їм відомо,

що великі за обсягом логіки програми можуть використовувати значні ресурси комп’ютера таким чином, що це може навіть впливати на продуктивність системи в цілому. Саме тому вони намагаються писати програмний код так, щоб він виконувався максимально ефективно, з мінімальними витратами пам’яті та ресурсів центрального процесору.

Проте, якщо стоїть задача виконувати програмний код, написаний ще не професіоналами, при цьому робити це паралельно й у великій кількості, то необхідно вдаватись до детального проектування рішення.

Перш за все, потрібно звернути увагу на використання ресурсів серверу, оскільки вони є обмеженими. Саме тому потрібно реалізувати щось на кшталт запобіжників, якщо програмний код виконується занадто довго, використовує занадто багато пам’яті чи ресурсів процесору, тому таке виконання переривається завчасно.

Крім цього, потрібно дбати про кількість програм, які виконуються паралельно на сервері, оскільки навіть якщо весь програмний код, що виконується, буде максимально оптимізований, але самих програм, що потрібно виконати, буде дуже багато, це теж може призвести до значного використання апаратних ресурсів серверу.

Іншим важливим аспектом є підбір мов програмування доступних для написання користувацького програмного коду. Тут варто враховувати кілька чинників:

- популярність мови програмування;
- вивчення мови програмування на факультеті/кафедрі;
- складність підтримки виконання завдань на цій мові;
- наявність готових рішень для даної мови програмування.

Розглянувши ці аспекти, стає зрозуміло, що почати необхідно з найбільш популярних мов програмування, таких як JavaScript, проте зробити це так, щоб добавлення нових мов програмування було доволі простим і не вимагало значних змін в системі.

Висновки до розділу 2

У цьому розділі проведено ґрунтовний аналіз існуючих програмних засобів, які мають схожий функціонал до запропонованої інформаційної системи моніторингу виконання тестових завдань. У процесі розгляду кожного з аналогів було окресленого його сильні та слабкі сторони в порівнянні із запропонованою інформаційною системою. Зроблено висновок, що наразі не існує програмного забезпечення, яке виконувало б весь перелік функцій, запропонованих в інформаційний системі моніторингу виконання тестових завдань.

Розглянуто проблему професійної орієнтації студентів для кращого розуміння основних потреб студентів в даному питання. На основі цього розроблено деякі з вимог до програмного забезпечення таким чином, щоб це сприяло професійній орієнтації студентів в майбутньому.

Детально проаналізовано технічні проблеми, пов’язані з виконання користувацького коду в хмарі, оскільки це одна з основних функцій інформаційної системи моніторингу виконання тестових завдань. Було окреслено основні проблеми, які існують при побудові такого роду систем та наведено потенційні їх вирішення.

3 ЗАСОБИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ

При виборі засобів для розробки програмного продукту було зроблено акцент на актуальності та сучасності технологій. При цьому було враховано особливості системи. Дуже важливо, щоб програмні засоби не унеможливлювали розробку тієї чи іншої функції системи.

3.1 Засоби розробки веб-застосунку

При виборі засобів розробки для тієї чи іншої галузі програмування необхідно керуватись насамперед трендами самої галузі. Оскільки обрані технології мають бути сучасними, добре підтримуватись їхніми розробниками, забезпечувати ефективне виконання поставлених задач було прийнято рішення провести аналіз найбільш популярних засобів для розробки веб-застосунків.

3.1.1 Мова програмування JavaScript

У сфері розробки веб-застосунків зазвичай виокремлюють три групи мов програмування для реалізації програмного забезпечення. Це мови програмування для написання клієнтської частини, мови для написання серверної частини. Іншими є мови програмування, на яких можна реалізовувати як серверну, так і клієнтську частину. Третя група мов найбільше підходить для інформаційної системи моніторингу виконання тестових завдань, оскільки це забезпечить швидку та ефективну розробку програмного продукту, а деякі частини коду можна буде використовувати одночасно і в клієнтському, і в серверному застосунку.

Найбільш сучасною та актуальною мовою програмування, на якій можна виконувати написання програмного коду і для клієнта, і для сервера, є JavaScript [6]. Її код виглядає доволі просто та зрозуміло (рис. 3.1).

```
let meetups = [
  {name:'JavaScript', isActive:true, members:700},
  {name:'Angular', isActive:true, members:900},
  {name:'Node', isActive:false, members:600},
  {name:'React', isActive:true, members:500}
];
let sumFPChain = meetups.filter((m)=>{
    return m.isActive;
  })
  .map((m)=>{
    return m.members- (0.1*m.members);
  })
  .reduce((acc, m)=>{
    return acc + m;
  },0);
console.log(sumFPChain);   // Output will be 1890

```

Рисунок 3.1 – Приклад коду на JavaScript

Ця мова програмування здобула популярність завдяки своїй універсальності, ефективності, зручності, функціональності та простоті в використанні. Таким чином, на сьогоднішній день це вже не просто мова програмування для сфери веб-додатків, а найпопулярніша мова програмування в цілому в галузі, оскільки на ній можна реалізовувати і мобільні застосунки, і стаціонарні застосунки, і навіть програмувати мікроконтролери.

3.1.2 Односторінковий веб-застосунок

За останні 10 років сфера розробки веб-застосунків отримала значний поштовх завдяки збільшенню потужності користувацьких комп'ютерів. Таким чином, стало простіше і ефективніше виконувати більше складних процедур на стороні клієнта. Одним з яскравих прикладів такого процесу є набуття популярності в односторінкових веб-застосунках [7].

Основною ідеєю таких веб-застосунків є перенесення всієї логіки, яка відповідає за відображення даних на сторону клієнта. При цьому взаємодіяти з сервером тільки для отримання якихось даних, чи як це найчастіше буває, частини даних. Цим і відрізняється такий тип застосунків від традиційних, які звертались до серверу кожного разу, коли потрібно було змінити щось на користувацькому інтерфейсі (рис. 3.2).

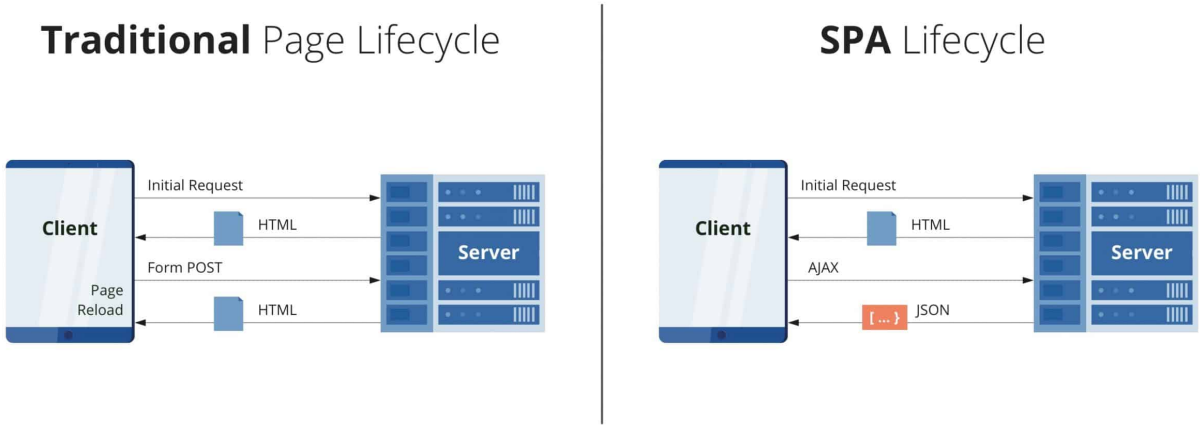


Рисунок 3.2 – Різниця між традиційним та одно сторінковим веб-застосунком

З набуттям популярності односторінкових веб-застосунків почали з'являтись й інструменти для зручної та ефективної їх розробки, оскільки серверні мови програмування вже не могли виконати цю задачу. Найбільш популярною в даному випадку стала мова програмування JavaScript, яка розрослась новим функціоналом впродовж останніх 10-15 років, а також отримала велику кількість сучасних

фреймворків та бібліотек для побудови одно сторінкових веб-застосунків. До них належать: AngularJS, Angular, React, Vue і т.д. Всі вони виконують одну функцію, проте застосовують для цього різні підходи та принципи. Наприклад, Angular базується на принципі надання всього необхідного функціоналу «з коробки», тобто всі бібліотеки, які можуть знадобитись будуть йти в комплекті з самим в фреймворком. React, з іншого ж боку, надає можливості тільки для побудови інтерфейсу, а решту функціоналу реалізується за допомогою сторонніх бібліотек на вибір програміста.

Найбільш популярним рішенням для написання односторінкових веб-застосунків наразі є бібліотека React [8]. Вона була розроблена технологічною компанією гігантом Facebook як внутрішня технологія для перебудови їхньої соціальної мережі, проте згодом почала розповсюджуватись як програмне забезпечення з відкритим кодом. Факт того, що дану бібліотеку розробляє та підтримує така велика компанія є безсумнівним плюсом даної технології, оскільки це робить її надійною і такою, на яку можна покладатись в довгостроковій перспективі.

React працює за принципом збільшення ефективності роботи клієнтського застосунку шляхом зменшення кількості перемальовування сторінки браузера. Оскільки процес перемальовування і перерахунку розташування елементів на сторінці є ресурс затратним, React застосовує свої внутрішні алгоритми, щоб цьому запобігти, коли не потрібно. Це дозволяє зробити його однією з найшвидших в плані відмальовування елементів клієнтського застосунку технологій, при цьому розмір самої бібліотеки є доволі скромний порівняно з іншими бібліотеками та фреймворками.

Саме тому React було обрано для написання клієнтської частини інформаційної системи тестування студентів, що є складовою комплексного програмного продукту.

3.1.3 Мова запитів GraphQL

Оскільки для реалізації веб-застосунку обрано клієнт-серверну архітектуру, де клієнтський і серверний застосунки – це дві окремі частини програмного забезпечення, які повинні комунікувати між собою певним чином, необхідно обрати засоби і технології для цієї комунікації. Основним протоколом у випадку розробки клієнт-серверних веб-застосунків є HTTP версії 1.1, оскільки саме цей протокол спілкування підтримують сучасні браузери [9]. Не зважаючи на те, що це єдиний протокол для такого спілкування, було придумано чимало способів взаємодії між частинами веб-застосунку на його основі.

Найстарішим та найменш актуальним наразі є архітектура SOAP, яка передбачала комунікацію за допомогою формату даних XML [10]. Така архітектура була популярна через популярність використання Java для написання класичних монолітних веб застосунків, а Java дуже добра працювала саме з форматом передачі даних XML [11]. Формат опису API передбачав використання в url запиту на сервер найменування дії яку хоче використати клієнт, наприклад “/getAllUsers”, чи “/getUserById&id=1”.

Більш сучасною та актуальною є архітектура REST, яка передбачає широке використання можливостей самого протоколу HTTP, насамперед його методів GET, POST, PUT, DELETE, PATCH для позначення виду операції яку клієнт хоче зробити на сервері, та стандартний кодів помилок для відображення невдач під час опрацювання запитів на сервері [12]. Філософією такого підходу є представлення серверних можливостей як ресурсів, наприклад ресурсом може бути користувач, над яким можна виконувати дії: створення (за допомогою POST запиту), оновлення (за допомогою PUT чи PATCH запиту), видалення (за допомогою DELETE запиту) та перегляд (за допомогою GET запиту). Крім цього, оскільки для вказання виду операції використовується не url, як в SOAP, а HTTP методи, то в url прийнято вказувати самі ресурси, над якими виконується операція, а або їхні під ресурси,

наприклад для отримання всіх користувачів GET “/users”, а для видалення одного користувача DELETE “/users/123”, де “123” це ідентифікатор користувача.

Третім і найбільш сучасним підходом є GraphQL, який передбачає використання тільки POST запитів як SOAP, проте JSON як формат передачі даних, аналогічно до REST [13]. Особливістю даного підходу є принцип описування API у вигляді графу, над яким клієнт має повний контроль при запиті. Під контролем мається на увазі вказання списку інформації про сутності, яку той бажає отримати. Для модифікації даних використовуються «мутації», які є чимось схожим до виклику функції в JavaScript. Такий підхід забезпечує найбільш гнучку та надійну розробку веб-застосунку (рис. 3.3).

```
schema {
  query: Query
  mutation: Mutation
}
type Mutation {
  createOffice(input: OfficeInput!): Office
}
type Query {
  company(id: ID!): Company
}
type Company {
  id: ID!
  name: String
  address: String
  age: Int @deprecated(reason: "No longer relevant.")
  offices(limit: Int!, after: ID): OfficeConnection
}
```

```
type OfficeConnection {
  totalCount: Int
  nodes: [Office]
  edges: [OfficeEdge]
}
type OfficeEdge {
  node: Office
  cursor: ID
}
type Office {
  id: ID!
  name: String
}
input OfficeInput {
  name: String!
}
```

Рисунок 3.3 – Приклад GraphQL схеми

Варто зауважити, що при використанні GraphQL значно простіше буде розширювати інформаційну систему моніторингу виконання тестових завдань в майбутньому, особливо з точки зору користувацького інтерфейсу.

3.1.4 Платформа Node.js

Завдяки неабиякій популярності мови програмування JavaScript почали активно розроблялись нові способи її застосування. Одним з таких способів є написання серверного коду на цій мові програмування. Для того, щоб це стало реальністю, необхідна спеціальна платформа, яка б добавила в JavaScript тих можливостей, яких немає при звичайному застосування в браузері. Такою платформою є Node.js [14]. На даний момент це основна найпопулярніша та найнадійніша платформа для написання серверних застосунків за допомогою мови програмування JavaScript.

Приклад серверного коду, а саме простого файлового серверу, можна переглянути на рисунку 3.4.

index.js

```
var static = require('node-static');

//
// Create a node-static server instance to serve the './public' folder
//
var file = new static.Server();

require('http').createServer(function(request, response) {
  request.addListener('end', function() {
    //
    // Serve files!
    //
    file.serve(request, response);
  }).resume();
}).listen(process.env.PORT || 3000);
```

Рисунок 3.4 – Приклад файлового сервера на Node.js

Основною концепцією Node.js є асинхронність, яка передбачає неблокуючий потік опрацювання операцій вводу виводу. Цей підхід покликаний компенсувати

один з великих недоліків Node.js, а саме однопоточність, яка не притаманна таким мовам програмування як Java, C# тощо.

За допомогою асинхронності Node.js дозволяє виконувати користувацький код, допоки відбувається очікування відповіді від стороннього сервісу чи бази даних. Це дозволяє отримати неабиякі продуктивність для високонавантажених систем, реалізованих за допомогою Node.js.

Іншою вагомою перевагою Node.js, в порівнянні з тими ж Java та C#, в контексті написання серверного коду для веб-застосунків є простота у вивченні та реалізації бажаних задумок. Коли в Java для реалізації HTTP серверу необхідно декілька класів та правильне налаштування, в Node.js для цього достатньо тільки кілька рядків коду.

3.1.5 Веб-фреймоворк NestJS для Node.js

Незважаючи на свою простоту та ефективність, розробка серверного застосунку на Node.js не є простою задачею, оскільки вимагає неабияких вмінь саме в цій галузі. На відміну від клієнтського коду, при розробці серверних застосунків потрібно враховувати можливість паралельного доступу до один і тих самих змінних. В такій ситуації складно ефективно розробляти програмні продукти, які будуть легко підтримуватись та супроводжуватись в майбутньому.

Тому для написання серверних застосунків на базі Node.js створена багато бібліотек та фреймворків які полегшують розробку, роблять код більш надійним та структурованим. До найпопулярніших фреймворків для Node.js відносять: Express.js, koa, fastify, NestJS.

Для інформаційної системи моніторингу виконання тестових завдань найбільше підходить останній, NestJS [15]. Він є найбільш сучасним на найбільш зрілим серед своїх конкурентів, оскільки побудований на базі філософії Spring Boot [16] для Java та Angular для клієнтської розробки на JavaScript. Даний фреймворк

включає в себе широкий набір інструментів для ефективної, швидкої та надійної розробки серверних застосунків на базі Node.js. Він одночасно є простим для розуміння та достатньо складним для виконання комплексних завдань. Побачити початкову структуру файлів в NestJS можна на рисунку 3.5.

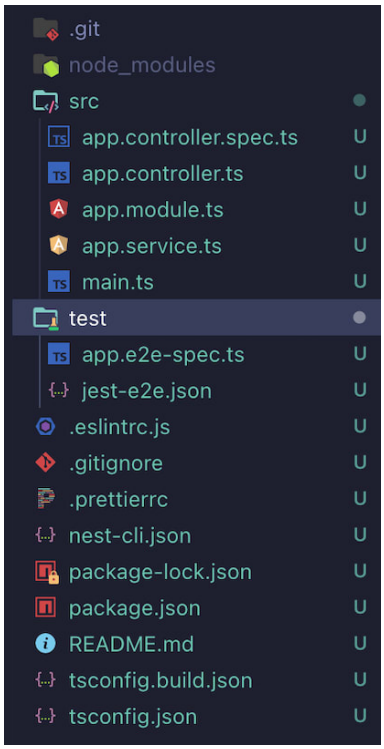


Рисунок 3.5 – Типова структура файлів в NestJS

Можна помітити, що основними компонентами NestJS є модулі, сервіси та контролери.

Модулі відповідають за організацію та інкапсуляцію коду, що допомагає в майбутньому масштабувати систему.

Контроллери відповідають за опрацювання клієнтських запитів, перевірку коректності запитів та формування відповіді.

Сервіси відповідають за основну логіку серверного застосунку, роботу з базою даних тощо.

3.2 Засоби збереження даних

Надійне збереження даних є дуже важливою частиною практично будь-якої інформаційної системи. Оскільки інформація може мати різні форми та види, то й бази даних бувають різних видів, мають різні принципи збереження та доступу до даних, працюють краще в тих чи інших ситуаціях та призначені для тих чи інших конкретних випадків використання.

3.2.1 Бази даних у веб-застосунках

Якщо дуже спростити зміст веб-застосунків в цілому, то можна дійти до висновку, що головно їхньою задачею є забезпечення надійного збереження даних з контролем доступу через браузер. Сервер в даному випадку виконує роль посередника між клієнтським застосунком та базою даних. Це звісно дуже спрощена модель, оскільки в сучасних програмних продуктах, серверні застосунки виконують широкий спектр задач.

У випадку використання баз даних у веб-застосунках варто, в першу чергу, звертати увагу на наступні чинники:

- масштабованість;
- пропускна здатність;
- можлива кількість паралельних підключень;
- можливості для моніторингу в режимі реального часу.

Перший пункт масштабованість пояснюється тим, що функціонал самого серверного застосунку, як і кількість його користувачів буде з часом рости. Це означає, що база даних як фундаментальна складова всієї системи повинна не тільки могли підтримувати існуючий функціонал, а й мати потенціал для його розширення в майбутньому.

Пропускна здатність та кількість паралельних підключень є важливими в контексті того, що користуватись веб-застосунком можуть десятки, сотні, а то й тисячі користувачів одночасно. Відповідно база даних повинна вміти справлятись з таким навантаженням та забезпечувати невеликий час опрацювання запитів для всіх вхідних з'єднань.

Можливість моніторингу в реальному часі є критично важливим чинником, який впливає на здатність швидко виявляти помилки в системі, досліджувати їх та усувати.

3.2.2 PostgreSQL

У якості бази даних для інформаційної системи моніторингу виконання тестових завдань було обрано сучасну систему керування базами даних PostgreSQL [17]. Дане програмне забезпечення являє собою об'єктно-реляційну базу даних, що зберігає дані у вторинній пам'яті та підтримує мову запиту SQL (Structured Query Language) з певними надбудовами, доступними тільки в PostgreSQL.

Великою перевагою даної СКБД, порівняно з популярними аналогами, є те, що вона поширюється на основі ліцензії програмного забезпечення з відкритим кодом та підтримується великою спільнотою ентузіастів-інженерів по всьому світі. Оскільки дана СКБД створена як універсальний інструмент, вона має дуже великі можливості для розширення. Наприклад, можна створювати свої типи даних, свої функції нові можливості і тд. PostgreSQL поєднує в собі найкращі сторони реляційних баз даних та об'єктних сховищ, що робить її дуже гнучкою у використанні з веб-застосунками.

Оскільки до інформаційної системи моніторингу виконання тестових завдань входять підсистеми статистики викладача та студента, база даних повинна володіти широким набором статистичних та аналітичних функцій задля підтримки побудови інформативних вибірок даних на основі вхідних параметрів. При правильному

проектуванні схеми бази даних, PostgreSQL чудово підходить під такі потреби, оскільки містить великий набір програмних засобів для побудови аналітичних запитів, при цьому оптимізує їх виконання.

3.3 Засоби розробки та розгортання

Якщо засоби реалізації програмного забезпечення, мови програмування, бібліотеки, підходи та інше впливає на роботу програмного забезпечення, то засоби розробки та розгортання впливають на те, як швидко та ефективно це програмне забезпечення буде розроблятись.

3.3.1 Середовище розробки програмного забезпечення IntelliJ IDEA

На сьогоднішній день існує величезна кількість засобів для написання програмного коду. Від простіших, які представляють звичайний редактор програного коду до складних, які володіють вбудованими компіляторами, штучним інтелектом на широким спектром налаштувань під себе.

Для написання програмного коду було використано середовище розробки програмного забезпечення IntelliJ IDEA [18]. Даний продукт вважається одним з найкращих для написання програмного для серверних застосунків. Хоч воно і спеціалізується на мові програмування Java, IDEA чудово підходить для написання програмного коду на JavaScript та TypeScript. Містить велику кількість зручних інструментів для покращення якості коду та спрощення його написання.

Завдяки своєму функціональному та зручному інтерфейсу є найпопулярнішим середовищем розробки програмного забезпечення в цілому (рис. 3.6).

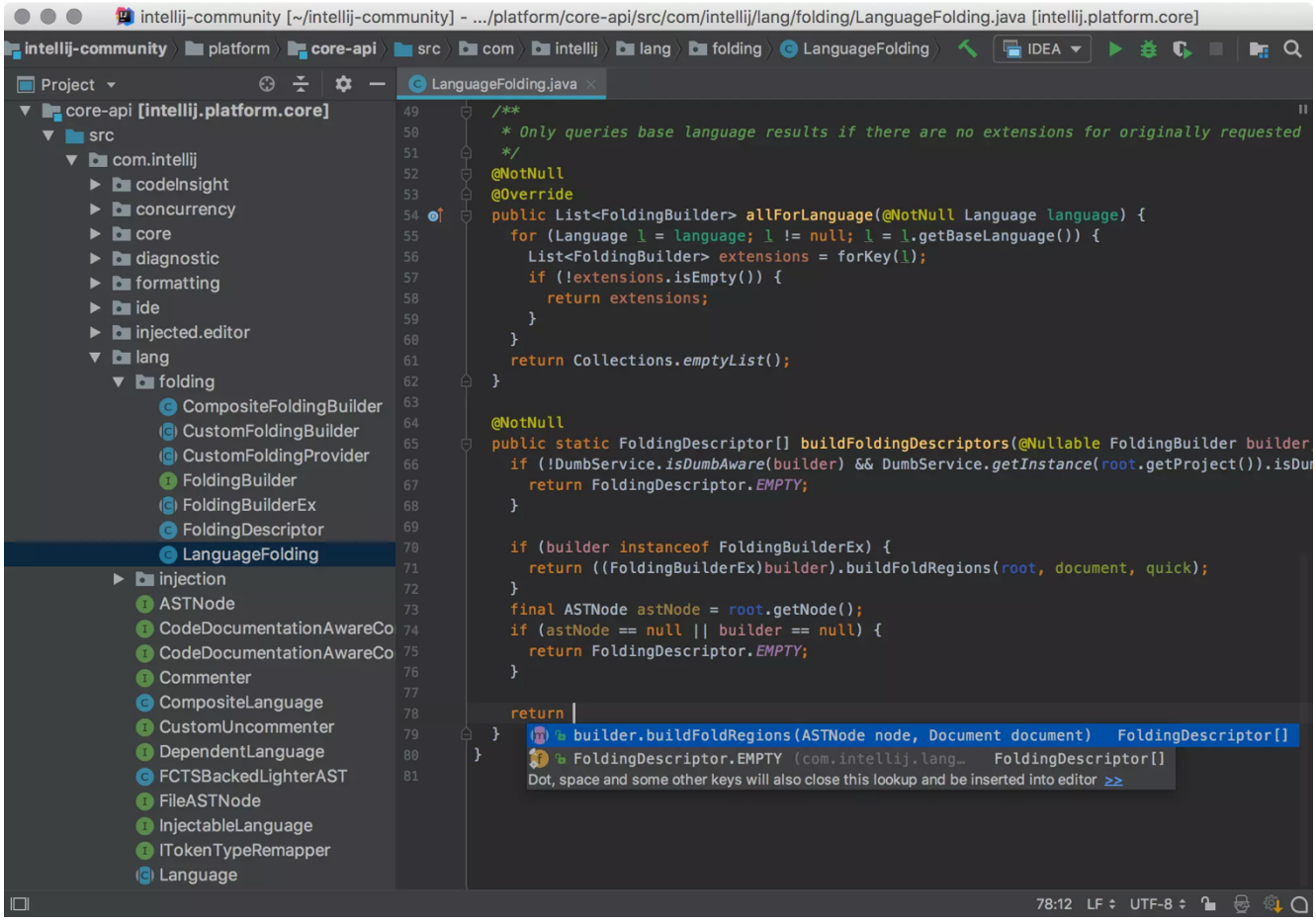


Рисунок 3.6 – Приклад роботи IntelliJ IDEA

Варто зазначити, що IntelliJ IDEA має вбудовану підтримку терміналу, який є дуже зручним в розробці.

3.3.2 Система контролю версій git

Оскільки інформаційна система моніторингу виконання тестових завдань є складовою частиною комплексної інформаційної системи, критично важливим є забезпечення можливості одночасного написання програмного коду кількома людьми.

У таких випадках на допомогу приходять системи контролю версій вихідного коду. Найпопулярнішим представником яких є система Git, яка широко використовується по всьому світу [19].

Проте задля одночасного доступу до вихідного програмного коду, його необхідно розмістити в такому місці, до якого буде доступ у всіх членів команди. У цьому випадку необхідно скористатись послугами спеціальних платформ, які виступають віддаленим репозиторієм для Git.

У випадку інформаційної системи моніторингу виконання тестових завдань такою платформою є GitHub (рис. 3.7).

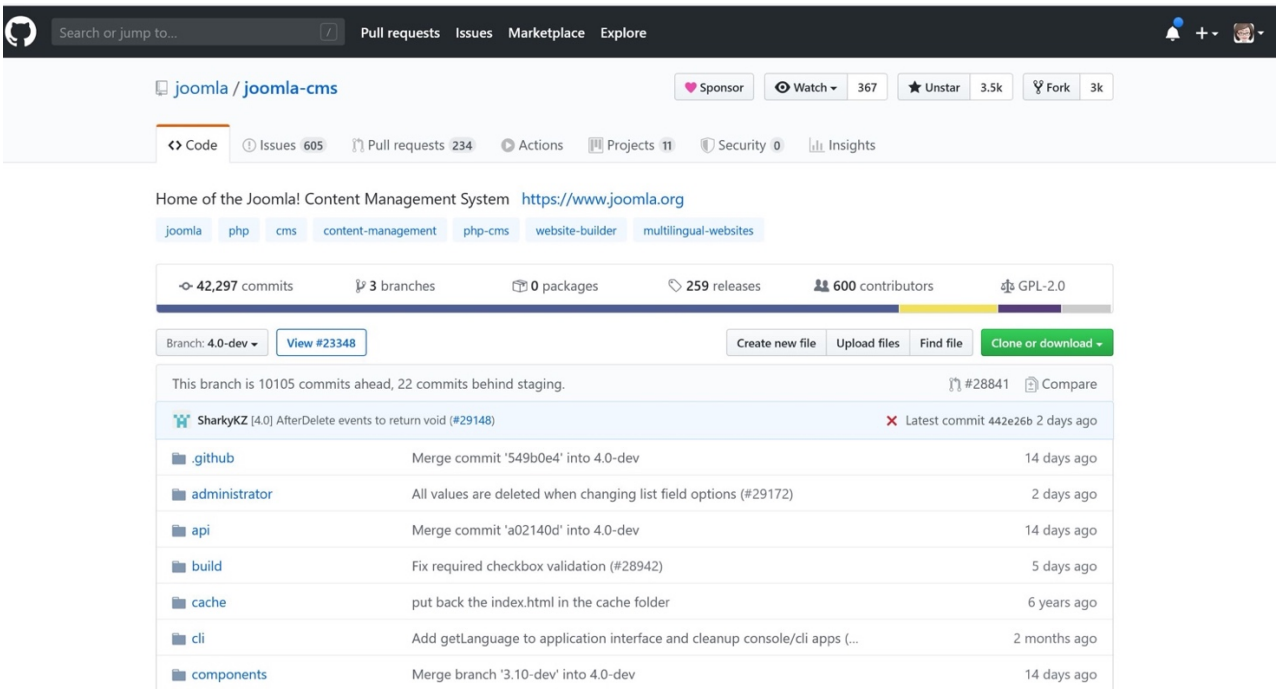


Рисунок 3.7 – Користувацький інтерфейс GitHub

Дана платформа пропонує широкий спектр функціоналу для ефективної роботи в команді, а також надає безплатні можливості для студентів.

Висновки до розділу 3

У даному розділі проведено детальний аналіз використаних програмних засобів та технологій при розробці інформаційної системи моніторингу виконання тестових завдань.

Проаналізовано сучасні підходи до розробки веб-застосунків, мови програмування, які використовуються, бібліотеки та фреймворки. В якості мови програмування для клієнтської та серверної частини було обрано JavaScript завдяки її зручності, універсальності та гнучкості. В якості основної бібліотеки для написання клієнтського застосунку було обрано React як найбільш надійний вибір серед можливих. Для написання серверного застосунка обрано Node.js з фреймворком NestJs, що забезпечить написання надійного, продуктивного коду, який буде легко підтримувати в майбутньому.

Описано потенційні методи і підходи до комунікації між клієнтським та серверним застосунками. Наведено їх порівняльну характеристику. Як спосіб комунікації між клієнтом та сервером обрано мову запитів та маніпуляції даними GraphQL як найбільш сучасне та гнучке рішення.

Виконано аналіз сховища даних для інформаційної системи моніторингу виконання тестових завдань, окреслено основні принципи використання баз даних у веб-застосунках. В якості СКБД обрано об’єктно-реляційну базу даних PostgreSQL. Наведено переліг її переваг перед аналогами.

Описано використані засоби для розробки програмного забезпечення як такого, тобто написання програмного коду, а також засобів для спільного написання програмного коду кількома учасника команди, оскільки дана робота є комплексною.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ

При розгляді архітектури додатку було обрано клієнт-серверну архітектуру, з монолітним принципом побудови сервера [20]. На відміну від мікросервісного підходу, даний підхід економить час на розробку та допомагає надійно побудувати систему, оскільки зменшується кількість помилок через комунікацією між мікросервісами [21].

4.1 Архітектура веб-застосунку

Інформаційна система моніторингу виконання тестових завдань є частиною системи тестування студентів та побудована на базі клієнт-серверного веб-застосунку, у якому є розділення клієнтську та серверну частини (рис. 4.1).

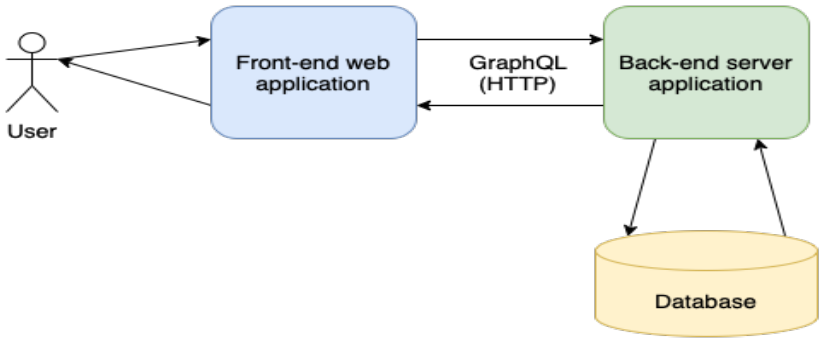


Рисунок 4.1 – Архітектура системи

Ці частини є незалежними та розгортаються окремо. Даний підхід додає гнучкості в програмний продукт, тому що декомпозована система дозволяє простіше підтримувати розробку та розширення інформаційної системи.

Клієнтська частина представлена односторінковим веб-застосунком та передбачає плавну зміну інтерфейсу без перезавантаження сторінки. Вона призначена насамперед для взаємодії з користувачами системи, а саме: адміністраторами, викладачами та студентами.

Серверна частина – це програмний застосунок, який розгортається на віддаленому сервері. Ця частина відповідає за опрацювання логіки інформаційної системи та комунікацію з базою даних, а також для збереження інформації.

4.2 Реалізація клієнтської частини

Як вже згадувалось в попередньому підрозділі, клієнтська частина являє собою односторінковий веб-застосунок, який реалізовано за допомогою бібліотеки React, яка створена для користувацьких інтерфейсів. Не зважаючи на свою назву «односторінковий», застосунок містить багато різних спеціалізованих сторінок-інтерфейсів для взаємодії з користувачем. Умовно його можна розділити на спільні елементи інтерфейсу та підсистеми:

- адміністратора;
- викладача;
- студента.

До підсистеми студента належать сторінки, які стосуються виконання завдань. Це і пошук курсів чи тестів. Різні елементи інтерфейсу відповідальні за написання програмного коду в завданнях, що такого вимагають, а також перегляд результату тестування (рис 4.2).

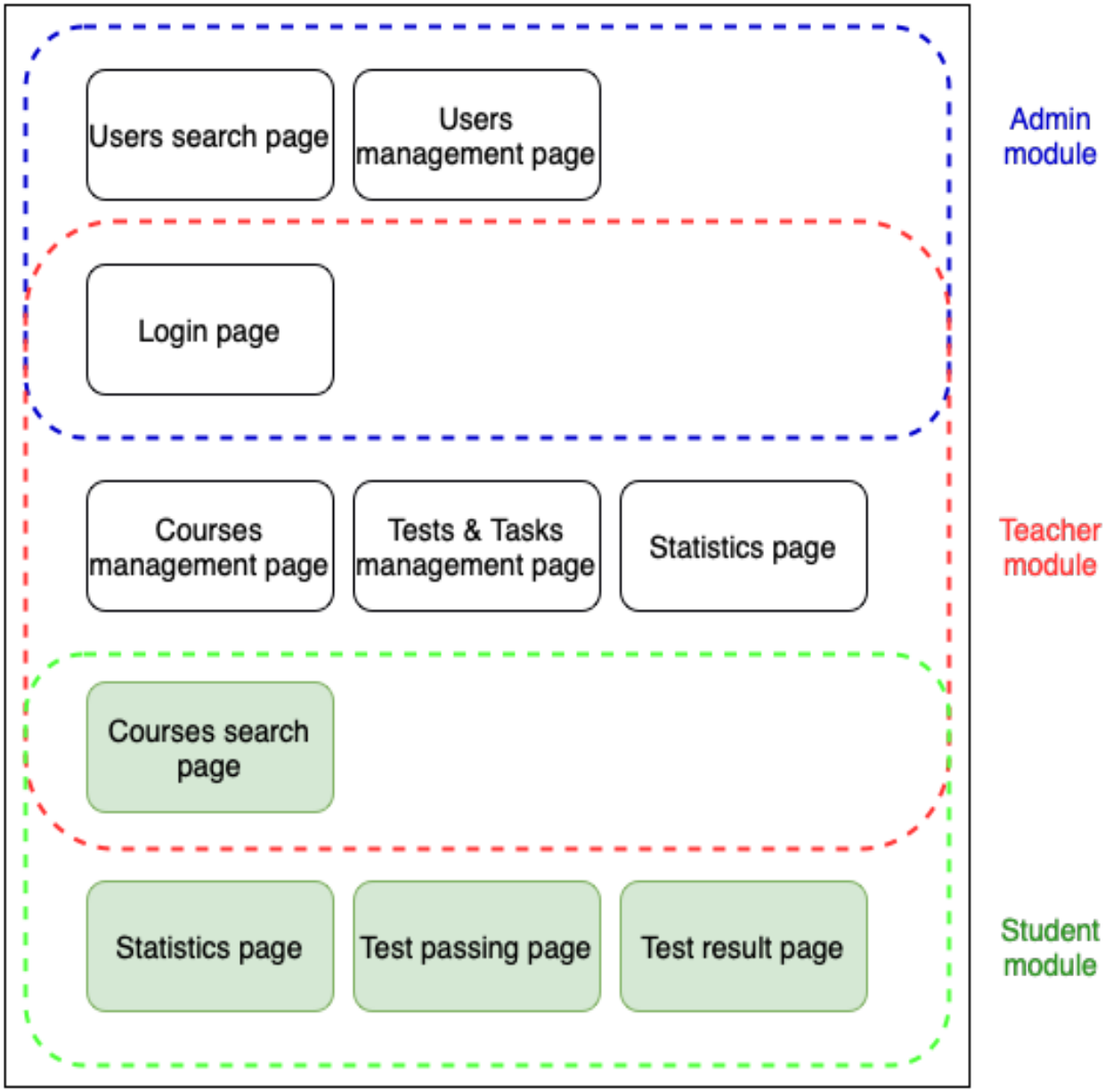


Рисунок 4.2 – Складові клієнтського застосунку

При цьому студенту доступна статистика виконання тестів. Вона відображається за попередні 6 місяців.

Варто зауважити, що статистика зберігається локально на пристрої студента, з якого він проходить тести, оскільки система не зберігає ідентифікаційних даних про користувачів.

4.3 Реалізація серверної частини

Серверний застосунок побудований на базі Node.js. При проектуванні було застосовано так звану кілька-шарову архітектуру додатку. Це означає, що програма поділена на кілька шарів, які мають різне призначення, є незалежними один від одного, при цьому комунікують між собою за допомогою API, що можна переглянути на рисунку 4.3.

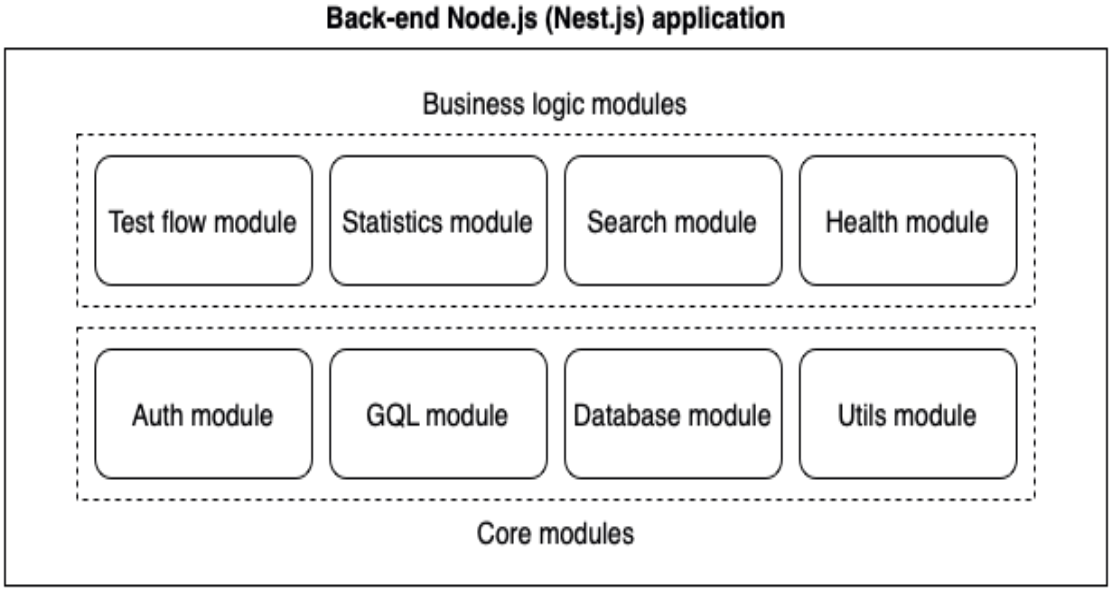


Рисунок 4.3 – Основні модулі серверного застосунку

Основними шарами є:

- API для комунікації з клієнтським застосунком;
- шар призначений для бізнес логіки системи;
- шар призначений для комунікації з базою даних.

У серверному застосунку для комунікації між клієнтською та серверною частиною використовується фреймворк GraphQL, який передбачає презентацію даних у вигляді графу, який описується схемою.

В якості контролерів, як в звичайному REST API, використовуються поняття резолверів (Resolver) для отримання даних та мутацій (Mutation) для модифікації даних.

4.3.1 Модуль виконання завдань що вимагають написання програмного коду

Модуль виконання завдань, які вимагають написання програмного коду є важливою частиною інформаційної системи моніторингу виконання тестових завдань оскільки реалізує одну з найважливіших особливостей системи – можливість виконання та перевірки написаного студентами програмного коду.

Незважаючи на свою складну внутрішню логіку, API цього модуля міститься тільки одну мутацію, яка відповідає, за власне виконання програмного коду на базі тих тестових даних, які задав викладач при створенні завдання (рис. 4.4).

```
extend type Mutation {
    executeTestCases(questionId: String!, input: TestCasesExecutionInput): TestCasesExecutionResult
}
```

Рисунок 4.4 – GraphQL схема модуля виконання програмного коду

Цей модуль є цікавим організацією внутрішнього коду та розподілом обов’язків між класами всередині. Оскільки виконання програмного коду є нетиповою задачею для веб-програмування, то й структура класів відрізняється від типової.

Можемо переглянути структуру класів даного модуля, яка складається з 8 класів (рис. 4.5).

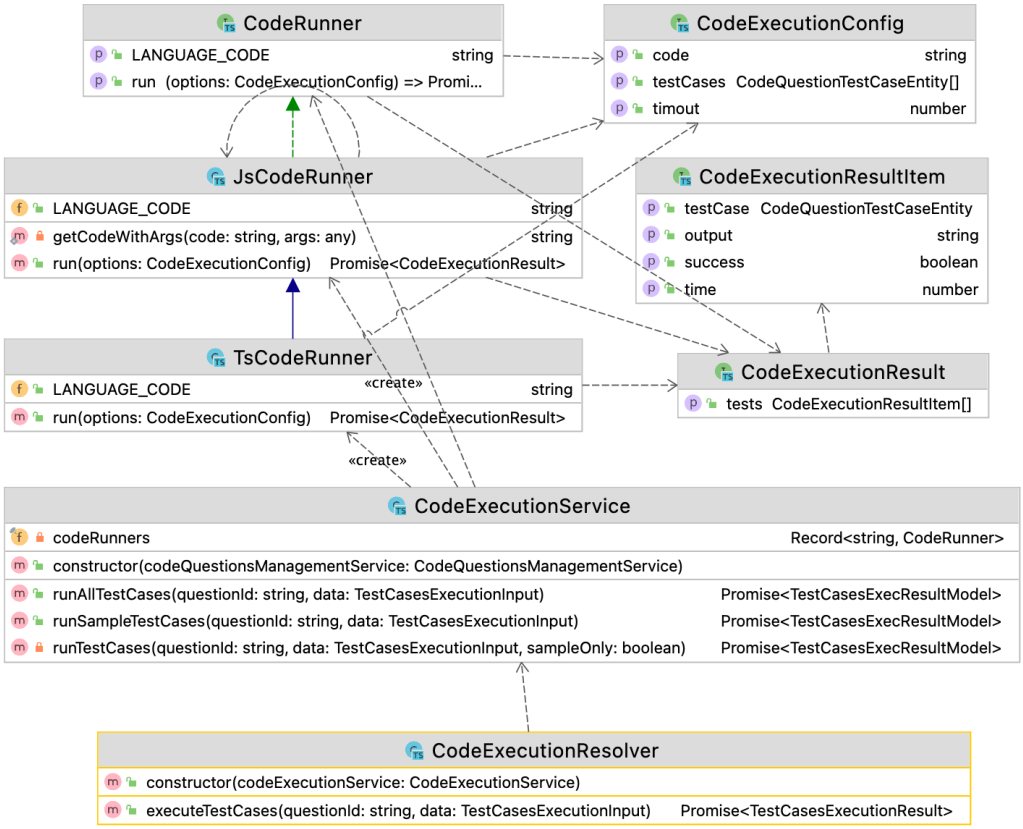


Рисунок 4.5 – Схема класів модуля виконання програмного коду

Серед них один резолвер, який надає доступ до API виконання коду, та 7 класів, які реалізують бізнес-логіку модуля.

4.3.2 Модуль статистики

В інформаційній системі моніторингу виконання тестових завдань модуль статистики відіграє ключову роль, оскільки забезпечує можливість спостереження за успішність виконання тестових завдань як для студентів, які їх проходять, так і для викладачів, які їх створюють. Можна переглянути структуру класів даного модуля на рисунку 4.6.

Рисунок 4.6 – Схема класів модуля статистики

До цього модуля входять 4 класи, два резолвери та два сервіси. Відповідно для статистики викладача один резолвер і відповідний сервіс та для студента аналогічно.

Варто зазначити, що для побудови статистики використовуються складні аналітичні запити в базу даних, які містять в собі кілька підзапитів та звертаються одночасно до декількох таблиць.

API даного модуля представлене чотирма запитами: отримання загальної статистики студента, списку виконаних тестів студентом, загальну статистику для викладача і статистику по певному тесту. Можливості API можна переглянути на рисунку 4.7.

```
extend type Query {
    studentStatisticGeneral(attempts: [String]): StudentGeneralStatistic
    studentTestsList(attempts: [String]): [StudentCompletedTest]

    teacherStatisticGeneral: TeacherStatisticGeneral
    testStatistic(testId: String!): TeacherTestStatistic
}
```

Рисунок 4.7 – GraphQL схема модуля статистики

Наразі даний модуль надає можливості отримання доволі базових статистичних даних, проте в майбутньому це можна буде з легкістю розширювати.

4.3.3 Модуль виконання тестових завдань

Основною задачею модуля виконання тестових завдань є надання студентам завдань для проходження тестів, опрацювання відповідей студентів на запитання, їх перевірка, та формування звіту з результатами. API модуля можна переглянути на рисунку 4.8.

```
extend type Query {
    testAttemptResult(attemptId: String!): CompletedTestAttempt
}

extend type Mutation {
    startTest(testId: String!): TestAttempt

    answerQuiz(attemptId: String!, questionId: String!, answers: [Int]): Boolean
    answerOpenText(attemptId: String!, questionId: String!, answer: String): Boolean
    answerCode(attemptId: String!, questionId: String!, code: String, language: String): Boolean
}
```

Рисунок 4.8 – GraphQL схема модуля виконання тестових завдань

До API входить один запит та чотири мутації. Запит повертає результати пройденого тесту, а мутації відповідають за отримання відповідей від студента.

Сам модуль складається з шести класів, серед яких один резолвер, три сервіси та два класи що відображають модель даних (рис. 4.9).

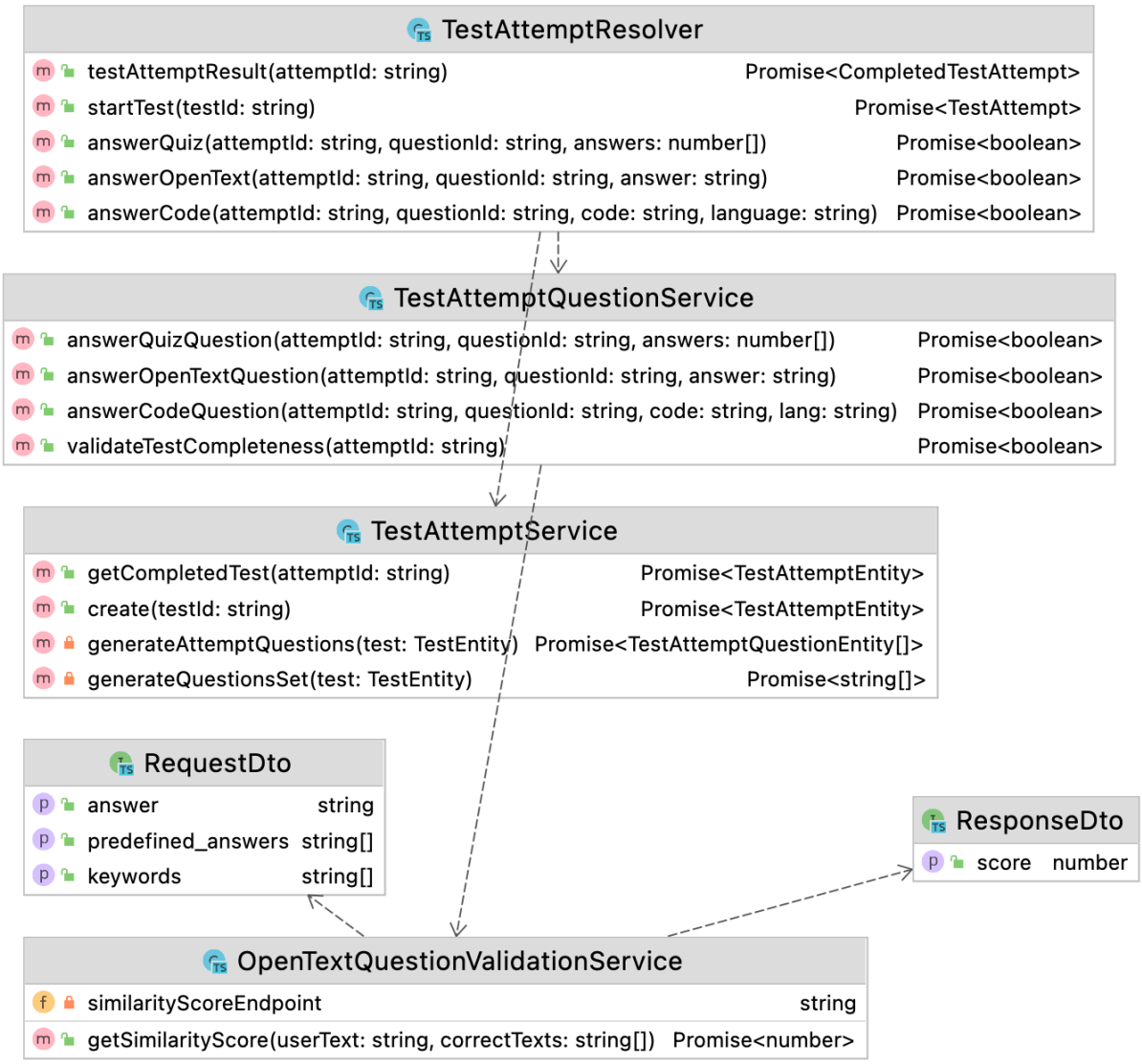


Рисунок 4.9 – Схема класів модуля виконання тестових завдань

В цілому, даний модуль побудований таким чином, щоб додавання нових видів завдань в систему було максимально простим та зручним. А будь-яка модифікація вже існуючих видів завдань могла бути виконано без змін, які б могли пошкодити працюючу систему.

4.4 База даних

У рамках проектування бази даних було змодельовано таблиці та зв’язок між ними, які призначені для підтримки збереження даних про систему моніторингу тестування студентів (рис. 4.10).

Рисунок 4.10 – Схема бази даних

При побудові схеми бази даних було враховано основі поради щодо використання реляційних баз даних [22]. Виконано нормалізацію даних. Враховано необхідність побудови індексів за полями, які беруть участь у вибірці.

4.4.1 Таблиця спроби проходження тесту

Коли студент розпочинає проходження тесту, тоді створюється запис в таблиці test_attemps (рис. 4.11).

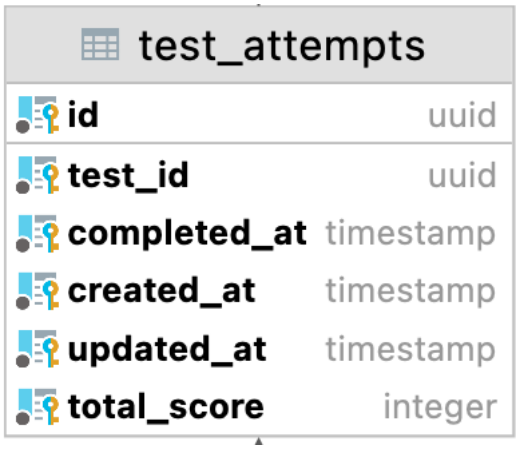


Рисунок 4.11 – Схема таблиці test_attempts

Дана таблиця містить наступні поля:

- ідентифікатор спроби;
- ідентифікатор тесту, до якого вона відноситься;
- час закінчення проходження тесту;
- час створення;
- час останнього оновлення;
- загальну оцінку за тест.

Після виконання тесту в неї записується час виконання, а також кількість балів, які отримав студент за розв’язання завдань.

Варто зазначити, що дана таблиця є важливою для майбутнього відображення статистики як для викладачів, так і для студентів, оскільки саме в ній містить інформація про результати проходження тих чи інших тестів студентами.

4.4.2 Таблиця запитання під час проходження тесту

Коли студент розпочинає тест, то серед всіх запитань, які є в тесті, обирається певна кількість, відповідно до того, як визначив викладач, та перемішується випадковим чином, а після цього вже потрапляє до студента. Кожне таке запитання, яке має бути виконане в рамках певної спроби проходження тесту, записується до таблиці test_attempt_questions (рис. 4.12).

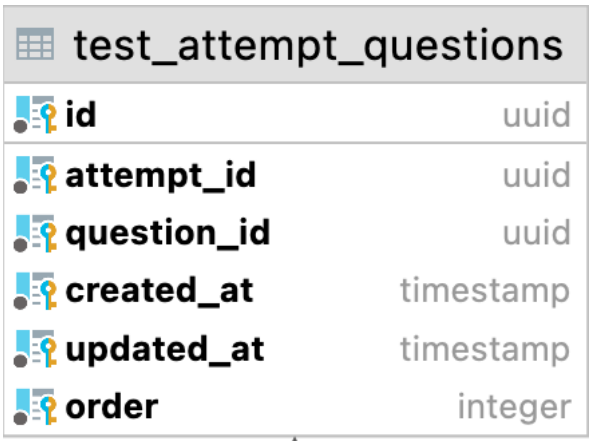


Рисунок 4.12 – Схема таблиці test_attempt_questions

Вона містить наступні поля:

- ідентифікатор запитання в рамках спроби;
- ідентифікатор спроби проходження тесту, до якої вона відноситься;
- ідентифікатор запитання, до якого вона відноситься;
- час створення;
- час останнього оновлення;
- порядковий номер під час відображення для студента.

Дана таблиця надає можливість реалізації додаткового функціоналу, який полягає в тому, що студентам будуть видаватись кожного разу різні запитання та в різному порядку, що є дуже зручним для проведення тестувань в цілому.

4.4.3 Таблиця відповідей студентів

Коли студент відповідає на запитання, його відповідь перевіряється на правильність, після чого і відповідь і результати перевірки записуються до таблиці test_attempt_question_answers (рис. 4.13).

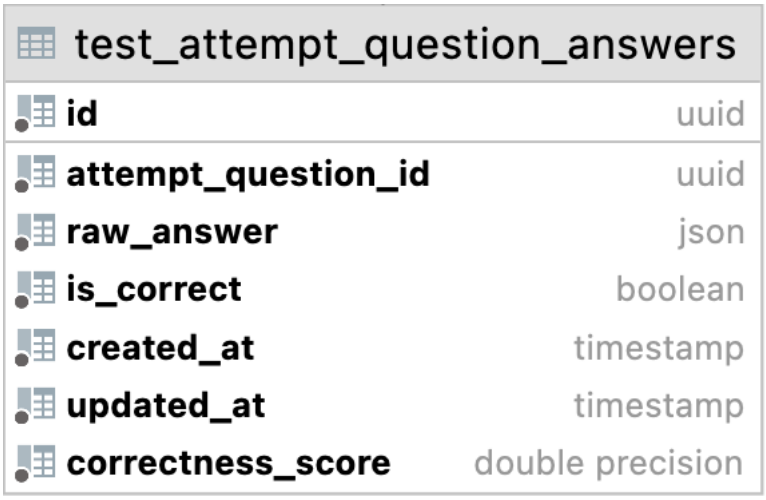


Рисунок 4.13 – Схема таблиці test_attempt_question_answers

Вона містить наступні поля:

- ідентифікатор відповіді студента;
- ідентифікатор запитання в рамках спроби проходження тесту, до якого вона відноситься;
- відповідь студента у вхідному форматі;
- позначення правильності відповіді;
- час створення;
- час останнього оновлення;
- відсоток правильності відповіді.

Дана таблиця грає важливу роль при побудові статистики виконання кожного конкретного тесту для викладача, оскільки за допомогою даних з неї можна проводити аналіз того, як добре студенти відповідають на це запитання.

Також дана таблиця є необхідно для відображення відповідей студента після завершення проходження тесту. Це дає частково зрозуміти, які відповіді були правильні, а які ні, що дозволяє краще засвоювати матеріал.

Висновки до розділу 4

У даному розділі описано загальну архітектуру веб-застосунку інформаційної системи моніторингу виконання тестових завдань. Було обґрунтовано вибір підходів під час проектування загальної концепції роботи програмного забезпечення.

Розглянуто структуру клієнтського застосунку, його функціонал та основні сторінки. Описано принцип поділу на сторінки та основні особливості користувацького інтерфейсу в системі. Продемонстровано принцип комунікації клієнтського та серверного застосунків, який відбувається на основі GraphQL.

Розглянуто архітектуру серверного застосунку, наведено діаграму модулів та підсистем. Окремо описано найважливіші модулі з переліком функціоналу, який вони реалізують, частинами схем GraphQL, які вони реалізують, а також з діаграмами класів, що до них входять.

Наведено діаграму бази даних, з детальним описом всіх таблиць системи та зв’язками між ними. Окремо розглянуто найважливіші таблиці, які стосуються процесу проходження тестів, а також грають важливу роль при побудові статистичних звітів для викладачів та студентів.

Даний розділ в цілому описує програмне забезпечення розроблене в рамках інформаційної системи моніторингу виконання тестових завдань, ілюструє структуру компонентів системи та взаємодію між ними.

5 РОБОТА КОРИСТУВАЧА З ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ

У даному розділі наведена інструкція використання системи студентами, які бажають пройти тестування для оцінки власного рівня знань з того чи іншого предмету чи технології. Насамперед, студенту доступний пошук курсів та тестів за текстом та категоріями. Після вибору бажаного тесту він може спробувати його виконати і після виконання подивитись на результат.

5.1 Інструкція студента

Студент є основним користувачем системи, оскільки основною метою розробки інформаційної системи моніторингу виконання тестових завдань є допомога студентам.

5.1.1 Пошук курсів та тестів

Студент повинен відкрити сторінку зі списком курсів.

Для пошуку курсу студент може використовувати поле для пошуку або фільтрацію за певними параметрами. Також користувач може сортувати отриманий результат.

Пошук курсів відбувається за:

- назвою курсу;
- назвою тесту;
- іменем викладача.

Фільтрація курсів відбувається за наступними параметрами:

- складністю;
- критерії;
- цільова аудиторія.

Приклад зображено на рисунку 5.1.

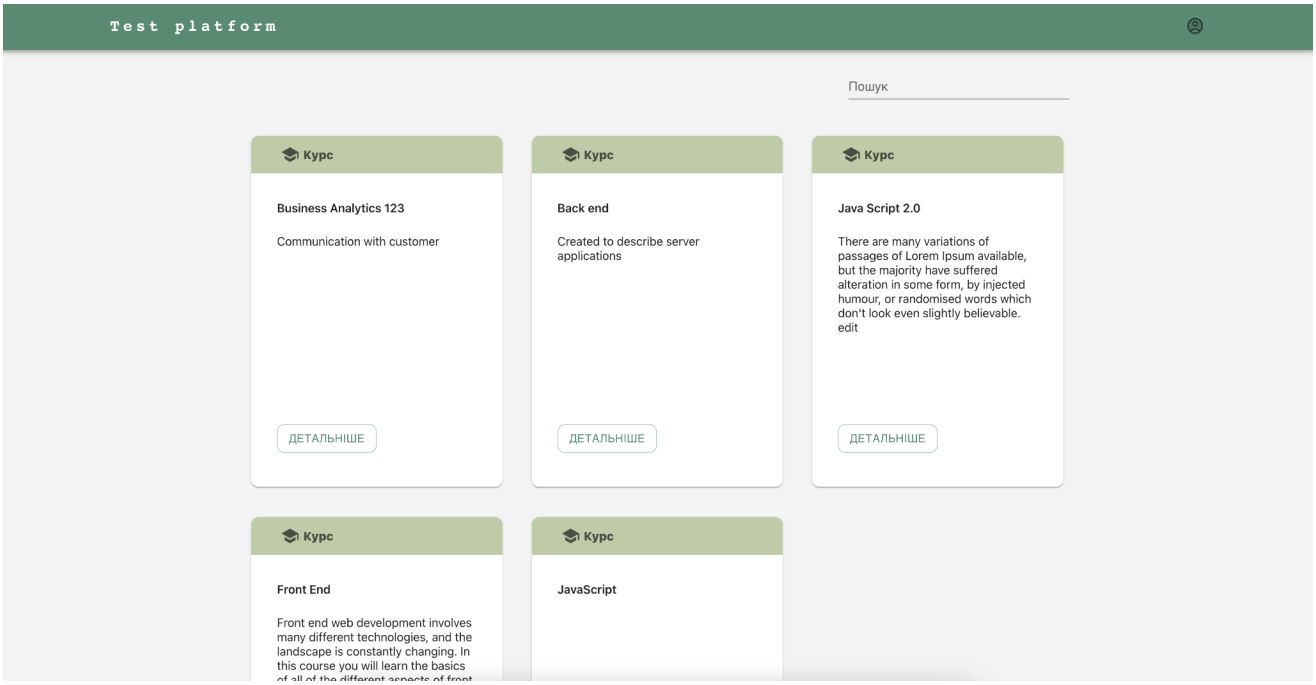


Рисунок 5.1 – Сторінка зі списком курсів

Сортування курсів відбувається за:

- датою створення;
- популярністю проходження;
- складністю.

Після виконання пошуку результати відображаються на екрані користувача. При цьому для кожного курсу відображається коротка інформація.

Натиснувши на кнопку «Детальніше», можна знайти більш детальну інформацію про курс.

5.1.2 Початок проходження курсів та тестів

Для проходження тесту студенту необхідно перейти на сторінку з списком тестів. При цьому можна переглянути детальну інформацію про даний тест:

- за ім’ям автора, якщо автор раніше дозволив його відображення;
- по категоріям курсу, які були обрані викладачем, що створив даний тест;
- окремо коротку інформацію по кожному тесту.

Після натискання на кнопку «Детальніше» користувача буде переадресовано на сторінку з конкретним тестом, де він вже зможе розпочати його безпосереднє виконання (рис. 5.2).

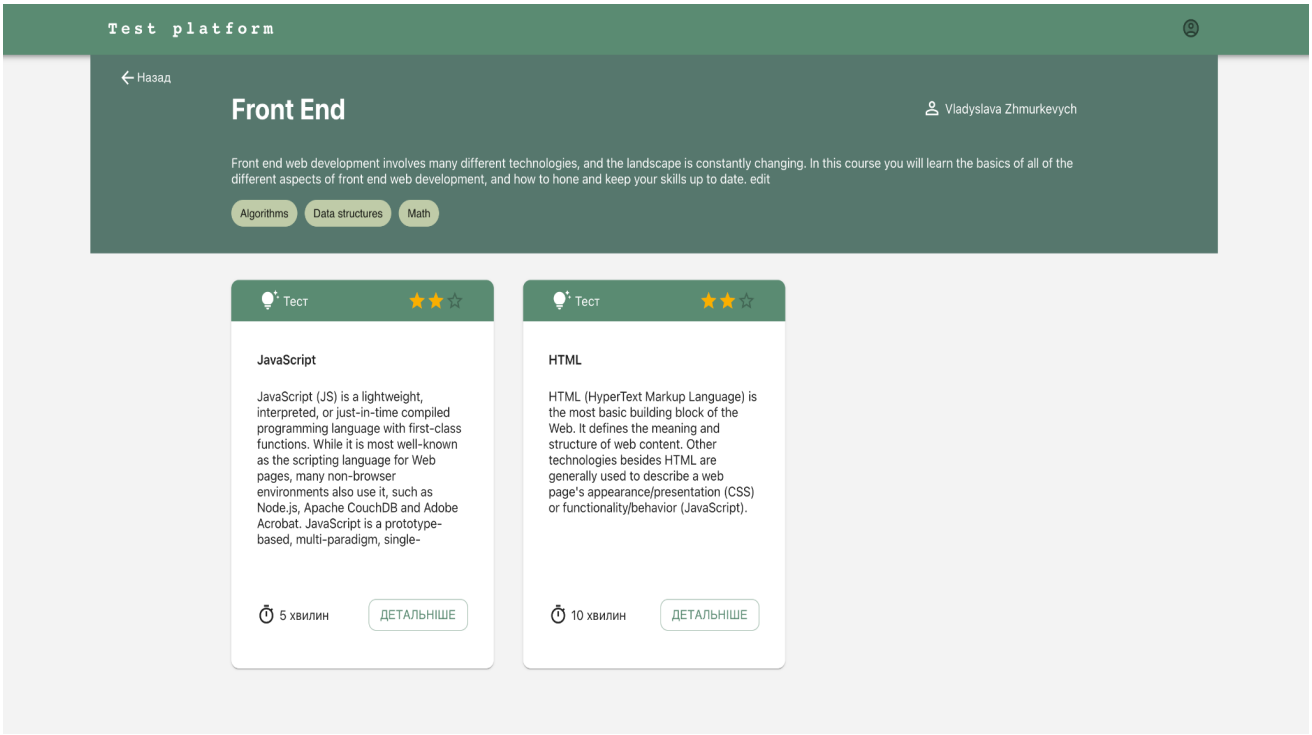


Рисунок 5.2 – Список доступних тестів в курсі та інформація про них

Для того, щоб розпочати проходження тесту, студенту необхідно натиснути на кнопку «Детальніше» на бажаному тесті. Після чого його буде переадресовано на сторінку з деталями проходження тесту (рис. 5.3).

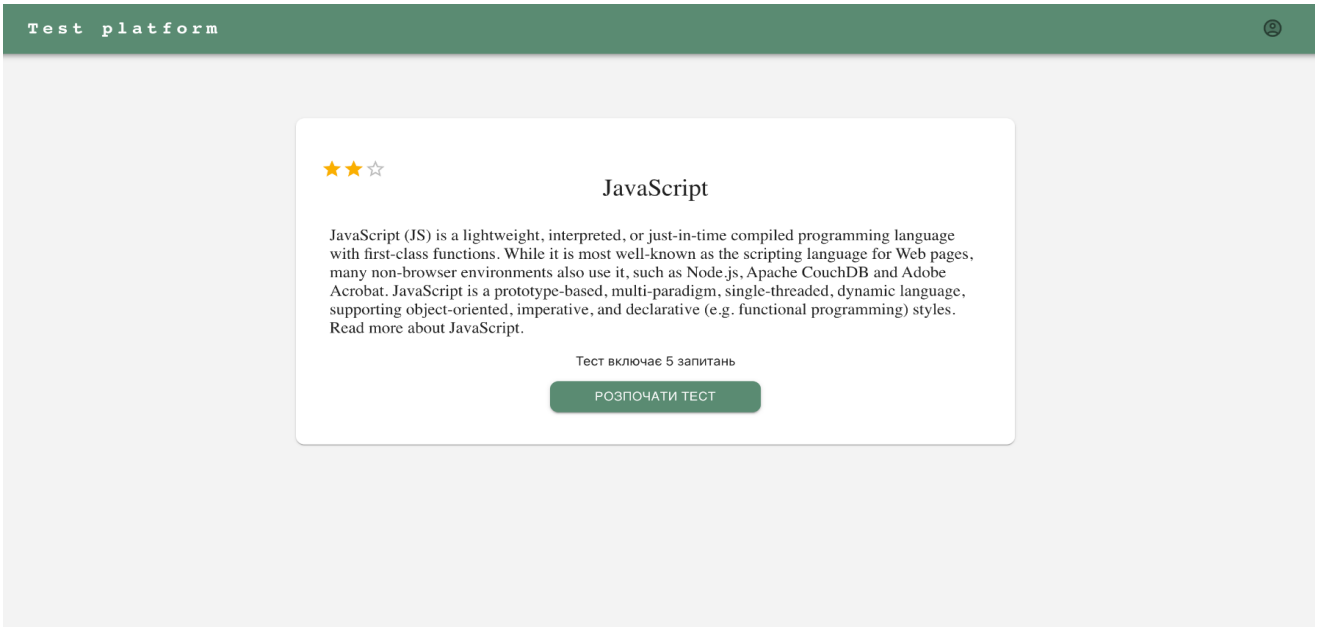


Рисунок 5.3 – Сторінка з початком проходження тесту

Потім студент може ознайомитись з умовами проходження тесту. У деталях тесту вказано кількість завдань, максимальний час виконання тесту (якщо такий було задано викладачем), ПІБ викладача автора тесту (якщо викладач дозволив відображення цієї інформації).

5.1.3 Виконання завдання в тесті

Для виконання завдання з набором варіантів відповідей студенту необхідно вибрати один або декілька варіантів відповідей серед запропонованих.

Якщо користувач вибере не всі правильні відповіді, то результатом виконання завдання буде вважатись відсоток вибраних правильних відповідей відносно кількості всіх правильних відповідей (рис. 5.4).

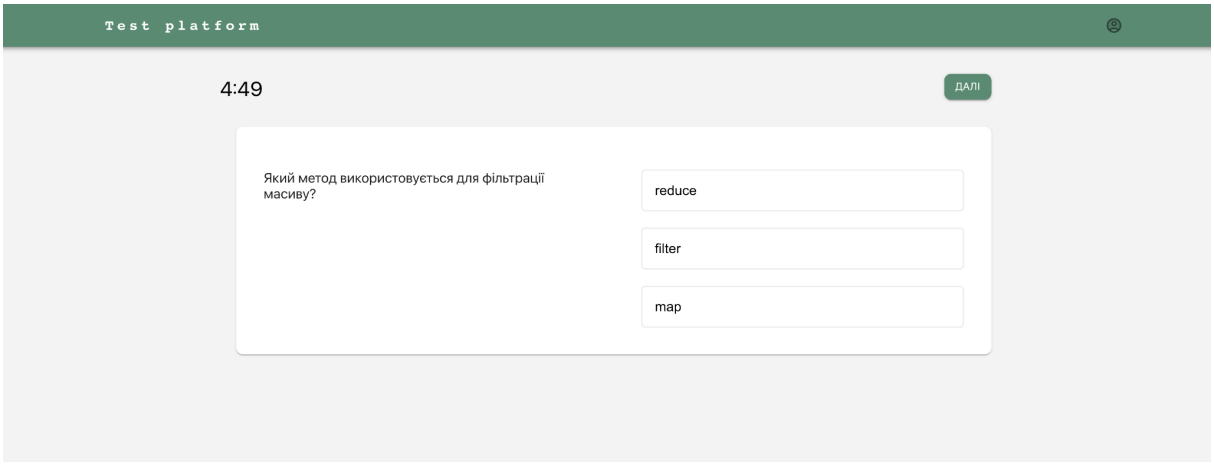


Рисунок 5.4 – Сторінка з виконанням завдання з набором варіантів відповідей

Для виконання завдання з відкритою відповіддю студенту необхідно надати відповідь у текстовому форматі. При цьому надана відповідь буде порівняна з правильною, а за результат виконання завдання буде взято до уваги відсоток співпадіння змісту відповідей (рис. 5.5).

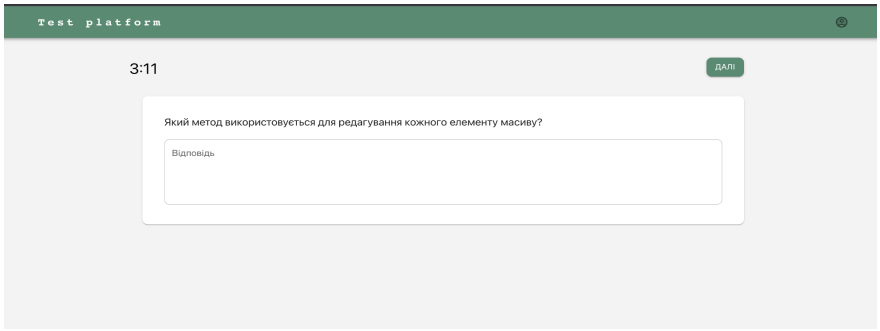


Рисунок 5.5 – Сторінка з виконанням завдання з відкритою відповіддю

Для виконання практичного завдання студенту необхідно написати програмний код, який буде задовольняти умову завдання, та для заданих вхідних даних повертати очікувані вихідні дані (рис. 5.6).

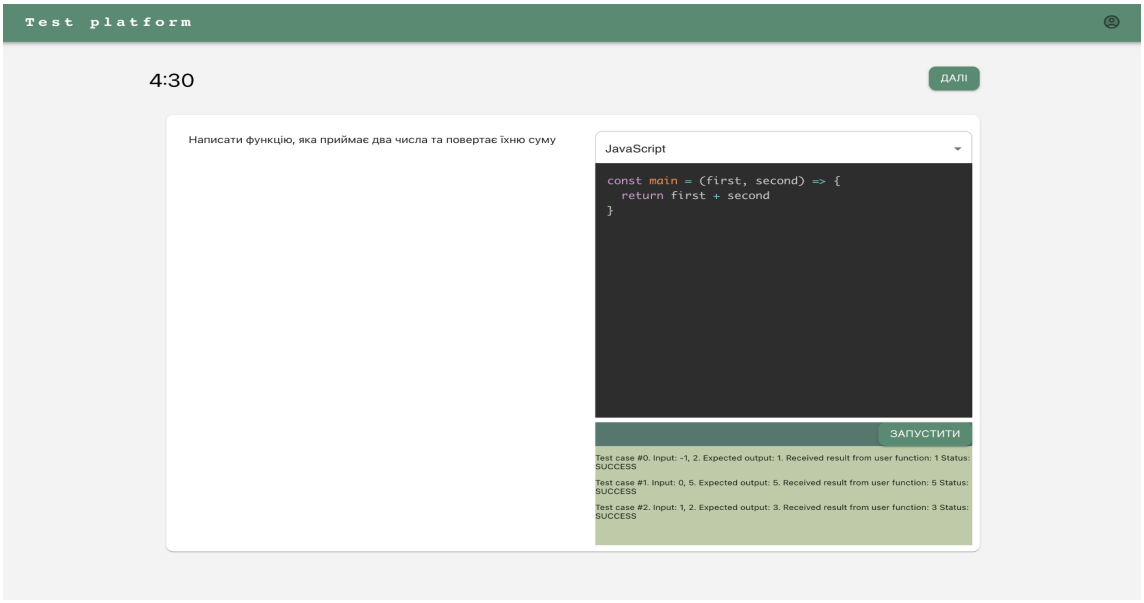


Рисунок 5.6 – Сторінка з виконанням практичного завдання

Студент має можливість вибрати мову програмування серед доступних (дозволених викладачем, який створив завдання).

Власний програмний код студент повинен писати в текстовому редакторі програмного коду.

Студенту доступна можливість запуску (компіляції і виконання чи інтерпретації, залежно від мови програмування) написання програмного коду на одному (текстовому) наборі даних безліч раз, для перевірки коректності роботи його програми та наявності помилок. При цьому він має обмежену кількість спроб запуску програми на всьому наборі вхідних даних, як додав викладач.

У разі виникнення помилок при компіляції, виконанні чи інтерпретації програмного коду такі помилки з детальним описом будуть відображені студенту для подальшого їх виправлення.

При перевірці правильності виконання завдання береться до уваги тільки результат виконання програми при заданих вхідних даних. При цьому кількість коду, стиль його написання та розмір самої програми значення для оцінювання роботи програми немає.

5.1.4 Перегляд результатів

Після проходження тесту студенту відображається загальний результат успішності у вигляді відсотків. При цьому є можливість переглянути результат кожного завдання окремо (рис. 5.7).

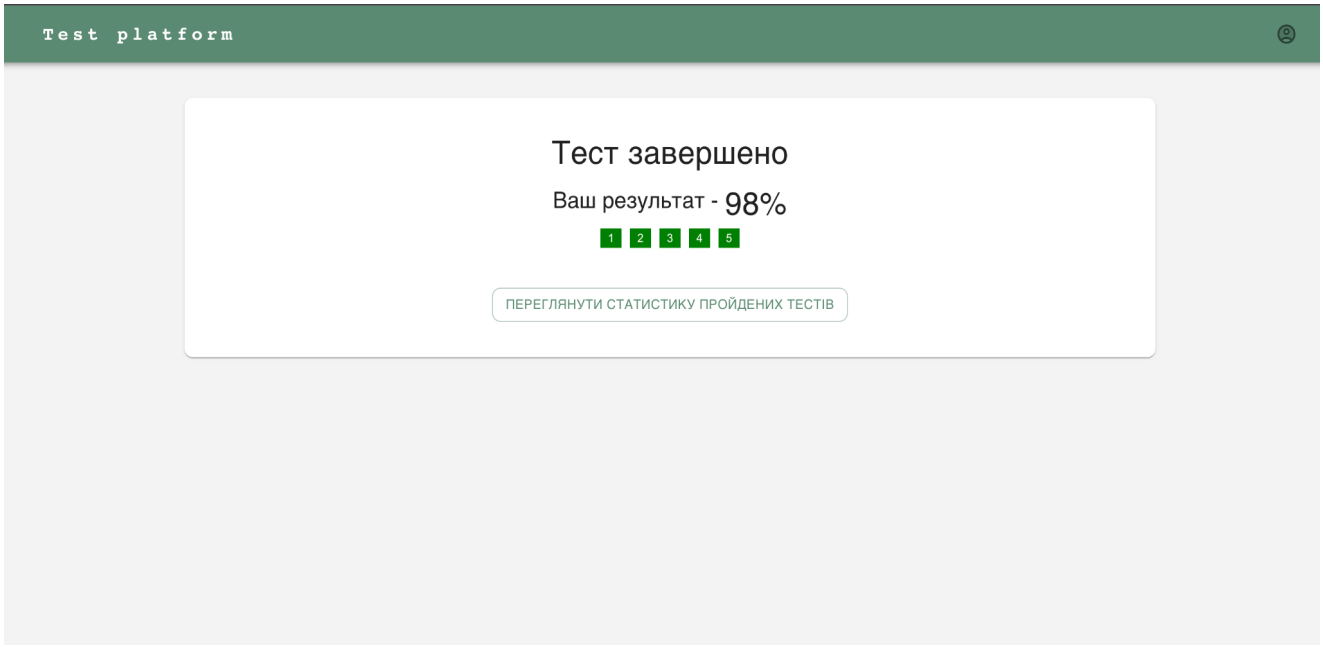


Рисунок 5.7 – Сторінка з результатом проходження тесту

Для того, щоб переглянути результат виконання кожного завдання необхідно натиснути на порядковий номер цього завдання. Варто зауважити, що в залежності від успішності виконання того чи іншого завдання, квадратик з його порядковим номером буде мати інший колір. Таким чином, якщо завдання виконане неправильно воно буде мати червоне забарвлення, якщо частково правильно – жовтим, а якщо повністю правильно – зелене.

Результат проходження завдання з набором варіантів відповідей відображено на рисунку 5.8.

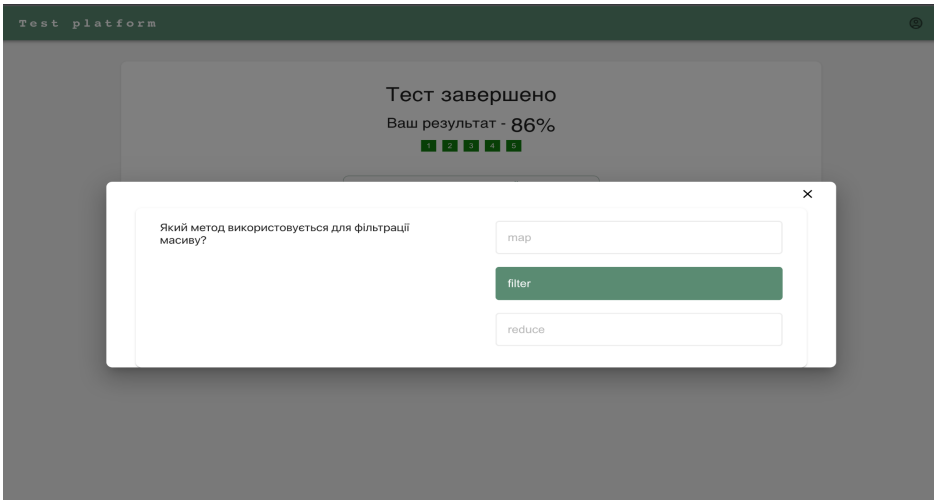


Рисунок 5.8 – Результат проходження завдання з набором варіантів відповідей

Студенту відображається підсвіченою відповідь, що він обрав раніше, при цьому не вказується, яка відповідь є правильною.

Результат проходження завдання з відкритою відповіддю відображено на рисунку 5.9.

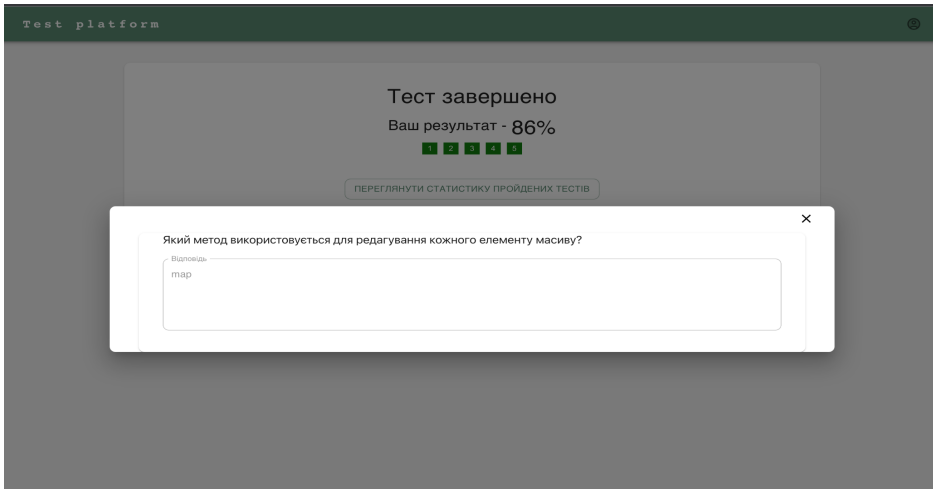


Рисунок 5.9 – Результат проходження завдання з відкритою відповіддю

Результат проходження практичного завдання відображено на рисунку 5.10.

Рисунок 5.10 – Результат проходження практичного завдання

Студенту відображається умова та заповнена ним частина коду, але не відображаються, як саме ця програма повинна була виглядати.

5.1.5 Історія проходження курсів та тестів

Студент має можливість переглянути історію проходження тестів та курсів. Інформація доступна тільки на пристрої, з якого курси були пройдені. При відкритті платформи з іншого пристрою історія проходжень не відобразиться.

В історії тесту відображається:

- дата проходження;
- результат проходження кожного курсу;
- результат проходження кожного завдання;
- час виконання кожного завдання.

Приклад сторінки зображений на рисунку 5.11.

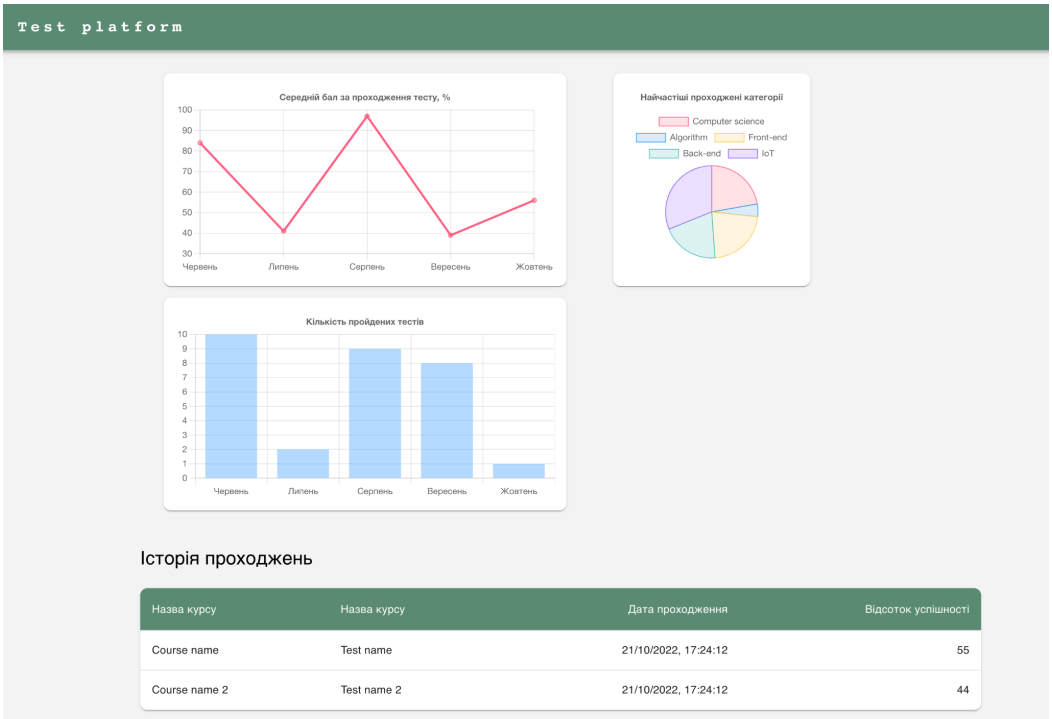


Рисунок 5.11 – Сторінка з відображенням статистики для студента

Студент має доступ до історії попередніх тестів, а також до статистики, створеної на основі цих проходжень.

5.2 Інструкція викладача

5.2.1 Перегляд загальної статистики

Для перегляду загальної статистики викладачу необхідно перейти на сторінку «Статистика», натиснувши відповідну кнопку меню в верхньому меню платформи.

Викладачу відобразиться сторінка з даними проходження створених ним курсів (рис. 5.12).

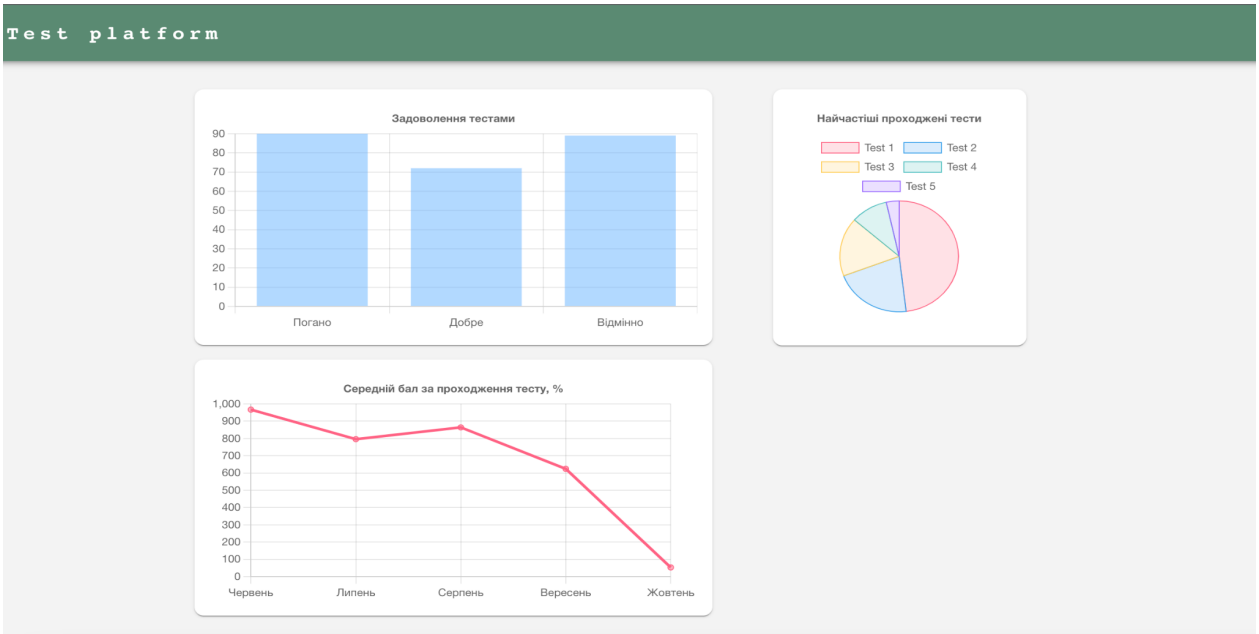


Рисунок 5.12 – Сторінка з відображенням статистики для викладача

При цьому він зможе побачити наступні показники у форматі графіків:

- оцінка задоволення проходження тестів;
- середній бал (у відсотковому співвідношенні) серед усіх тестів викладача протягом останніх шести місяців.

5.2.2 Перегляд статистики окремого тесту

Для перегляду детальної статистики по окремому тесту викладачу необхідно перейти на сторінку «Статистика», натиснувши відповідну кнопку меню в верхньому меню платформи, після цього вибрати тест по якому він бажає переглянути статистику.

Викладачу відобразяться дані щодо проходження створеного ним обраного тесту (рис. 5.13).

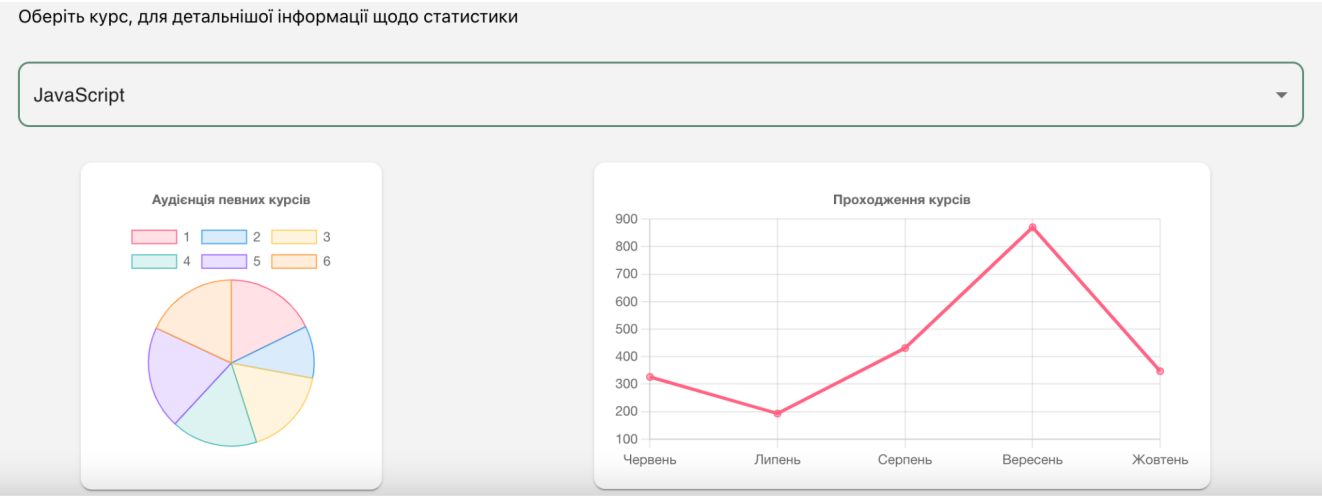


Рисунок 5.13 – Сторінка з відображенням статистики для викладача по одному тесту

При цьому він зможе побачити статистику виконання тесту та статистику проходження тесту студентами різних курсів.

Висновки до розділу 5

У даному розділі було наведено детальні приклади роботи з застосунком. Розглянуто основні сценарії використання зі сторони викладача та студента. Таким чином, для студента було розглянуто весь процес проходження тестових завдань. Окремо розглянуто запитання, що має перелік відповідей, запитання що вимагає відкритої відповіді, та запитання що вимагає написання програмного коду.

У деталях було продемонстровано можливості використання статистики, як з боку викладача, так і зі сторони студента. Таким чином, для студента доступна історія проходження тестів та загальна статистика впродовж шести місяців. Для викладача доступна загальна статистика по його тестам, а також конкретно по кожному тесту.

6 СТАРТАП ПРОЄКТ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ВИКОНАННЯ ТЕСТОВИХ ЗАВДАНЬ

6.1 Опис ідеї проекту

Метою даного стартап проєкту є створення системи для моніторингу виконання тестових завдань, за допомогою якої студенти можуть за допомогою самоконтролю перевіряти свої знання з певних дисциплін.

Цільовою аудиторією такої системи є студенти закладів вищої освіти, а також викладачі та представники компаній, які зацікавлені в розвитку студентів та запрошенні їх на роботу. Найбільш націлений даний проєкт на студентів технічних спеціальностей в сфері інженерії програмного забезпечення, так як окрім можливості проходження звичайних тестових завдань, користувачам пропонується також редактор коду, де вони можуть написати вирішення певної задачі та запустити його. Окрім цього, система є автоматизованою, так як зовсім не потребує менеджменту зі сторони користувача після створенням ним певних даних. Власне, сама ідея проєкту зображена в таблиці 6.1.

Таблиця 6.1 – Опис ідеї проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
1	2	3
Інформаційна система моніторингу виконання тестових завдань	Перевірка рівня знань студентів, отриманих в університеті.	Студенти можуть самостійно перевіряти свій рівень знань, визначати, які теми їм варто повторити, а з чим вони вже знайомі, готуватися до працевлаштування.

Як бачимо, основним напрямом системи є перевірка знань. Після переходу на дистанційне навчання університети почали співпрацювати з певними компаніями, використовувати інші уже існуючі системи, які мають схожий функціонал, так як це базова потреба при процесі навчання. Тим не менш, деякі з систем зараз вважаються дещо застарілими, інші ж просто не мають достатньо функціоналу, тому доводиться користуватися ще сторонніми системами, з неможливістю поєднати це все в одному місці. Наприклад, на технічних спеціальностях для виконання коду зазвичай немає спеціального поля, а використовується звичайний текстовий ввід, або ж надсилається окремо файл з готовим варіантом рішення. Якщо ж студент хоче підготуватися до іспиту чи інтерв'ю в компанію самостійно, то йому доводиться шукати весь необхідний матеріал в інтернеті та використовувати для цього велику кількість платформ.

Серед інших конкурентів дана система відрізняється тим, що зберігає весь цей функціонал в одному місці, відповідно, студенту не потрібно витрачати зайвий час на пошуки, натомість він може зосередитися на самій підготовці.

Порівняльний аналіз даної системи відносно інших конкурентів наведено в таблиці 6.2.

Таблиця 6.2 – Порівняння системи з іншими аналогами

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/ концепції конкурентів			Слабка сторона	Нейтральна сторона	Сильна сторона
		Моя система	Leetcode	Test Gorilla			
1	2	3	4	5	6	7	8
1	Можливість проходити тести	Так	Ні	Так			+

Продовження таблиці 6.2

1	2	3	4	5	6	7	8
2	Наявність аналітики	Так	Ні	Так		+	
4	Можливість виконувати практичні завдання з залученням коду	Так	Так	Так		+	
5	Автоматична перевірка виконання завдань	Так	Ні	Ні			+
6	Ціна	Так	Так	Ні			+

З таблиці 6.2 видно, що дана система моніторингу тестування покриває весь функціонал систем-аналогів та додає власні можливості, такі як автоматизована перевірка відповідей, статистика проходжень, виконання завдань різного типу.

6.2 Технологічний аудит ідеї проєкту

Проведемо аудит системи, перевіривши доступність технологій, що використовуються при імплементації системи. Результат аналізу наведено в таблиці 6.3.

Таблиця 6.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	2	3	4	5
1	Сервер	NodeJS, Nest, GraphQL	Наявна	Безкоштовна
2	Клієнт	TypeScript, React, Apollo	Наявна	Безкоштовна
3	Алгоритм порівняння тексту	Python, PyTorch	Наявна	Безкоштовна
4	База даних	PostgreSQL	Наявна	Безкоштовна

Проаналізувавши технології, що використовуються в проєкті та їх доступність, можемо побачити, що реалізація та підтримка самого проєкту є безкоштовною з точки зору технологій.

6.3 Аналіз ринкових можливостей запуску стартап проєкту

Перед запуском проєкту в ринок варто проаналізувати, наскільки проєкт може бути там успішним, що варто зробити, щоб збільшити успіх та кількість користувачів тощо [23]. Для цього потрібно детально розібратися, ким саме буде використовуватися проєкт, з якою метою, які загрози можуть виникнути після запуску проєкту та вже на основі цього робити певні висновки.

Аналіз цільової аудиторії наведено у таблиці 6.4.

Таблиця 6.4 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	2	3	4
Перевірка знань студентів	Студенти та викладачі в університеті	Перевірка засвоєння матеріалу студентом, підготовка до важливих іспитів.	Простота, можливість конфігурації
Профорієнтація студентів	Представники зацікавлених в студентах компаній	Надання завдань для перевірки знань перед пошуком робочого місця, новий матеріал для вивчення	Простота, можливість конфігурації, автоматизація процесів

Як бачимо, що цілі всієї цільової аудиторії є приблизно схожими. Їхня загальна мета – перевірити рівень знань студентів. Студенти – можуть в одному місці знайти весь матеріал для перевірки, що їм потрібно. Викладачі можуть створити тести, та вказати, що на їхню думку повинен знати студент після завершення курсу. Компанії вказують рівень очікуваних ними знань від студента та підказують, на що на їхню думку варто ще звернути увагу.

Звісно ж, при запуску даної системи можуть виникнути і певні загрози. Список можливих варіантів показано в таблиці 6.5.

Таблиця 6.5 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	2	3	4
1	Необхідність розбиратися з функціоналом	Користувачам потрібно розбиратися з системою, її функціоналом	Спрощення інтерфейсу користувача, стандартизація різного функціоналу
2	Наявність інших схожих систем	На ринку можуть існувати інші системи, які виконують практично такий ж фунціонал, і перебувають на ринку довше, відповідно мають більшу аудиторію	Добавлення в систему нового унікального функціоналу, який може стати необхідним для користувачів та змусить їх перейти на дану систему.

Згідно з аналізом можна зрозуміти, що система має переваги серед інших систем на ринку, але також після її запуску можуть виникнути певні загрози. Деякі з них пов’язані з конкурентним ринком і складністю знайти нових користувачів. Інші ж пов’язані з проблемою адаптації нових користувачів та необхідністю розбиратися з новим функціоналом.

Також при виході на ринок варто врахувати, що окрім факторів ризику, існують також фактори можливостей, які при правильному використанні, можуть навпаки принести проєкту більше переваг та допомогти знайти більшу аудиторію [24]. Такі фактори зображено в таблиці 6.6.

Таблиця 6.6 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	2	3	4
1	Автоматизація системи	Система є повністю автоматизованою, відповідно користувачам не потрібно витрачати час на керування певними даними, такими як проходження тестів.	Покращення алгоритмів перевірки відповідей студентів.
2	Виконання коду	Можливість виконувати практичні завдання шляхом реалізації це в коді.	Розширення системи шляхом додавання нові варіанти мов програмування для використання під час написання коду.

Можемо побачити, що дана система містить певні можливості, які зараз є актуальними на ринку, а, відповідно, має перевагу над іншими системами, в яких це відсутнє. Тим не менш, також система ще має куди розширюватися та розвиватися, наприклад, в покращенні автоматичних перевірок відповідей, добавленню нових варіантів мов програмування. Такі покращення зроблять систему більш точною, а також зможуть охопити більшу аудиторію. Відповідно більша кількість користувачів будуть бажати нею користуватися, що і є фінальною ціллю проєкту.

Далі потрібно визначитися з типом конкуренції, для цього проведемо дослідження конкуренції в галузі за Портером (табл. 6.7).

Таблиця 6.7 – Дослідження конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
1	2	3	4	5	6
Статус	Відсутні	Присутні	Відсутні	Споживачі диктують умови.	Присутні, зокрема LeetCode, TestGorilla.
Висновки	Прямих конкурентів немає.	Вихід на ринок можливий попри присутність потенційних конкурентів.	Постачальників немає, необхідні інженери розробники ПЗ.	Споживачі визначають умови користування системою.	Товари замінники не представляються собою обмежень.

На основі даних таблиці можна зробити висновок, що вихід на ринок є можливим, незважаючи на присутність на ньому конкурентів. Основні фактори конкурентоспроможності можна побачити в таблиці 6.8.

Таблиця 6.8 – Фактори конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	2	3
1	Мультиформенність	Оскільки інформаційна система є веб-застосунком, вона не вимагає скачування та може бути використана на будь якому пристрої який підтримує браузер.
2	Зручність у використанні	Інтерфейс користувачів системи є інтуїтивно зрозумілим, що пришвидшує звикання користувача до продукту.
3	Підтримка інтеграції з навчальним процесом	Продукт розробляється для впровадження в закладах вищої освіти, а тому враховує їхні особливості та потреби

На основі вище згаданих факторів можна виконати порівняльний аналіз інформаційної системи з потенційними конкурентами (табл. 6.9).

Таблиця 6.9 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бал	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	2	3	4	5	6	7	8	9	10
1	Мультиформенність	20			К2	К1			П
2	Зручність у використанні	19				К2	К1	П	

Продовження таблиці 6.9

1	2	3	4	5	6	7	8	9	10
3	Підтримка інтеграції з навчальним процесом	18				К1	К2	П	

Іншим популярним способом аналізу слабких та сильних сторін продукту є SWOT матриця. Для інформаційної системи моніторингу виконання тестових завдань SWOT аналіз наведено в таблиці 6.10.

Таблиця 6.10 – SWOT-аналіз стартап проєкту

Сильні сторони: Мультиформенність, підтримка інтеграції з навчальним процесом, зручність у використанні	Слабкі сторони: кількість наповнення матеріалом
1	2
Можливості: виконання завдань з написанням програмного коду, самоконтроль знань, анонімність.	Загрози: необхідність у поглибленій інтеграції з системами університету.

На основі SWOT-аналізу можна побудувати план вирішення потенційних загроз система, для того щоб покращити вихід продукту на ринок, а також зробити його більш конкурентоспроможним.

Крім цього, такий аналіз дозволять краще зрозуміти продукт, як його розвивати в подальшому та які сильні сторони потрібно підкреслювати під час рекламної кампанії.

Альтернативи ринкового впровадження стартап проєкту можна переглянути в таблиці 6.11.

Таблиця 6.11 – Альтернативи ринкового впровадження стартап проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність Отримання ресурсів	Строки реалізації
1	2	3	4
1	Розробка мінімальної версії продукту, з реалізацією запланованого функціоналу	Висока	До 3 місяців
2	Розробка версії продукту який включатиме розширений функціонал	Низька	До 9 місяців
3	Створення мобільної версії продукту	Середня	До 5 місяців
4	Впровадження сучасних інтелектуальних технологічних рішень	Низька	До 7 місяців

Після детального аналізу альтернатив було обрано перший варіант, завдяки швидкості реалізації та надійносні результатів.

6.4 Розробка ринкової стратегії проєкту

Визначення стратегії охоплення ринку є першочерговою задачею при побудові ринкової стратегії продукту (табл. 6.12).

Таблиця 6.12 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів.	Готовність споживачі в сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	2	3	4	5
Заклади вищої освіти та фірми які зацікавлені в працевлаштуванні студентів.	Споживачі готові сприйняти продукт.	Цільовому сегменту властивий середній попит.	Присутня конкуренція з товарами-замінниками	Продукт просто впроваджувати на ринок

Враховуючи те, що в інформаційної системи моніторингу виконання тестових завдань присутній тільки один цільовий сегмент – обрана стратегія концентрованого маркетингу. Базову стратегію розвитку наведено в таблиці 6.13.

Таблиця 6.13 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	2	3	4
Розробка мінімальної версії продукту, з реалізацією запланованого функціоналу	Стратегія розширення первинного попиту	Конкурентами є лише товари-замінники. Платформа має переваги в зручності та універсальності	Стратегія диференціації

Після проведеного аналізу потрібно виконати визначення базової стратегії конкурентної поведінки. Це допоможе зрозуміти, чи є сартап-проект першопроходцем на ринку, чи дана ніша в ринку вже є заповненою, та наскільки дана галузь є досліджено.

Результат аналізу можна переглянути на таблиці 6.14. Одразу ж можна зробити висновок, що інформаційна система моніторингу виконання тестових завдання не є першопрохоцем на ринку, оскільки в даній галузі вже давно існують схожі продукти комерційного та некомерційного плану.

Таблиця 6.14 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	2	3	4
Ні	Шукати нових споживачів	Ні	Стратегія лідера

Серед наявних стратегій було обрано стратегію лідера.

Небідно розробити стратегію позиціонування, базуючись на вимогах користувачів, стратегій розвитку та конкурентної поведінки.

Результати можна переглянути в таблиці 6.15.

Таблиця 6.15 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	2	3	4	5
1	Зручність, надійність, користь.	Стратегія диференціації	Зручність, надійність, простота використання	Сучасність, анонімність, самоконтроль

6.5 Розроблення маркетингової програми

У таблиці 6.16 зображена маркетингова концепція продукту, що є першим кроком при розробці маркетингової програми.

Таблиця 6.16. – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	2	3	4
1	Інформативність статистики	Інформаційна система надає викладачам та студентам точну статистику по великій кількості показників.	Більш детально статистика ніж у конкурентів.

Продовження таблиці 6.16.

1	2	3	4
2	Можливість виконання завдань з програмування	Краще засвоювання матеріалу який стосується інженерії програмного забезпечення.	Більшість конкурентів не пропонує даної можливості, оскільки це не входить до концепції їхнього продукту.
3	Зручність використання	Інтуїтивно зрозумілий користувацький інтерфейс	Більшість конкурентів мають складний інтерфейс.

Якщо порівнювати інформаційну систему моніторингу виконання тестових завдань з конкурентами, можна зауважити що вона пропонує більш зрозумілий користувацький інтерфейс. Вона забезпечує інформативну статистику для викладачів та студентам, що дозволяє їм ефективніше досягти своїх цілей.

Далі необхідно розробити трирівневу маркетингову модель товару (табл. 6.17). Тут варто розглянути такі рівні товару, як задум і реальне виконання.

Таблиця 6.17 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1	2
I. Товар за задумом	Інформаційна система моніторингу виконання тестових завдань. Можемо виокремити наступні вигоди від використання продукту: — інформативність статистики; — доступність; — простота використання;

Продовження таблиці 6.17.

1	2	
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм
	1. Доступність	М
	2. Інформативність статистики	М
	3. Анонімність для студентів	М
	4. Простота використання	М
	Якість: відповідає нормам розробки програмного забезпечення.	
	Пакування: Продукт представлений у форматі веб-додатку, який включає в себе: — загальна назва продукту, власна назва; — система адміністратора; — система викладача; — система студента; — інструкція користувача.	
	Марка: “TestPlatform”	

Цінові межи визначено в таблиці 6.18.

Таблиця 6.18 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар
1	2	3	4
74$	Безкоштовно/31$	10000UAH	0-123

Кінцевим кроком маркетингової програми є виконання моделювання концепції маркетингових комунікацій, що базується на завчасно обраній основі для позиціонування. Результат можна побачити в таблиці 6.19.

Таблиця 6.19 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій цільових клієнтів	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	2	3	4	5
Використання інтернет ресурсів для інформації про систему	Інтернет, месенджери	Зручність використання, універсальність, допомога з навчанням.	Донести зручність та користь продукту до клієнта	Анонімна система самоконтролю знань для студентів.

На основі даних з таблиці можна зробити висновок, що користувачі в основному дізнаються про існування продукту з інтернетом, який є один з найважливіших каналів комунікації. Основною ідеєю програмного продукту є сучасний підхід то тестування як виду самоконтролю знань студентів, що дозволяє покращувати рівень матеріалів викладачів а також підвищувати рівень знань студентів.

Протягом час рекламної кампанії увагу користувачів потрібно акцентувати на різноманітний функціонал, можливість адаптації продукту, зручність використання та відсутність авторизації для студентів.

ВИСНОВКИ

Проведено детальний аналіз предметної області в рамках обраної теми магістерської роботи. Проаналізовано наявні проблеми в сфері дистанційної освіти закладів вищої освіти та професійної орієнтації студентів.

Після аналізу предметної області було виконано моделювання програмного забезпечення для інформаційної системи моніторингу тестування студентів, що складається з клієнтської та серверної частини. Було розроблено схему бази даних.

На основі результаті проєктування було реалізовано комплекс програмних продуктів, що є веб-застосунком з функціоналом для моніторингу тестування студентів. Даний веб-застосунок призначений для використання студентами та викладачами університету.

Основною метою розробленого програмного забезпечення є допомога студентам в оцінюванні власних знань, без можливості отримати негативний відгук чи погану оцінку від викладача за неправильно виконані завдання. Також система призначення для викладачів, які зацікавлені в дослідження ефективності своїх навчальних матеріалів, адже розділ статистики покаже графічні та числові результати, на основі яких можна буде зробити певні висновки. Після кількох тижнів чи місяців виконання студентами тестів, викладач зможе тверезо оцінити якість викладених матеріалів.

Інформаційна система моніторингу виконання тестових завдань була реалізована за допомогою JavaScript, NodeJS, NestJS, PostgreSQL, GitHub, GraphQL, Heroku.

Після реалізації необхідного програмного забезпечення було проведено тестування на предмет недоліків системи, а також складено коротку інструкцію для користувачів системи залежно від ролей.

Після тестування програмного забезпечення було проведено виправлення недоліків системи а також модернізацію інфраструктури для більш ефективної роботи в хмарі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Центр інформаційних технологій в освіті ІПО КПІ ім. Ігоря Сікорського. Дистанційне навчання. URL: https://do.ipo.kpi.ua/ (дата звернення: 21.11.2022).
2. Susan S. N. Moodle 4 E-Learning Course Development. Packt Publishing. 2022. 436 c.
3. LeetCode. URL: https://leetcode.com/ (дата звернення: 21.11.2022).
4. Jay Wengrow, A Common-Sense Guide to Data Structures and Algorithms. Pragmatic Bookshelf. 2020. 508 c.
5. TestGorilla. URL: https://www.testgorilla.com/ (дата звернення: 21.11.2022).
6. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. O'Really. 2020. 704 p.
7. Scott E. SPA Design and Architecture: Understanding single-page web applications. Manning Publications. 2015. 563 p.
8. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. O'Really. 2020. 310 p.
9. Gourley D. HTTP: The Definitive Guide. O'Reilly Media. 2002. 656p.
10. Snell J. Programming Web Services with SOAP: Building Distributed Applications. O'Reilly Media. 2001. 337p.
11. T. Ray E. Learning XML. O'Really. 2003. 432 p.
12. Masse M. REST API Design Rulebook. O'Really. 2011. 112 p.
13. Buna S. GraphQL in Action. Manning. 2021. 384 p.
14. Casciaro M., Mammino L. Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques. 3rd Edition. 2020. 664 p.
15. NestJS – A progressive Node.js . URL: https://nestjs.com/ (дата звернення: 21.11.2022).

16. Craig Walls. Spring Boot in Action. Manning. 2016. 264p.

17. Obe R., Hsu L. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database. O'Really. 2017. 312 p.

18. IntelliJ IDEA – the Leading Java and Kotlin. URL: https://www.jetbrains.com/idea/ (дата звернення: 21.11.2022).

19. Chatson S., Straub B. Pro Git. Apres. 2014. 487 p.

20. C. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. 2017. O'Really. 432 p.

21. Richardson C. Microservices Patterns: With Examples in Java. *Journal of Software Engineering and Applications*. 2021. Vol.14 No.8. P. 34-43.

22. Nadeau T., Lightstone S. Database Modeling and Design: Logical. Series Editor: Jim Gray, Microsoft Research. 2011. 289 p.

23. Masters B., Thiel P. Zero to One: Notes on Startups, or How to Build the Future. Penguin Random House Company, New York. 2014. 160 p.

24. Гавриш О.А. Розроблення стартап-проекту. Київ: НТУУ «КПІ ім. Ігоря Сікорського». 2016. 30 с.

ДОДАТОК А

Апробація

Інформаційна система моніторингу виконання тестових завдань

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ ТВ-**1**2мп

Аркушів 3

2022

Міністерство освіти і науки України
Одеський національний технологічний університет
Інститут комп'ютерних систем і технологій
"Індустрія 4.0" ім.П.Н.Платонова

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2022»

МАТЕРІАЛИ
XV МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

20 - 21 ЖОВТНЯ 2022 р.

м.ОДЕСА

Україна)	
Артеменко В. Б., Артеменко О. В., Давида Н. М. Інструментарій вироблення веб-аналітики для онлайн-навчання**.** (Львівський торговельно-економічний університет, Україна)	102
Вода А.В., Юрченко А.О. Цифрові інструменти для супроводу професійної діяльності вчителя інформатики. (Сумський державний педагогічний університет імені А.С.Макаренка, Україна)	105
Воінова С.О. Роль іноваційних освітніх технологій у підготовці здобувачів вищої освіти до іноваційоної діяльності. (Одеський національний технологічний університет, Україна)	108
Гнатишин М.С, Жмуркевич В.І., Свинчук О.В. Інформаційна система тестування студентів. («Київський політехнічний інститут імені Ігоря Сікорського», Україна)	110
Заріцька С.І., Литвиненко Н.І. Завдання розвитку освітніх технологій в контексті євроінтеграції. (Міжнародний науково-навчальний центр інформаційних технологій і систем НАН України та МОН України, Україна)	111
Кочкодан О.Д. Використання ресурсу CISCO WEBEX в дистанційному навчанні. (Національний університет біоресурсів і природокористування України, Україна)	114
Мельников О. Ю. Додаток для роботи із системами класифікацій галузей знань та спеціальностей. (Донбаська державна машинобудівна академія, Україна)	115
Селіванова А. В., Каліта М. В. Моніторинг працевлаштування випускників закладів вищої освіти. (Одеський національний технологічний університет**, Україна**)	118
Середюк Г. В., Паламарчук Є. А. Мобільний додаток для роботи з архітектурними планами Будівель і обробкою даних з використанням штучного інтелекту. (Вінницький національний технічний університет, Україна)	120
Слуковська А. Ю., Бабюк Н. П. Розробка методу і програмного засобу оптимізації робочих завдань ІТ-команди (Вінницький національний технічний університет, Україна)	123
Шершень О.В., Шамоня В.Г. Інтернет-ресурси як інструмент реалізації неформальної освіти. (Сумський державний педагогічний університет імені А.С.Макаренка, Україна)	124
Щиров О. С., Паламарчук Є. А., Коваленко О. О. Особливості формування адаптивного контенту в електронних навчальних системах. **(**Вінницький національний технічний університет, Україна)	127
Юрченко К.В., Юрченко А.О. Розробка вебресурсу як навчального проєкту STEM-освіти. (Комунальна установа Сумська спеціалізована школа I-III ступенів №25, м. Суми Сумської області, Україна) , Сумський державний педагогічний університет імені А.С.Макаренка, Україна)	129
Розділ 5. Проектування інформаційних систем та програмних комплексів	133
Avramchuk V. V. System to getting related videos based on text topic with ml.net and youtube data api. (Taras Shevchenko National University of Kyiv, Ukraine)	133
Dosanalieva A.T. Based on android operating system " beat.development of mobile application "maker". (Turan University, Almaty, Republic of Kazakhstan)	136
Kopp A.M., Orlovskyi D.L., El Arbaouti I. The software tool for error probability evaluation in business process models. (National Technical University «Kharkiv Polytechnic Institute», Ukraine)	141
Mamenko P. P., Zinchenko S. M., Nosov P. S., Kyrychenko K. V., Mateichuk V. M. Automation of the exit to the ellipse of the given risk. (Kherson State Maritime Academy, Ukraine)	144
Seksenali A.K., Ismailova R.T. Using the distributed database systems as a cybersecurity improvement for fintech companies. (Turan University, Almaty, Republic	147

ІНФОРМАЦІЙНА СИСТЕМА ТЕСТУВАННЯ СТУДЕНТІВ

Гнатишин М.С, Жмуркевич В.І., Свинчук О.В. (gnatyshynmisha@gmail.com, vladagmyrka1@gmail.com, 7011990@ukr.net)

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (Україна)

Дана робота спрямована на створення інформаційної веб-системи для тестування та профорієнтації студентів закладів вищої освіти з метою покращення якості дистанційної освіти в університеті.

Сучасні реалії вже вкотре кидають виклик класичній освіті та звичній для нас організації навчального процесу. Якщо раніше можливість віддалено здати лабораторну роботу чи пройти тест було приємним доповненням, то з початком пандемії в 2020 році це стало критичною необхідністю. Для сфери освіти, як і для багатьох інших сфер людської діяльності, можливість забезпечувати виконання всіх своїх процесів в дистанційному режимі стало актуальним питанням сьогодення.

З перших днів переходу на дистанційне навчання заклади вищої освіти активно впроваджували різноманітні онлайн-засоби в навчальний процес. Хтось зупинився на старих, але перевірених платформах, таких як Moodle [1], хтось почав відкривати для себе сучасні засоби, такі як Google Classroom [2]. Використання сучасних інформаційних систем допомагає у вирішенні актуальних проблем дистанційної освіти, проте цього виявляється недостатньо для досягнення максимальної ефективності онлайн-навчання порівняно з класичним. Це пов'язано з тим, що в кожного закладу вищої освіти є свої унікальні підходи до навчального процесу, які вимагають відповідного функціоналу.

Метою даної роботи є створення інформаційної веб-платформи для викладачів та студентів, яка буде виконувати функції порталу для тестування та профорієнтації студентів.

Дана система надаватиме можливість викладачам зручно створювати практичні завдання до своїх нових чи вже існуючих курсів в наступному форматі:

- формат вікторини з переліком відповідей, серед яких є одна або декілька правильних;
- формат відкритого запитання, коли студент відповідає своїми словами на задане запитання, а система порівнює його відповідь з тими, які попередньо задав викладач;
- формат запитання, яке вимагає написання тексту коду програми для вирішення запропонованої задачі.

Особливістю даної інформаційної системи є те, що завдання можна створювати в рамках освітнього процесу та у вигляді партнерських курсів в співробітництві з компаніями, які зацікавлені в наймі студентів на роботу ще під час їхнього навчання. Таким чином, студенти зможуть постійно перевіряти свої знання на основі актуальних, перевірених реальним життям задач та краще розуміти свій рівень готовності до працевлаштування в тій чи іншій компанії. Самі ж компанії та університет могли отримувати краще підготовлених студентів та в цілому аналізувати вміння та навички студентів завдяки аналітичному розподілу на платформі.

Таким чином, створення спеціалізованої інформаційної веб-системи для тестування та допомоги в професійній орієнтації студентів покращить якість сучасної освіти та допоможе студентам почувати себе більш впевнено на ринку праці.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Відкрита платформа дистанційної освіти Moodle \\ Вікіпедія: [Веб-сайт]. URL: https://uk.wikipedia.org/wiki/Moodle Дата звернення: Жовтень 10. 2022.

[2] Google Classroom: інструкція, як самостійно створювати онлайн-курси \\ Освіторія медіа: [Веб-сайт]. URL: https://osvitoria.media/news/google-classroom-instruktsiya-yak-samostijno-stvoryuvaty-onlajn-kursy/ Дата звернення: Жовтень 10. 2022.

ДОДАТОК Б

Лістинг розробленої програми

Інформаційна система моніторингу виконання тестових завдань

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ ТВ-12мп

Аркушів 9

2022

test-attempt-question.service.ts

```
@Injectable()
export class TestAttemptQuestionService {
  constructor(
    @InjectRepository(TestAttemptQuestionAnswerEntity)
    private answersRepository: Repository<TestAttemptQuestionAnswerEntity>,
    @InjectRepository(TestAttemptQuestionEntity)
    private attemptQuestionsRepository: Repository<TestAttemptQuestionEntity>,
    @InjectRepository(TestAttemptEntity)
    private testAttemptRepository: Repository<TestAttemptEntity>,
    private testService: TestSearchService,
    private questionsService: QuestionsManagementService,
    private openTextQuestionValidationService: OpenTextQuestionValidationService,
    private codeExecutionService: CodeExecutionService,
  ) {}

  async answerQuizQuestion(attemptId: string, questionId: string, answers: number[]): Promise<boolean> {
    const originalQuestion = await this.questionsService.findById(questionId);
    const allAnswers = originalQuestion.quizQuestionDetails.answers;
    const correctAnswers = allAnswers.filter((answer) => answer.isCorrect);
    const attemptQuestions = await this.attemptQuestionsRepository.findOne({
      where: { attemptId: attemptId, questionId: questionId },
    });

    const matchedAnswers = correctAnswers.filter((correctAnswer) => answers.includes(correctAnswer.id));
    const correctnessScore = matchedAnswers.length / correctAnswers.length;

    const answerRecord = new TestAttemptQuestionAnswerEntity();
    answerRecord.correctnessScore = correctnessScore;
    answerRecord.isCorrect = correctnessScore === 1;
    answerRecord.attemptQuestion = attemptQuestions;
    answerRecord.rawAnswer = JSON.stringify(answers);

    await this.answersRepository.save(answerRecord);

    await this.validateTestCompleteness(attemptId);

    return true;
  }

  async answerOpenTextQuestion(attemptId: string, questionId: string, answer: string): Promise<boolean> {
    const originalQuestion = await this.questionsService.findById(questionId);
    const allAnswers = originalQuestion.openTextQuestionDetails.answers;
    const attemptQuestions = await this.attemptQuestionsRepository.findOne({
      where: { attemptId: attemptId, questionId: questionId },
    });

    const correctnessScore = await this.openTextQuestionValidationService.getSimilarityScore(
      answer,
      allAnswers.map((it) => it.text),
    );

    const answerRecord = new TestAttemptQuestionAnswerEntity();
```

```
    answerRecord.correctnessScore = correctnessScore;
    answerRecord.isCorrect = correctnessScore > originalQuestion.openTextQuestionDetails.minSimilarityScore;
    answerRecord.attemptQuestion = attemptQuestions;
    answerRecord.rawAnswer = JSON.stringify(answer);

    await this.answersRepository.save(answerRecord);

    await this.validateTestCompleteness(attemptId);

    return true;
  }

  async answerCodeQuestion(attemptId: string, questionId: string, code: string, lang: string): Promise<boolean>
{
    const attemptQuestions = await this.attemptQuestionsRepository.findOne({
      where: { attemptId: attemptId, questionId: questionId },
    });

    const testExecutionResult = await this.codeExecutionService.runAllTestCases(questionId, {
      code,
      language: lang,
    });

    const correctAnswers = testExecutionResult.testCases.filter((it) => it.isSuccessful).length;
    const correctnessScore = correctAnswers / testExecutionResult.testCases.length;

    const answerRecord = new TestAttemptQuestionAnswerEntity();
    answerRecord.correctnessScore = correctnessScore;
    answerRecord.isCorrect = correctnessScore === 1;
    answerRecord.attemptQuestion = attemptQuestions;
    answerRecord.rawAnswer = JSON.stringify(code);

    await this.answersRepository.save(answerRecord);

    await this.validateTestCompleteness(attemptId);

    return true;
  }

  async validateTestCompleteness(attemptId: string): Promise<boolean> {
    const questions = await this.attemptQuestionsRepository.find({
      where: { attemptId: attemptId },
      relations: ['answer'],
    });

    if (questions.some((question) => question.answer === null)) {
      return false;
    }

    const originalQuestionIds = questions.map((it) => it.questionId);
    const originalQuestions = await this.questionsService.findManyLight(originalQuestionIds);

    const questionsHashMap = createIdToEntityHashMap(originalQuestions);

    const { totalScore, maxScore } = questions.reduce(
```

```
      (acc, curr) => ({
        totalScore: acc.totalScore + curr.answer.correctnessScore * questionsHashMap[curr.questionId].weight,
        maxScore: acc.maxScore + questionsHashMap[curr.questionId].weight,
      }),
      { totalScore: 0, maxScore: 0 },
    );

    const scoreInPercents = Math.floor((totalScore / maxScore) * 100);

    await this.testAttemptRepository.update(
      { id: attemptId },
      { completedAt: new Date().toISOString(), totalScore: scoreInPercents },
    );
  }
}
```

test-attempt.service.ts

```
@Injectable()
export class TestAttemptService {
  constructor(
    @InjectRepository(TestAttemptEntity)
    private testAttemptRepository: Repository<TestAttemptEntity>,
    private testService: TestSearchService,
    private questionsService: QuestionsManagementService,
  ) {}

  async getCompletedTest(attemptId: string): Promise<TestAttemptEntity> {
    const test = await this.testAttemptRepository.findOne({
      where: { id: attemptId },
      relations: ['questions', 'questions.answer'],
    });

    if (!test) {
      throw Error(`Can\'t find test attempt by id ${attemptId}`);
    }

    if (!test.completedAt) {
      throw Error(`Test attempt with id ${attemptId} was not completed!`);
    }

    return test;
  }

  async create(testId: string): Promise<TestAttemptEntity> {
    const test = await this.testService.getById(testId);

    if (!test) {
      throw Error(`Can\'t find test by id ${testId}`);
    }

    const testAttempt = new TestAttemptEntity();
    testAttempt.test = test;
    testAttempt.questions = await this.generateAttemptQuestions(test);
    return this.testAttemptRepository.save(testAttempt);
```

```
  }

  private async generateAttemptQuestions(test: TestEntity): Promise<TestAttemptQuestionEntity[]> {
    const questionIds = await this.generateQuestionsSet(test);

    return questionIds.map((id, index) => {
      const question = new TestAttemptQuestionEntity();
      question.questionId = id;
      question.order = index + 1;
      return question;
    });
  }

  private async generateQuestionsSet(test: TestEntity): Promise<string[]> {
    const questions = await this.questionsService.getTestQuestionIds(test.id);

    const questionsAmountInAttempt = test.questionsAmountInAttempt || questions.length;

    return shuffleArray(questions.slice(0, questionsAmountInAttempt));
  }
}
```

code-execution.service.ts

```
@Injectable()
export class CodeExecutionService {
  private readonly codeRunners: Record<string, CodeRunner>;

  constructor(private codeQuestionsManagementService: CodeQuestionsManagementService) {
    this.codeRunners = {
      js: new JsCodeRunner(),
      ts: new TsCodeRunner(),
    };
  }

  async runAllTestCases(questionId: string, data: TestCasesExecutionInput):
Promise<TestCasesExecResultModel> {
    return this.runTestCases(questionId, data, false);
  }

  async runSampleTestCases(questionId: string, data: TestCasesExecutionInput):
Promise<TestCasesExecResultModel> {
    return this.runTestCases(questionId, data, true);
  }

  private async runTestCases(
    questionId: string,
    data: TestCasesExecutionInput,
    sampleOnly: boolean,
  ): Promise<TestCasesExecResultModel> {
    const testCases = await this.codeQuestionsManagementService.getTestCases(questionId, sampleOnly);

    const runner = this.codeRunners[data.language];

    if (!runner) {
```

```
      throw Error(`Failed to find runner for programming language ${data.language}`);
    }

    const result = await runner.run({
      code: data.code,
      testCases: testCases,
      timout: -1,
    });

    return new TestCasesExecResultModel(result);
  }
}
```

js.code-runner.ts

```
export class JsCodeRunner implements CodeRunner {
  LANGUAGE_CODE = 'js';

  private static getCodeWithArgs(code: string, ...args: any[]): string {
    return code + `;main(${args.join(', ')})`;
  }

  async run(options: CodeExecutionConfig): Promise<CodeExecutionResult> {
    const testCasesOutput = options.testCases.map((testCase) => {
      const formattedInput = JSON.parse(`[${testCase.input}]`);
      const finalCode = JsCodeRunner.getCodeWithArgs(options.code, formattedInput);

      const startTime = performance.now();

      const testCaseResult = eval(finalCode);

      const endTime = performance.now();

      return {
        testCase,
        output: Number.isNaN(testCaseResult) ? 'NaN' : testCaseResult,
        success: testCaseResult == testCase.expectedOutput,
        time: endTime - startTime,
      };
    });

    return { tests: testCasesOutput };
  }
}
```

student-statistic.service.ts

```
@Injectable()
export class StudentStatisticService {
  constructor(
    @InjectRepository(TestAttemptEntity)
    private testAttemptRepository: Repository<TestAttemptEntity>,
    @InjectDataSource()
    private readonly connection: DataSource,
  ) {}
```

```
  async getStudentCompletedTests(
    attempts: string[],
    page: number = 1,
    pageSize: number = 10000,
  ): Promise<TestAttemptEntity[]> {
    return this.testAttemptRepository.find({
      where: { id: In(attempts), totalScore: Not(IsNull()) },
      relations: ['test', 'test.course'],
      skip: (page - 1) * pageSize,
      take: pageSize,
    });
  }

  async getCategoriesStatistics(attempts: string[]): Promise<StudentCategoryStatisticItem[]> {
    const result = await this.connection.query(
      `
          select count(categories.name) as amount, categories.name, categories.id from categories
          inner join courses_categories_categories ccc on categories.id = ccc."categoriesId"
          inner join tests on tests.course_id = ccc."coursesId"
          inner join test_attempts ta on tests.id = ta.test_id
          where ta.id = any ($1)
          group by categories.name, categories.id
        `,
      [attempts],
    );

    return result as StudentCategoryStatisticItem[];
  }

  async getTestsStatistics(attempts: string[]): Promise<[TimeSeriesStatisticItem[], TimeSeriesStatisticItem[]]> {
    const result = (await this.connection.query(
      `
          with dates as (select * FROM   generate_series(
                        CURRENT_DATE - INTERVAL '5 months',
                        CURRENT_DATE,
                        interval  '1 month')),
              smth as (select count(id), to_char(completed_at, 'YYYY-MM') as date
                        from test_attempts
                        where id = any($1)
                        and completed_at >= CURRENT_DATE - INTERVAL '3 months'
                        group by date),
              scores as (select avg(total_score) as evaluation, to_char(completed_at, 'YYYY-MM') as date
                        from test_attempts
                        where id = any($1)
                        and completed_at >= CURRENT_DATE - INTERVAL '3 months'
                        and total_score > 0
                        group by date)
          select to_char(dates.generate_series,'YYYY-MM') as month, smth.count as amount, scores.evaluation as
evaluation
          from dates
          left join smth on smth.date = to_char(dates.generate_series,'YYYY-MM')
          left join scores on scores.date = to_char(dates.generate_series,'YYYY-MM')
        `,
      [attempts],
```

```
    )) as { month: string; amount: number; evaluation: number }[];

    return result.reduce(
      (acc, curr) => {
        return [
          [...acc[0], { date: curr.month, value: curr.amount }],
          [...acc[1], { date: curr.month, value: Math.floor(curr.evaluation) }],
        ];
      },
      [[], []],
    );
  }
}
```

dataloader.service.ts

```
@Injectable()
export class DataloaderService {
  constructor(private readonly testsService: TestSearchService) {}

  getLoaders(): IDataloaders {
    const testsLoader = this._createTestsLoader();
    return {
      testsLoader,
    };
  }

  private _createTestsLoader() {
    return new DataLoader<string, TestEntity[]>(
      async (keys: readonly string[]) => await this.testsService.getAllTestsByCoursesIds(keys as string[]),
    );
  }
}
```

code-questions-management.service.ts

```
@Injectable()
export class CodeQuestionsManagementService {
  constructor(
    @InjectRepository(ProgrammingLanguageEntity)
    private programmingLanguagesRepository: Repository<ProgrammingLanguageEntity>,
    @InjectRepository(CodeQuestionDetailsEntity)
    private codeQuestionDetailsRepository: Repository<CodeQuestionDetailsEntity>,
    @InjectRepository(CodeQuestionTestCaseEntity)
    private readonly testCaseRepository: Repository<CodeQuestionTestCaseEntity>,
  ) {}

  async createWithoutSaving(data: CreateQuestionInput): Promise<CodeQuestionDetailsEntity> {
    const details = new CodeQuestionDetailsEntity();

    details.maxExecutionTime = data.maxTimeMinutes;
    details.supportedLanguages = await this.getProgrammingLanguagesByIds(data.supportedProgrammingLanguages);
    details.testCases = this.createInitialTestCases(data.codeAnswers.testCases);
```

```
    return details;
  }

  async getProgrammingLanguagesByIds(ids: string[]): Promise<ProgrammingLanguageEntity[]> {
    return this.programmingLanguagesRepository.find({
      where: { id: In(ids) },
    });
  }

  async getTestCases(questionId: string, sampleOnly = false): Promise<CodeQuestionTestCaseEntity[]> {
    const query: FindOptionsWhere<CodeQuestionDetailsEntity> = { questionId: questionId };

    if (sampleOnly) {
      query['testCases'] = { isExample: true };
    }

    const entity = await this.codeQuestionDetailsRepository.findOne({
      where: query,
      relations: ['testCases'],
    });

    return entity.testCases;
  }

  private createInitialTestCases(data: CreateCodeQuestionTestCaseInput[]): CodeQuestionTestCaseEntity[] {
    return data.map(({ input, expectedOutput, isSample }) => {
      const testCase = new CodeQuestionTestCaseEntity();
      testCase.input = input;
      testCase.expectedOutput = expectedOutput;
      testCase.isExample = isSample;

      return testCase;
    });
  }

  async updateWithoutSaving(questionId: string, data: CreateQuestionInput):
Promise<CodeQuestionTestCaseEntity[]> {
    const existingAnswers = await this.testCaseRepository.find({
      where: { question: { questionId: questionId } },
      relations: ['question'],
    });

    const remainingIds = data.codeAnswers.testCases.filter((it) => !!it.id).map((it) => it.id);
    const deletedAnswers = existingAnswers.filter((it) => !remainingIds.includes(it.id));

    if (deletedAnswers.length > 0) {
      await this.testCaseRepository.softRemove(deletedAnswers);
    }

    const existingAnswersIdToEntityHashMap = createIdToEntityHashMap(existingAnswers);

    return data.codeAnswers.testCases.map((testCase) => {
      const entity = !!testCase.id ? existingAnswersIdToEntityHashMap[testCase.id] : new
CodeQuestionTestCaseEntity();
```

```
      if (isPresent(testCase.input)) {
        entity.input = testCase.input;
      }

      if (isPresent(testCase.expectedOutput)) {
        entity.expectedOutput = testCase.expectedOutput;
      }

      if (isPresent(testCase.isSample)) {
        entity.isExample = testCase.isSample;
      }

      return entity;
    });
  }
}
```

test-management.service.ts

```
@Injectable()
export class TestManagementService {
  constructor(
    @InjectRepository(TestEntity)
    private testsRepository: Repository<TestEntity>,
  ) {}

  async create(courseId: string, data: CreateTestInput): Promise<TestEntity> {
    const test = new TestEntity();
    test.name = data.name;
    test.description = data.description;
    test.duration = data.duration;
    test.complexity = data.complexity;
    test.courseId = courseId;

    return this.testsRepository.save(test);
  }

  async update(testId: string, data: CreateTestInput, userId: string): Promise<TestEntity> {
    const test = await this.getTestForUpdate(testId, userId);

    test.name = getUpdatedValueIfPresent(data.name, test.name);
    test.complexity = getUpdatedValueIfPresent(data.complexity, test.complexity);
    test.duration = getUpdatedValueIfPresent(data.duration, test.duration);
    test.description = getUpdatedValueIfPresent(data.description, test.description);

    return this.testsRepository.save(test);
  }

}
```

ДОДАТОК В

Презентація роботи

Інформаційна система моніторингу виконання тестових завдань

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ ТВ-12мп

Аркушів 21

2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

Інформаційна система моніторингу виконання тестових завдань

Студент: Гнатишин Михайло Степанович
Керівник: доцент, кандидат фізико-математичних наук Свинчук Ольга Василівна

Актуальність роботи

- Необхідність впровадження сучасних інформаційних технологій в освітній процес закладів вищої освіти.
- Удосконалення існуючих інструментів для оцінки рівня знань студентів, отриманих в університеті.

Мета, об’єкт та предмет дослідження

Метою дослідження є створення інформаційної системи моніторингу виконання тестових завдань для самоконтролю та оцінки якості знань.

Об’єктом дослідження є моніторинг виконання тестових завдань.

Предметом дослідження є інформаційна система моніторингу виконання тестових завдань.

Постановка задачі

1) Проаналізувати існуючі програмні системи моніторингу виконання тестових завдань в освітньому процесі.

2) Розробити перелік вимог до системи, провести моделювання та проектування програмного забезпечення.

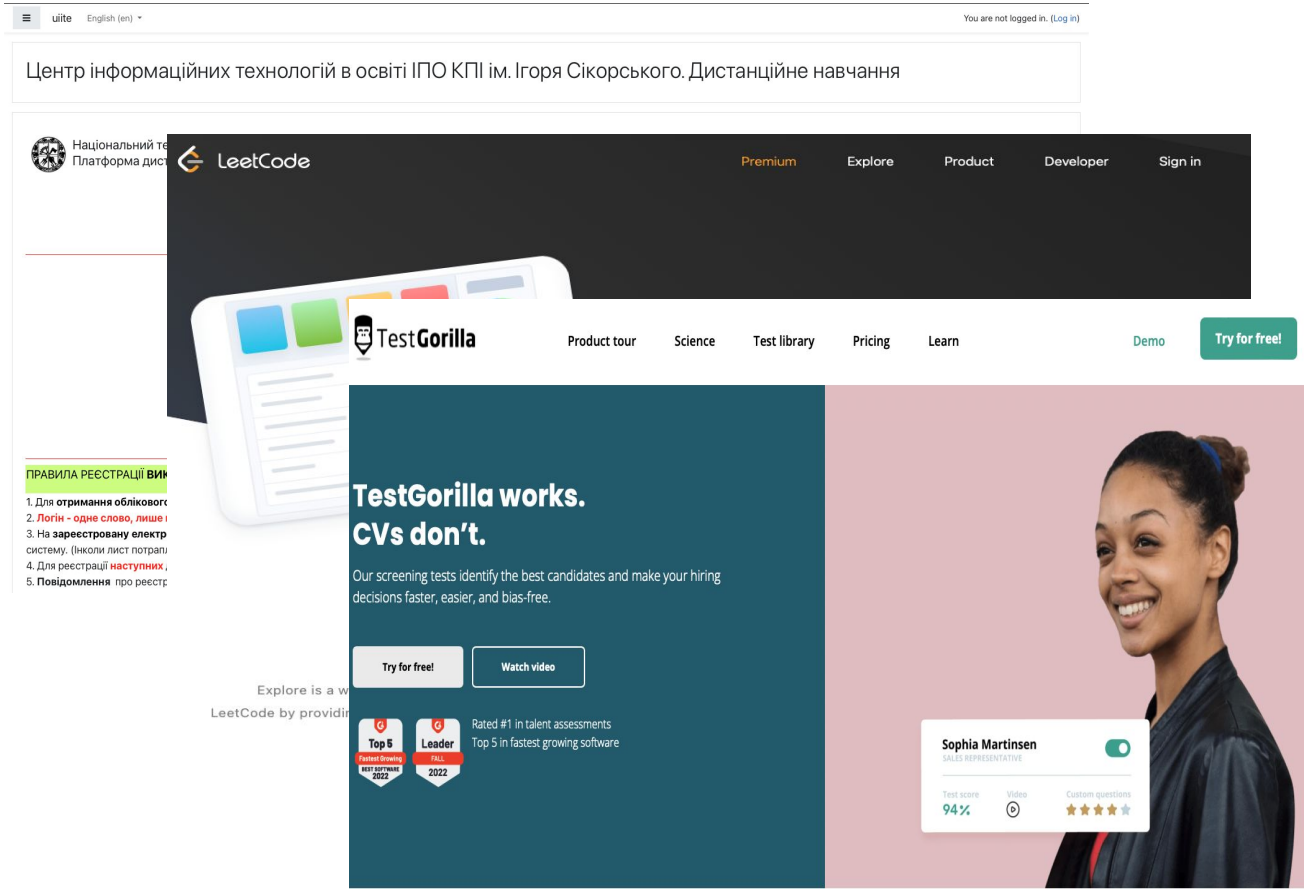
3) Розробити інформаційну систему, що включає наступні підзадачі:

- Створення системи побудови та відображення статистики проходження тестів для студентів.
- Створення системи проходження тестів.
- Створення системи побудови та відображення статистики проходження тестів для викладача.
- Створення підсистеми запуску та перевірки завдань, що вимагають написання програмного коду.

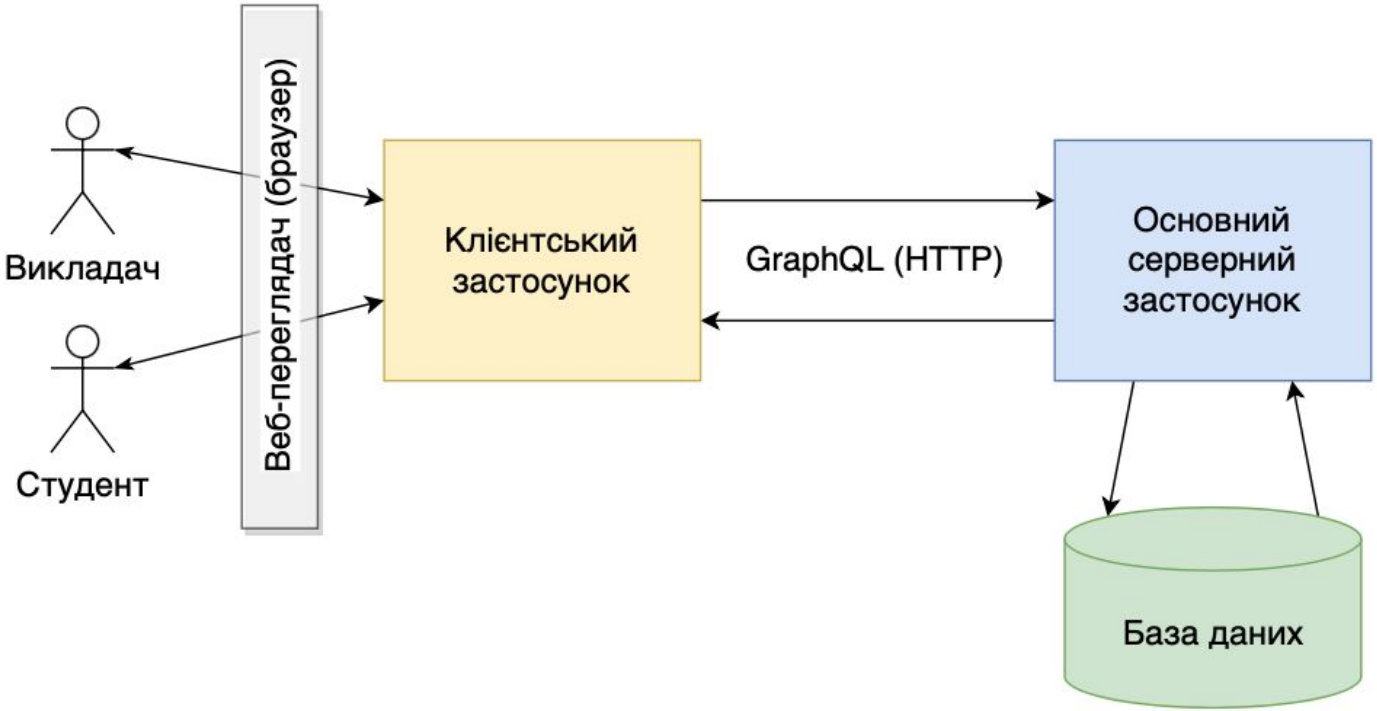
4) Виконати тестування системи.

Існуючі аналоги

- Moodle
- LeetCode
- Test Gorilla



Архітектура системи



Засоби розробки системи

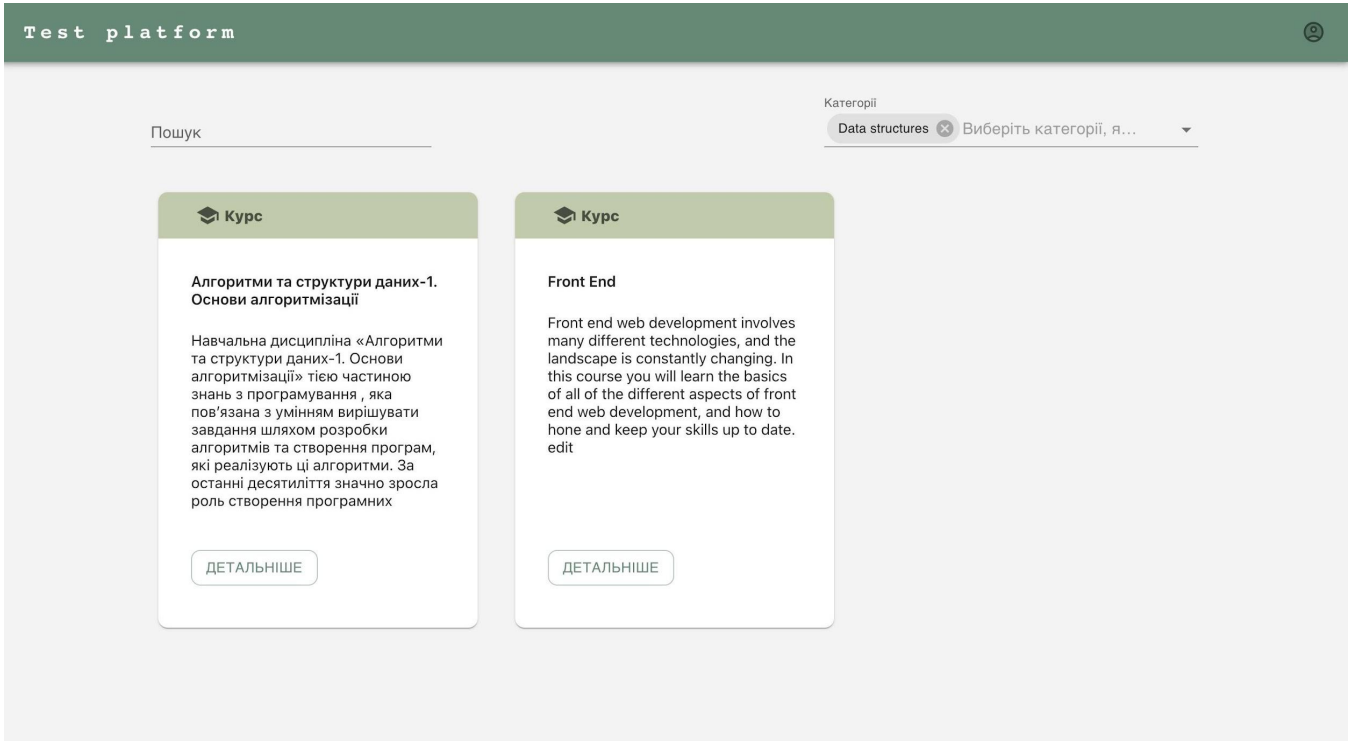
Клієнтська частина

Серверна частина

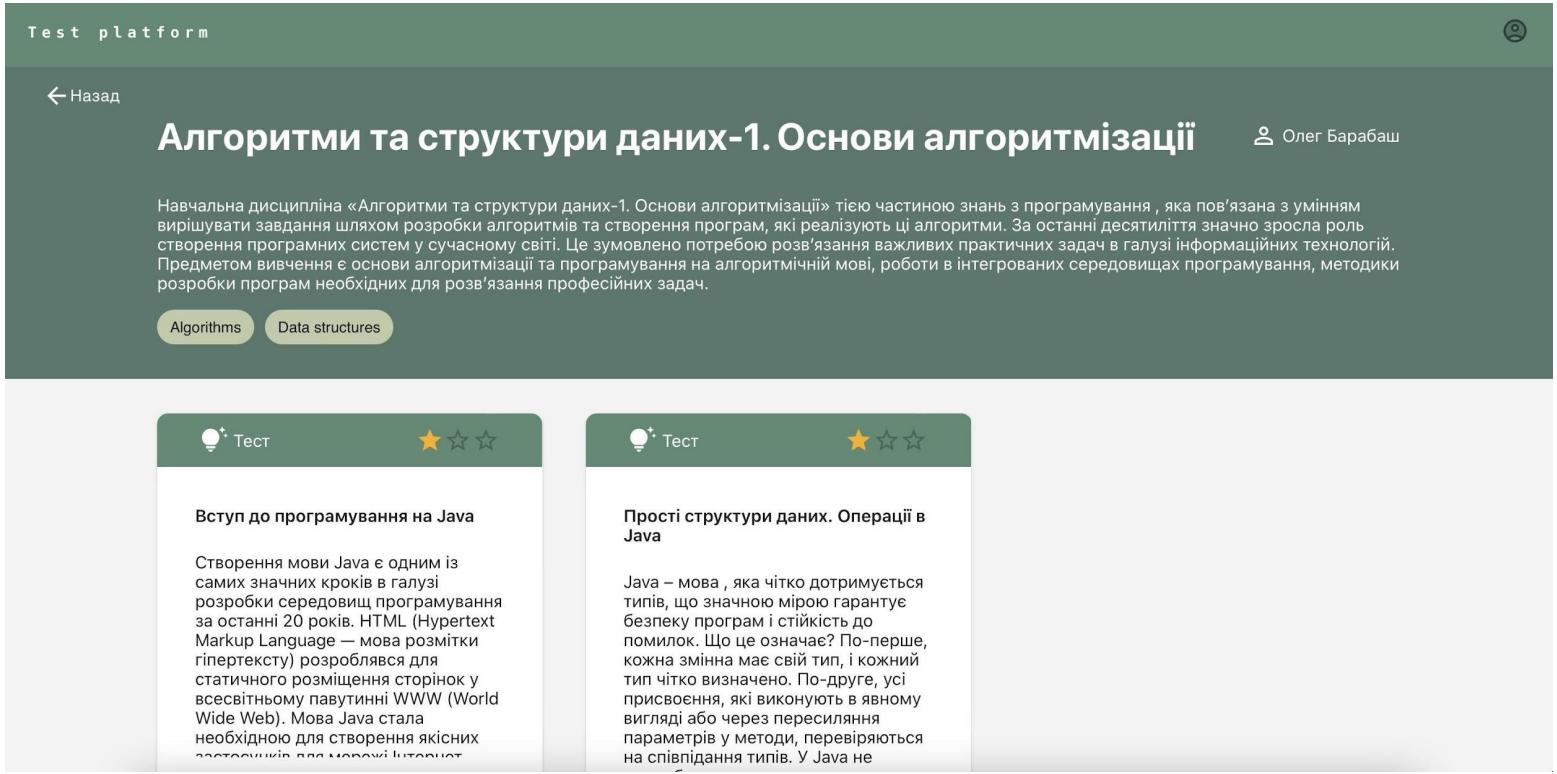
Бази даних

Розгортання

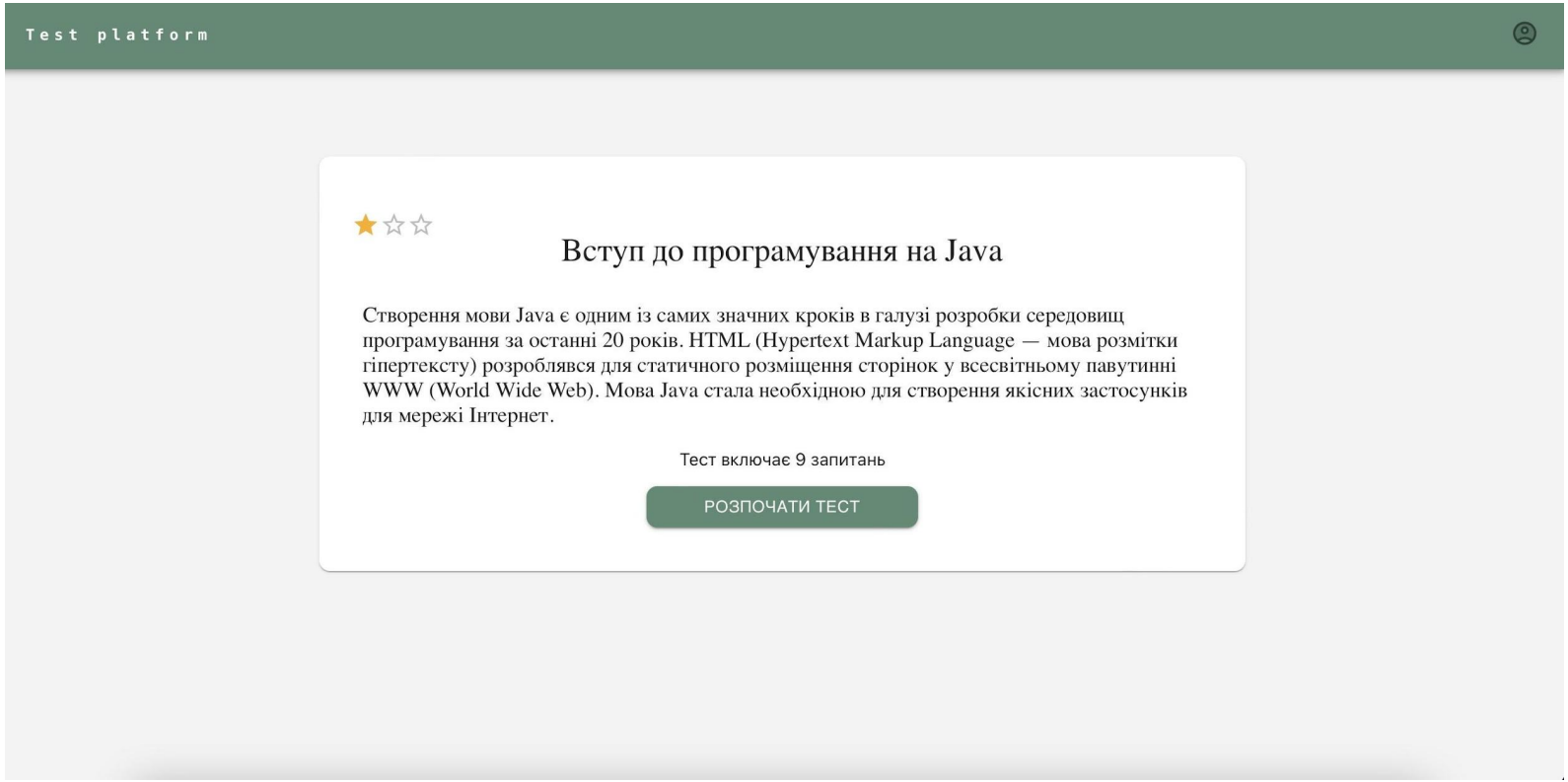
Сторінка списку курсів для студента



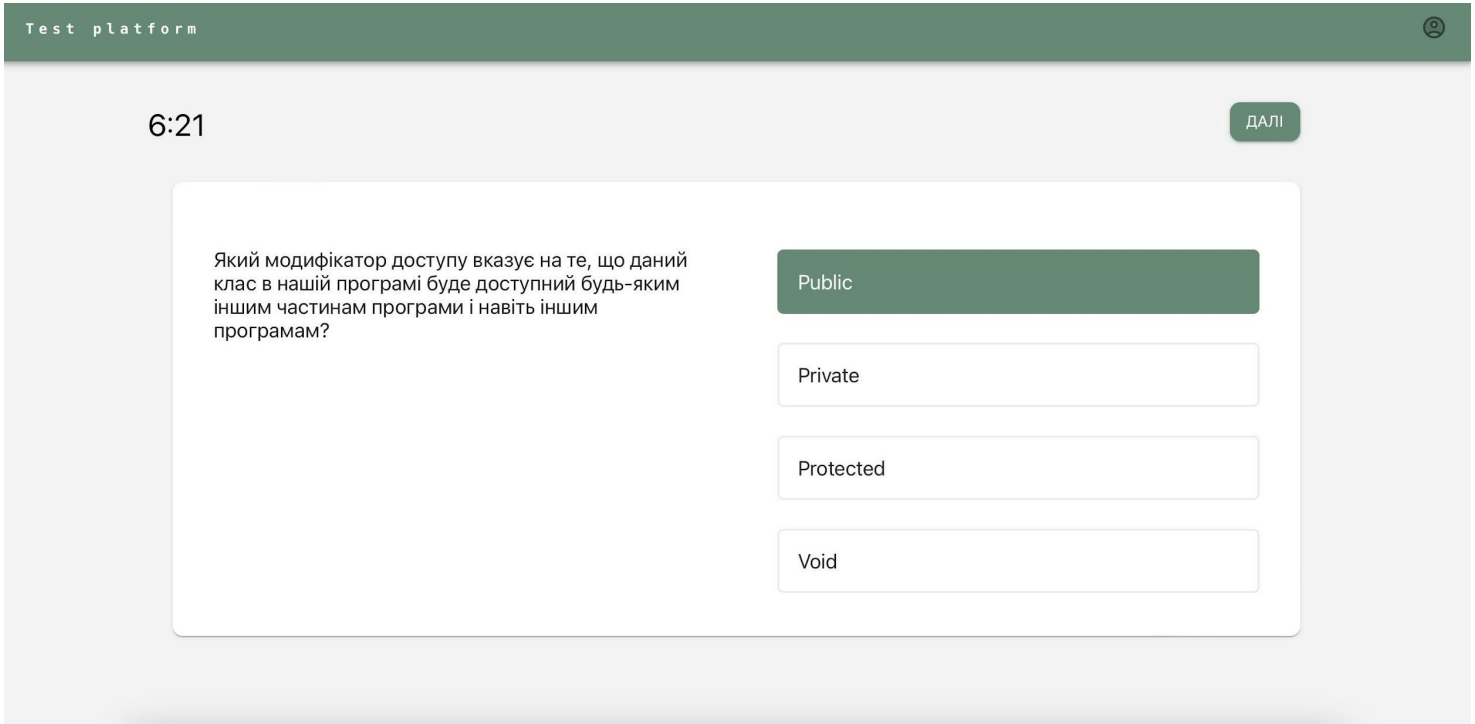
Сторінка курсу зі списком тестів



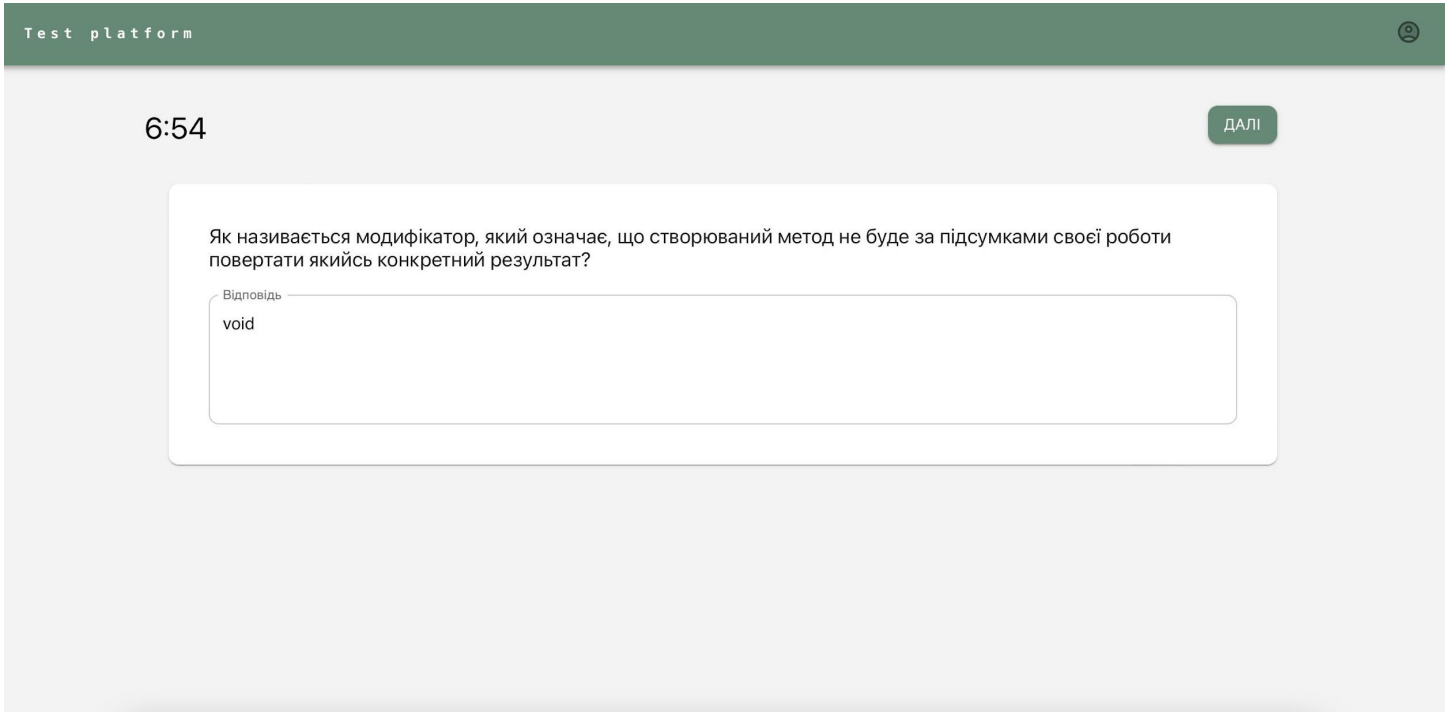
Початок проходження тесту



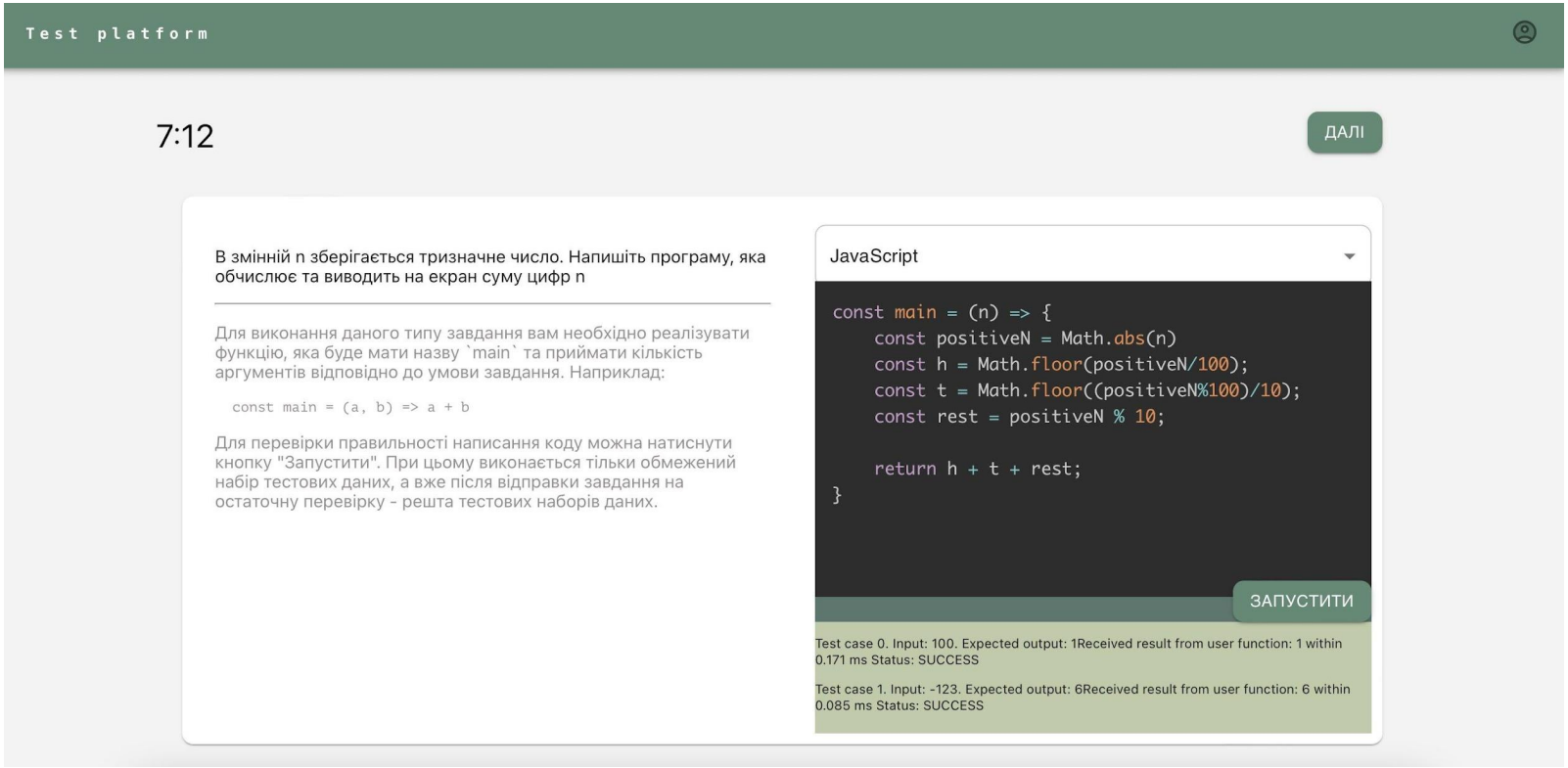
Виконання завдання з набором варіантів відповідей



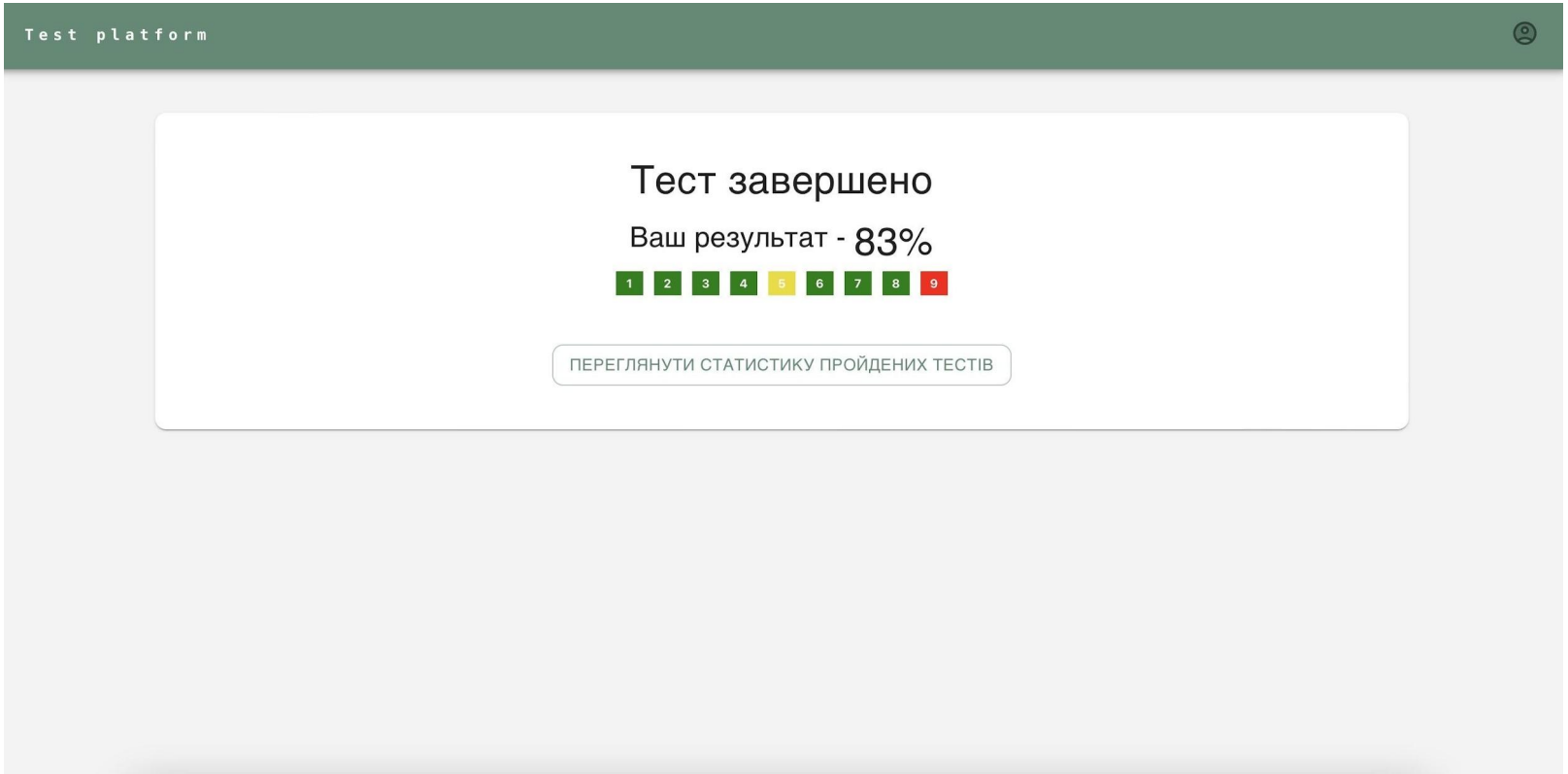
Виконання завдання з відкритою відповіддю



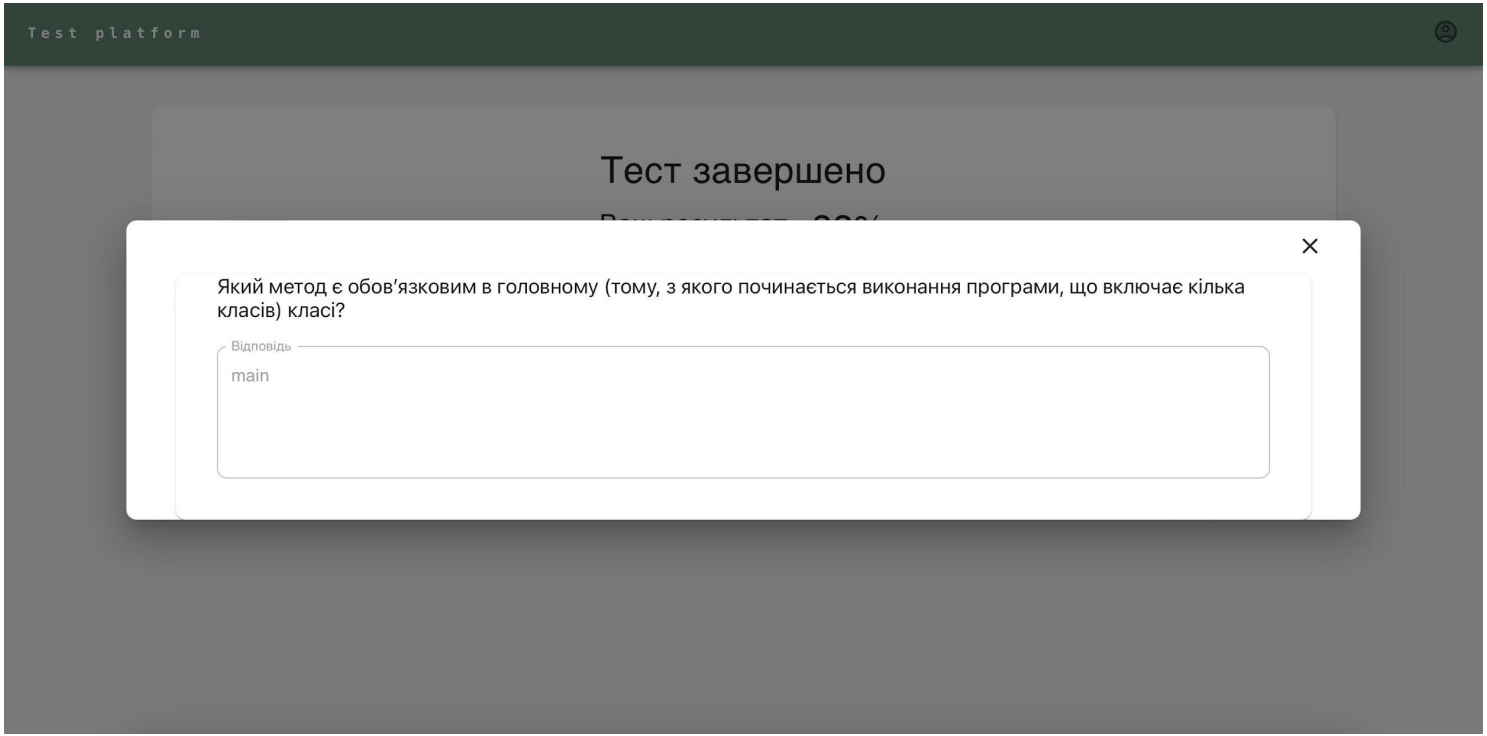
Виконання практичного завдання



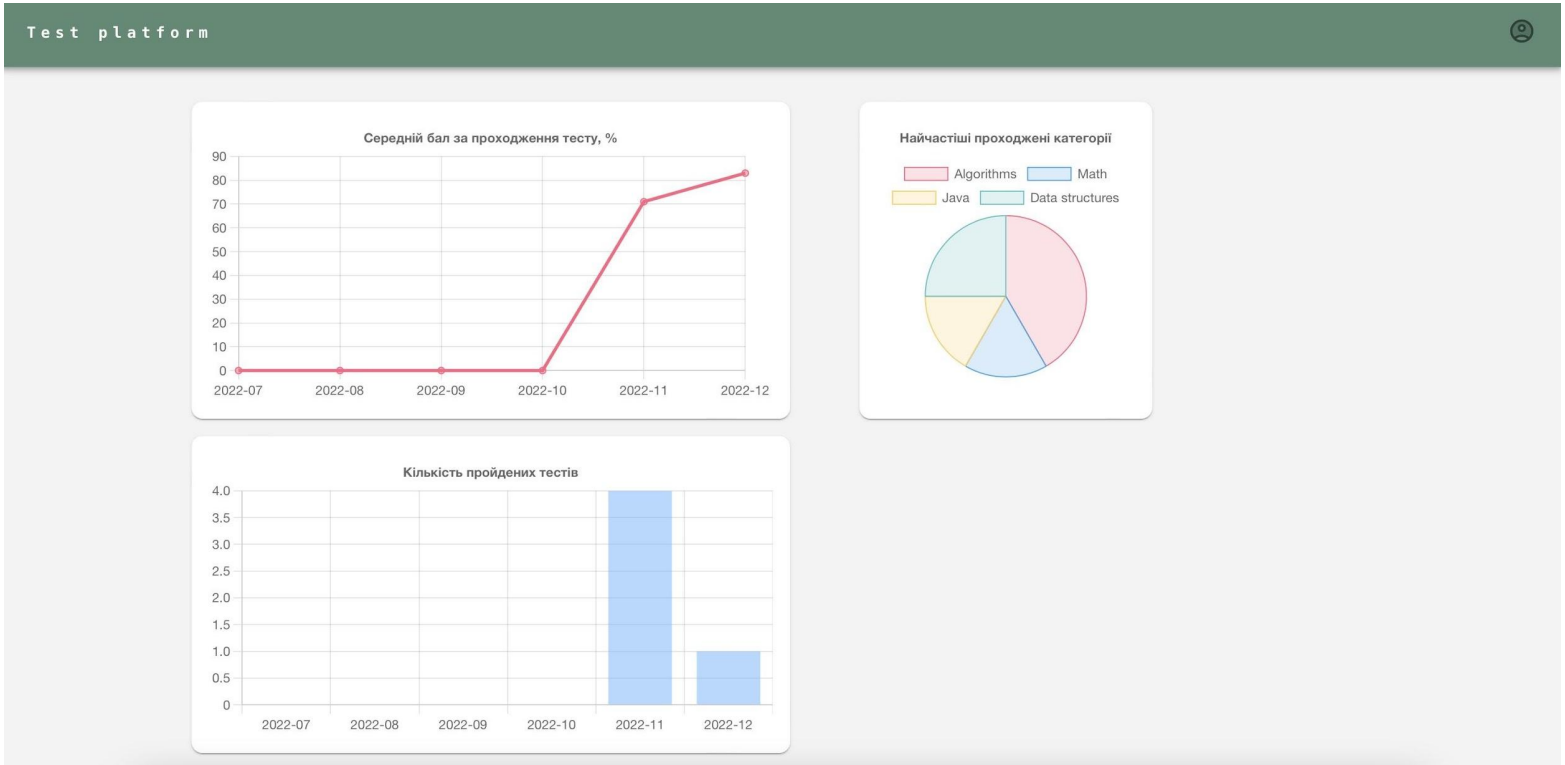
Результат проходження тесту



Приклад пройденого завдання з набором варіантів відповідей



Відображення статистики для студента

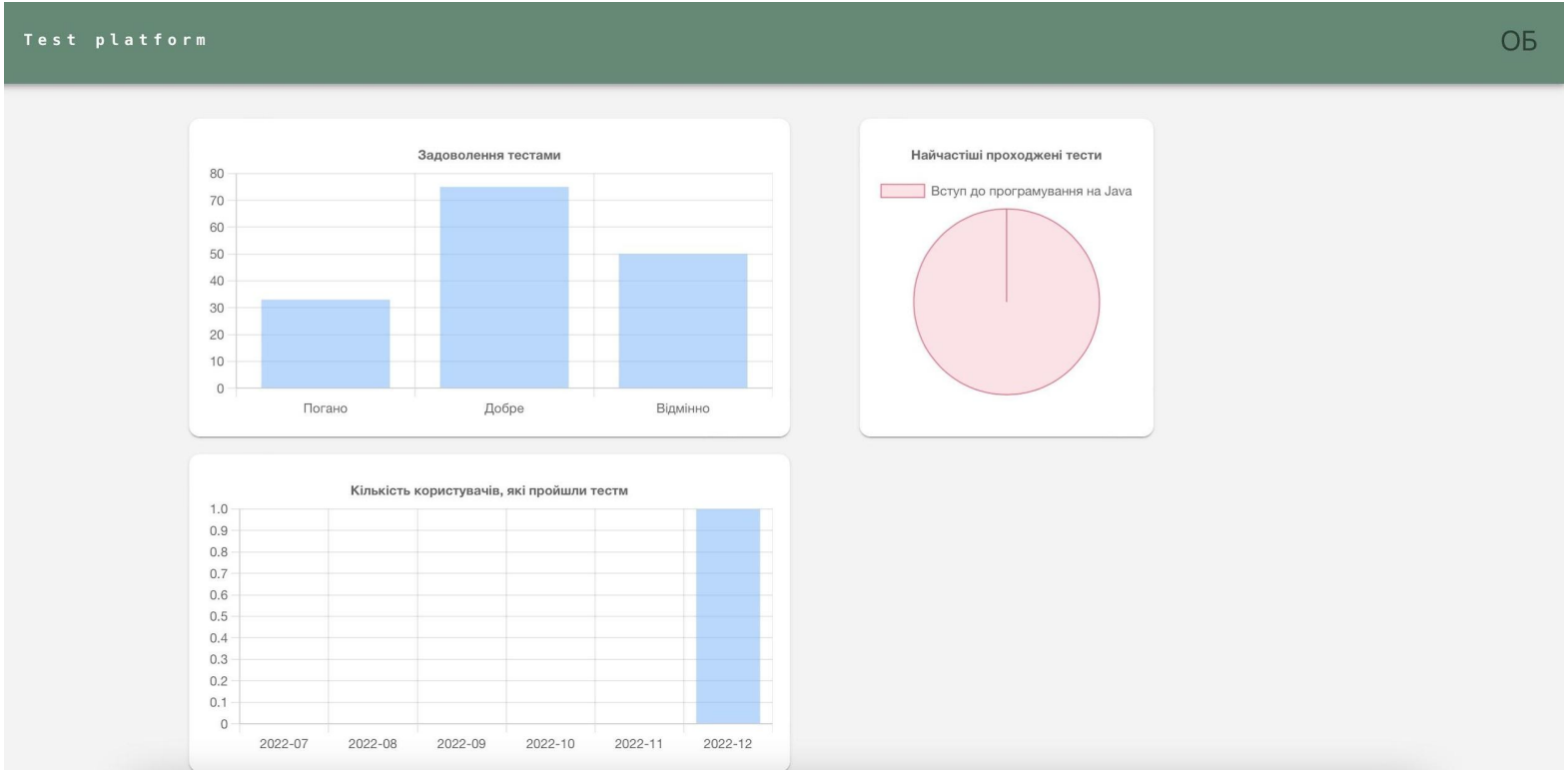


Історія проходжень тестів студентом

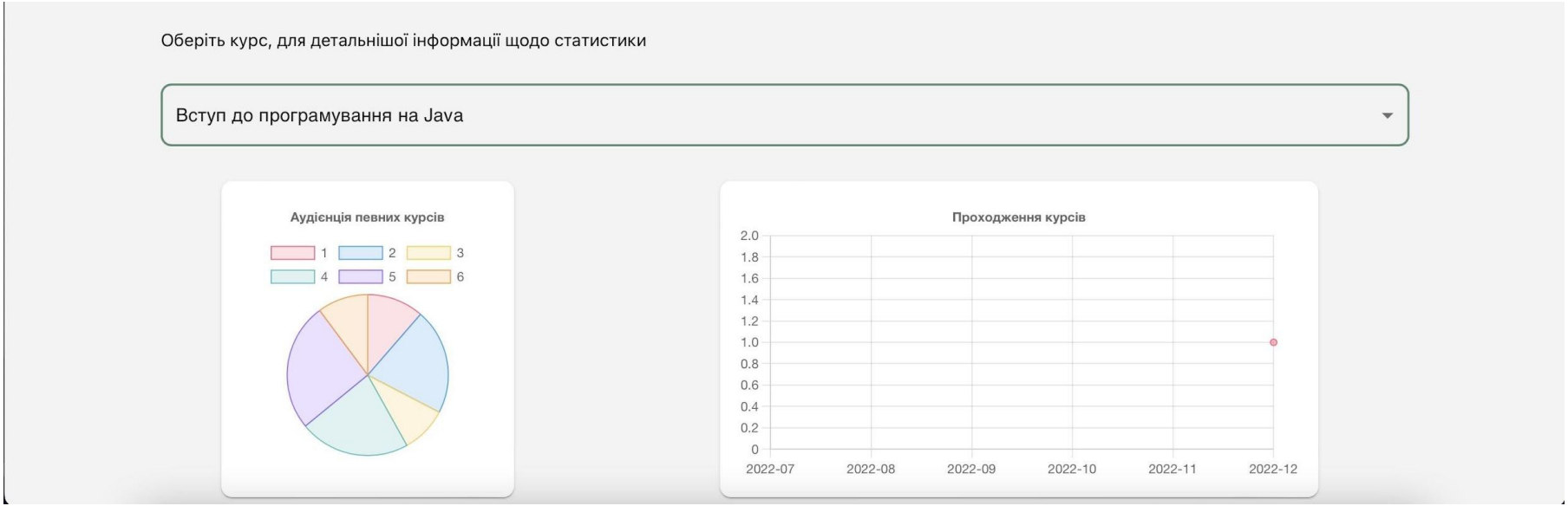
Історія проходжень

Назва курсу	Назва тесту	Дата проходження	Відсоток успішності
Java Script 2.0	Arrays	13/11/2022, 10:53:19	19
Front End	JavaScript	13/11/2022, 11:05:11	87
Java Script 2.0	Arrays	19/11/2022, 15:58:24	99
Front End	JavaScript	19/11/2022, 16:00:22	80
Алгоритми та структури даних-1. Основи алгоритмізації	Вступ до програмування на Java	18/12/2022, 22:01:00	83

Відображення статистики для викладача



Відображення статистики для викладача по одному тесту



Висновки

1) Проаналізовано існуючі програмні системи системи моніторингу виконання тестових завдань в освітньому процесі.

2) Розроблено перелік вимог до системи, проведено моделювання та проектування програмного забезпечення.

3) Розроблено систему, що включає наступні підзадачі:

- Система побудови та відображення статистики проходження тестів для студентів.
- Система проходження тестів.
- Система побудови та відображення статистики проходження тестів для викладача.
- Система запуску та перевірки завдань, що вимагають написання програмного коду.

4) Виконано тестування системи.

Публікації

Гнатишин М.С, Жмуркевич В.І., Свинчук О.В. Інформаційна система тестування студентів. (Тези)

Інформаційні технології і автоматизація – 2022 / XV міжнародна науково-практична конференції. Одеса, 20-21 жовтня 2022 р.