НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«До захисту допущено»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *Платформа для спільної роботи викладача та студента під час іспиту*

Виконав: студент 4 курсу, групи ІП-52

Шателюк Олександр Дмитрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Ковтунець О.В* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н. Ткаченко В.В.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Шателюку Олександру Дмитровичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Платформа для спільної роботи викладача та студента під час іспиту*»
керівник проекту *Ковтунець Олесь Володимирович, старший викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с
2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: опис предметної області, аналіз, аналіз існуючих проектів, розроблення функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, Конструювання програмного забезпечення, аналіз безпеки даних.
3) Впровадження та супровід програмного забезпечення,
4) Керівництво користувача

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна бізнес процесів

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	*Вивчення предметної області*	18.03.2019	
2	*Аналіз існуючих методів розв'язання задачі*	23.03.2019	
3	*Постановка та формалізація задачі*	25.03.2019	
4	*Аналіз вимог до програмного забезпечення*	01.04.2019	
5	*Моделювання програмного забезпечення*	08.04.2019	
6	*Розробка архітектури програмного забезпечення*	15.04.2019	
7	*Розробка програмного забезпечення*	30.04.2019	
8	*Оформлення пояснювальної записки*	17.05.2019	
9	*Виконання графічних документів*	23.05.2019	
10	*Подання ДП на попередній захист*	28.05.2019	
11	*Подання ДП рецензенту*	03.05.2019	
12	*Подання ДП на основний захист*	08.06.2019	

Студент _____ Шателюк О.Д.
(підпис)

Керівник проекту _____ Ковтунець О.В.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-5226.045440.01.81	Пояснювальна записка	72	
3	А4	КПІ.ІП-5226.045440.02.91	Технічне завдання	11	
4	А4	КПІ.ІП-5226.045440.03.34	Керівництво користувача	6	
5	А3	КПІ.ІП-5226.045440.04.51	Програма та методика тестування	6	
6	А3	КПІ.ІП-5226.045440.05.99 СБД	Схема бази даних	1	
7	А3	КПІ.ІП-5226.045440.05.99 СС	Схема структурна варіантів використання	1	
8	А3	КПІ.ІП-5226.045440.05.99 СС	Схема структурна бізнес процесів	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5226.045440.00.90		
Розробн.		Шателюк О.Д.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-52	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка до дипломного проекту складається з трьох розділів, містить 3 таблиці, 10 джерел та додаток з текстами програмного коду – загалом 72 сторінки.

Об`єкт дослідження: системи, що використовуються для код-шерингу між багатьма користувачами та онлайн-редактори для програмування, що дозволяють компілювати та інтепритувати програми, використовуючи потужності сторонніх сервісів.

Мета дипломного проекту: застосування можливостей сучасних веб-технологій для спрощення процесу складання іспитів з програмування в університетах та інших навчальних закладах.

У першому розділі було розроблено архітектуру застосунку та діаграму послідовності.

У другому розділі було проведено тестування платформи для спільної роботи викладача та студента під час іспиту за розробленим планом. Процес тестування був описаний.

У третьому розділі наведене керівництво для розгортання та впровадження створеного додатку.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-РЕДАКТОР, КОД-ШЕРИНГ, ВЕБ-СОКЕТИ

ABSTRACT

Explanatory note of the diploma project consists of 3 sections, 3 tables, 10 sources, and an annex with code sources – total 72 pages.

The object of study: code-sharing systems for multiple users, online-editors for programming which allow to compile or interpret code with a use of remote resources.

The aim of the diploma project: use all the capabilities of modern web-techologies to simplify and optimize passing of the coding exams in universities.

In the first section, the architecture of current web-application was described and created thereafter, also a sequence diagram was constructed.

The second section covers the testing process of the resulting web application according to the test plan created before.

In the third section the deployment and implementation of the system for a mutual work of a student and a lecturer was described in details.

KEYWORDS: ONLINE-EDITOR, WEB SOCKETS, CODE SHARING

Пояснювальна записка
до дипломного проекту

на тему: Платформа для спільної роботи викладача та студента під час іспиту

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – набір чітко визначених методів для взаємодії різних компонентів.

Компонент – незалежний блок інтерфейсу, що може бути ізольованим від іншої частини програми та повторо використаним на ішних сторінках чи в іншому додатку.

Кімната – у контексті додатку – ізольований простір імен для виконання однієї екзаменаційної роботи багатьма студентами.

Judge0 – сервіс, що має Web API для інтерпритації та компіляції коду на усіх популярних мовах програмування.

WebSocket – протокол, що створений для передачі даних між двома кінцевими точками (наприклад, між веб-браузером та веб-сервером) у режимі реального часу. Він реалізує двонаправлений канал зв'язку, який працює через один TCP-сокет.

MonacoEditor – плагін для підсвічування синтаксису мов програмування та підтримки IntelliSense у текстових редаткорах. Використовується популярною IDE Visual Studio Code.

ВСТУП

Зі стрімким розвитком інтернету багато звичних процесів зазнали автоматизації та оптимізації. Можливості інтернет-технологій розширюються кожного дня і тому веб як платформа стрімко зростає, поступово витісняючи більшість десктопних додатків з ринку. Розвиток мережі відчутно впливає і на сферу освіти. Студенти та викладачі спеціальностей, що пов’язані з інформаційними технологіями, зазвичай використовують системи контролю версій чи електронну пошту для здачі робіт за допомогою мережі Інтернет протягом семестру. При цьому для здачі іспитів з програмування можливості веб-технологій не застосовуються, хоча вони могли б значно оптимізувати цей процес. Можна сказати, що веб-платформи для перевірки знань з програмування поділяються на три види:

– сервіси з можливістю парної роботи онлайн, що зазвичай використовуються під час технічних співбесід(наприклад, CodeInterview);

– сервіси, що дозволяють зберігати власноруч створені програми та ділитися ними у мережі (наприклад, JSFiddle);

– сервіси, що пропонують свої обчислювальні потужності для використання їх платформ у якості повноцінних IDE.

Метою написання даного дипломного проекту є застосування широких можливостей сучасних веб-технологій для спрощення процесу складання іспитів з програмування в університетах. В тому числі надання змоги одночасно працювати багатьом студентам та лише одному викладачу, контролю за виконанням роботи у режимі реального часу, реалізації оцінювання і штрафів, а також – збереження вихідних даних у зручному для доступу форматі.

Призначенням розробки є впровадження нового формату здачі екзаменів в онлайн-режимі. Проект повинен оптимізувати взаємну роботу викладача і студента під час іспиту та інтегрувати готовий продукт у цей процесс його складання.

Задачі:

– запровадження реєстрації студентів та викладачів;

– створення викладачами нових «кімнат» для проходження екзаменів;

– встановлення часових обмежень при проходженні іспиту;

– вибір зручної для студента мови програмування;

– онлайн-перегляд прогресу студентів викладачем;

– виставлення штрафів та оцінок викладачем;

– створення особистого кабінету студента з можливістю перегляду усіх зданих іспитів, реалізованих програм та отриманих оцінок;

– створення особистого кабінету викладача з можливістю перегляду усіх зданих іспитів, реалізованих програм та отриманих оцінок студентами.

Цілі:

– реалізація централізованого контролю над проходженням іспитів програмування;

– запобігання дисциплінарним порушенням під час роботи;

– зменшення часового навантаження на викладача;

– підвищення рівня інформованості студентів;

– створення електронного архіву для викладачів.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Програмний продукт, що задовольняє вимогам повинен бути написаний на мові, що дозволяє запуск на сучасних операційних системах, таких як Windows 10, Ubuntu 18.04 та macOS 10. Він повинен підтримувати HTTP запити по протоколу REST з кодуванням данних за допомогою JSON та підтримку протоколу WebSocket. Для підтримки роботи з MongoDB необхідне встановлення відповідного програмного забезпечення. Реалізований додаток має підтримувати можливість звернення до сторонніх API.

Для забезпечення високої доступності та зручності масштабування програмний продукт повинен бути розділений на окремі модулі, що відповідають за вирішення окремих логічниз задач. Клієнтська та серверна частина додатку повинні мати змогу обмінюватися даними за протоколами HTTP та WebSocket.

1.2 Змістовний опис і аналіз предметної області

Комплекси для онлайн-взаємодії під час вирішення завдань з програмування мають підтримувати хоча б декілька можливостей з такого переліку функцій: вибір мови програмування для реалізації, підсвічування синтаксису, можливість спільної роботи над кодом, фіксація результатів у базі даних та зручний доступ до історії. Обов’язковими функціями є можливість виконання та запуску коду і підсвічування синтаксису.

1.3 Аналіз успішних ІТ-проектів

В інтернеті можна знайти велику кількість систем для програмування у онлайн-редакторі з можливістю код-шерингу. Слід дослідити кожен з аналогів на можливість їх застосування для одночасної колективної здачі іспиту, що є ключовою функцією платформи, що розроблюється у рамках дипломного

проекту. В ході пошуку схожих за функціональністю платформ було виявлено чотири системи, що містять схожі функції.

- Codeshare;
- CodeInterview;
- JSFiddle;
- Coderpad.

Codeshare – онлайн-редактор для технічних співбесід, індивідуальної роботи та парного програмування. Один з найбільш популярних додатків для код-шерингу. Мінусом даного сервісу є те, що історія реалізованих завдань зберігається лише на 24 години. Важливою перевагою є можливість налаштувати відеозв'язок.

CodeInterview за функціями дуже схожий на сервіс Codeshare. Відрізняється він можливістю інтерв'юерів чи викладачів залишати текстові коментарі під час виконання завдання та описувати його у текстовому форматі до початку реалізації. Крім того, працює він значно швидше та стабільніше, бо використовує потужності хмарних технологій.

JSFiddle – сервіс для шерингу програм, що написані на веб-технологіях: HTML, CSS та JavaScript. Він не дозволяє переглядати прогрес у режимі онлайн та обирати технології реалізації. Важливими плюсами є можливість переглядати мініатюру створеної веб-сторінки та зберігати створені програми впродовж необмеженого часу.

SyncFiddle єдиний з близьких аналогів дозволяє приєднуватись до кімнати багатьом користувачам одночасно. Даний продукт використовується під час навчання для шерингу екрану викладача багатьом студентам. На жаль, SyncFiddle обмежений у виборі технологій і не вирішує проблему одночасного написання коду багатьма користувачами.

Скорочену порівняльну характеристику розглянутих продуктів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика проектів

Аналог / Функція	Codeshare	CodeInterview	JSFiddle	SyncFiddle
Текстовий опис завдання для реалізації	–	+	–	–
Можливість встановити часовий ліміт для реалізації	–	+	–	+
Вибір мови програмування	+	+	–	–
Додавання багатьох користувачів до спільного іспиту	–	–	–	–
Онлайн-перегляд прогресу виконання	+	–	–	+
Збереження історії пройдених/створених екзаменів	+	–	+	–

З порівняння представленого в таблиці 1.1, можна зробити висновок, що всі продукти мають майже однаковий функціонал, але всі вони пристосовані для парної роботи і тому не відповідають ключовій вимозі для використання у якості екзаменаційної платформи. Існуючі аналоги також не передбачують можливості оцінювання реалізованих завдань.

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Розроблення функціональних вимог

Варіанти використання у системі наведені у таблиці 1.2.

Таблиця 1.2 – Варіант використання UC01

Назва	Авторизація користувача
Опис	Будь-який користувач має можлівість авторизуватися у додатку
Учасники	Користувач
Передумови	
Постумови	Користувач пройшов авторизацію у комплексі
Основний сценарій	Система демонструє вікно авторизації яке містить поле вводу логіну та паролю Користувач заповнює поля Користувач натискає кнопку “Login” Система авторизує користувача
Розширення сценаріїв	Система не може авторизувати користувача через неправильні дані Система демонструє повідомлення про помилку

Таблиця 1.3 – Варіант використання UC02

Назва	Перегляд списку поточних кімнат
Опис	Будь який користувач має можлівість перегляду списку кімнат
Учасники	Користувач
Передумови	Користувач пройшов авторизацію у комплексі
Постумови	Надано список кімнат

Продовження таблиці 1.3

Основний сценарій	Користувач надсилає запит на отримання списку кімнат Додаток відображає список усіх кімнат

Таблиця 1.4 – Варіант використання UC03

Назва	Створення нової кімнати
Опис	Будь який викладач має можлівість створити нову кімнату
Учасники	Викладач
Передумови	Викладач пройшов авторизацію у комплексі Усі поточні кімнати були відображені
Постумови	Створена нова кімната
Основний сценарій	Адміністратор вводить дані у необхідна поля та натискає кнопку «Створити кімнату» Система створює нову кімнату

Таблиця 1.5 – Варіант використання UC04

Назва	Приєднання до кімнати
Опис	Будь-який користувач має можливість приєднатися до кімнати
Учасники	Користувач
Передумови	Користувач пройшов авторизацію Відображені усі поточні кімнати
Постумови	Користувач приєднався до кімнати

Продовження таблиці 1.5

Основний сценарій	Користувач надсилає запит на додавання до кімнати Додаток приєднує користувача

Таблиця 1.6 – Варіант використання UC05

Назва	Перегляд списку учасників кімнати
Опис	Будь який користувач у кімнаті має можливість перегляду списку користувачів у ній
Учасники	Користувач
Передумови	Користувач пройшов авторизацію Користувач приєднався до кімнати
Постумови	Відображено список усіх користувачів у кімнаті
Основний сценарій	Користувач приєднується до кімнати Додаток відображає усіх користувачів, що до неї приєдналися

Таблиця 1.7 – Варіант використання UC06

Назва	Налаштування опцій кімнати
Опис	Викладач, який створив кімнату, має можливість встановити її налаштування
Учасники	Викладач
Передумови	Викладач пройшов авторизацію у комплексі Викладач створив нову кімнату Викладач приєднався до кімнати

Продовження таблиці 1.7

Постумови	Створені опції кімнати
Основний сценарій	Адміністратор вводить налаштування кімнати у відповідні поля Додаток зберігає зміни

Таблиця 1.8 – Варіант використання UC07

Назва	Початок екзамену
Опис	Викладач, який створив кімнату, має можливість розпочати екзамен у ній
Учасники	Викладач
Передумови	Викладач налаштував опції кімнати
Постумови	Екзамен у кімнаті розпочато
Основний сценарій	Викладач натискає на кнопку «Розпочати екзамен» Додаток починає зворотній відлік таймеру

Таблиця 1.9 – Варіант використання UC08

Назва	Написання коду студентом
Опис	Будь-який студент, що приєднався до кімнати, має можливість писати код у текстовому редакторі
Учасники	Студент
Передумови	Студент авторизувався у системі Студент приєднався до кімнати
Постумови	

Продовження таблиці 1.9

Основний сценарій	Система демонструє вікно текстового редактору Студент реалізує поставлене завдання шляхом написання коду у текстовому редакторі

Таблиця 1.10 – Варіант використання UC09

Назва	Виконання коду
Опис	Будь який студент у кімнаті має можливість запускати код, написаний у текстовому редакторі
Учасники	Студент
Передумови	Студент приєднався до кімнати Студент написав код у текстовому редакторі Екзамен у кімнаті було розпочато
Постумови	Виведено результат виконання коду
Основний сценарій	Користувач натискає на кнопку «Виконати код»

Таблиця 1.11 – Варіант використання UC10

Назва	Виставлення оцінок та штрафів за виконання
Опис	Викладач, що створив кімнату, має можлівість виставляти оцінки та штрафи за роботи студентів
Учасники	Викладач

Продовження таблиці 1.11

Передумови	Екзамен у кімнаті було розпочато Екзамен було завершено
Постумови	Студенти можуть побачити свої оцінки та свої штрафи
Основний сценарій	Викладач вводить необхідні дані у відповідні поля Додаток зберігає дані про кімнату

Таблиця 1.12 – Варіант використання UC11

Назва	Перегляд історії екзаменів
Опис	Будь який користувач має можлівість переглянути історію складених або створених екзаменів
Учасники	Користувач
Передумови	Користувач пройшов авторизацію у системі
Постумови	Відображена історія екзаменів
Основний сценарій	Користувач пройшов авторизацію у системі Користувач переходить на сторінку з відображеними кімнатами Користувач натискає кнопку «Переглянути історію» Додаток відображає історію для певного користувача
Розширення сценаріїв	Система демонструє користувачу повідомлення про помилку

Таблиця 1.13 – Варіант використання UC12

Назва	Онлайн-перегляд прогресу певного студента
Опис	Викладач, який створив кімнату, має можливість у режимі онлайн переглянути поточний прогрес у виконанні завдання будь-якого студента з кімнати
Учасники	Викладач
Передумови	Екзамен було розпочато
Постумови	Виведення поточного стану програми студента на екран викладача
Основний сценарій	Викладач розпочинає екзамен Викладач у списку студентів натискає на будь-якого з них Додаток відображає поточний прогрес користувача

Також система має такі функціональні вимоги:

Таблиця 1.14 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Реєстрація користувачів в системі
Опис	Додаток дає можливість реєстрації

Таблиця 1.15 – Опис функціональної вимоги REQ002

Номер	REQ002

Продовження таблиці 1.15

Назва	Створення екзаменаційних кімнат
Опис	Система надає можливість зареєстрованому користувачу, що є викладачем, створювати екзаменаційні кімнати

Таблиця 1.16 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Перегляд створених екзаменаційних кімнат
Опис	Система надає можливість зареєстрованому користувачу переглядати створені активні кімнати та приєднуватись до них у разі відповідності групи екзаменаційної кімнати та користувача

Таблиця 1.17 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Написання коду в онлайн-редакторі
Опис	Система надає можливість користувачу, що є студентом писати код на обраній мові програмування у вікні текстового редактору.

Таблиця 1.18 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Виконання коду з онлайн-редактору
Опис	Система надає можливість користувачу, що є студентом, виконувати написаний в онлайн-редакторі код та переглядати результат інтерпритації/компіляції.

Таблиця 1.19 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Налаштування таймеру та опис завдання реалізації
Опис	Система дозволяє користувачу, що є викладачем, встановлювати таймер для виконання екзаменаційного завдання

Таблиця 1.20 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Виставлення оцінок та штрафів
Опис	Система дозволяє користувачу, що є викладачем, виставляти оцінки та штрафи студентам під час виконання екзаменаційного завдання

Таблиця 1.21 – Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Вхід до особистого аккаунту
Опис	Додаток дозволяє входити у власний особистий аккаунт

Таблиця 1.22 – Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Вихід з особистого аккаунту
Опис	Додаток надає можливість вийти з особистого акканту

Наведемо результуючу матрицю трасування на Рисунку 1.1.

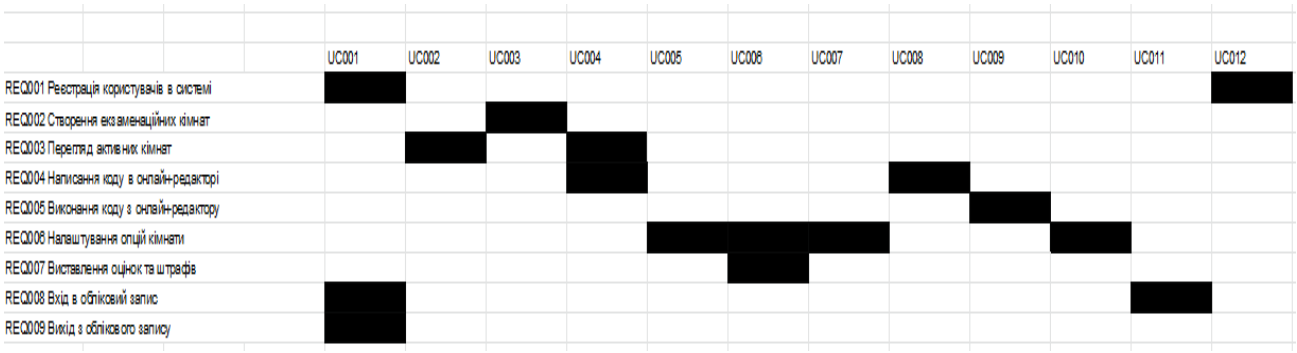

Рисунок 1.1. – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Створений програмний продукт повинен задовольняти таким нефункціональним вимогам:

– підтримувати сучасні версії операційних систем та браузерів;

– підтримувати усі версії екранів, у тому числі і Retina;

1.4.3 Постановка комплексу завдань модулю

Розроблене програмне забезпечення створюється для подальшого впровадження нового формату здачі екзаменів в онлайн-режимі.

Мета створення даної работи – надання змоги одночасно працювати багатьом студентам та лише одному викладачу, контролю за виконанням роботи у режимі реального часу, реалізації оцінювання і штрафів, а також – збереження вихідних даних у зручному для доступу форматі. Проект повинен оптимізувати взаємну роботу викладача і студента під час іспиту та інтегрувати готовий продукт у цей процесс його складання.

Для того, щоб досягнути поставленої мети, додаток має вирішувати такі задачі:

- виконання коду з онлайн-редактору;
- надання можливості викладачу контролювати іспит;
- надання необхідної інформації для студентів та викладачів;
- збереження історії.

Додаток повинен працювати на операційних системах з будь-яким зі встановлених сучасних веб-браузерів.

1.5 Висновки по розділу

У цьому розділі було описано предметну область розробки та зроблений її детальний аналіз. Виділені успішні та відомі проекти, що стосуються даної предметної області області, виконано їх порівняння за різними характеристиками: розглянуті переваги, недоліки та можливості. Визначено варіанти використання, функціональні та нефункціональні вимоги. Результуючі дані представлені у вигляді порівняльної таблиці.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Перед етапом безпосередньої реалізації програмного забезпечення потрібно провести детальний аналіз та включити етап моделювання. Для цього буде використано методологію створення діаграм BPMN та буде відображено основні процеси використання проекту.

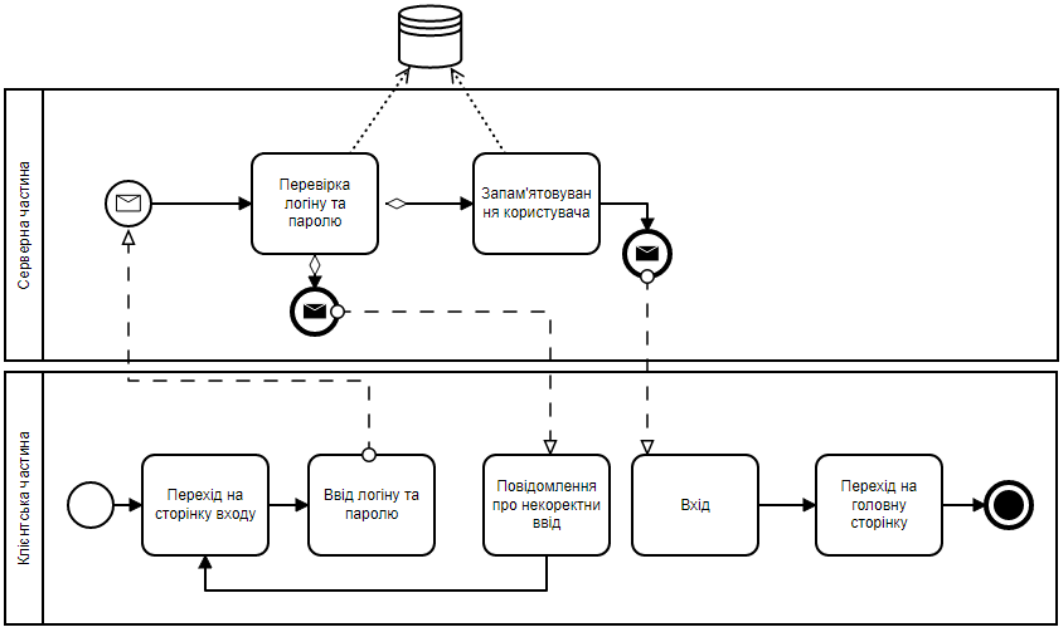

Рисунок 2.1 – Вхід користувача на сайт

Послідовність етапів авторизації користувача на сайті:

– користувач потрапляє на сторінку авторизації;
– користувач вводить дані для авторизації;
– введені дані відправляються на сервер;
– сесія запам’ятовується, повідомляється про успішну авторизацію;
– якщо дані неправильні, повідомляється про помилку.

Рисунок 2.2 – Створення екзаменаційної кімнати

Опис процесу створення екзаменаційної кімнати:

– викладача заходить на сторінку перегляду кімнат;
– викладача вводить дані нової кімнати у відповідні поля;
– на сервер відправляється запит на створення нової кімнати;
– сервер повідомляє викладача про успішне створення кімнати;
– інакше виводиться повідомлення про помилку.

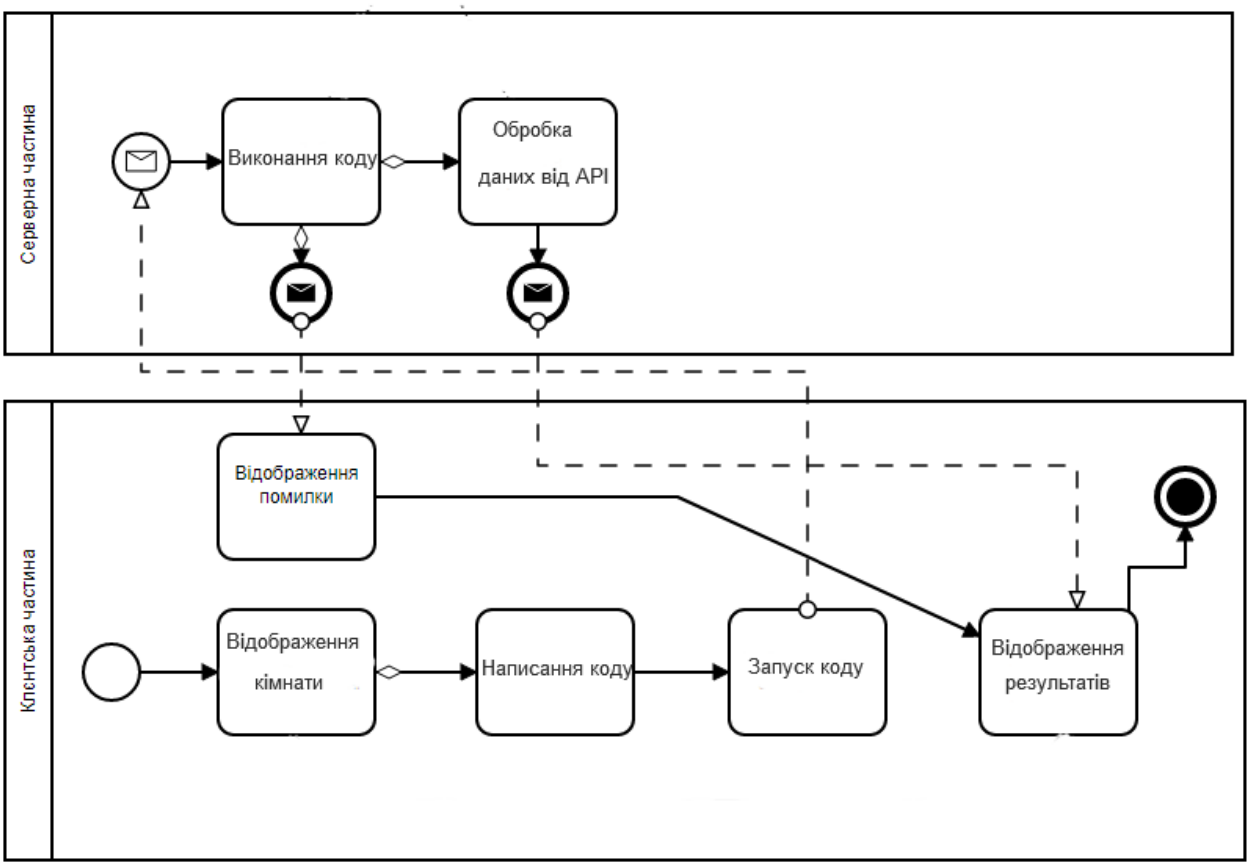

Рисунок 2.3 – Процес проходження екзамену

Опис процесу проходження екзамену:

– студент заходить на сторінку екзаменаційної кімнати;

– він може взаємодіяти з текстовим редактором та писати код у ньому;

– код, що запускає користувач, відправляється на сервер;

– сервер перевіряє правильність введених даних, виконує код за домогою стороннього API, повертає результат користувачу;

– інакше користувач отримує повідомлення про помилку.

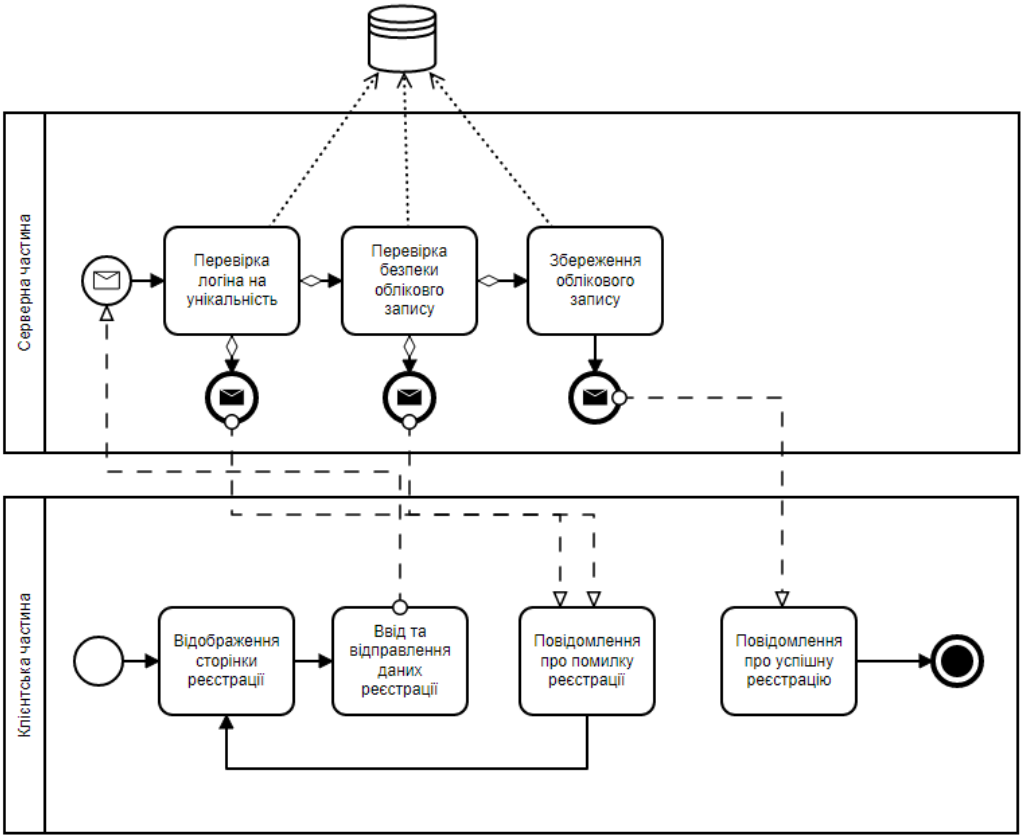

Рисунок 2.4 – Реєстрація користувача

Наведено послідовний опис процесу реєстрації користувача:

– відкривання сторінки реєстрації;

– вводення даних, потрібних для реєстрації;

– користувач відправляє запит на реєстрацію на сервер;

– якщо дані не є унікальними, то користувач отримує відповідне повідомлення;

– якщо дані не відповідають параметрам валідації чи вимогам безпеки, то користувач отримує відповідне повідомлення;

– створюється особистий аккаунт, виводиться відповідне повідомлення.

2.2 Архітектура програмного забезпечення

Розроблена система виконується під сучасними операційними системами, що мають встановлені останні версії браузерів Google Chrome, Mozilla Firefox

або Safari. Використання браузерами Internet Explorer та Edge можливе, але не було протестоване.

Для розробки браузерної частини системи було застосовано мову JavaScript з використанням можливостей стандарту EcmaScript 7, з використанням бібліотеки React. Використано паттерн Observer, який дозволяє зручно та швидко виконувати асинхронний обмін повідомленнями та даними між різними частинами додатку за допомогою механізму функцій зворотнього виклику. Взаємодія з серверною частиною відбувається по протоколу HTTP REST з кодуванням даних за допомогою JSON та за допомогою протоколу WebSocket з використанням бібліотеки Socket.io.

Для розробки серверної частини було застосовано платформу Node.js та мікрофреймворк для веб-додатків Express. Це дозволило використати одну мову програмування для створення як фронтенду, так і бекенду. Використано паттерни MVC, Observer та Facade. Окремі функціональні компоненти розділені на модулі, які, в свою чергу, можуть бути імпортовані іншими модулями. Даний підхід дозволяє розділити програму на окремі частини, що інкапсулюють складність та надають зручний API для взаємодії.

У якості сховища даних використовується документоорієнтована база даних MongoDB. Документи у колекціях MongoDB зберігаються у форматі JSON, що значно спрощує її інтеграцію з Node.js. Крім того, NoSQL бази даних є гнучкими та більш придатні для горизонтального масштабування і дозволяють спростити написання запитів і пришвидшити доступ до даних.

2.3 Конструювання програмного забезпечення

Опис класів наведено у таблиці 2.1. Більш детальний опис методів наведено у таблиці 2.2. Опис таблиць бази даних наведено у таблиці 2.3. Схема бази даних наведена у окремому документі.

Таблиця 2.1 – Опис класів системи

Клас/Модуль	**Опис**
MainPageComponent	Клас, який містить бізнес-логіку головної сторінки додатку та є компонентом вищого порядку
LoginComponent	Клас, який містить бізнес-логіку компоненту авторизації
RegistrationComponent	Клас, який містить бізнес-логіку компоненту реєстрації
RoomsListComponent	Клас, який містить бізнес-логіку компоненту списку поточних кімнат
StudentsListComponent	Клас, який містить бізнес-логіку компоненту списку студентів, що приєднались до кімнати та є компонентом вищого порядку
MonacoEditorComponent	Клас, який містить бізнес-логіку компоненту текстового редактору для реалізації завдань
EditorComponent	Клас, який містить бізнес-логіку компоненту для взаємодії з текстовим редактором та є компонентом вищого порядку
RoomComponent	Клас, який містить бізнес-логіку компоненту екзаменаційної кімнати та є компонентом вищого порядку
RoomSetupComponent	Клас, який містить бізнес-логіку компоненту для створення налаштувань кімнати для екзамену та є компонентом вищого порядку

Продовження таблиці 2.1

RoomInfoComponent	Клас, який містить бізнес-логіку компоненту для виведення інформації про поточну кімнату та є компонентом вищого порядку
StudentInfoComponent	Клас, який містить бізнес-логіку компоненту для виведення інформації про студента
StudentAccountComponent	Клас, який містить бізнес-логіку компоненту для виведення інформації про особистий кабінет студента
LecturerAccountComponent	Клас, який містить бізнес-логіку компоненту для виведення інформації про особистий кабінет викладача
RoomResultComponent	Клас, який містить бізнес-логіку компоненту для виведення результатів екзамену поточної кімнати
App	Головний клас програми
MongoClient	Клас, який надає методи для взаємодії з клієнтом MongoDB
LecturerModel	Клас, який містить бізнес-логіку моделі Лектора
ProgramModel	Клас, який містить бізнес-логіку моделі Програми
StudentModel	Клас, який містить бізнес-логіку моделі Студента

Продовження таблиці 2.1

RoomModel	Клас, який містить бізнес-логіку моделі Екзамену
Router	Клас, який містить бізнес-логіку обробки маршрутів серверу
Middlewares	Клас, який містить бізнес-логіку проміжних обробників запитів до сервера
Auth	Клас, який містить бізнес-логіку авторизації
Judge0Client	Клас, який надає методи для взаємодії з Judge0 API
SocketClient	Клас, який надає методи для взаємодії з бібліотекою Socket.IO

Таблиця 2.2 – Опис методів класів та інтерфейсів системи

Клас	**Метод класу**	**Опис методу**
App	use(middlewareFunction)	Додає проміжний обробник запитів до сервера
App	listen(port)	Запускає сервер на відповідному порті
App	getConfig(options)	Повертає дані про конфігурацію додатку
MongoClient	connect(username, password, databaseName)	Створює підключення до бази даних MongoDB

Продовження таблиці 2.2

MongoClient	close(connection)	Закриває створене підключення до бази даних
MongoClient	createSchema(schema)	Створює нову схему коллекції у ORM Mongoose
MongoClient	generateHash(password)	Генерує хеш паролю за криптографічним алгоритмом bcrypt
MongoClient	generateSalt(size)	Генерує рядок випадкових символів у кодуванні base-16 (hexidecimal)
MongoClient	comparePasswords(password, originalHash, salt)	Порівнює пароль, введений при авторизації з оригінальним паролем користувача, що збережений у базі
LecturerModel	create(data)	Створює нового викладача у базі даних
LecturerModel	findOne(data)	Знаходить певного викладача у базі даних за його іменем
LecturerModel	findById(id)	Знаходить певного викладача у базі даних за його ідентифікатором

Продовження таблиці 2.2

StudentModel	create(data)	Створює нового студента у базі даних
StudentModel	findOne(data)	Знаходить певного студента у базі даних за його іменем
StudentModel	findById(id)	Знаходить певного студента у базі даних за його ідентифікатором
RoomsModel	find(data)	Знаходить усі створені екзаменаційні кімнати у базі даних
RoomsModel	findOne(data)	Знаходить певну кімнату у базі даних
RoomsModel	findById(id)	Знаходить певну кімнату у базі даних за її ідентифікатором
RoomsModel	create(data)	Створює нову кімнату у базі даних
RoomsModel	getStudents(room, socket)	Знаходіть усіх студентів у базі даних, що підключені до даної кімнати
RoomsModel	removeStudent(socket, callback)	Видаляє підключення студента з кімнати у базі даних
RoomsModel	addStudent(room, socket)	Додає студента до кімнати у базі даних

Продовження таблиці 2.2

RoomsModel	findByLecturerId(id)	Знаходить усі кімнати у базі даних, що створені певним викладачем
ProgramModel	create(data)	Створює новий програмний код у базі даних
ProgramModel	findByRoomId(id)	Знаходить у базі даних всі програмні коди, що були написані під час екзамену у певній кімнаті
ProgramModel	findByStudentId(id)	Знаходить усі програми, що були написані певним студентом за його індентифікатором
Auth	saveUserToken(user, token)	Зберігає токен авторизації для певного користувача
Auth	verifyToken(user, token)	Порівнює токен авторизації, що прийшов від користувача, зі збереженим токеном авторизації для цього користувача

Продовження таблиці 2.2

Auth	generateToken(user)	Генерує новий токен авторизації для користувача
Middlewares	checkAuthentication(request, response)	Перевіряє авторизацію користувача перед кожним запитом
Middlewares	handle404(request, response)	Перехоплює помилки звернення клієнту до неіснуючих маршрутів у додатку
Middlewares	logRequest(request, response)	Логує усі запити до серверу у консоль серверу
Router	createRouteHandler(router, middlewares, handler)	Асоціює URL маршрут з певною функцією-обробником
Judge0Client	getSubmissionToken(languageId, code)	Створює новий токен подання у сервісі для виконання коду у режимі онлайн Judge0
Judge0Client	getTokenResult(token)	Повертає результат виконання програмного коду для попередньо створеного токена подяння у Judge0

Продовження таблиці 2.2

Judge0Client	interpret(languageName, code)	Повертає результат інтерпритації програмного коду клієнту
Judge0Client	compile(languageName, code)	Повертає результат компіляції програмного коду клієнту
SocketClient	use(socket, next)	Надає серверу WebSocket доступ до запитів на основний сервер
SocketClient	set(protocol)	Конфігурує протокол для використання бібліотекою Socket.io
SocketClient	adapter(adapterType)	Визначає та конфігурує тип адаптера для використання бібліотекою Socket.io
SocketClient	registerEvent(namespace, handler)	Створює певну подію та реєструє функцію-обробник для неї
SocketClient	createNamespace(name)	Створює простір імен для використання сервером Socket.IO за його назвою

Продовження таблиці 2.2

SocketClient	emitEvent(eventName)	Викликає певну подію за її назвою, тобто ініціалізує функцію-обробник події
MainPageComponent	render()	Відображає головний компонент додатку на сторінці
MainPageComponent	onStartBtnClick()	Функція-обробник натискання на кнопку «Почати». Відображає компонент авторизації.
LoginComponent	render()	Відображає компонент авторизації користувача на сторінці
LoginComponent	onSubmitBtnClick()	Функція-обробник натискання на кнопку «Авторизуватися». Надсилає дані користувача на сервер для авторизації
LoginComponent	onRegisterBtnClick()	Функція-обробник натискання на кнопку «Перейти до реєстрації». Відображає компонент реєстрації.

Продовження таблиці 2.2

RegistrationComponent	render()	Відображає компонент реєстрації користувача на головній сторінці.
RegistrationComponent	onSubmitBtnClick()	Функція-обробник натискання на кнопку «Зареєструватися». Надсилає дані користувача на сервер для реєстрації.
RoomsListComponent	render()	Відображає компонент списку доступних екзаменаційних кімнат на сторінці
RoomsListComponent	componentDidMount()	Отримує дані про поточні активні екзаменаційні кімнати з серверу
RoomsListComponent	createRoom(name, group)	Створює нову екзаменаційну кімнату для певної групи студентів
RoomsListComponent	onListElementClick()	Функція-обробник вибору однієї з кімнат у списку. Приєднує користувача до кімнати.

Продовження таблиці 2.2

StudentsListComponent	render()	Відображає компонент зі списком приєднаних до кімнати студентів на сторінці
StudentsListComponent	componentDidMount()	Отримує дані про приєднаних до кімнати студентів з серверу
StudentsListComponent	onListElementClick()	Функція-обробник вибору одного зі студентів у списку
MonacoEditorComponent	render()	Відображає компонент текстового редактору для програмування на сторінці
MonacoEditorComponent	getEditorValue()	Повертає поточний текст програми у вікні текстового редактора
MonacoEditorComponent	setEditorValue(value)	Змінює текст програмного коду у вікні текстового редактора
MonacoEditorComponent	onEditorValueChange()	Функція-обробник події зміни тексту у вікні текстового редактора для програмування

Продовження таблиці 2.2

EditorOutputComponent	render()	Відображає компонент з інформацію про результат виконання програмного коду на сторінці
EditorOutputComponent	setStdout(value)	Змінює текст у вікні для виведення інформації результату виконання коду
EditorOutputComponent	setStderr(value)	Змінює текст у вікні для виведення інформації помилок виконання програмного коду
EditorOutputComponent	onRunBtnClick()	Функція-обробник натискання на кнопку «Виконати». Компілює або інтерпритує код з текстового редактору для програмування
RoomComponent	render()	Відображає компонент екзаменаційної кімнати на сторінці
RoomComponent	setPenalty(user)	Встановлює штраф за виконання екзамену для певного студента

Продовження таблиці 2.2

RoomSetupComponent	render()	Відображає компонент з даними налаштування опцій екзаменаційної кімнати на сторінці
RoomSetupComponent	setTimer()	Встановлює часовий таймер для роботи екзаменаційної кімнати
RoomSetupComponent	setTask()	Встановлює екзаменаційне завдання, що буде виконуватись у кімнаті
RoomInfoComponent	render()	Відображає компонент з інформацією про екзаменаційну кімнату на сторінці
RoomResultComponent	render()	Відображає компонент з інформацією про результат екзамену на сторінці
RoomResultComponent	setMark(user)	Встановлює оцінку за екзамен для певного студента
RoomResultComponent	setPenalty(user)	Встановлює штраф за екзамен для певного студента

Продовження таблиці 2.2

StudentInfoComponent	render()	Відображає компонент з інформацією про поточного студента на сторінці
StudentAccountComponent	render()	Відображає компонент з інформацією про обліковий запис студента на сторінці
StudentAccountComponent	componentDidMount()	Завантажує з бази даних інформацію про екзамени, складені певним студентом
LecturerAccountComponent	render()	Відображає компонент з інформацією про обліковий запис лектора на сторінці
LecturerAccountComponent	componentDidMount()	Завантажує з бази даних інформацію про екзаменаційні кімнати, що створені певним викладачем

Таблиця 2.3 – опис моделі бази даних

Таблиця	**Запис**	**Тип**	**Опис**
Students	ID (первиний ключ)	ObjectID	Унікальний ідентифікатор
Students	firstName	String	Ім’я студента

Продовження таблиці 2.3

Students	lastName	String	Прізвище студента
Students	groupCode	String	Група студента
Students	picture	String	Адреса аватару користувача
Students	hash	String	Хеш паролю
Students	salt	String	Сіль для хешу
Students	login	String	Логін
Lecturers	ID (первиний ключ)	ObjectID	Унікальний ідентифікатор
Lecturers	login	String	Логін
Lecturers	firstName	String	Ім’я викладача
Lecturers	lastName	String	Прізвище викладача
Lecturers	picture	String	Адреса аватару користувача
Lecturers	hash	String	Хеш пароля
Lecturers	salt	String	Сіль для хешу пароля
Rooms	ID (первиний ключ)	ObjectID	Унікальний ідентифікатор
Rooms	lecturerID	ObjectID	Посилання на викладача, що створив кімнату

Продовження таблиці 2.3

Rooms	title	String	Назва
Rooms	groupCode	String	Код групи у екзаменаційній кімнаті
Rooms	task	String	Завдання у екзаменаційній кімнаті
Rooms	timer	Number	Таймер виконання
Rooms	connections	Array	Масив підключених користувачів
Rooms	programs	Array	Масив програм, що були написані у кімнаті
Programs	ID (первиний ключ)	ObjectID	Унікальний ідентифікатор
Programs	text	String	Текст програми
Programs	mark	Number	Оцінка за виконання програми
Programs	penalty	Number	Штраф за виконання програми

Продовження таблиці 2.3

Programs	roomID	ObjectID	Посилання на екзаменаційну кімнату
Programs	studentID	ObjectID	Посилання на користувача, що створив програму

2.4 Аналіз безпеки даних

Паролі користувачів у базі зберігаються у вигляді результату обробки криптографічно стійкою функцією bcrypt, тому оригинальний пароль майже неможливо знайти без використання повного перебору чи застосування райдужних таблиць. Подібні спроби будуть значно сповільнені через особливісті роботи bcrypt, що змушує витрачати багато часу на подібні криптографічні атаки. Таким чином цінність ресурсів, що будуть затрачені на знаходження одного паролю, несумісне з цінністю самого паролю.

Налаштування зв’язку з базою даних MongoDB відбувається за допомогою імені користувача та паролю.

Механізм авторизації у додатку використовує технологію JWT(JSON Web Token), що включає використання секретного ключа на боці сервера.

Взаємодія з Judge0 API відбувається з використанням токену авторизації, завдяки чому перевіряється що запит надійшов від зареєстрованого у сервісі користувача.

2.5 Висновки по розділу

У цьому розділі проведено моделювання архітектури веб-додатку для взаємодії викладача та студента під час іспиту. Обрано браузери, що найкраще

підходять у якості програми для взаємодії зі створеним додатком. Було знайдено найбільш зручну мову програмування для реалізації, підібрано відповідні бібліотеки та фреймворки для пришвидшення та спрощення розробки. Були визначені протоколи взаємодії між клієнською та серверною частиною системи та обрані паттерни програмування з урахуванням їх переваг та недоліків. У таблицях описані класи, методи класів, їх вхідні параметри та задачі, що вони виконують. Також було проаналізовано безпеку даних у застосунку з урахуванням широко відомих та надійних підходів.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Етап тестування є важливим в процесі розробки даного проекту у зв`язку з особливостями роботи: інтенсивним обміном даними між клієнтом та сервером. Також додаток передбачає постійну взаємодію зі стороннім API.

Тестування програмного забезпечення — проведення контролю над якістю продукту, перевірка відповідності між фактичною та запланованою поведінкою програми з використанням кінцевого набору тестів.

Тестування, як процес виявлення помилок та багів не може повністю забезпечити правильність роботи програмного забезпечення у всіх можливих випадках. Тестування лише порівнює поведінку програми у різних ситуаціях зі специфікацією.

Поняття описується технічними вимогами, котрі описані у стандарті ISO 9126. План тестування ПЗ включає в себе обсяг, підхід, ресурси та план усіх методів тестування. План описує об’єкти та функції що будуть протестовані, тип тестів, ресурси та план необхідний для виконання тестування.

Буде проведено тестування усіх функціональних частин додатку.

Будуть протестовані такі можливості та функції:

- реєстрація користувачів;
- вхід користувачів у систему;
- створення екзаменаційних кімнат викладачами;
- налаштування опцій екзаменаційної кімнати;
- можливість приєднання студентами до доступних екзаменаційних кімнат;
- реалізація можливості перегляду коду студентів викладачем у режимі онлайн;
- реалізація можливості компіляції та інтерпритації коду;

– виставлення оцінок та штрафів;
– перегляд історії складених та створених іспитів.

Налаштовані наступні тестові модулі:

– реєстрація користувача;
– реєстрація користувача якщо логін вже зайнято;
– реєстрація користувача з невалідним паролем;
– авторизація користувача;
– авторизація користувача з неправильним паролем;
– створення екзаменаційної кімнати;
– створення екзаменаційної кімнати з невалідною назвою;
– можливість перегляду коду студентів викладачем у режимі онлайн;
– використання токену авторизації користувача;
– використання невалідного токену авторизації користувача;
– налаштування опцій екзаменаційної кімнати;
– налаштування опцій екзаменаційної кімнати з невалідними даними;
– виконання задачі компіляції та інтерпритації коду;
– виконання задачі компіляції та інтерпритації коду з помилками.

3.2 Підходи до тестування

Застосуємо наступні методи тестування у цьому плані:

– компонентне;
– інтеграційне;
– продуктивності.

3.2.1 Компонентне тестування

Компонентне тестування перевірить окремі частини API сервера, такі як:

– оброблювачі шляхів API;

– проміжкові оброблювачі шляхів API;

– методи доступу до бази даних.

3.2.2 Інтеграційне тестування

Інтеграційне тестування перевірить взаємодію між частинами додатку:

– взаємодія сервера з базою даних;

– взаємодія клієнтської частини з стороннім API;

– взаємодія викладача та студента за протоколом WebSocket;

– взаємодія клієнтської та серверної частини за допомогою протоколу HTTP.

3.2.3 Тестування продуктивності

Тестування продуктивності перевірить швидкість роботи наступних частин додатку:

– звернення до бази даних;

– обмін даними між викладачем та студентом;

– одночасна робота декількох створених екзаменаційних кімнат у додатку.

3.3 Критерії проходження тестування

3.3.1 Компонентне тестування

Умова успішного складення компонентного тестування – виконання усіх

елементів тесту. Якщо хоча б один пункт з переліку не був успішно виконаний – тестування вважається не пройденим.

3.3.2 Інтеграційне тестування

Умова успішного складення інтеграційного тестування – успішне виконання усіх елементів тесту. Якщо хоча б один пункт з переліку не був успішно виконаний – тестування вважається не пройденим.

3.3.3 Тестування швидкодії

Умова успішного складення тестування швидкодії – успішне виконання тесту з усіма варіантами параметрів з доступного набору за оптимальний час. Якщо хоча б один елемент тесту не пройшов або виконувався довше, ніж потрібно, тестування не вважається пройденим.

3.4 Процес тестування

3.4.1 Дані до тестів

Вхідні дані компонентного тестування – набори параметрів, на яких очікується конкретний результат, який водночас є і вихідними даними.

Вхідними даними інтеграційного тестування – варіанти даних, що можуть передаватись від користувача-студента до користувача-викладача. Вихідні дані – відображення отриманих даних клієнтською частиною додатку з боку викладача.

Вхідними даними тестування швидкодії є певні множини даних, що відповідають можливим варіантам роботи системи у конкретному випадку. Вихідними даними можна вважати швидкість обробки запитів до серверу та кількість оброблених запитів до серверу, а також середнє навантаження на апаратну частину.

3.4.2 Задачі тесту

Випробування повинні перевірити коректність роботи програми відповідно до умов виконання, а також виявити дефекти у її роботі.

3.4.3 План виконання

Компонентне тестування має виконуватись перед інтеграційним. Інтеграційне тестування, у свою чергу, виконується переж тестуванням швидкодії.

3.5 Вимоги до середовища

3.5.1 Апаратна частина

Вимоги апаратного середовища співвідносяться з вимогами, що наведені у технічному завданні.

3.5.2 Програмна частина

Для виконання тестування апаратна платформа повинна мати операційну систему на базі Linux або іншу Unix-сумісну зі встановленою платформою Node.js та базою даних MongoDB.

3.5.3 Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

- assert (стандартний модуль Node.js);
- Postman;
- Locust.io (https://locust.io/) – для тестування швидкодії.

3.6 Опис контрольного прикладу

У якості прикладу можна навести тестування логіну користувача за допомогою утиліти Postman.

Вхідні дані:

– пара логін та пароль *nonexistinguser nonexistingpassword* (не збережені у базі даних);

– пара логін та пароль *shulder nonexistongpassword* (неправильний пароль, існуючий користувач);

– пара логін та пароль *shulder 822729* (існуючий користувач і правильний пароль).

Вихідні дані:

– повідомлення про помилку у вигляді JSON стрічки *{“err”: “Failed to login”}*;

– повідомлення з токеном доступа при успішному логіні у вигляді JSON стрічки *{ “token”: “TOKEN”}* де TOKEN – токен доступу до API.

Для виконання тесту потрібно в Postman створити POST запит за адресою http://localhost:PORT/login де PORT – номер порту, на якому запущений локальний сервер.

У якості тіла запиту використати JSON об’єкт наступного формату: *{“login”: “LOGIN”, “password”: “PASSWORD”}* де LOGIN та PASSWORD – логін та пароль з однієї з пар вхідних даних. Після введення даних натиснути кнопку “Send” та перевірити відповідність вихідних даних до наведених вище (Рисунок 3.1).

Рисунок 3.1 – Приклад виконаного тестового прикладу

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для того, щоб розгорнути середовище для запуску розробленого додатку потрібно виконати наступні етапи:

- встановлення та налаштування MongoDB;
- встановлення Node.js;
- встановлення сучасного веб-браузера;
- запуск додатку.

4.1.1 Встановлення та налаштування MongoDB

Для встановлення бази даних MongoDB потрібно перейти на офіційний сайт MongoDB та завантажити версію для необхідної платформи. Можливо налаштувати запуск бази даних у вигляді фонового демона. Після цього база даних буде готова для роботи та доступна для підключення на порті, який визначить користувач, або на порті за замовчуванням.

4.1.2 Встановлення Node.js

Версії платформи Node.js з парними порядковими номерами – версії LTS релізу. Саме LTS реліз рекомендується для використання. Мінімальної версією для стабільної роботи додатку є версія 8.2.4. Для встановлення Node.js потрібно перейти на офіційний сайт платформи та завантажити версію для необхідної платформи. Існує також альтернативний спосіб встановлення з використанням NVM – Node Version Manager.

4.1.3 Встановлення сучасного веб-браузера

Для надійного функціонування системи необхідно взаємодіяти з нею за домогою одного з сучасних веб-браузерів. До таких належать Firefox версії 65+,

Google Chrome версії 70+ та Safari версії 11+. Кожен з них можна встановити на відповідному офіційному сайті. Зазвичай, один з сучасних браузерів за замовчуванням встановлений на популярних операційних системах.

4.1.4 Запуск додатку

Для того, щоб почати користуватись додатком необхідно запустити базу даних MongoDB. Далі потрібно перейти у директорію з файлами додатку та запустити у консолі команду *npm install*, яка встановить необхідні модулі для проекту з архіву пакетів NPM – Node Package Manager. Після успішного виконання цієї команди потрібно запустити скрипт *npm start*, який запустить проект.

4.2 Інструкція користувача

Інструкція користувача приведиться у керівництві користувача.

ВИСНОВКИ

У першому розділі “Аналіз вимог до програмного забезпечення” було описано та проаналізовано предметну область дипломного проекту.

У другому розділі “Моделювання та конструювання програмного забезпечення” було розроблено архітектуру платформи для спільної роботи викладача та студента під час іспиту.

У третьому розділі “Аналіз якості та тестування програмного забезпечення” розроблено план тестування додатку та виконано приклад тестування.

У четвертому розділі “Впровадження та супровід програмного забезпечення” містяться інструкції з встановлення та впровадження системи на апаратну платформу. Розроблено інструкції для користувача.

У рамках даного дипломного проекту було застосовано набуті знання з розробки баз даних, клієнт-серверної архітектури програмного забезпечення, веб розробки, безпеки даних, комп’ютерних мереж та використання паттернів програмування.

Розроблений комплекс задач є повноцінним програмним продуктом, готовим для використання. Він може бути доповнений новими функціями та горизонтально масштабований.

ПЕРЕЛІК ПОСИЛАНЬ

1) Node.js https://nodejs.org/en/ – офіційна сторінка у мережі Інтернет.
2) React https://reactjs.org/ – офіційна сторінка у мережі Інтернет.
3) MDN JavaScript https://developer.mozilla.org/ru/docs/Web/JavaScript – офіційна документація мови JavaScript.
4) JSON Web Token https://tools.ietf.org/html/rfc7519 – офіційний опис стандарту.
5) WebSocket https://ru.wikipedia.org/wiki/WebSocket – сторінка на Wikipedia.
6) Monaco Editor https://github.com/microsoft/monaco-editor – сторінка на GitHub.
7) Judge0 Web API https://api.judge0.com/ – документація Judge0 API.
8) MongoDB https://www.mongodb.com/ – офіційна сторінка у мережі Інтернет.
9) Socket.IO https://socket.io/docs/ – документація бібліотеки SocketIO.
10) Express.js https://expressjs.com/ – офіційна сторінка у мережі Інтернет.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Платформа для спільної роботи викладача та студента під час іспиту

(Найменування програми (документа))

DVD-R

(Вид носія даних)

11 арк, 560 Кб

(Обсяг програми (документа) , арк.,) Кб)

Server.js

```
const express = require('express');

const app = express();
const morgan = require('morgan');
const routes = require('./app/routes');
const ioServer = require('./app/socket')(app);
const { handle404 } = require('./app/middlewares');

// Middlewares
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(express.static('public'));
app.use(morgan('dev'));
app.use('/', routes);
app.use(handle404);

ioServer.listen(process.env.PORT || 3000);
```

databases/index.js

```
const mongoose = require('mongoose');
const userSchema = require('./schemas/user.js');
const roomSchema = require('./schemas/room.js');
const {
  username,
  password,
  host,
  port,
  name,
} = require('../config').db;
```

```
const dbURI = username && password
  ? `mongodb://${username}:${password}@${host}:${port}/${name}`
  : `mongodb://${host}:${port}/${name}`;

mongoose.connect(dbURI, { useNewUrlParser: true });

mongoose.connection.on('error', (err) => {
  if (err) {
    console.log('database connection error!');
    throw err;
  }
});

mongoose.connection.once('open', () => {
  console.log('successfull database connection!');
});

// mongoose's default promise library is deprecated
mongoose.Promise = global.Promise;

module.exports = {
  models: {
    user: userSchema,
    room: roomSchema,
  },
};
```

Schemas/room.js

```
const mongoose = require('mongoose');

// Each connection object represents a user connected through a
unique socket.
```

```
// Each connection object composed of {userId + socketId}. Both of them together are unique.

const RoomSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  connections: {
    type: [
      {
        userId: String,
        socketId: String,
      },
    ],
  },
});

module.exports = mongoose.model('room', RoomSchema);
```

Schemas/user.js

```
const mongoose = require('mongoose');
const crypto = require('crypto');

const DEFAULT_USER_PICTURE = '/img/user.jpg';

const generateSalt = () => crypto.randomBytes(16).toString('hex');

const hashPassword = (password, salt) => crypto.pbkdf2Sync(password, salt, 2048, 32, 'sha512').toString('hex');


const UserSchema = new mongoose.Schema({
```

```
    username: {
      type: String,
      required: true,
    },
    firstName: {
      type: String,
      default: null,
      required: true,
    },
    secondName: {
      type: String,
      default: null,
      required: true,
    },
    picture: {
      type: String,
      default: DEFAULT_USER_PICTURE,
    },
    password: {
      type: String,
      required: true,
    },
    salt: {
      type: String,
    },
  });

  // using 'function' keyword is a must here to keep 'this' context
  UserSchema.pre('save', function (next) {
    // ensure user picture is set
    if (!this.picture) {
      this.picture = DEFAULT_USER_PICTURE;
    }
    this.salt = generateSalt();
```

```
  this.password = hashPassword(this.password, this.salt);
  next();
});

// using 'function' keyword is a must here to keep 'this' context
UserSchema.methods.validatePassword = function (password) {
  return this.password === hashPassword(password, this.salt);
};

module.exports = mongoose.model('user', UserSchema);
```

Judge0/index.js

```
const axios = require('axios');
const { judge0 } = require('../config');

const getSubmissionToken = async (languageID, code) => {
  const response = await axios.post(judge0.baseURL, {
    language_id: languageID,
    source_code: code,
  }).catch((err) => {
    throw err;
  });
  return response.data.token;
};

const getTokenResult = async (token) => {
  const response = await axios.get(`${judge0.baseURL}/${token}`)
    .catch((err) => {
    throw err;
  });
  // return properties only suitable for interpret languages
  return {
    stdout: response.data.stdout,
```

```
    stderr: response.data.stderr,
  };
};

const interpret = async (languageName, code) => {
  const languageID = judge0.languagesID[languageName];
  const token = await getSubmissionToken(languageID, code);
  const result = await getTokenResult(token);
  console.log(result);
  return result;
};

module.exports.interpret = interpret;
```

Middlewares/index.js

```
const jwt = require('jsonwebtoken');
const { secret } = require('../config');

const checkAuth = (req, res, next) => {
  const token = req.body.token
    || req.query.token
    || req.headers['x-access-token']
    || req.cookies.token;
  console.log('checked token!');
  console.log(token);
  if (token) {
    jwt.verify(token, secret, (err, decoded) => {
      if (err) {
        res.status(401)
          .send('Unauthorized: Invalid token');
      } else {
        req.username = decoded.username;
        next();
```

```
      }
    });
  } else {
    res.status(401).send('Unauthorized: No token provided');
  }
};

module.exports = {
  checkAuth,
};
```

Routes/index.js

```
const express = require('express');
const jwt = require('jsonwebtoken');
const User = require('../models/user');
const Room = require('../models/room');
const judge0 = require('../judge0');
const { secret } = require('../config');
const { checkAuth } = require('../middlewares');

const router = express.Router();

router.get('/check', checkAuth, (req, res) => {
  console.log('/checktoken route!');
  res.sendStatus(200);
});

router.post('/login', async (req, res) => {
  const username = req.body.username;
  const password = req.body.password;
  console.log('ROUTE HANDLER INVOKED');
  // const user = await User.findOne({ username: new
RegExp(username, 'i') })
```

```
        const user = await User.findOne({ username: new RegExp(username,
'i') })
          .catch((error) => {
            res.status(500).json({
              error: 'Internal error, try again later',
            });
            console.error(error);
          });
        if (user) {
          const matches = user.validatePassword(password);
          if (matches) {
            const payload = { username };
            const token = jwt.sign(payload, secret, { expiresIn: '2h' });
            console.log(1);
            res.cookie('token',     token,     {     httpOnly:     true
}).sendStatus(200);
          } else {
            console.log(2);
            res.status(401)
             .json({
              error: 'Incorrect login or password',
            });
          }
        } else {
          console.log(3);
          res.status(401)
           .json({
            error: 'Incorrect login or password',
          });
        }
      });

      router.post('/register', async (req, res) => {
        const { username, password } = req.body;
```

```
  // Check if the username already exists
  const user = await User.findOne({
    username: new RegExp(`^${username}$`, 'i') })
    .catch((error) => {
      // throw error;
      res.status(500).json({
        error: 'Internal error, try again later'
      });
      console.error(error);
    });


  if (user) {
    res.status(409).json({
      error: 'Login already exists'
    });
  } else {
    await User.create({ username, password })
    .catch((error) => {
      res.status(500).json({
        error: 'Internal error, try again later'
      });
      console.error(error);
    });
    res.status(200)
     .send('Your account has been created. Please log in');
  }
});


// add checkAuth middleware?
router.get('/rooms', async (req, res) => {
  const rooms = await Room.find({})
```

```
    .catch((error) => {
      res.status(500).json({
        error: 'Internal error, try again later'
      });
      console.error(error);
    });
    res.status(200).json({ rooms: rooms });
});

router.post('/rooms', async (req, res) => {
  const name = req.body.name;
  const group = req.body.group;
console.log('rooms post!');
});

// add checkAuth middleware?
router.get('/room/:id', checkAuth, async (req, res, next) => {
  const roomId = req.params.id;
  const room = await Room.findById(roomId)
    .catch((error) => {
    res.status(500).json({
      error: 'Internal error, try again later'
    });
    console.error(error);
  });
  if (room) {
    res.status(200)
     .json({ room: room });
  } else {
    return next();
  }
});

router.get('/interpret', async (req, res) => {
```

```
  const lang = req.body.lang;
  const code  = req.body.code;
  const { stdout, stderr } = await judge0.interpret(lang, code);
  // res.send?
  res.status(200).json({
    stdout,
    stderr,
  });
});

module.exports = router;
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПЛАТФОРМА ДЛЯ СПІЛЬНОЇ РОБОТИ ВИКЛАДАЧА ТА СТУДЕНТА ПІД ЧАС ІСПИТУ

Технічне завдання

КПІ.ІП-5226.045440-02-91

“ПОГОДЖЕНО”

Керівник проекту:

_____ О. В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____О.Д. Шателюк

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Платформа для спільної роботи викладача та студента під час іспиту

Галузь застосування:

Наведене технічне завдання поширюється на розробку платформи для спільної роботи викладача та студента під час іспиту, котра використовується для здачі екзаменів з програмування у режимі онлайн та призначена для викладачів та студентів спеціальностей, що пов’язані з інформаційними технологіями.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмних засобів для розробки інтерактивного кіно є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена використання навчальними закладами у навчальому процесі, а саме для викладачів та студентів спеціальностей, що пов’язані з інформаційними технологіями.

Метою розробки є впровадження нового формату здачі іспитів з програмування в онлайн-режимі.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– можливість централізованого контролю над проходженням іспиту для викладача;

– можливість перегляду історії складених іспитів;

– можливість вибору мови програмування для реалізації екзаменаційного завдання для студента;

– наявність зручного інструменту для написання, запуску та перегляду програмного коду;

4.1.2 Розробку виконати на платформі Linux

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

Даний продукт не потребує регулярного обслуговування.

4.3.3 Обслуговуючий персонал

Даний продукт не потребує залучення обслуговуючого персоналу.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

Тип процесору x86_64 або ARM v8 64.

Об‘єм ОЗП 1024 Мб.

1 Гб постійного дискового простору

Клавіатура

ЖК-дісплей

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейств Unix та Windows.

4.5.2 Вхідні дані є набором текстових документів, що являють собою сирцевий програмний код.

4.5.3 Результати представлені у форматі JSON-подібних документів колекцій MongoDB.

4.5.4 Програмне забезпечення повинно надавати графічний інтерфейс користувача та REST API в будь-якому текстовому чи бінарному форматі.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Спеціальні вимоги не пред’являються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Модулі розробленого програмного забезпечення мають бути ретельно задокументовані та мати достатню для розуміння кількість коментарів

5.2 У склад супроводжувальної документації повинні входити наступні документи:

5.2.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.2.2 Технічне завдання.

5.2.3 Керівництво користувача (включається в ПЗ).

5.2.4 Програма та методика тестування.

5.3 Графічна частина повинна бути виконана на 3 листах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.3.1 Схема структурна бізнес процесів.

5.3.2 Схема структурна варіантів використання.

5.3.3 Схема бази даних.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк,	Звітність
1	Розробка технічного завдання	25.03.2019	Технічне завдання
2	Аналіз вимог та уточнення специфікацій	01.04.2019	Специфікації програмного забезпечення
3	Проектування структури програмного забезпечення, проектування компонентів	15.04.2019	Схема структурна варіантів використання, схема структурна баз даних, схема структурна потоків даних
4	Програмна реалізація програмного забезпечення	30.04.2019	Тексти програмного забезпечення
5	Тестування програмного забезпечення	05.05.2019	Тести, результати тестування
6	Розробка матеріалів текстової частини проекту	17.05.2019	Пояснювальна записка.
7	Розробка матеріалів графічної частини проекту	23.05.2019	Графічний матеріал проекту
8	Оформлення технічної документації проекту	23.05.2019	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування готових програмних засобів має бути виконано згідно документа «Програми та методики тестування».

.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПЛАТФОРМА ДЛЯ СПІЛЬНОЇ РОБОТИ ВИКЛАДАЧА ТА СТУДЕНТА ПІД ЧАС ІСПИТУ

Керівництво користувача

КПІ.ІП-5226.045440.03.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____О.Д. Шателюк

Київ – 2019 року

Після того, як користувач відкриє відповідний URL у своєму веб-браузері, він отримує доступ до усього функціоналу додатку. Користувач не має потреби встановлювати додаткове програмне забезпечення для роботи з продуктом. Спочатку система демонструє сторінку регістрації.

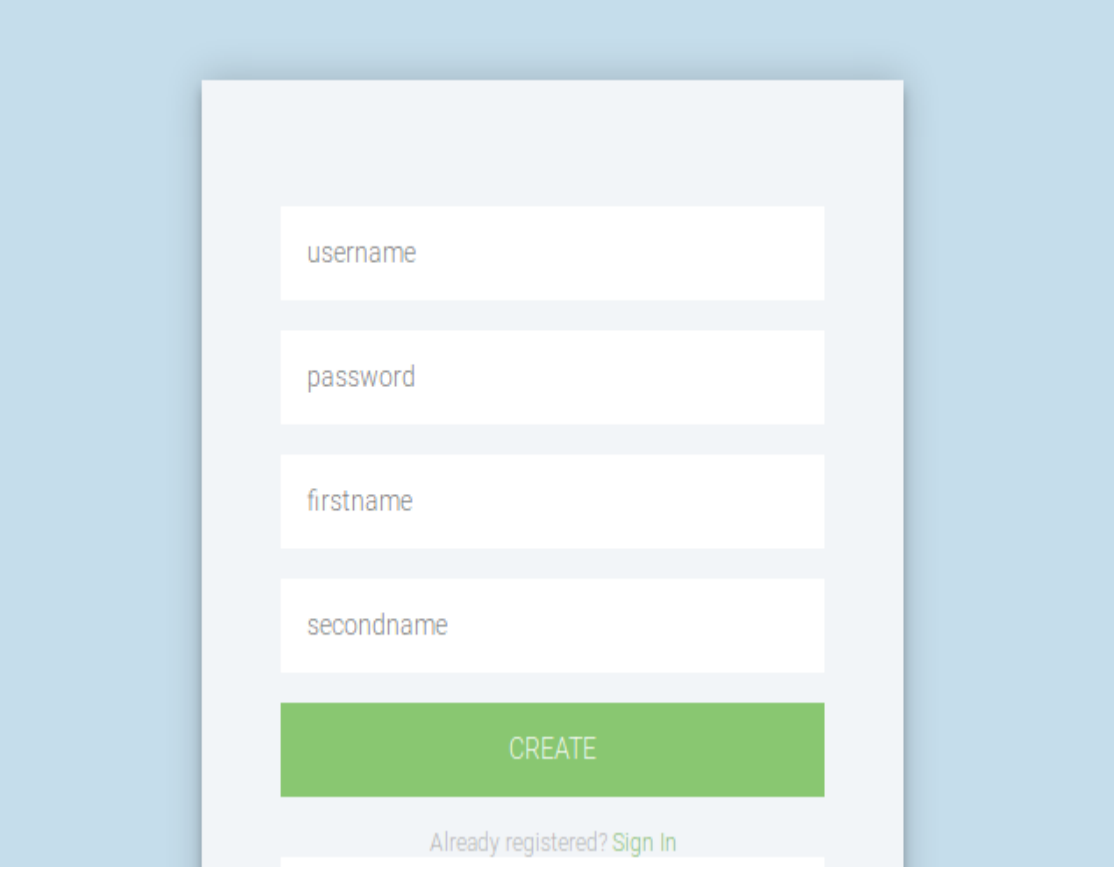

Рисунок 1.1 – Сторінка реєстрації

У цьому вікні користувач має можливість зареєструватися, якщо ще не має акаунту. Відповідні текстові поля мають бути валідними рядками, пароль містити не менш ніж 6 символів.

По натисканню на кнопку «Create» система створить новий аккаунт користувача та, за відсутності помилок, перенаправить його на сторінку авторизації, де потрібно буде ввести необхідні дані для входу. Додаток сповістить користувача відповідними повідомленнями у разі виникнення помилки реєстрації.

По натисканню на кнопку «Sign in» система одразу перенаправить користувача на сторінку авторизації. Сторінка авторизації зображена на рисунку 1.2.

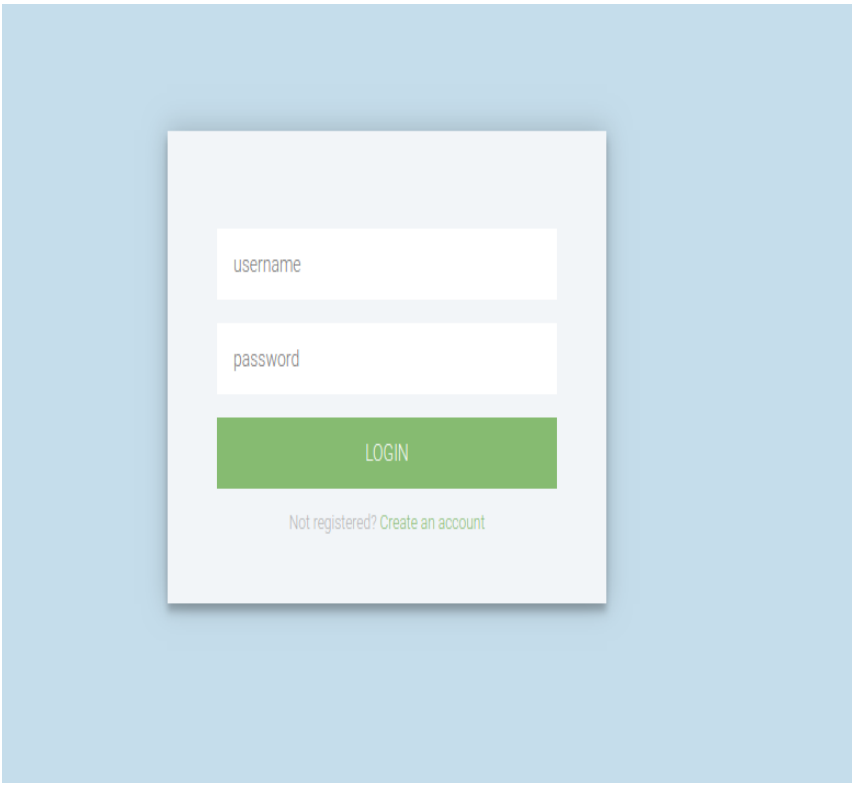

Рисунок 1.2 – Сторінка авторизації

У відповідних полях користувачу потрібно ввести дані про свій обліковий запис. По натисканню на кнопку «Login» система спробує авторизувати користувача, та, за відсутності помилок, перенаправить його на сторінку з переліком усіх активних екзаменаційних кімнат. Додаток сповістить користувача відповідними повідомленнями у разі виникнення помилки авторизації. По натисканню на кнопку «Create an account» користувач буде перенаправлений на сторінку реєстрації. Сторінка, що буде відображена після авторизації зображена на рисунку 1.3.

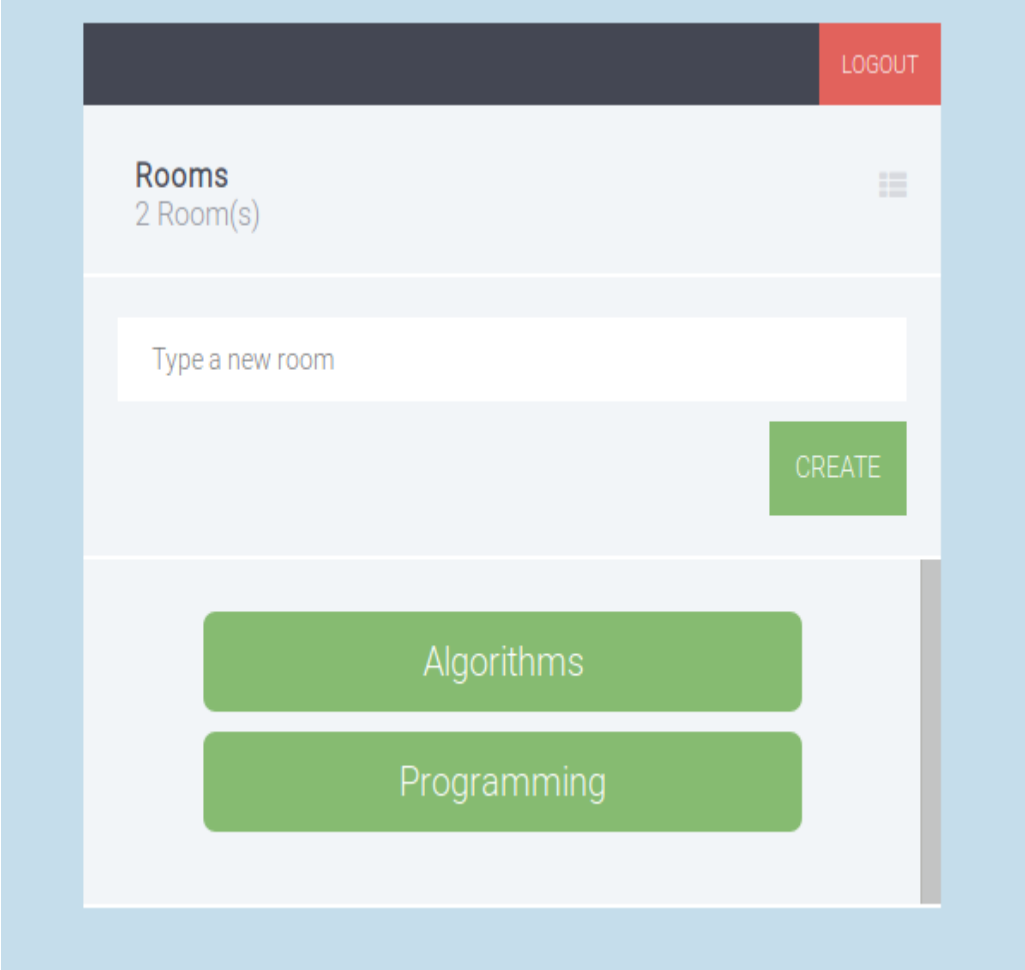

Рисунок 1.3 – Меню додатку

Після успішної авторизації користувач потрапить на сторінку зі списком активних екзаменаційних кімнат. Для того, щоб створити нову кімнату, потрібно ввести дані про неї у відповідні текстові поля та натиснути кнопку «Create». У верхній частині сторінки є блоки з зображенням загальної інформації про активні кімнати, наприклад, про їх кількість. По натисканню на кнопку «Logout» користувача буде перенаправлено на сторінку авторизації. По натисканню на кнопку з назвою кімнати, користувач буде підключений до відповідної кімнати. Коли користувач потрапляє на сторінку екзаменаційної кімнати, він може переглянути інформацію про підключених користувачів у відповідному блоці на правій частині сторінки. У ньому відображаються користувачі у вигляді списку. У верхній частині наявні кнопки «Rooms» та «Logout».

По натисканню на кнопку «Rooms» користувач повернеться на попередню сторінку, де зможе підключитися до іншої екзаменаційної кімнати. По натисканню на кнопку «Logout» користувач вийде з системи. На рисунку 1.4 зображений блок з інформацією про підключення екзаменаційної кімнати.

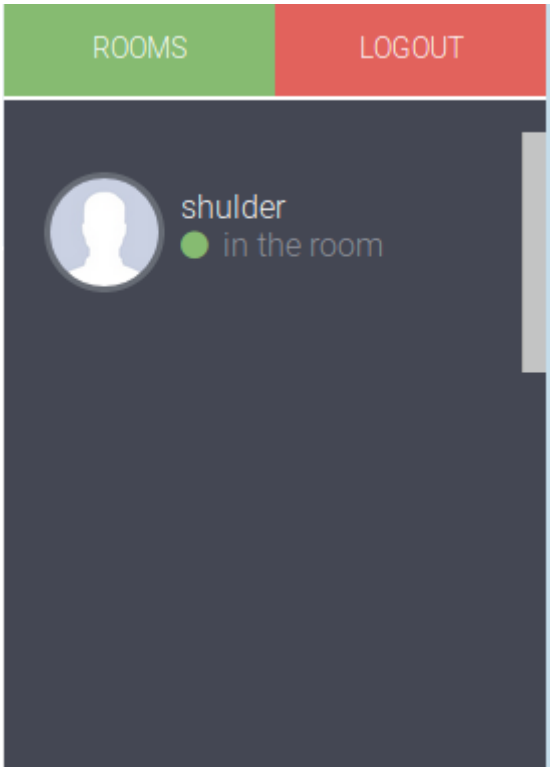

Рисунок 1.4 – Список підключених користувачів

На сторінці екзаменаційної кімнати, він може взаємодіяти з текстовим редактором для написання коду у лівій частині сторінки. У верхній частині текстового редактору можно побачити інформацію про кімнату: кількість підключених користувачів та її назву. Достатньо лише почати написання сирцевого коду у відповідному вікні, а по завершенню реалізації натиснути на кнопку «Run code».

Рисунок 1.5 – Текстовий редактор для коду

Якщо користувач натискає на кнопку запуску коду, результат виконання виводиться у відповідних текстових блоках. У блоці «STDOUT» знаходиться результат виконання коду, а у блоці «STDERR» - повідомлення про помилки. Відповідні текстові блоки зображені на рисунку 1.6.

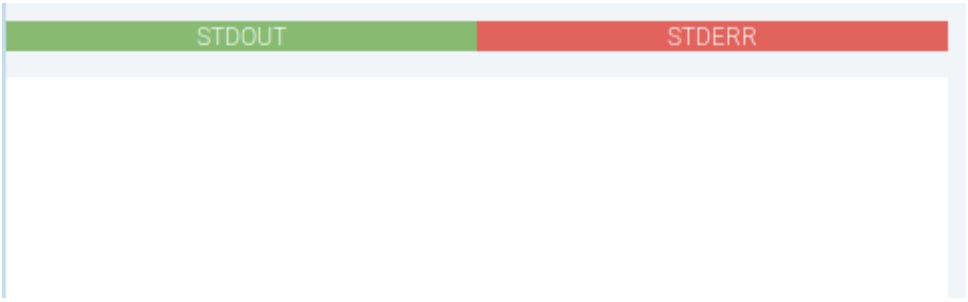

Рисунок 1.6 – Вікно чату

Для виходу з системи користувачу треба натиснути кнопку «Logout» у правій верхній частині сторінки.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПЛАТФОРМА ДЛЯ СПІЛЬНОЇ РОБОТИ ВИКЛАДАЧА ТА СТУДЕНТА ПІД ЧАС ІСПИТУ

Програма та методика тестування

КПІ.ІП-5226.045440.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____О.Д. Шателюк

Київ – 2019 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом тестових випробувань є платформа для спільної роботи викладача та студента під час іспиту з програмування, створена як веб-додаток за допомогою фреймворку Express.js, бібліотек React та Socket.io, з використанням сховища Redis, бази даних MongoDB, сервісу Judge0 та плагіну MonacoEditor.

2 МЕТА ТЕСТУВАННЯ

Процес тестування полягає у перевірці програмних засобів за наступними критеріями:

– повна та цілісна інтуітивна робота кожної сторінки веб-додатку;

– швидкий та якісний метод інтерпритації та компіляції коду;

– забезпечення стабільної взаємодії між студентами та викладачем у режимі реального часу;

– наочність та швидкодія роботи текстового редактору;

– належний та безпечний спосіб аутентифікації;

– відповідність програмних засобів до технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування було виконане методом закритої коробки, таким чином перевіряючи як код, так і програмні засоби на повну відповідність функціональним та іншим вимогам до програмних засобів.

Використовуються наступні методи тестування:

- компонентне тестування;
- інтеграційне тестування;
- швидкодійністне тестування.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується інструментаріями:

– assert (стандартний модуль Node.js);

– Locust.io (https://locust.io/) – для тестування швидкодії;

– Postman.

Надійність веб-ресурсу перевіряється методами:

– ручного тестування у відповідності до функціональних вимог;

– ручного тестування валідації;

– юніт-тестування коду;

– тестування зовнішнього вигляду та загального функціоналу у різних веб-переглядачах;

– тестування навантаженням;

– UX-тестування.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПЛАТФОРМА ДЛЯ СПІЛЬНОЇ РОБОТИ ВИКЛАДАЧА ТА СТУДЕНТА ПІД ЧАС ІСПИТУ

Графічний матеріал

КПІ.ІП-5226.045440-05-99

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____О.Д. Шателюк

Київ – 2019 року