

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ (підпис)

Едуард ЖАРИКОВ
(ім'я прізвище)

“ _____ ”

2023 р.

Дипломний проект
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-застосунок для моніторингу та обробки результатів тестувань
програмного забезпечення

Виконав студент IV курсу, групи ІІІ-91
(шифр групи)

Кочев Геннадій Геннадійович
(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник доцент, к.т.н., доц., Ліщук К. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

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

_____ (підпис)

Рецензент доцент, к.т.н., доц., Писаренко А.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ –2023

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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри

Едуард ЖАРИКОВ

(підпис)

(ім'я прізвище)

“ ____ ”

2023 р.

ЗАВДАННЯ
на дипломний проект студента

Кочеву Геннадію Геннадійовичу

(прізвище, ім'я, по батькові)

1. Тема проєкту Веб-застосунок для моніторингу та обробки результатів тестувань програмного забезпечення

керівник проєкту Лішук Катерина Ігорівна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснівальної записки

1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.

2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.

3) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.

4) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв'язання задачі	25.03.2023	
3	Постановка та формалізація задачі	28.03.2023	
4	Розробка інформаційного забезпечення	04.04.2023	
5	Алгоритмізація задачі	10.04.2023	
6	Обґрунтування вибору використаних технічних засобів	15.04.2023	
7	Розробка програмного забезпечення	15.05.2023	
8	Налагодження програми	18.05.2023	
9	Виконання графічних документів	23.05.2023	
10	Оформлення пояснівальної записки	29.05.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент

(підпис)

Геннадій КОЧЕВ

(ініціали, прізвище)

Керівник

(підпис)

Катерина ЛІЩУК

(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 43 таблиці, 17 рисунків, 12 джерел та 1 додаток – загалом 67 сторінок.

Мета дипломного проекту: оптимізація робочого процесу QA-інженерів під час тестування програмного забезпечення.

У першому розділі наведено загальні положення, змістовний опис та аналіз обраної предметної області. Проведено аналіз засобів розробки та відомих технічних рішень, присутній також порівняльний аналіз проекту з вже запровадженими програмними продуктами. На основі дослідження було сформовано функціональні та нефункціональні вимоги до застосунку, наведено призначення та задачі розробки.

Другий розділ містить опис моделювання та конструювання програмного забезпечення. У розділі присутні детальний опис архітектури веб-застосунку, опис сутностей бази даних, опис найважливіших методів та класів ПЗ. Проведено аналіз безпеки даних у створюваному застосунку.

Третій розділ присвячено опису аналізу якості програми, опису етапів тестування та опису контрольного прикладу.

У четвертому розділі деталізовано процес розгортання програмного забезпечення. Описано також процес підтримки програмного забезпечення та доставки нових версій до кінцевого користувача.

Також наведено: інструкцію користувача, програму та методику тестування, схему структурну варіантів використання, схему бази даних, креслення вигляду екранних форм.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ТЕСТУВАННЯ, КЛІЄНТ-СЕРВЕР, REACT.JS, FLASK, POSTGRESQL, БАЗА ДАНИХ, MESSAGE QUEUE, PDF-ЗВІТИ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 43 tables, 17 figures, 12 sources and 1 appendix – in total 67 pages.

The purpose of the diploma project is optimization of the workflow of QA engineers during software testing.

The first section contains general provisions, a meaningful description and analysis of the selected subject area. An analysis of development tools and known technical solutions was carried out, there is also a comparative analysis of the project with already implemented software products. Functional and non-functional requirements for the application were formed on the basis of the study, the purpose and tasks of the development were given.

The second section contains a description of software modeling and construction. The section contains a detailed description of the architecture of the web application, a description of the database entities, a description of the most important methods and software classes. An analysis of data security in the application being created was carried out.

The third section is devoted to the description of the analysis of the quality of the program, the description of the stages of testing and the description of the control example.

The fourth section details the software deployment process. The process of software support and delivery of new versions to the end user is also described.

Also provided: user manual, program and testing methodology, structural diagram of use cases, database diagram, drawing of screen forms.

KEYWORDS: WEB APPLICATION, TESTING, CLIENT-SERVER, REACT.JS, FLASK, POSTGRESQL, DATABASE, MESSAGE QUEUE, PDF REPORTS.

					КПІ.ІП-9113.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.	Кочев Г.Г.					Літ.	Аркуш
Перевір.	Ліщук К.І.					1	Аркушів
					Відомість дипломного проекту		
Н.контр.	Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-91		
Затв.	Жаріков Е.В.						

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРИКОВ

“ ____ ” _____ 2023 р.

**ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ
РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Технічне завдання

КП.ІП-9113.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Геннадій КОЧЕВ

Київ – 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
4.1	Вимоги до функціональних характеристик	6
4.1.1	Користувацького інтерфейсу.....	6
4.1.2	Для користувача:.....	9
4.2	Вимоги до надійності	10
4.3	Умови експлуатації.....	11
4.3.1	Вид обслуговування	11
4.3.2	Обслуговуючий персонал	11
4.4	Вимоги до складу і параметрів технічних засобів	11
4.5	Вимоги до інформаційної та програмної сумісності	11
4.5.1	Вимоги до вхідних даних.....	12
4.5.2	Вимоги до вихідних даних	12
4.5.3	Вимоги до мови розробки.....	12
4.5.4	Вимоги до середовища розробки	13
4.5.5	Вимоги до представленню вихідних кодів	13
4.6	Вимоги до маркування та пакування	13
4.7	Вимоги до транспортування та зберігання	13
4.8	Спеціальні вимоги	13
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	14
5.1	Попередній склад програмної документації	14
5.2	Спеціальні вимоги до програмної документації	14
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ	15
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	16

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

Галузь застосування: QA-інженерія у програмному забезпеченні.

Наведене технічне завдання поширюється на розробку програмного забезпечення веб-застосунку для моніторингу та обробки результатів тестування програмного забезпечення КПІ.ІП-9113.045440.01.91, котра використовується для створення тест-кейсів, їх групування у тест-сьюти, за якими проводяться тест-рани, та для автоматичної генерації звітів у форматі PDF та призначена для QA-інженерів, що займаються перевіркою програмного забезпечення як у великих IT-компаніях, так і у меншого розміру проектах.

Змін.	Арк.	№ докум.	Підп.	Дата.	KPI.IP-9113.045440.01.91

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для моніторингу та обробки результатів тестування програмного забезпечення є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

						Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.01.91	4

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації роботи тестувальника, а саме моніторингу та обробки результатів тестувань програмного забезпечення.

Метою розробки є оптимізація роботи тестувальника за рахунок надання можливості групувати тест-кейси у тест-сьюти, запускати тест-рани за створеними тест-сьютами та автоматизовано звітувати за результатами виконання тест-рану, що надасть можливість виконувати його завдання швидше та ефективніше.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.01.91

Арк.

5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

- наявність сторінки реєстрації;

Create an account

First Name * _____

Last Name * _____

Email Address * _____

Password * _____

I agree to the Rules of Test Result App usage.

SIGN UP

Рисунок 4.1 – Прототип екранної форми сторінки реєстрації

- наявність сторінки логіну до аккаунту;

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Sign in

Рисунок 4.2 – Прототип екранної форми сторінки логіну

- наявність сторінки з проектами;

Active project: Welcome

TEST RESULT APP

PROJECTS TEST CASES TEST SUITES TEST RUNS

Projects +

Project Name	Description	Is Active	Change state
Project A	Description A	Active	MAKE ACTIVE
Project B	Description B	Inactive	MAKE ACTIVE

Рисунок 4.3 – Прототип екранної форми сторінки проєктів

- наявність сторінки з тест-кейсами;

						Арк.
Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.01.91	7

Active project: Welcome

TEST RESULT APP

PROJECTS TEST CASES TEST SUITES TEST RUNS

Test Cases

Creating New Item

Pre-Conditions	Test Steps	Post-Conditions

Рисунок 4.4 – Прототип екранної форми сторінки тест-кейсів

- наявність сторінки з тест-сьютами;

The screenshot shows the 'Test Result APP' interface. At the top, there's a green header bar with 'Active project:' on the left and 'Welcome' on the right. Below it is a blue navigation bar with the app logo, 'TEST RESULT APP', and links for 'PROJECTS', 'TEST CASES', 'TEST SUITES', and 'TEST RUNS'. On the far right of the blue bar is a user profile icon. The main content area has a white background. It displays two sections: 'Test Suites' and 'Test Case Details'. The 'Test Suites' section contains two collapsed items, 'Test suite 1' and 'Test suite 2'. Each suite has a '+' button to its right. The 'Test Case Details' section is expanded for 'Test suite 1', showing four cards arranged in a 2x2 grid. Each card has 'Case:' and 'About:' fields. To the right of this grid are three icons: a blue pencil for 'CASE+', a blue play button for execution, and a blue trash can for deletion. A similar set of icons is located at the bottom right of the page.

Test Suites

Test suite 1

Case:
About:

Case:
About:

Test suite 2

Case:
About:

Case:
About:

CASE+

CASE+

Рисунок 4.5 – Прототип екранної форми сторінки тест-сьютів

КПІ.ІП-9113.045440.01.91

Арк.

8

- наявність сторінки з тест-ранами.

The screenshot shows a web application interface for a 'TEST RESULT APP'. At the top, there's a green header bar with 'Active project:' and 'Welcome' on the right. Below it is a blue navigation bar with the app logo, 'TEST RESULT APP', and links for 'PROJECTS', 'TEST CASES', 'TEST SUITES', and 'TEST RUNS'. A user profile icon is also in the top right. The main content area has a title 'Test Runs' with a '+' button. It displays two separate sections for 'Run for at 2023-05-01' and 'Run for at 2023-05-02'. Each section shows a table with columns 'Test Case Name', 'Test Case Description', and 'Status'. The first row in both tables has a 'PDF' export icon. The first table has one row with data: 'Test case 1 name' (Test Case Name), 'Some Description' (Test Case Description), and 'Successful' (Status). The second table has no visible rows.

Рисунок 4.6 – Прототип екранної форми сторінки тест-ранів

4.1.2 Для користувача:

- можливість зареєструватися;
- можливість увійти в створений аккаунт;
- можливість створити проект;
- можливість створити тест-кейс;
- можливість створити тест-сьют;
- можливість додати створені тест-кейси до тест-сьюту;
- можливість провести тест-ран за створеним тест-сьютом;
- можливість експортувати результат тест-рану до PDF-звіту.

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9113.045440.01.91

Арк.

9

Test Run Report

Test Suite

About suite

Date

Result of Run

Test Cases List

1

Case Name

Case Description

Status

Pre-Conditions

Test Steps

Post-Conditions

2

Case Name

Case Description

Status

Pre-Conditions

Test Steps

Post-Conditions

Рисунок 4.7 – Прототип екранної форми PDF-звіту

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних, захист від SQL-ін'єкцій для безпеки даних.

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9113.045440.01.91

Арк.

10

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах з підтримкою сучасних браузерів (Chrome, Firefox, Edge, Safari та інші).

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i3;
- об'єм ОЗП: 2 Гб;
- підключення до мережі Інтернет зі швидкістю від 1 мегабіт;

Рекомендована конфігурація технічних засобів, на якій виконувалась розробка:

- тип процесору: Intel Core i7;
- об'єм ОЗП: 8 Гб;
- підключення до мережі Інтернет зі швидкістю від 50 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows XP, Windows NT і т.д.) або Unix.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.01.91

Арк.

11

4.5.1 Вимоги до вхідних даних

Вхідні дані, що може ввести користувач у веб-застосунок, є повністю текстовими та можуть бути будь-якого змісту в залежності від потреб користувача. Такими даними є:

- назва, опис проєкту;
- назва, опис, передумови, кроки проведення, постумови тест-кейсу;
- назва, опис тест-сьюту;
- результати тест-рану та результати для кожного тест-кейсу в ньому.

4.5.2 Вимоги до вихідних даних

Результати, що надає веб-застосунок, користувач бачить або у браузері на відповідній сторінці, або у звіті, що було згенеровано. Таким чином, маємо дані такого формату:

- таблиця з переліком проектів;
- список тест-кейсів з drop-down меню для перегляду деталей кейса;
- список тест-сьютів з drop-down меню для перегляду тест-кейсів, що до нього входять;
- список тест-ранів з результатом та таблицею результатів для кожного тест-кейсу у drop-down меню;
- звіт у форматі PDF, що відповідає даним на сторінці тест-ранів для обраного тест-рану.

4.5.3 Вимоги до мови розробки

Розробку виконати мовами програмування Python (для серверної частини коду) та JavaScript (для клієнтської частини коду).

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Linux за допомогою середовищ JetBrains PyCharm (для серверної частини коду) та JetBrains WebStorm (для клієнтської частини коду).

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді двох репозиторіїв – серверної та клієнтської частин коду.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальних вимог не висувається.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.01.91

Арк.

13

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема бази даних;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КП.ІП-9113.045440.01.91

Арк.

14

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.03	
2.	Розробка технічного завдання	28.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	30.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	10.04	Схема структурна програмного забезпечення та специфікація компонентів
5.	Програмна реалізація програмного забезпечення	15.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	18.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	26.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	28.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до документу КПІ.ІП-9113.045440.04.51 “Програма та методика тестування”.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.01.91

Арк.

16

Пояснювальна записка до дипломного проекту

на тему: Веб-застосунок для моніторингу та обробки результатів тестувань
програмного забезпечення

КПІ.ІП-9113.045440.02.81

Київ – 2023

ЗМІСТ

ВСТУП 5

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	6
1.1 Загальні положення.....	6
1.2 Змістовний опис і аналіз предметної області	7
1.3 Аналіз існуючих технологій та успішних ІТ-проєктів.....	8
1.3.1 Аналіз відомих алгоритмічних та технічних рішень	8
1.3.2 Аналіз допоміжних програмних засобів та засобів розробки	9
1.3.3 Аналіз відомих програмних продуктів	10
1.4 Аналіз вимог до програмного забезпечення.....	11
1.4.1 Розроблення функціональних вимог.....	20
1.4.2 Розроблення нефункціональних вимог	24
1.5 Постановка задачі	24
Висновки до розділу	25

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

27

2.1 Моделювання та аналіз програмного забезпечення.....	27
2.2 Архітектура програмного забезпечення	28
2.3 Конструювання програмного забезпечення	30
2.4 Аналіз безпеки даних.....	41
Висновки до розділу	41

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 43

3.1 Аналіз якості ПЗ	43
3.2 Опис процесів тестування	44
3.3 Опис контролального прикладу.....	52
Висновки до розділу	59
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .	60
4.1 Розгортання програмного забезпечення	60

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.02.81

Арк.
2

4.2 Підтримка програмного забезпечення	61
Висновки до розділу	62
ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	65
ДОДАТКИ	67
ДОДАТОК А. ЗВІТ ПРО ПЕРЕВІРКУ НА СПІВПАДІНЯ	67

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.02.81

Арк.

3

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT	– Інформаційні технології
API	– Application Programming Interface, прикладний програмний інтерфейс
IDE	– Integrated Development Environment – інтегроване середовище розробки
ПЗ	– Програмне забезпечення
QA/QC	– Quality Assurance/Quality Control
UI/UX	– User Interface/User Experience
SQL	– Structured Query Language
HTTP	– HyperText Transfer Protocol
REST	– Representational State Transfer
ORM	– Object-Relational Mapping
PDF	– Portable Document Format
РСУБД	– Реляційна система управління базами даних
ERD	– Entity Relationship Diagram
CSRF	– Cross-Site Request Forgery
LTS	– Long-Term Support
PK/FK	– Primary Key/Foreign Key

Змін.	Арк.	№ докум.	Підп.	Дата.

ВСТУП

У сучасному світі стрімко набирає популярність використання цифрових технологій для вирішення задач, автоматизації рутинних справ та створення зручного та продуктивного середовища для ефективної праці людей. Маючи таку тенденцію, кількість програмних застосунків, утиліт, веб-сайтів постійно зростає.

У кожному нормальному циклі розробки програмного забезпечення присутня важлива частина - тестування, автоматизоване або мануальне. Оскільки тестувати доводиться все більше і більше функціоналу, виникає потреба у засобах для збереження та обробки результатів перевірок функціоналу ПЗ.

У сфері ІТ стає більш розвиненим напрям QA/QC Engineering[1, с.52], потреба у спеціалістах з тестування програмного забезпечення неуклінно зростає, і для їх роботи потрібні нові software-рішення, що дозволять збільшити продуктивність праці.

На даний момент існує не такий широкий вибір застосунків, що дозволяють відстежувати стан перевірки коректності роботи ПЗ, а тим паче формувати автоматичні звіти щодо фінальних результатів та статистики перевірок. Більшість таких програм не мають за собою конкретної направленості для використання Quality Assurance інженерами. Саме через актуальність такої проблеми для тестувальників було прийнято рішення про виконання проекту, що дозволить у форматі веб-застосунку зручно моніторити, аналізувати та обробляти результати тестів, а також генерувати репорти для звітності.

Серед сфер застосування такого застосунку можна розглядати будь-яку з ділянок ІТ-індустрії, оскільки майже для будь-якого програмного продукту, веб-сайту, консольної утиліти потрібно проводити як мінімум мануальне тестування, а найчастіше також автоматизоване, інтеграційне, та інші види тестів.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Почнемо з базового огляду предметної області - Software Testing.

Тестування програмного забезпечення - це процес перевірки програмного забезпечення на відповідність вимогам, виявлення помилок та недоліків, а також перевірка, що програмне забезпечення працює згідно з очікуваннями користувачів. Тестування зазвичай проводиться спеціалістами з QA (Quality Assurance) або тестерами. Під час тестування вони перевіряють різні аспекти програмного забезпечення, включаючи його функціональність, продуктивність, надійність, безпеку, зручність використання та інші.

Тестування програмного забезпечення почалося приблизно в 1940-х роках[2, с. 34], коли були створені перші електромеханічні комп'ютери. Програми перевірялися вручну, без використання спеціальних інструментів чи методологій. У 1960-х роках з'явилися перші методики, які включали в себе процес планування тестування та оцінку його результатів. У 1980-х роках з'явилися перші спеціальні програмні засоби для тестування, такі як фреймворки для тестування інтерфейсів користувача та інструменти для автоматизації тестування. По наш час тестування програмного забезпечення продовжує розвиватися та вдосконалюватися.

Тестування ПЗ може проводитися на різних етапах життєвого циклу програмного забезпечення, включаючи ранні етапи розробки, перед випуском продукту, а також після випуску для перевірки нових функцій та виправлення виявлених помилок. Мета таких перевірок полягає в тому, щоб забезпечити якість та надійність програмного забезпечення, зменшити кількість помилок та недоліків, а також забезпечити високий рівень задоволеності користувачів.

З основної термінології QA/QC-спеціалісти найчастіше використовують такі поняття, як Test Plan, Test Case, Test Suite[3, с. 3]. Зупинимось детальніше на кожному з них.

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9113.045440.02.81

Арк.

6

Тест-план - це документ, що декларує весь об'єм робіт по тестуванню, об'єкти перевірок, критеріїв початку та кінця перевірки, а також обладнання для проведення тестів. Тест-план є важливою частиною організованого процесу тестування.

Тест-кейс - це тестовий артефакт, суть якого полягає у виконанні деякої кількості дій або умов, що необхідні для перевірки визначеної функціональності програмної системи. Структура його складається з виконуемої дії (Action), очікуваного результату (Expected Result), та фактичного результату (Test Result). Сам тест-кейс складається трьох частин: передумови (дії, що призводять систему в належний для тестування стан), опису тест-кейсу (дії, за допомогою яких відбувається основна перевірка функціоналу), постумови (дії, які повертають систему до початкового стану).

Тест-сьют - це набір тест-кейсів, які об'єднані тим, що відносяться до одного тестованого модуля, функціональності, пріоритету, або одного типу тестування. Тест-сьюти забезпечують більшу зручність при проходженні тест-кейсів, для послідовної та коректної перевірки, щоб усунути неперевірені місця та недоліки тестування.

1.2 Змістовний опис і аналіз предметної області

На поточний момент розвитку ІТ-технологій тестування програмного забезпечення є дуже поширеним як у великих компаніях, так і у стартапів, пет-проектів та іншого. Загальним принципом тестування є:

- 1) побудова базового плану тестування;
- 2) формування тест-кейсів (назва, короткий опис, передумова, кроки для виконання, очікуваний результат);
- 3) розміщення тест-кейсів у тест-сьютах для логічного розділення процесу тестування;
- 4) прогін тест-сьютів, отримання результатів - Test Run;
- 5) формування звіту щодо успішності проведених робіт по тестуванню.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Саме такий процес QA-спеціалісти проводять ітеративно для забезпечення коректних перевірок різних груп функціоналу застосунків. Часто для ведення обліку результатів використовують непризначене для цього ПЗ, що сповільнює роботу тестувальників. Веб-застосунок, що стане результатом дипломного проекту, може покращити поточну ситуацію, оскільки він підпорядковується загальному принципу тестування, описаному вище. Спеціаліст з тестування матиме можливість провести названі кроки у одному застосунку, що підвищить продуктивність та комфорт його роботи.

1.3 Аналіз існуючих технологій та успішних IT-проектів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для моніторингу та обробки результатів тестувань програмного забезпечення. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Найчастіше веб-застосунки[4, с. 92] складаються з таких складових частин:

- сховище даних (SQL або NoSQL рішення);
- backend сервер (HTTP RESTful API);
- frontend сервер (веб-сторінка, на яку заходить користувач);
- message queue (черга для виконання однотипних задач).

Скористаємося саме цим відомим підходом до розробки та складемо загальну схему вигляду програмного забезпечення цього проекту.

На рисунку нижче представлена загальна схема веб-застосунку:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						8

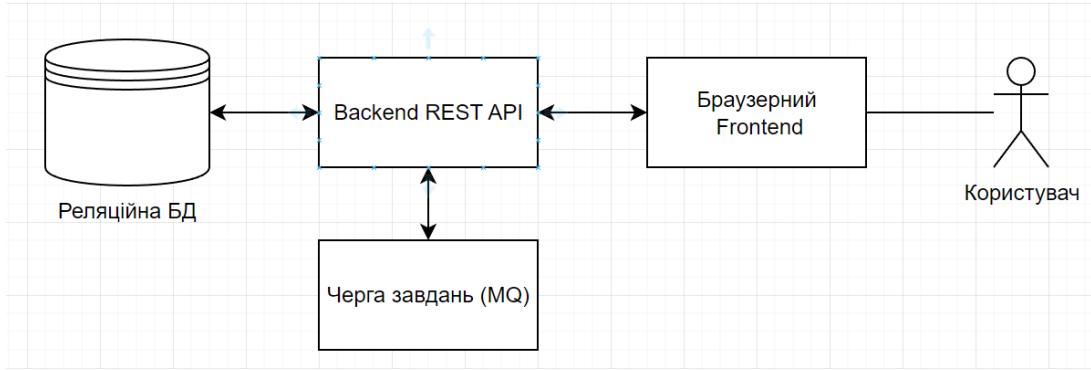


Рисунок 1.1 – Загальна конструкція веб-застосунку

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Основою реалізації проекту є мова програмування Python. Це високорівнева мова програмування загального призначення з динамічною суворою типізацією та автоматичним керуванням пам'яттю, що орієнтована на підвищення продуктивності розробника, читабельності коду та його якості [5, с. 112]. Python є кросплатформенною мовою, що дозволяє розгорнати додатки майже будь-де (Windows, Mac OS, Linux).

Для створення веб-застосунку на базі Python зазвичай використовують спеціалізовані фреймворки, найбільш популярними є Django, Flask, FastAPI. Порівняємо їх та оберемо найбільш підходящий для нашого проекту.

Django є веб-фреймворком, в якому вбудовано дуже багато на початку, як ORM, конфігурація middlewares та інше, через це фреймворк негнучкий та вимагає підлаштовуватися під його правила. FastAPI - це доволі новий веб-фреймворк, який знаходиться в стані розробки, тому спільнота, що його використовує, не дуже розвинена у кількості, через що можуть виникати складності у розробці. На думку автора роботи, оптимальним рішенням буде застосувати Flask - мінімалістичний веб-фреймворк, що легко налаштовується та є гнучким до змін, водночас надаючи розробнику типові потреби, як ORM, кешування, аутентифікація. Flask є доволі простим у використанні, що надає розробнику можливість навчатися його використанню у процесі розробки для кращого його розуміння.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

З вибору ORM для роботи з базою даних маємо такі варіанти: Django ORM, SQLAlchemy, Peewee. Перший варіант не підходить для проєкту, оскільки обрано Flask. Peewee - доволі нове рішення, тому мало розробників його використовують, а також є обмеження на використання лише декількох відомих СУБД, що зробить проєкт негнучким. Найкращим вибором у такому випадку є SQLAlchemy - ORM, що добре зарекомендувала себе у спільноті Python, та що використовується найчастіше, тому з пошуком інформації проблем не виникне. Перевагами SQLAlchemy також є простота використання, швидкість виконання запитів, та можливість роботи з БД на високому рівні, та низькому (рівень SQL-запитів).

Для генерування звітів - регулярної задачі веб-застосунку, прийнято рішення про використання Celery - розподіленої системи для обробки однотипних задач у реальному часі задля того, щоб оптимізувати навантаження на застосунок.

Графічний інтерфейс користувача буде у браузері, для його реалізації потрібно задіяти окремий фронтенд-сервер, який буде робити запити на бекенд задля отримання потрібної для користувача інформації. Найбільш зручним рішенням для розробки такого є бібліотека React.js, створена компанією Facebook. Ця бібліотека дозволяє створювати графічні інтерфейси, які відповідають світовим стандартам UI/UX, а також підтримувати високу якість та структурованість коду, що дає можливість підтримувати проєкт протягом часу.

1.3.3 Аналіз відомих програмних продуктів

За предметною областю тестування програмного забезпечення вже існують деякі software-рішення. Такими є Kualitee та TestRail, обидва ці застосунки дозволяють проводити менеджмент тест-кейсів, Kualitee також допомагає QA-інженерам в плані баг-трекінгу[6, с. 4], а TestRail дозволяє організовувати тест-плани та результати тестувань.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Для порівняння дипломного проекту з аналогами можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	Дипломний проект	Kualitee	TestRail	Пояснення
Збереження тест-кейсів, тест-сьютів	+	+	+	В усіх додатках можливо створювати та зберігати тест-кейси та тест-сьюти.
Проведення тест-ранів на базі тест-сьютів	+	-	+	Зі створених тест-сьютів надається можливість провести тест-ран у проєкті, подібний функціонал є у TestRail.
Експорт результатів тест-ранів до PDF	+	+	-	Можливість зберігати результат проходу тест-рану є у проєкті та у Kualitee.
Графіки, дашборди	-	+	+	Побудова автоматичних дашбордів відсутня у проєкті, але наявна у аналогах.
Темна тема застосунку	-	-	+	Dark mode присутній лише у нових версіях TestRail, у проєкті на даний час відсутній.

1.4 Аналіз вимог до програмного забезпечення

Перш за все, функціонал розробки має під собою реєстрацію користувача, вход в аккаунт, вихід з аккаунту, налаштування аккаунта. Користувачу надається можливість створювати проекти, до яких додаються

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						11

тест-кейси з налаштуваннями. Основні функції програмного забезпечення - можливість групувати тест-кейси у тест-сьюти, проводити Test Runs на базі тест-сьютів, переглядати історію Test Runs та генерувати автоматичні звіти у форматі PDF за результатами Test Run.

Детальніше функціонал описано на схемі структурній варіантів використань, що представлено у документі КПІ.ІП-9113.045440.06.99.CCB.

В таблицях 1.2 – 1.12 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися для початку роботи із застосунком
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: ім'я та прізвище, пошта користувача, пароль в системі та його повтор для підтвердження. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з'являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу.

Продовження таблиці 1.2

Extension	В випадку введення некоректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним.
Post-condition	Створення аккаунту користувача, перехід на сторінку входу

Таблиця 1.3 – Варіант використання UC-02

Use case name	Вхід користувача до аккаунту
Use case ID	UC-02
Goals	Вхід користувача до системи, початок користування
Actors	Зареєстрований користувач
Trigger	Зареєстрований користувач бажає почати користуватись застосунком
Pre-conditions	Користувач є зареєстрованим, до застосунку вхід не виконано
Flow of Events	Користувач переходить на сторінку входу. В поля вводить пошту та пароль. Після заповнення натискає кнопку входу, при введених коректних даних перехід на сторінку з проектами.
Extension	В випадку введення некоректних даних, вхід не відбувається, підсвічуються червоним поля, з'являється повідомлення щодо невірних credentials.
Post-condition	Вхід до аккаунту, перехід на сторінку проектів

Таблиця 1.4 – Варіант використання UC-03

Use case name	Вихід з аккаунту користувача
Use case ID	UC-03
Goals	Вихід з аккаунту, припинення роботи з застосунком
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає припинити роботу з застосунком
Pre-conditions	Вхід до застосунку виконано
Flow of Events	Користувач натискає кнопку “Log out”, після чого здійснюється перехід на сторінку входу в аккаунт.
Extension	-
Post-condition	Вихід виконано, для роботи потрібно знову виконати вхід.

Таблиця 1.5 – Варіант використання UC-04

Use case name	Створення проекту
Use case ID	UC-04
Goals	Створення проекту для роботи користувача
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає почати роботу з проектами
Pre-conditions	Вхід здійснено.

Продовження таблиці 1.5

Flow of Events	Користувач переходить на сторінку проектів, натискає кнопку “+”, вводить назву та опис проекту, підтверджує створення проекту. Після цього створений проект стає активним для користувача, відбувається перехід на сторінку проектів.
Extension	-
Post-condition	Новий проект є активним, перехід на сторінку проектів.

Таблиця 1.6 – Варіант використання UC-05

Use case name	Встановлення активного проекту
Use case ID	UC-05
Goals	Зміна активного проекту для користувача
Actors	Користувач, що здійснив вход до аккаунту
Trigger	Користувач бажає змінити активний проект
Pre-conditions	Вхід виконано
Flow of Events	Користувач переходить на сторінку проектів, обирає потрібний йому, натискає навпроти нього кнопку для встановлення активним.
Extension	-
Post-condition	Активний проект змінено.

Таблиця 1.7 – Варіант використання UC-06

Use case name	Створення тест-кейса
Use case ID	UC-06
Goals	Створення тест-кейса для тестування функціоналу
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає створити новий тест-кейс
Pre-conditions	Вхід виконано.
Flow of Events	Користувач переходить на сторінку тест-кейсів, натискає кнопку “+”, користувач у модальному вікні вводить назву кейсу, короткий опис, передумови, кроки для тестування та очікуваний результат. Після цього підтверджує створення тест-кейса. Відбувається перехід на сторінку з тест-кейсами.
Extension	-
Post-condition	Тест-кейс створено, перехід на сторінку з тест-кейсами

Таблиця 1.8 – Варіант використання UC-07

Use case name	Модифікація тест-кейсу
Use case ID	UC-07
Goals	Зміна змісту тест-кейсу
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає внести зміни до тест-кейсу
Pre-conditions	Вхід виконано.

Продовження таблиці 1.8

Flow of Events	Користувач переходить на сторінку тест-кейсів, обирає потрібний, натискає кнопку “Редагування”, змінює значення в полях тест-кейса, підтверджує зміни. Після цього відбувається перехід на сторінку з тест-кейсами.
Extension	-
Post-condition	Тест-кейс змінено. Перехід на сторінку з тест-кейсами

Таблиця 1.9 – Варіант використання UC-08

Use case name	Видалення тест-кейсу
Use case ID	UC-08
Goals	Видалення тест-кейсу зі списку створених
Actors	Користувач, що здійснив вход до аккаунту
Trigger	Користувач бажає видалити тест-кейс
Pre-conditions	Вхід виконано.
Flow of Events	Користувач переходить на сторінку тест-кейсів, обирає потрібний, натискає навпроти нього кнопку “Видалити”, тест-кейс зникає зі списку, видаляється з БД. Оновлена сторінка не має видаленого тест-кейса.
Extension	-
Post-condition	Тест-кейс видалено.

Таблиця 1.10 – Варіант використання UC-09

Use case name	Створення тест-сьюту
Use case ID	UC-08
Goals	Створення Test Suite з набором тест-кейсів
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає створити тест-сюtot
Pre-conditions	Вхід виконано.
Flow of Events	Користувач переходить на сторінку тест-сьютів, натискає кнопку “+”, вводить у поля назву та опис тест-сьюту, обирає тест-кейси, що будуть додані до нього, підтверджує створення. Після цього відбувається перехід на сторінку тест-сьютів.
Extension	-
Post-condition	Тест-сюtot створений, перехід на сторінку Test Suites.

Таблиця 1.11 – Варіант використання UC-10

Use case name	Виконання Test Run
Use case ID	UC-10
Goals	Прогін тест-рану з відмітками пройдених/непройдених тестів
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає виконати тест-ран

Продовження таблиці 1.11

Pre-conditions	Вхід виконано.
Flow of Events	Користувач переходить на сторінку тест-сьютів, обирає потрібний, натискає навпроти нього кнопку “Виконати”, відкривається вікно з переліком тест-кейсів з тест-сьюта, навпроти кожного користувач обирає стан тесту (успішно/невдача/пропущено), підтверджує створення тест-рану. Після цього відбувається перехід на сторінку Test Runs, де у переліку першим стоїть новий тест-ран.
Extension	Якщо користувач не обирає стан для тесту (успішно/невдача пропущено), кнопка підтвердження тест-рану є неактивною.
Post-condition	Тест-ран створений, перехід на сторінку тест-ранів.

Таблиця 1.12 – Варіант використання UC-11

Use case name	Генерація звіту за тест-раном
Use case ID	UC-11
Goals	Створення PDF-файлу зі змістом тест-рану
Actors	Користувач, що здійснив вхід до аккаунту
Trigger	Користувач бажає згенерувати звіт тест-рану
Pre-conditions	Вхід виконано.

Продовження таблиці 1.12

Flow of Events	Користувач переходить на сторінку тест-ранів, обирає потрібний, натискає навпроти нього кнопку “Згенерувати”, після чого створюється файл у форматі PDF, який користувачу пропонується завантажити. Після вказання користувачем шляху для завантаження, сторінка оновлюється.
Extension	-
Post-condition	Звіт згенеровано, завантажено користувачем.

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.3 наведено загальну модель вимог, а в таблицях 1.13 – 1.22 наведений опис функціональних вимог до програмного забезпечення.

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						20

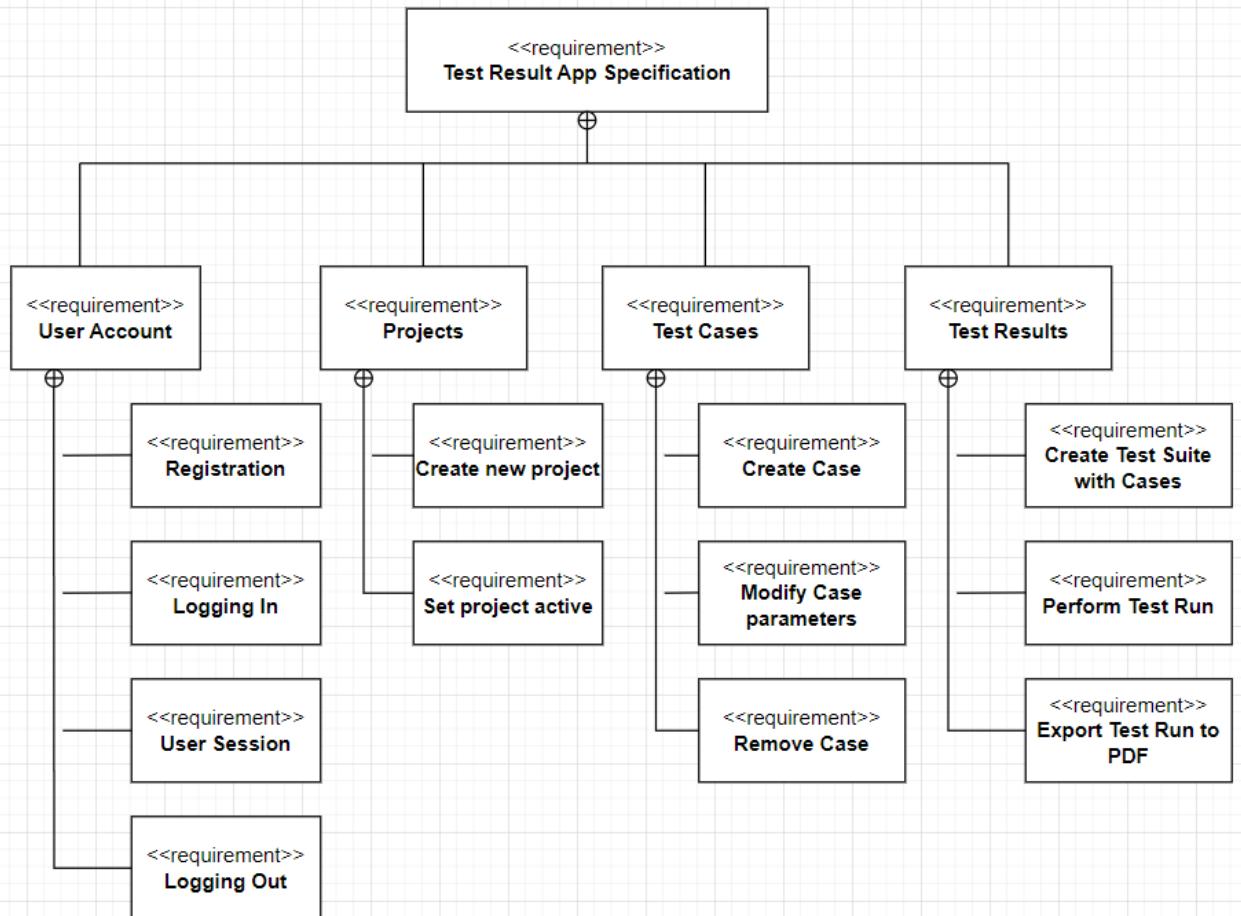


Рисунок 1.3 – Модель вимог у загальному вигляді

Таблиця 1.13 – Функціональна вимога FR-1

Назва	Додавання нового користувача до бази даних
Опис	Система повинна надавати можливість створювати новий аккаунт для користувача.

Таблиця 1.14 – Функціональна вимога FR-2

Назва	Авторизація користувача
Опис	Система повинна надавати можливість користувачу увійти до застосунку з відповідними логіном та паролем, а також вийти з нього.

Таблиця 1.15 – Функціональна вимога FR-3

Назва	Додавання нового проєкту до бази даних
Опис	Система повинна надавати можливість користувачу створити та зберегти новий проєкт для роботи у застосунку.

Таблиця 1.16 – Функціональна вимога FR-4

Назва	Змінення стану проєкту на “активний” для користувача
Опис	Система повинна надавати можливість користувачу встановити для одного з проектів статус “активний” для поточного аккаунта.

Таблиця 1.17 – Функціональна вимога FR-5

Назва	Додавання тест-кейсу до бази даних
Опис	Система повинна надавати можливість користувачу створити та зберегти тест-кейс з такими параметрами: назва, опис, передумови, кроки виконання, постумови.

Таблиця 1.18 – Функціональна вимога FR-6

Назва	Змінення параметрів тест-кейсу у базі даних
Опис	Система повинна надавати можливість користувачу змінити будь-який параметр тест-кейса на інше довільне значення.

Таблиця 1.19 – Функціональна вимога FR-7

Назва	Видалення тест-кейсу з бази даних
Опис	Система повинна надавати можливість користувачу видалити тест-кейс, що було створено ним раніше.

Таблиця 1.20 – Функціональна вимога FR-8

Назва	Додавання тест-сьюту до бази даних
Опис	Система повинна надавати можливість користувачу створити та зберегти тест-сьют, встановивши параметри та прикріпивши до нього потрібні тест-кейси.

Таблиця 1.21 – Функціональна вимога FR-9

Назва	Додавання тест-рану до бази даних
Опис	Система повинна надавати можливість користувачу провести тест-ран, встановити для кожного тест-кейса значення результату проходу кейсу та загальний результат тест-рану.

Таблиця 1.22 – Функціональна вимога FR-10

Назва	Створення PDF-файлу зі змістом тест-рану
Опис	Система повинна надавати можливість користувачу згенерувати PDF-файл для завантаження, у якому міститься інформація щодо проведеного тест-рану.

Встановимо відповідність між варіантами використання та функціональними вимогами за допомогою матриці трасування вимог у таблиці нижче:

Таблиця 1.23 – Матриця трасування вимог

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10
UC-01	+									
UC-02		+								
UC-03		+								

Продовження таблиці 1.23

UC-04			+								
UC-05				+							
UC-06					+						
UC-07						+					
UC-08							+				
UC-09								+			
UC-10									+		
UC-11											+

1.4.2 Розроблення нефункціональних вимог

Веб-застосунок повинен бути інтуїтивно зрозумілим для користувача, повинен працювати з оптимальною для користувача швидкістю, завантаження сторінок не повинно займати багато користувацького часу. Застосунок має бути зручним в плані User Experience та працювати у браузерах як десктопних, так і мобільних/планшетних.

1.5 Постановка задачі

Призначення дипломного проекту — веб-застосування для автоматизації роботи тестувальника, а саме моніторингу та обробки результатів тестувань програмного забезпечення.

Метою розробки програмного забезпечення є оптимізація роботи тестувальника за рахунок надання можливості групувати тест-кейси у тест-сьюти, запускати тест-рани за створеними тест-сьютами та автоматизовано звітувати за результатами виконання тест-рану, що надасть можливість виконувати його завдання швидше та ефективніше.

Для досягнення поставленої мети необхідно вирішити комплекс наступних взаємопов'язаних задач:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						24

- створення тест-кейсів;
- групування тест-кейсів у тест-сьюти;
- виконання тест-ранів за змістом тест-сьюту;
- генерація звітів у форматі PDF.

Висновки до розділу

У процесі виконання першого розділу дипломного проєкту було проведено загальну аналітику щодо вимог до програмного забезпечення, яке буде створене у результаті дипломної роботи.

Роботу почато з короткого екскурсу до історії предметної області - тестування ПЗ та надання визначень найважливішим термінам, що використовуються спеціалістами у цій галузі, та з якими буде тісно зв'язаний бізнес-процес застосунку. Після цього було зазначено поточну ситуацію в ІТ-сфері з тестуванням та загальний принцип його проведення на різних проектах.

Було проведено аналіз технологій та обрано найбільш оптимальні бібліотеки та фреймворки для виконання дипломного проєктування. Деталізовано варіанти використання веб-застосунку та зазначено різні випадки ситуацій з боку користувача.

Проаналізовано успішні ІТ-проекти за схожою тематикою, такі як TestRail та Kualitee. Дипломний проєкт передбачає, як і його аналоги, можливість створення тест-кейсів та тест-сьютів, але його особливістю серед них виступає можливість зручного проведення Test Runs та експорту результатів до PDF-звіту.

Було поставлено чіткий набір функціональних вимог до створюваного програмного забезпечення:

- реєстрація та авторизація користувача;
- створення проєктів, встановлення активного проєкту;
- додавання, редагування та видалення тест-кейсів;
- створення тест-сьютів, додавання до них тест-кейсів;

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

- виконання тест-ранів та експорт результатів до PDF-файлу.

Таким чином, маємо зрозумілі критерії та вимоги, яким повинен відповісти цей проект по закінченню його написання.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.02.81

Арк.

26

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес-процесів веб-застосунку скористаємося BPMN-діаграмами[8, с. 56].

Першим продемонструємо процес створення тест-кейсів користувачем.

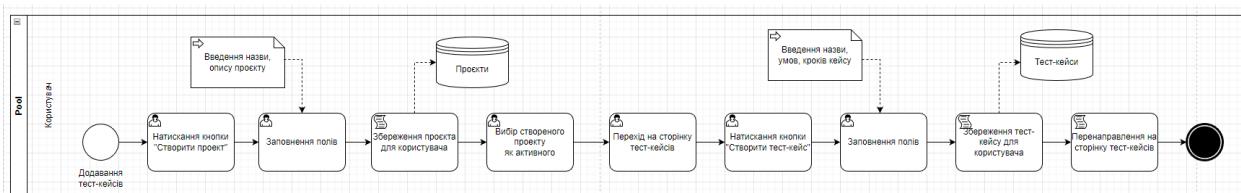


Рисунок 2.1 – Схема бізнес-процесу створення тест-кейсів

Зі схеми можна зрозуміти, що користувач повинен створити проект, встановити його активним, після чого заповнювати тест-кейси відповідними даними.

Наступним користувачеві надається можливість групувати тест-кейси у тест-съоти, опишемо діаграмою детально процес:

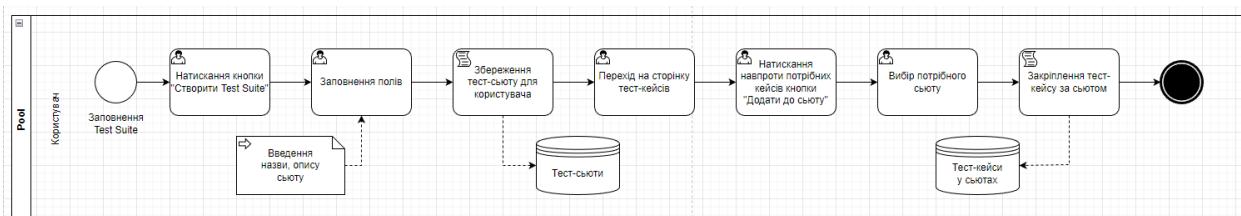


Рисунок 2.2 – Схема бізнес-процесу створення та наповнення тест-съотів

Центральним процесом у використанні веб-застосунку є проведення Test Runs для тест-съотів з подальшим експортом результату у звіт.

Детальніше на діаграмі:

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

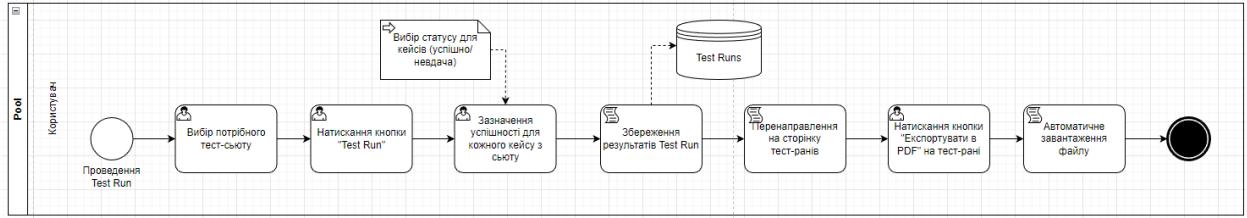


Рисунок 2.3 – Схема бізнес-процесу проведення Test Run та експорту

2.2 Архітектура програмного забезпечення

Визначимося з архітектурою веб-застосунку [9, с. 32], оберемо багатошарову (layered architecture). Таким чином, застосунок буде складатися з таких слоїв:

- Database Layer (реляційна база даних);
- Persistence Layer (ORM, побудова запитів до бази);
- Business Layer (backend, authentication, REST API, черга завдань, бізнес-логіка);
- Presentation Layer (frontend, графічний інтерфейс користувача у браузері).

Таким чином, Database Layer, Persistence Layer, Business Layer забезпечуються серверною частиною проекту, Presentation Layer забезпечується клієнтською частиною проекту.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

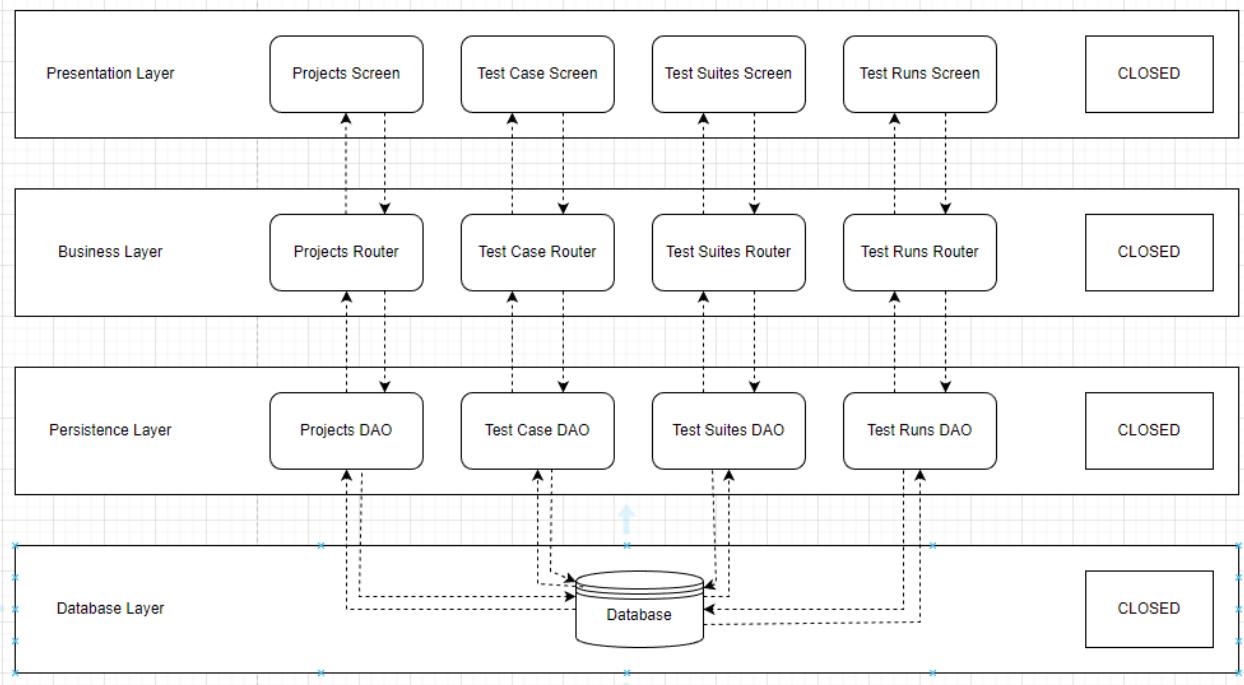


Рисунок 2.4 – Схема багаторівневої архітектури проєкту

Детальніше про компоненти:

- Database Layer утворює база даних на основі реляційної СУБД PostgreSQL;
- Persistence Layer утворює функціонал доступу до БД, написаний за допомогою ORM SQLAlchemy;
- Business Layer утворює функціонал контролерів-роутів, написаний за допомогою фреймворку Flask;
- Presentation Layer утворюють екранні форми користувачького інтерфейсу, написані за допомогою веб-фреймворку React.js.

Зв’язки між компонентами продемонструємо також за допомогою діаграмами компонентів на рисунку нижче:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						29

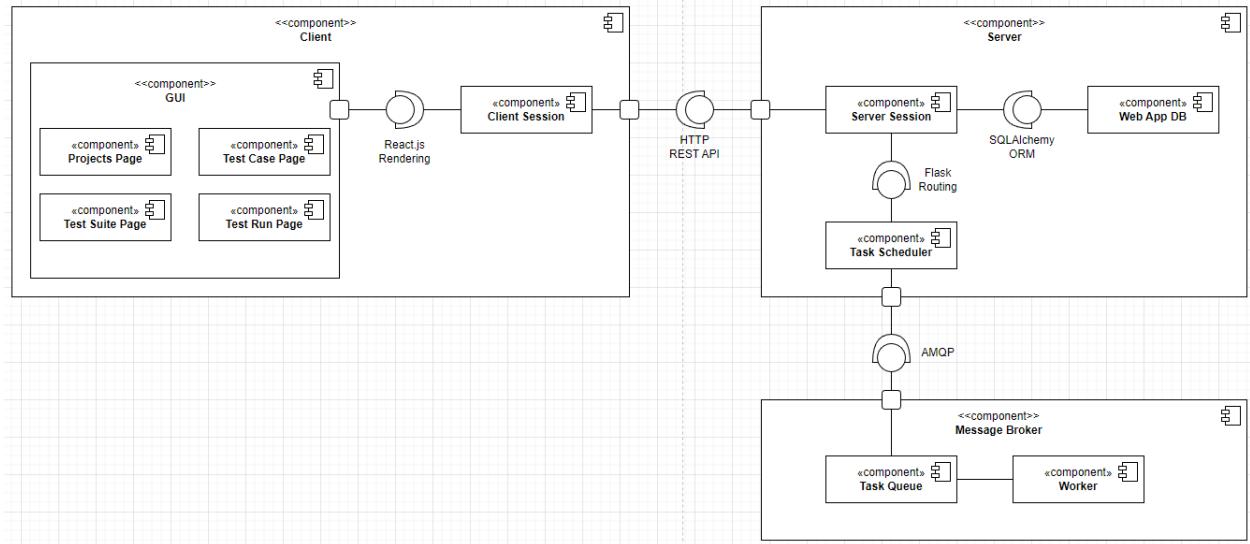


Рисунок 2.5 – Діаграма компонентів

2.3 Конструювання програмного забезпечення

Серверна частина програмного забезпечення представляє собою REST API[10, с. 157], що взаємодіє з базою даних та брокером повідомлень для виконання регулярних задач з генерації звітів.

Отримуючи HTTP-запити, сервер обробляє їх, та надає відповідь, що містить код статусу HTTP, заголовки(headers) та тіло відповіді у форматі JSON.

Серверна частина обробляє такі HTTP-запити:

- GET /projects (отримання усіх)
- GET /test_cases
- GET /test_suites
- GET /test_runs
- POST /projects (створення нового)
- POST /test_cases
- POST /test_suites
- POST /test_runs
- DELETE /projects/id (видалення одного)
- DELETE /test_cases/id
- DELETE /test_suites/id

Змін.	Арк.	№ докум.	Підп.	Дата.

- DELETE /test_runs/id
- POST /test_runs/id (експорт у PDF-звіт)

Коди відповідей сервера на запити:

- GET
 - a) успіх: 200 OK
 - b) не знайдено: 404 Not Found
- POST
 - a) успіх: 201 Created
 - b) невдача: 400 Bad Request
- DELETE
 - a) успіх: 204 No Content
 - b) невдача: 400 Bad Request

Тіло відповіді має формат JSON – об'єкт або масив об'єктів.

Проект повертається в такому форматі:

```
{
  "name": "Project name",
  "description": "Project description",
  "is_active": true,
}
```

Тест-кейс повертається в такому форматі:

```
{
  "name": "Case name",
  "description": "Case description",
  "precondition": "Case precondition content",
  "test_steps": ["Step 1 content", "Step 2 content"],
  "postcondition": "Case postcondition content"
}
```

Тест-сьют повертається в такому форматі:

Змін.	Арк.	№ докум.	Підп.	Дата.

```
{
    "name": "Suite name",
    "description": "Suite description",
    "test_cases": [
        {"name": "Case 1 name", "description": "Case 1 description"}
    ]
}
```

Тест-ран повертається в такому форматі:

```
{
    "test_run_id": "1",
    "result": "success",
    "test_cases_results": [
        {"name": "Case 1", "status": "success"}
    ]
}
```

Клієнтська частина звертається до серверного REST API за встановленими запитами та отримує результат, який відображає на екрані браузера після форматування.

Щодо цілісності даних, в якості системи управління базами даних використовується PostgreSQL[11, с. 34]. База даних серверу призначена для зберігання користувачів, а також даних про їх проекти, тест-кейси, тест-сьюти, тест-рани. Для наочності нижче наведемо ER-діаграму сутностей у БД.

Змін.	Арк.	№ докум.	Підп.	Дата.

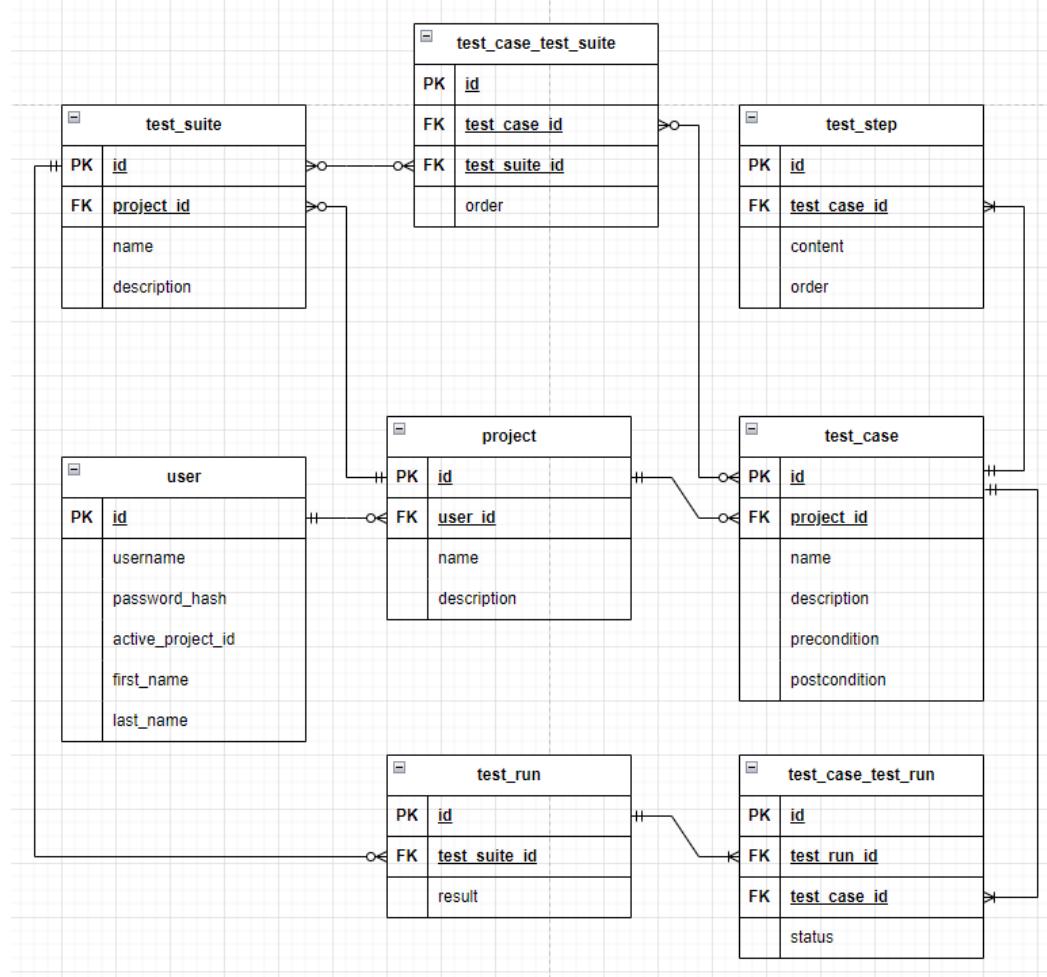


Рисунок 2.6 – ЕР-діаграма бази даних

Фізична схема БД представлена у документі КПІ.ІП-9113.045440.06.99.СБД.

Докладний опис таблиць для сутностей можна побачити у таблицях нижче:

Таблиця 2.1 – Опис таблиці user

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
username	varchar			Ім'я для входу
password_hash	varchar			Хеш паролю для перевірки користувача

Продовження таблиці 2.1

active_project_id	int			Ідентифікатор проєкту, що є активним для користувача
first_name	varchar			Ім'я користувача
last_name	varchar			Прізвище користувача

Таблиця 2.2 – Опис таблиці project

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
user_id	int		X	Ідентифікатор користувача, що володіє проєктом
name	varchar			Назва проєкту
description	varchar			Опис проєкту

Таблиця 2.3 – Опис таблиці test_case

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
project_id	int		X	Ідентифікатор проєкту, у якому знаходитьться тест-кейс
name	varchar			Назва тест-кейсу
description	varchar			Опис тест-кейсу

Продовження таблиці 2.3

precondition	varchar			Передумова тест-кейсу
postcondition	varchar			Постумова тест-кейсу

Таблиця 2.4 – Опис таблиці test_step

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
test_case_id	int		X	Ідентифікатор тест-кейсу, до якого відноситься цей крок
content	varchar			Зміст кроку
order	int			Порядковий номер кроку

Таблиця 2.5 – Опис таблиці test_suite

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
project_id	int		X	Ідентифікатор проекту, до якого відноситься тест-сьют
name	varchar			Назва тест-сьюту
description	varchar			Опис тест-сьюту

Таблиця 2.6 – Опис таблиці test_run

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
test_suite_id	int		X	Ідентифікатор тест-сьюту, для якого проводиться тест-ран
result	varchar			Результат тест-рану загальний

Таблиця 2.7 – Опис таблиці test_case_test_suite

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
test_case_id	int		X	Ідентифікатор тест-кейсу
test_suite_id	int		X	Ідентифікатор тест-сьюту
order	int			Порядковий номер тест-кейсу у тест-сьюті

Таблиця 2.8 – Опис таблиці test_case_test_run

Назва поля	Тип даних	PK	FK	Опис
id	serial	X		Ідентифікатор
test_run_id	int		X	Ідентифікатор тест-кейсу
test_case_id	int		X	Ідентифікатор тест-сьюту
status	varchar			Результат одного тест-кейсу в рамках тест-рану

Опишемо методи та класи написаного програмного забезпечення.

Почнемо з клієнтської частини, вона складається з функцій-компонентів React (далі "функції") графічного інтерфейсу.

Найважливіші функції зберігаються у директорії `src/apps`, та являють собою мікрододатки, з яких складається веб-застосунок. Тобто:

- `TestResultApp` - містить у собі головний `AppBar` – меню застосунку, `AppBar` для позначення активного проєкту та користувача, а також роутер, що відображає потрібну вкладку з обраних користувачем у головному `AppBar`;
- `AuthApp` - містить роутер, що відображає сторінку логіну або реєстрації в залежності від стану користувача.

Далі по значущості йдуть функції, що представляють логічні сторінки, вони зберігаються у директорії `src/pages`. Маємо такий перелік функцій:

- `SignInPage` - сторінка логіну користувача до системи;
- `SignUpPage` - сторінка реєстрації нового користувача;
- `ProjectsPage` - сторінка з таблицею проєктів користувача;
- `TestCasesPage` - сторінка з переліком тест-кейсів користувача;
- `TestSuitesPage` - сторінка з тест-сьютами користувача;
- `TestRunsPage` - сторінка з проведеними тест-ранами.

Складовими частинами вищепереліканих сторінок є функції, що базуються на компонентах з бібліотеки `MaterialUI`, вони зберігаються у каталозі `src/components`. Ці функції було згруповано за батьківськими компонентами у такі групи:

- Аппбари (`src/components/appbars`);
- Акордеони (`src/components/accordions`);
- Таблиці (`src/components/tables`);
- Кнопки (`src/components/buttons`);
- Роутери (`src/components/routers`).

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Розглянемо наявні компоненти детальніше:

1) Аппбари:

- MainAppBar - головне робоче меню користувача для вибору сторінки;
- ActiveProjectUserAppBar - аппбар, що демонструє активний проект та ім'я користувача;
- PageHeadingAppBar - аппбар-заголовок для кожної сторінки сутностей з кнопкою додавання нового екземпляру.

2) Акордеони - для кожного наведено функцію для одного (Accordion) та списку (AccordionList):

- TestCaseAccordion - містить контент кожного окремого тест-кейсу;
- TestSuiteAccordion - складається з сітки тест-кейсів у кожному з тест-сьютів;
- TestRunAccordion - зберігається таблицю результатів тест-рану для конкретного тест-сьюту.

3) Таблиці:

- ProjectsTable - таблиця з даними усіх проєктів, кнопками для редагування даних;
- TestRunContentTable - таблиця з результатами кожного тест-кейсу у тест-рані.

4) Кнопки:

- SetActiveProjectButton - кнопка для встановлення обраного проєкту активним.

5) Роутери:

- TestResultAppRouter - дозволяє обирати між сторінками проєктів, тест-кейсів, тест-сьютів, тест-ранів;
- AuthAppRouter - дозволяє обирати сторінку логіну або реєстрації.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Наступною розглянемо серверну частину коду.

У директорії migration/versions зберігаються файли міграцій реляційної бази даних з функціями upgrade, downgrade для зручного застосування та відміни міграцій та відстежування змін у БД.

Основна частина програми знаходиться у директорії src.

Функціонал для генерації PDF-звітів знаходиться у каталозі src/core/pdf, основна логіка у файлі report.py. Базовим класом для звіту є ReportPDF, що містить методи prepare(), build_pdf(), make(). Дочірній клас TestRunReportPDF, має також такі методи:

- build_report_heading() - відображає у звіті основний заголовок;
- build_test_suite_info() - відображає базову інформацію про тест-сьют
- build_test_cases_list_heading() - відображає заголовок перед списком тест-кейсів
- build_test_case_index() - відображає черговий номер тест-кейсу
- build_test_case_name_status() - відображає назву та статус тест-кейсу
- build_test_case_conditions_steps() - відображає передумову, постумову, кроки виконання тест-кейсу
- build_one_test_case() - створює у звіті повністю один тест-кейс з номером, назву, статусом та усією інформацією
- build_test_cases() - рендерить усі тест-кейси, передані класу.

У методі TestRunReportPDF.build_pdf відбувається перевизначення з рендерингом заголовку звіту, інформації про тест-сьют та списком тест-кейсів з результатами.

У директорії src/models зберігаються класи, що відображають таблиці бази даних у якості Object-Relational Mapping, а саме наявні такі класи: Project, User, TestCase, TestStep, TestSuite, TestRun, TestCaseTestSuite, TestCaseTestRun. Наведені класи мають властивості, що відповідають полям у таблицях, а також relationships для зв'язування таблиць за ORM best practices.

Змін.	Арк.	№ докум.	Підп.	Дата.

У директорії src/api зберігаються роутери, які надають ендпоїнти для авторизації, а також взаємодії з сущностями бази даних за допомогою HTTP REST API. Маємо такі роутери: auth, projects, test_cases, test_suites, test_runs. Вони містять відповідні методи GET, POST, DELETE для кожної з сущностей.

Мінімальна конфігурація технічних засобів для використання ПЗ:

Процесор: Intel i3 або аналогічні

ОЗП: 2 ГБ

Інтернет: 1 мбіт/с

ОС: Windows, MacOS, Linux

Сумісні браузери: Chrome, Firefox, Opera, Edge, Safari

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.9.

Таблиця 2.9 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	JetBrains PyCharm	Середовище розробки програмного забезпечення серверної частини дипломного проєкту.
2	JetBrains WebStorm	Середовище розробки програмного забезпечення клієнтської частини дипломного проєкту.
3	Postman	Програмне забезпечення, необхідне для тестування REST запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
4	JetBrains DataGrip	Програмне забезпечення, яке надає графічний інтерфейс для доступу до РСУБД PostgreSQL.

2.4 Аналіз безпеки даних

Розглянемо можливі загрози для створюваного програмного забезпечення, а також засоби боротьби з ними.

Оскільки ПЗ працює з базою даних, очевидною загрозою стає SQL Injection[12, с. 103]. Ця загроза дозволяє користувачу отримати дані, до яких доступу в нього бути не повинно, або навіть пошкодити дані, що зберігаються у базі. Цей застосунок при роботі з БД використовує ORM SQLAlchemy, котрий автоматично форматує вхідні дані від користувача у формат, який позбавляє можливості виконати ін'єкцію SQL-коду.

Також веб-застосунки часто вразливі до CSRF-атак[13, с. 14], коли атакований користувач може здійснити те, що він не планував робити, будучи залогіненим у веб-застосунку і зробити запит, що поставить користувача в небезпечну ситуацію. Для боротьби з цими атаками у проектному застосунку використовуються CSRF-токени, що підтверджують дію користувача саме з графічного інтерфейсу нашого застосунку, а не з інших серверів або хостів.

Таким чином, застосунок, що створюється у ході дипломного проєкту, є захищеним від відомих та широко розповсюджених загроз.

Висновки до розділу

В другому розділі дипломного проєкту було розглянуто та проаналізовано моделі та конструкція програмного забезпечення.

Основними бізнес-процесами проєктного веб-застосунку є:

- створення тест-кейсів;
- групування тест-кейсів у тест-сьюти;
- виконання тест-ранів та експорт їх результатів у PDF-звіт.

За основу архітектури проєкту взято шаблон проєктування багатошарового ПЗ, імплементовано 4 шари програмного забезпечення –

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						41

Database Layer, Persistence Layer, Business Layer, Presentation Layer. Перші 3 шари реалізовані у серверній частині проекту, останній - Presentation Layer – на клієнтській частині.

Конструкція програмного забезпечення складається з бази даних, серверу, чергі завдань та клієнту. База даних зберігає цілісність даних. Сервер звертається до неї та формує REST API, за яким можна звертатись до сервера для здійснення запитів до БД та встановлення задач на генерацію звітів. Клієнт згідно правил REST звертається до сервера та відображає отримані дані у браузері.

Системні вимоги програмного забезпечення мінімальні, застосунок не потребує великих обчислювальних потужностей, оскільки працює у браузері та навантажує сервер, а не комп'ютер користувача.

Найважливіші програмні утиліти, використані у ході розробки - інтегровані середовища розробки - PyCharm, WebStorm, DataGrip від компанії JetBrains, а також утиліта Postman, що використовується для тестування REST API.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення (Software Quality Analysis) - це процес визначення та оцінки відповідності програмного забезпечення встановленим стандартам якості, вимогам та очікуванням користувачів. Цей процес включає оцінку різних аспектів програмного забезпечення, таких як його функціональність, продуктивність, надійність, безпеку, зручність використання та інші.

До такого аналізу відносяться:

- визначення критеріїв якості - становлення метрик та критеріїв, які використовуються для оцінки якості програмного забезпечення. Це можуть бути такі критерії, як час відгуку системи, кількість виявлених помилок, швидкість роботи програми;
- виконання тестування для перевірки функціональності, продуктивності та надійності програмного забезпечення. Це можуть бути модульні тести, інтеграційні тести, системні тести, регресійні тести та інші;
- аналіз результатів тестування, виявлення помилок, недоліків та слабких місць у програмному забезпеченні. Це допомагає виправити проблеми та покращити якість продукту;
- використання метрик якості, таких як кількість виявлених помилок, час відгуку, продуктивність програмного забезпечення тощо. Це допомагає зрозуміти, наскільки добре програмне забезпечення відповідає встановленим стандартам та вимогам.

Для перевірки якості написаного коду використаємо статичний аналізатор Embold, що має зручний веб-інтерфейс для демонстрації результатів. На рисунках нижче представлені результати аналізу:

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

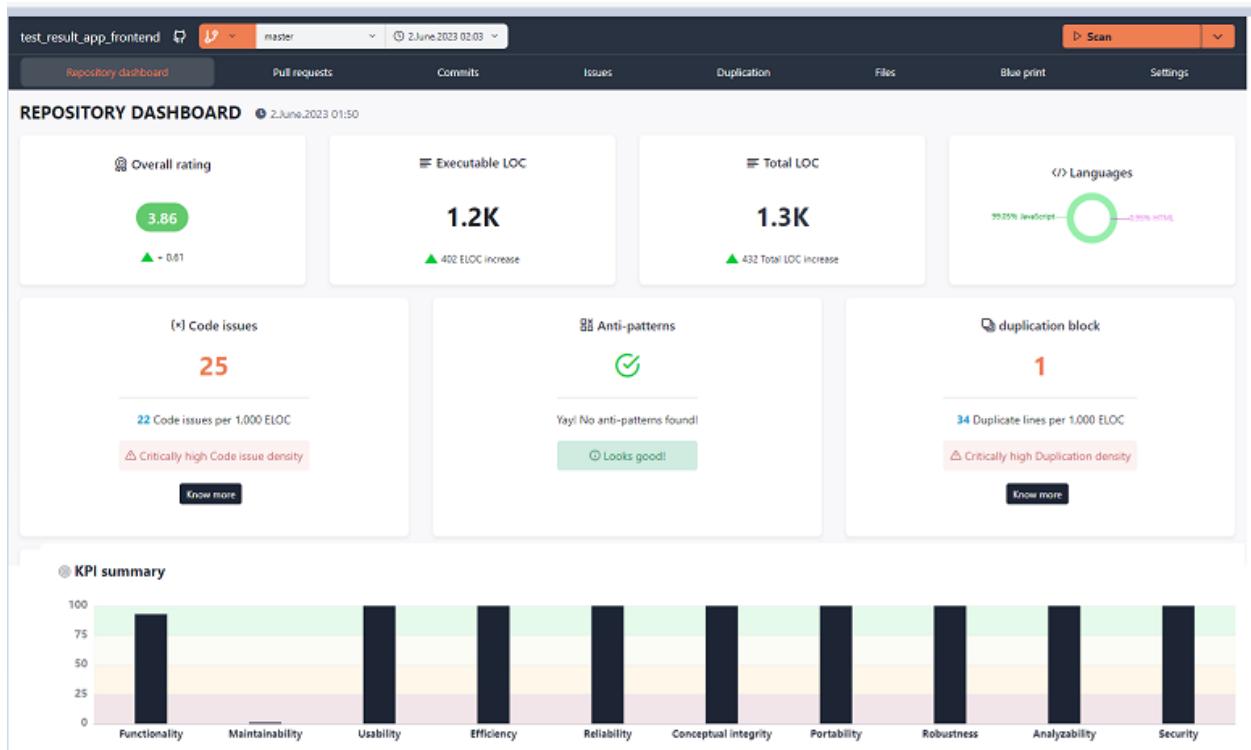


Рисунок 3.1 – Результат статичного аналізу клієнтської частини коду

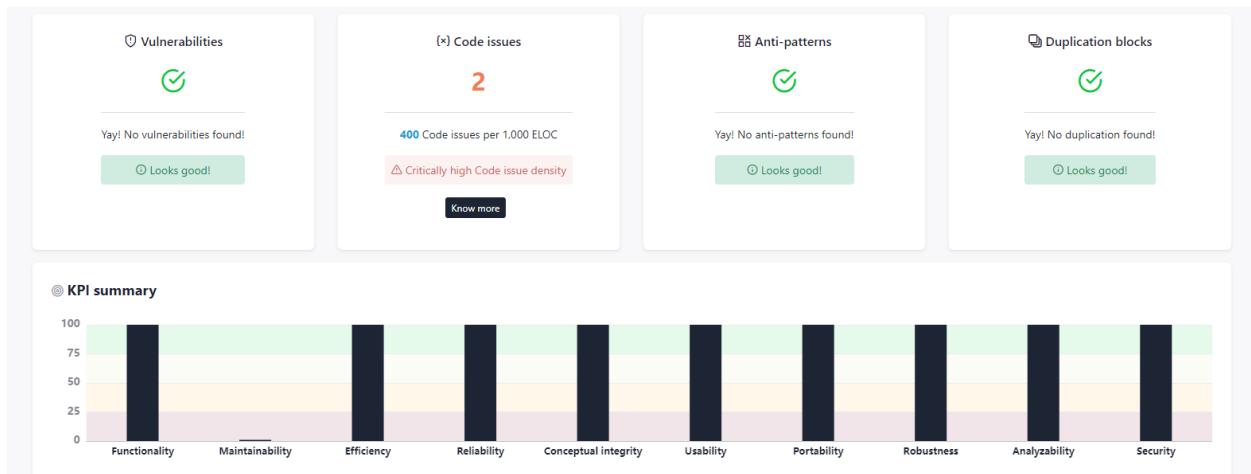


Рисунок 3.2 – Результат статичного аналізу серверної частини коду

3.2 Опис процесів тестування

Для перевірки коректності роботи функціоналу створеного програмного забезпечення, було проведено мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.11.

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						44

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Аутентифікація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Ім'я, прізвище, електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: ім'я та прізвище користувача, коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.

Таблиця 3.2 – Тест 1.2

Тест	Авторизація користувача
Модуль	Аутентифікація користувача
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці входу, є зареєстрованим раніше
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться електронна пошта та пароль. Після цього натискається кнопка підтвердження входу.
Очікуваний результат	Вхід успішний, користувач перенаправляється на головну робочу сторінку веб-застосунку.

Продовження таблиці 3.2

Фактичний результат	Вхід успішний, користувач перенаправляється на головну робочу сторінку веб-застосунку.
---------------------	----------------------------------------------------------------------------------------

Таблиця 3.3 – Тест 1.3

Тест	Вихід з системи
Модуль	Аутентифікація користувача
Номер тесту	1.3
Початковий стан системи	Користувач є авторизованим до системи за електронною поштою та паролем.
Вхідні дані	-
Опис проведення тесту	На головному App Bar натискається іконка користувача, обирається пункт “Log Out” та натискається кнопка.
Очікуваний результат	Користувач стає неавторизованим, відбувається перенаправлення на сторінку авторизації.
Фактичний результат	Користувач стає неавторизованим, відбувається перенаправлення на сторінку авторизації.

Таблиця 3.4 – Тест 2.1

Тест	Створення проєкту
Модуль	Робота з проєктами
Номер тесту	2.1
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	Назва проєкту, опис проєкту.

Продовження таблиці 3.4

Опис проведення тесту	На головному App Bar обирається вкладка «Projects», натискається кнопка. Після цього навпроти заголовку сторінки натискається кнопка «+», в вікно вводяться відповідно назва та опис проекту, натискається кнопка підтвердження.
Очікуваний результат	Створюється новий проект, з'являється у списку створених проектів.
Фактичний результат	Створюється новий проект, з'являється у списку створених проектів.

Таблиця 3.5 – Тест 2.2

Тест	Встановлення активного проекту
Модуль	Робота з проектами
Номер тесту	2.2
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	-
Опис проведення тесту	На головному App Bar обирається вкладка «Projects», натискається кнопка. Обирається потрібний проект зі створених раніше, навпроти нього натискається кнопка «Make active».
Очікуваний результат	Обраний проект стає активним, що відображається записом у таблиці проектів.
Фактичний результат	Обраний проект стає активним, що відображається записом у таблиці проектів.

Таблиця 3.6 – Тест 3.1

Тест	Створення тест-кейса
Модуль	Робота з тест-кейсами
Номер тесту	3.1
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	Назва кейсу, опис кейсу, передумови, післяумови, кроки виконання
Опис проведення тесту	На головному App Bar обирається вкладка «Test Cases», натискається кнопка. Після цього навпроти заголовку сторінки натискається кнопка «+», в вікно вводяться відповідно назва, опис, передумови, післяумови, кроки виконання тест-кейсу, натискається кнопка підтвердження.
Очікуваний результат	Створюється новий тест-кейс з заданими параметрами, що відображається у загальному списку на сторінці усіх тест-кейсів.
Фактичний результат	Створюється новий тест-кейс з заданими параметрами, що відображається у загальному списку на сторінці усіх тест-кейсів.

Таблиця 3.7 – Тест 3.2

Тест	Зміна тест-кейса
Модуль	Робота з тест-кейсами
Номер тесту	3.2
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	Параметри тест-кейсу, що потрібно змінити.

Продовження таблиці 3.7

Опис проведення тесту	На головному App Bar обирається вкладка «Test Cases», натискається кнопка. Після цього обирається потрібний тест-кейс, навпроти нього натискається кнопка «Edit», в потрібні полі у вікні вводяться змінені дані, натискається кнопка підтвердження змін.
Очікуваний результат	У обраного тест-кейса на сторінці змінюються параметри, у які було введено нову інформацію.
Фактичний результат	У обраного тест-кейса на сторінці змінюються параметри, у які було введено нову інформацію.

Таблиця 3.8 – Тест 3.3

Тест	Видалення тест-кейса
Модуль	Робота з тест-кейсами
Номер тесту	3.3
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	-
Опис проведення тесту	На головному App Bar обирається вкладка «Test Cases», натискається кнопка. Після цього обирається тест-кейс для видалення, навпроти нього натискається кнопка «Delete».
Очікуваний результат	Видалений тест-кейс зникає зі сторінки усіх тест-кейсів та видаляється з бази даних.
Фактичний результат	Видалений тест-кейс зникає зі сторінки усіх тест-кейсів та видаляється з бази даних.

Таблиця 3.9 – Тест 4.1

Тест	Створення тест-сьюту
Модуль	Групування тестів, виконання, звітність
Номер тесту	4.1
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	Назва тест-сьюту, опис тест-сьюту, потрібні тест-кейси наявні в системі.
Опис проведення тесту	На головному App Bar обирається вкладка «Test Suites», натискається кнопка. Після цього навпроти заголовку натискається кнопка «+», у поля вводиться назва та опис тест-сьюту, з випадаючого списку обираються потрібні тест-кейси, що додаються до тест-сьюту. Після цього натискається кнопка підтвердження.
Очікуваний результат	Створюється новий тест-сьют з параметрами та зазначеними користувачем тест-кейсами, відображається на сторінці усіх тест-сьютів.
Фактичний результат	Створюється новий тест-сьют з параметрами та зазначеними користувачем тест-кейсами, відображається на сторінці усіх тест-сьютів.

Таблиця 3.10 – Тест 4.2

Тест	Виконання Test Run
Модуль	Групування тестів, виконання, звітність
Номер тесту	4.2
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	Обраний тест-сьют, статуси для кожного тест-кейсу у сьюті.

Продовження таблиці 3.10

Опис проведення тесту	На головному App Bar обирається вкладка «Test Suites», натискається кнопка. Навпроти потрібного тест-сьюту натискається кнопка «Run». Заповнюються відповідні статуси для кожного тесту в залежності від результату виконання, натискається кнопка підтвердження.
Очікуваний результат	На вкладці “Test Runs” з’являється новий тест-ран з усіма зазначеними деталями та статусами тест-кейсів.
Фактичний результат	На вкладці “Test Runs” з’являється новий тест-ран з усіма зазначеними деталями та статусами тест-кейсів.

Таблиця 3.11 – Тест 4.3

Тест	Генерація PDF-звіту
Модуль	Групування тестів, виконання, звітність
Номер тесту	4.3
Початковий стан системи	Користувач є авторизованим до системи.
Вхідні дані	-
Опис проведення тесту	На головному App Bar обирається вкладка «Test Runs», натискається кнопка. Навпроти обраного тест-рану на сторінці натискається кнопка «Export to PDF».
Очікуваний результат	Згенерований PDF-файл завантажується на комп’ютер користувача, зміст файлу відповідає обраному тест-рану.
Фактичний результат	Згенерований PDF-файл завантажується на комп’ютер користувача, зміст файлу відповідає обраному тест-рану.

3.3 Опис контрольного прикладу

Проведемо контрольний приклад ефективного використання веб-застосунку, що було створено у процесі виконання дипломного проекту.

Почнемо з реєстрації користувача – потрібно ввести ім'я та прізвище, електронну пошту та пароль, після чого підтвердити реєстрацію. Проілюструємо процес на рисунку нижче:

The screenshot shows a web browser window with the following details:
Address bar: localhost:3000/signup
Title bar: Create an account
Form fields:

- First Name *: Hennadii
- Last Name *: Kochev
- Email Address *: myemail@mail.com
- Password *: *****

A checkbox labeled "I agree to the Rules of Test Result App usage." is checked.
Buttons:

- SIGN UP (blue button)
- ALREADY HAVE ACCOUNT? LOG IN (link)

Copyright notice: Copyright © Test Result App 2023.

Рисунок 3.3 – Інтерфейс реєстрації користувача

Після натискання кнопки «Sign Up» аккаунт користувача додається до бази даних, а інтерфейс перенаправляє на сторінку входу до системи, де потрібно ввести електронну пошту та пароль, за якими проведено реєстрацію. Демонстрація на рисунку нижче:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						52

localhost:3000/login

Email Address *

myemail@mail.com

Password *

.....

Remember me

LOG IN

DON'T HAVE AN ACCOUNT? [SIGN UP](#)

Copyright © [Test Result App](#) 2023.

Рисунок 3.4 – Інтерфейс авторизації користувача

Після натискання кнопки «Log In» інтерфейс переводить користувача до головної сторінки веб-застосунку. На сторінці присутній головний App Bar, де можна обрати потрібну сторінку з наданих, а також поточний активний проект та повне ім’я користувача. Обираємо пункт меню “Projects” та переходимо на сторінку проектів. Демонстрація сторінки нижче на рисунку:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						53

The screenshot shows a web browser window with the URL `localhost:3000/projects`. At the top, there's a green header bar with the text "Active project: Central Database" on the left and "Welcome, Hennadii Kochev!" on the right. Below this is a blue navigation bar with the title "TEST RESULT APP" and links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS". On the far right of the navigation bar is a user profile icon. The main content area has a white background and is titled "Projects". It features a table with four columns: "Project Name", "Description", "Is Active", and "Change state". The table contains three rows of data:

Project Name	Description	Is Active	Change state
Central Database	Main application of company	Yes	
Affiliate Management Portal	Secondary for influencers	No	<button>MAKE ACTIVE</button>
Supplier Order App	Secondary for suppliers	No	<button>MAKE ACTIVE</button>

Рисунок 3.5 – Інтерфейс сторінки проектів

На сторінці можна створити новий проект, а також змінити активний проект на інший. Наступним кроком переглянемо тест-кейси, що належать до активного проекту, екран сторінки наведено нижче:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						54

The screenshot shows the 'TEST RESULT APP' interface at localhost:3000/test_cases. The top navigation bar includes 'PROJECTS', 'TEST CASES', 'TEST SUITES', and 'TEST RUNS'. The main content area is titled 'Test Cases' and lists three entries:

- Creating New Item**: Need to create some item detail. Pre-Conditions: User is logged in. Test Steps: 1: Step 1 C content, 2: Step 2 C content, 3: Step 3 C content. Post-Conditions: Item is created.
- Reading New Item**: Need to Reading some item detail.
- Updating New Item**: Need to Updating some item detail.

Рисунок 3.6 – Інтерфейс сторінки тест-кейсів

На сторінці можна переглянути вже створені тест-кейси, створити нові, а також відредактувати та видалити тест-кейси за потреби користувача.

Наступним кроком створимо тест-сьют. Для цього потрібно перейти на вкладку “Test Suites” та натиснути кнопку «+». Заповнимо назву та опис, а також оберемо потрібні до цього сьюту тест-кейси. Покажемо результат на рисунку нижче:

Змін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.02.81	Арк.
						55

The screenshot shows the 'TEST RESULT APP' interface at localhost:3000/test_suites. The top navigation bar includes 'PROJECTS', 'TEST CASES', 'TEST SUITES', and 'TEST RUNS'. A green header bar displays 'Active project: Central Database' and 'Welcome, Hennadii Kochev !'. Below the header, a section titled 'Test Suites' lists two entries:

- Test suite 1 name:** Suite 1 description
 - Case: Test case 1 name
 - About: Some1 Description
- Test suite 2 name:** Suite 2 description
 - Case: Test case 2 name
 - About: Some2 Description
- Test suite 3 name:** Suite 3 description
 - Case: Test case 3 name
 - About: Some3 Description
- Test suite 4 name:** Suite 4 description
 - Case: Test case 4 name
 - About: Some4 Description

Each entry has a 'CASE+' button with a plus sign, a blue circular arrow icon, a pencil icon, and a delete icon.

Рисунок 3.7 – Інтерфейс сторінки тест-сьютів

Тепер проведемо Test Run для фіксації результатів по кожному тест-кейсу, що входить у тест-сьют. Для цього на сторінці тест-сьютів оберемо створений тест-сьют та натиснемо кнопку «Run». Встановимо для кожного тест-кейсу свій статус з наданих для вибору.

Зафіксуємо проведений тест-ран за допомогою кнопки підтвердження. Тепер маючи готовий тест-ран, з нього можна згенерувати звіт у форматі PDF. Перейдемо на сторінку “Test Runs”, оберемо наш створений тест-ран та навпроти нього натиснемо кнопку «PDF». Нижче наведено екран сторінки тест-ранів.

Змін.	Арк.	№ докум.	Підп.	Дата.

The screenshot shows a web application titled "TEST RESULT APP" with a blue header bar. The header includes a logo, the title, and navigation links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS". A user profile icon is also present. Below the header, a green banner displays "Active project: Central Database" and "Welcome, Hennadii Kochev !". The main content area is titled "Test Runs" and contains two entries. The first entry is for a run on "Test suite name" at "2023-05-01 12:34:56" with a "Successful" result. It lists two test cases: "Test case 1 name" with "Some Description" and "Successful" status, and "Test case 2 name" with "Some 2 Description" and "Successful" status. A "PDF" button is available for this entry. The second entry is for a run on "Test suite 2 name" at "2023-05-02 22:34:56" with a "Re-Test" result. A "PDF" button is also present for this entry.

Рисунок 3.8 – Інтерфейс сторінки Test Runs

Отримуємо PDF-звіт, що містить інформацію про тест-ран, приклад звіту нижче на рисунку:

Змін.	Арк.	№ докум.	Підп.	Дата.		КПІ.ІП-9113.045440.02.81		Арк.
								57

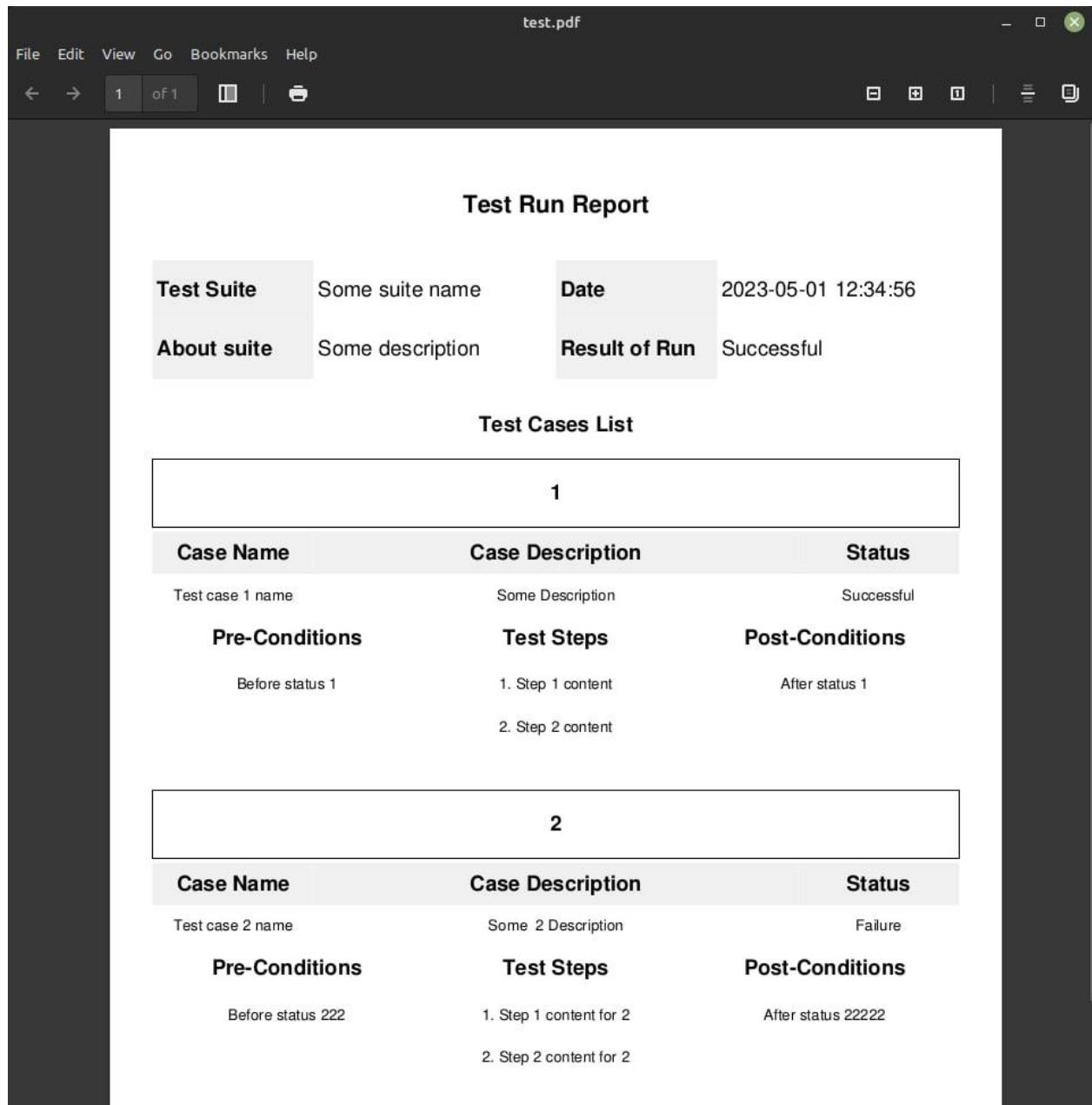


Рисунок 3.9 – Приклад згенерованого PDF-звіту

Таким чином, цей процес являє собою основний спосіб використання дипломного програмного забезпечення та виконується успішно.

Змін.	Арк.	№ докум.	Підп.	Дата.

Висновки до розділу

У ході виконання третього розділу дипломного проекту було проведено перевірку якості роботи та функціонування веб-застосунку.

Перш за все, було наведено базову теорію щодо аналізу якості ПЗ, і для перевірки якості сирцевого коду було використано статичний аналізатор для репозиторіїв коду Embold. Цей веб-додаток надає оцінку щодо присутності проблем у коді, а також наявності антипаттернів та повторів коду тощо. Було продемонстровано результати аналізу для серверної та клієнтської частин коду.

Наступним кроком стала перевірка функціоналу написаного програмного забезпечення, для цього було проведено мануальне тестування усіх функціональних вимог до веб-застосунку, як результат маємо повну відповідность роботи до заданих вимог.

Після цього було наведено контрольний приклад використання створеного ПЗ, що є основним способом його застосування. Процес складається з таких кроків:

- реєстрація користувача;
- авторизація користувача;
- створення проекту, встановлення його активним;
- створення деяких тест-кейсів;
- групування тест-кейсів до тест-сьютів;
- проведення тест-рану на базі тест-сьюту;
- генерація звіту з проведеного тест-рану.

Таким чином, веб-застосунок успішно пройшов тестування, відповідає поставленим вимогам, та має зазначені показники якості коду.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для роботи з готовим застосунком його потрібно розгорнути.

Перечислимо основні залежності проекту, що повинні бути встановлені:

- РСУБД PostgreSQL 12+ версії;
- Python 3.8+, pip 22+;
- Node.js 18 LTS, npm 9+.

Для запуску потрібно стягнути останні версії репозиторіїв клієнтської та серверної частин програмного забезпечення з платформи GitHub.

Наведемо нижче посилання на репозиторії:

- клієнт: https://github.com/genndy007/test_result_app_frontend ;
- сервер: https://github.com/genndy007/test_result_app_backend .

Перш за все потрібно створити базу даних для збереження інформації застосунку. Створення відбувається командою:

```
createdb -U postgres test_result_app_db
```

Детальніше про розгортання серверної частини:

- 1) встановити залежності-бібліотеки за допомогою команд:
 - a) pip install poetry;
 - b) poetry install;
- 2) вказати необхідні ключі у файлі .env (шлях до серверу PostgreSQL, секретний API-ключ для перевірки доступу та інше);
- 3) запустити командою flask run.

Також потрібно запустити чергу завдань для генерації звітів за допомогою команди:

```
celery -A report_app worker --loglevel=INFO -n worker1@%h
```

Детальніше про розгортання клієнтської частини:

- 1) потрібно встановити залежності-бібліотеки за допомогою команди
npm install;

Змін.	Арк.	№ докум.	Підп.	Дата.

- 2) вказати у файлі .env посилання на серверну частину та API-ключ для підтвердження валідності фронтенду;
- 3) запустити за допомогою команди npm start.

Також зазначимо, що цей веб-застосунок можливо розгорнути у веб-середовищі для публічного доступу користувачів у Інтернеті. Для цього потрібно скористатися хмарними сервісами, а саме:

- ElephantSQL – сервіс, що надає можливість безкоштовно розгорнути РСУБД PostgreSQL з мінімальною конфігурацією;
- PythonAnywhere – сервіс, що дозволяє безкоштовно розгорнути Flask-застосунок серверної частини коду;
- Render – веб-сервіс для хостингу React.js застосунку клієнтської частини коду, що також дозволяє утримувати Celery воркери для генерації звітів;
- Freenom – реєстратор безкоштовних доменних імен для веб-застосунків.

4.2 Підтримка програмного забезпечення

Подальша розробка веб-застосунку буде продовжена для введення нового функціоналу, який може стати у нагоді потенціальним користувачам. Тому в користувачів повинна бути можливість отримати нову версію програмного забезпечення. Кожна нова версія зі змінами буде доставлена до відповідного репозиторію на GitHub у гілку master, та комміт буде мати тег формату “v*.*.*”.

У випадку клієнтської частини застосунку нові версії зібраного застосунку для допомогою команди npm run build будуть розміщуватись на сторінці репозиторію у GitHub, в секції Releases, де версія буде відповідати тегу комміта за форматом, зазначеним вище.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Висновки до розділу

У ході виконання четвертого розділу дипломного проекту було проведено опис процесу розгортання програмного забезпечення, що було створено, а саме запуск серверної та клієнтської частин програми.

Найважливіші моменти розгортання – необхідність мати на комп’ютері встановленими PostgreSQL, Python, Node.js, оскільки вони є базою, на якій ґрунтуються проект. Також потрібно завантажити серверну та клієнтську частини коду, встановити залежності, налаштувати конфігурацію секретних значень у файлі змінних оточення та запустити сервер з клієнтом. Було також описано, що цей веб-застосунок за допомогою хмарних сервісів можливо розгорнути для публічного доступу з Інтернету.

Наступним кроком була деталізація підтримки створеного веб-застосунку. Оскільки код застосунку зберігається у публічних репозиторіях на платформі GitHub, то нові версії будуть додаватися до центральної гілки master з тегом версії, а також на боці клієнтської частини будуть додаватися зібрани нові версії у вкладку Releases на сторінці репозиторію.

Таким чином, було проведено розгортання веб-застосунку з його подальшою підтримкою у розробці.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

ВИСНОВКИ

В результаті виконання дипломного проекту було реалізовано веб-застосунок для моніторингу та обробки результатів тестувань програмного забезпечення.

Оцінюючи результати роботи, можна зазначити, що веб-застосунок відповідає сучасному рівню наукових і технічних знань у галузі створення програмного забезпечення. Він успішно пройшов тестування та відповідає всім поставленим вимогам.

Ступінь впровадження та можливі галузі використання результатів роботи полягають у використанні веб-застосунку для створення та управління тест-кейсами в різних сферах діяльності, де необхідно забезпечити якість програмного забезпечення. Веб-застосунок може бути застосований у компаніях, які займаються розробкою програмного забезпечення, а також у будь-яких проектах, де необхідно проводити тестування та аналіз якості програмного забезпечення.

Наукова, технічна та соціально-економічна значущість роботи полягають у поліпшенні процесу тестування та контролю якості програмного забезпечення, що може привести до зниження кількості помилок та збільшення задоволеності користувачів. Розроблений веб-застосунок спрощує та автоматизує процеси створення та управління тест-кейсами, що покращує продуктивність тестилюльників програмного забезпечення.

В ході розгортання програмного забезпечення було показано, що веб-застосунок може бути успішно запущений на комп'ютерах з встановленими PostgreSQL, Python та Node.js. Його також можна розгорнути для публічного доступу з Інтернету за допомогою хмарних сервісів.

Щодо підтримки створеного веб-застосунку, було вказано, що нові версії будуть додаватися до центральної гілки master з тегом версії, а також будуть доступні у вкладці Releases на сторінці репозиторію. Це забезпечує

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

зручний спосіб оновлення та розповсюдження нових версій програмного забезпечення.

Розроблений веб-застосунок успішно відповідає поставленим вимогам, пройшов тестування та демонструє високу якість коду. Впровадження веб-застосунку може поліпшити процеси тестування та контролю якості програмного забезпечення в різних галузях. Продовження досліджень у цій тематиці може включати розширення функціональності веб-застосунку, удосконалення процесу розгортання та підтримки, а також дослідження нових методів тестування та аналізу якості програмного забезпечення.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.02.81

Арк.

64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Whittaker J. A. How to Break Software: A Practical Guide to Testing / James A. Whittaker., 2002. – 208 c. – (1st edition).
- 2) Weinberg G. M. Computer Programming Fundamentals / G. M. Weinberg, H. D. Leeds., 1961. – 368 c.
- 3) Volokh E. Test Suites: A Tool for Improving Student Articles [Електронний ресурс] / Eugene Volokh // Journal of Legal Education. – 2002. – Режим доступу до ресурсу: <https://www.jstor.org/stable/42893766>.
- 4) Northwood C. The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer / Chris Northwood. – Pune, India: Apress, 2018. – 365 c. – (1st ed.).
- 5) Lutz M. Learning Python / Mark Lutz. – Sebastopol, California: O'Reilly Media, 2013. – 1643 c. – (Fifth edition).
- 6) Radha S. Comparative analysis of bug tracking tools [Електронний ресурс] / S. Radha, S. Bhattacharya, D.S. Jat // International Journal of Pharmacy and Technology. – 2016. – Режим доступу до ресурсу: https://www.researchgate.net/publication/316888056_Comparative_analysis_of_bug_tracking_tools.
- 7) Korab J. Understanding Message Brokers / Jakub Korab. – Sebastopol, California: O'Reilly Media, 2017. – 70 c.
- 8) Rucker B. Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company / B. Rucker, J. Freund. – Scotts Valley, California: CreateSpace Independent Publishing Platform, 2012. – 232 c. – (1st edition).
- 9) Richards M. Software Architecture Patterns / Mark Richards. – Sebastopol, California: O'Reilly Media, Inc., 2015. – 47 c. – (1st ed.).
- 10) Richardson L. RESTful Web APIs: Services for a Changing World / L. Richardson, M. Amundsen, S. Ruby., 2013. – 406 c. – (1st edition).

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9113.045440.02.81

Арк.

65

- 10) Obe R. PostgreSQL: Up and Running / R. Obe, L. Hsu., 2017. – 312 c. – (3rd edition).
- 11) Clarke-Salt J. SQL Injection Attacks and Defense / Justin Clarke-Salt. – Oxford, UK: Syngress, 2012. – 576 c. – (2nd Edition).
- 12) Henthorn M. Detection and Prevention of Cross-Site Request Forgery Attacks: A Browser-Side Solution / Mary Henthorn. – Saarbrucken, Germany: VDM Verlag Dr. Müller, 2008. – 36 c.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.02.81

Арк.

66

ДОДАТКИ

ДОДАТОК А. Звіт про перевірку на співпадіння



Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015404637

Дата перевірки:
03.06.2023 11:10:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 11:17:18 EEST

ID користувача:
76913

Назва документа: ІП-91_Кочев_ПЗ

Кількість сторінок: 60 Кількість слів: 8586 Кількість символів: 67215 Розмір файлу: 1.99 MB ID файлу: 1015068311

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.3% Схожість

Найбільша схожість: 6.38% з джерелом з Бібліотеки (ID файлу: 1015067687)

4.5% Джерела з Інтернету 182 Сторінка 62

13.2% Джерела з Бібліотеки 260 Сторінка 65

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 6

Підозріле форматування 12 сторінок

Змін.	Арк.	№ докум.	Підп.	Дата.

КПІ.ІП-9113.045440.02.81

Арк.

67

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

Едуард ЖАРИКОВ

“ ____ ” _____ 2023 р.

**ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ
РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Текст програми

КПІ.ІП-9113.045440.03.12

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Геннадій КОЧЕВ

Київ – 2023

Клієнтська частина коду

Файл public/index.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Test Result App</title>
</head>
<body>
  <noscript>You need JS to run this app.</noscript>
  <div id="root"></div>
</body>
</html>
```

Файл src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";
import App from "./App";

ReactDOM.render(
  <BrowserRouter>
    <App/>
  </BrowserRouter>,
  document.getElementById("root")
)
```

Файл src/pages/ProjectsPage.js

```
import React from 'react';
import ProjectsTable from "../components/tables/ProjectsTable";
import {projects} from "../utils/TestingItems";
import PageHeadingAppBar from "../components/appbars/PageHeadingAppBar";

const ProjectsPage = () => {
  return (
    <div className="projects">
      <PageHeadingAppBar name={"Projects"} />
      <ProjectsTable projects={projects}></ProjectsTable>
    </div>
  );
};

export default ProjectsPage;
```

Файл src/pages/SignInPage.js

```
import * as React from 'react';
import { useNavigate } from "react-router-dom";

import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

import TextField from '@mui/material/TextField';
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import { createTheme, ThemeProvider } from '@mui/material/styles';

import MyLink from "../components/appbars/MainAppBar/MyLink";
import Copyright from "../components/common/Copyright";

const defaultTheme = createTheme();

const SignInPage = ({loginFunc}) => {
  let navigate = useNavigate();
  const handleSubmit = (event) => {
    event.preventDefault();
    const data = new FormData(event.currentTarget);
    loginFunc();
    navigate('/settings');
    console.log({
      email: data.get('email'),
      password: data.get('password'),
    });
  };
}

return (
  <ThemeProvider theme={defaultTheme}>
    <Container component="main" maxWidth="xs">
      <CssBaseline />
      <Box
        sx={{
          marginTop: 8,
          display: 'flex',
          flexDirection: 'column',
          alignItems: 'center',
        }}
      >
        <Avatar sx={{ m: 1, bgcolor: '#008080' }}>
          <LockOutlinedIcon />
        </Avatar>
        <Typography component="h1" variant="h5">
          Sign in
        </Typography>
        <Box component="form" onSubmit={handleSubmit} noValidate sx={{ mt: 1 }}>
          <TextField
            margin="normal"
            required
            fullWidth
            id="email"
            label="Email Address"
            name="email"
            autoComplete="email"
            autoFocus
          />
          <TextField
            margin="normal"
            required
            fullWidth
            name="password"
            label="Password"
          />
        </Box>
      </Box>
    </Container>
  </ThemeProvider>
)

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        type="password"
        id="password"
        autoComplete="current-password"
      />
      <FormControlLabel
        control={<Checkbox value="remember" color="primary" />}
        label="Remember me"
      />
      <Button
        type="submit"
        fullWidth
        variant="contained"
        sx={{ mt: 3, mb: 2 }}
      >
        Log In
      </Button>
    </Box>

    <Box>
      <Button variant="outlined">
        <MyLink color="#1976D2" to="/signup" name="Don't have an
account? Sign up"/>
      </Button>
    </Box>
  </Box>
  <Copyright sx={{ mt: 8, mb: 4 }} />
</Container>
</ThemeProvider>
);
}

export default SignInPage;

```

Файл src/pages/SignUpPage.js

```

import * as React from 'react';
import { useNavigate } from "react-router-dom";

import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import { createTheme, ThemeProvider } from '@mui/material/styles';

import Copyright from "../components/common/Copyright";
import MyLink from "../components/appbars/MainAppBar/MyLink";

const defaultTheme = createTheme();

const SignUpPage = () => {
  let navigate = useNavigate();
  const handleSubmit = (event) => {
    event.preventDefault();

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        const data = new FormData(event.currentTarget);
        navigate('/login');
        console.log({
            email: data.get('email'),
            password: data.get('password'),
        });
    };

    return (
        <ThemeProvider theme={defaultTheme}>
            <Container component="main" maxWidth="xs">
                <CssBaseline />
                <Box
                    sx={{
                        marginTop: 8,
                        display: 'flex',
                        flexDirection: 'column',
                        alignItems: 'center',
                    }}
                >
                    <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
                        <LockOutlinedIcon />
                    </Avatar>
                    <Typography component="h1" variant="h5">
                        Create an account
                    </Typography>
                    <Box component="form" noValidate onSubmit={handleSubmit} sx={{ mt: 3 }}>
                        <Grid container spacing={2}>
                            <Grid item xs={12} sm={6}>
                                <TextField
                                    autoComplete="given-name"
                                    name="firstName"
                                    required
                                    fullWidth
                                    id="firstName"
                                    label="First Name"
                                    autoFocus
                                />
                            </Grid>
                            <Grid item xs={12} sm={6}>
                                <TextField
                                    required
                                    fullWidth
                                    id="lastName"
                                    label="Last Name"
                                    name="lastName"
                                    autoComplete="family-name"
                                />
                            </Grid>
                            <Grid item xs={12}>
                                <TextField
                                    required
                                    fullWidth
                                    id="email"
                                    label="Email Address"
                                    name="email"
                                    autoComplete="email"
                                />
                            </Grid>
                            <Grid item xs={12}>
                                <TextField
                                    required
                                    fullWidth
                                    id="password"
                                    label="Password"
                                    name="password"
                                    type="password"
                                />
                            </Grid>
                        </Grid>
                    </Box>
                </Box>
            </Container>
        </ThemeProvider>
    );
}

export default Register;

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        name="password"
        label="Password"
        type="password"
        id="password"
        autoComplete="new-password"
    />
</Grid>
<Grid item xs={12}>
    <FormControlLabel
        control={<Checkbox value="allowExtraEmails" color="primary">
/>
        label="I agree to the Rules of Test Result App usage."
    />
</Grid>
</Grid>
<Button
    type="submit"
    fullWidth
    variant="contained"
    sx={{ mt: 3, mb: 2 }}
>
    Sign Up
</Button>
</Box>

<Box>
    <Button variant="outlined">
        <MyLink color="#1976D2" to="/login" name="Already have account? Log in"/>
    </Button>
</Box>

</Box>
<Copyright sx={{ mt: 5 }} />
</Container>
</ThemeProvider>
);
}

export default SignUpPage;

```

Файл src/pages/TestCasesPage.js

```

import React from 'react';

import {testCases} from "../utils/TestingItems";
import TestCaseAccordionList from
"../components/accordions/TestCaseAccordionList";
import PageHeadingAppBar from "../components/appbars/PageHeadingAppBar";


const TestCasesPage = () => {
    return (
        <div className="testCases">
            <PageHeadingAppBar name="Test Cases"/>
            <TestCaseAccordionList testCases={testCases}/>
        </div>
    );
};

export default TestCasesPage;

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

Файл src/pages/TestRunsPage.js

```
import React from 'react';

import {testRuns} from "../utils/TestingItems";
import TestRunAccordionList from
"../components/accordions/TestRunAccordionList";
import PageHeadingAppBar from "../components/appbars/PageHeadingAppBar";


const TestRunsPage = () => {
    return (
        <div className="testRuns">
            <PageHeadingAppBar name="Test Runs"/>
            <TestRunAccordionList testRuns={testRuns}/>
        </div>
    );
};

export default TestRunsPage;
```

Файл src/pages/TestSuitesPage.js

```
import React from 'react';
import PageHeadingAppBar from "../components/appbars/PageHeadingAppBar";
import TestSuiteAccordionList from
"../components/accordions/TestSuiteAccordionList";
import {testSuites} from "../utils/TestingItems";


const TestSuitesPage = () => {
    return (
        <div className="testSuites">
            <PageHeadingAppBar name="Test Suites"/>
            <TestSuiteAccordionList testSuites={testSuites}/>
        </div>
    );
};

export default TestSuitesPage;
```

Файл src/apps/TestResultApp.js

```
import React from 'react';
import ActiveProjectUserAppBar from
"../components/appbars/ActiveProjectUserAppBar";
import {projects} from "../utils/TestingItems";
import MainAppBar from "../components/appbars/MainAppBar/MainAppBar";
import TestResultAppRouter from "../components/routers/TestResultAppRouter";


const TestResultApp = ({logoutFunc}) => {
    return (
        <div className="TestResultApp">
            <ActiveProjectUserAppBar projectName={projects[0].name}
userNames="Hennadii Kochev"/>
            <br/>
            <MainAppBar logoutFunc={logoutFunc}/>
            <br/>
            <TestResultAppRouter/>
        </div>
    );
};
```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        );
    };

    export default TestResultApp;

```

Файл src/apps/AuthApp.js

```

import React from 'react';
import AuthAppRouter from "../components/routers/AuthAppRouter";

const AuthApp = ({loginFunc}) => {
    return (
        <div className="AuthApp">
            <AuthAppRouter loginFunc={loginFunc}/>
        </div>
    );
};

export default AuthApp;

```

Файл src/components/accordions/TestCaseAccordion.jsx

```

import * as React from 'react';
import Accordion from '@mui/material/Accordion';
import AccordionDetails from '@mui/material/AccordionDetails';
import AccordionSummary from '@mui/material/AccordionSummary';
import Typography from '@mui/material/Typography';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';
import {Fragment} from "react";
import Box from "@mui/material/Box";

const TestCaseAccordionContent = ({testCase}) => {
    return (
        <Fragment sx={{display: "flex"}}>
            <Box sx={{display: 'flex', flexDirection: 'column'}}>
                <Typography sx={{fontWeight: 700 , mb: 2}}>Pre-Conditions</Typography>
                <Typography sx={{fontWeight: 400}}>{testCase.precondition}</Typography>
            </Box>
            <Box sx={{display: 'flex', flexDirection: 'column'}}>
                <Typography sx={{fontWeight: 700 , mb: 2}}>Test Steps</Typography>
                {testCase.testSteps.map((testStep) => (
                    <Typography>{testStep.order}: {testStep.content}</Typography>
                )));
            </Box>
            <Box sx={{display: 'flex', flexDirection: 'column'}}>
                <Typography sx={{fontWeight: 700 , mb: 2}}>Post-Conditions</Typography>
                <Typography sx={{fontWeight: 400}}>{testCase.postcondition}</Typography>
            </Box>
        </Fragment>
    );
}

const TestCaseAccordion = ({testCase, open, onChange}) => {
    return (
        <Accordion expanded={open === testCase.id}
        onChange={onChange(testCase.id)}>

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

<AccordionSummary
  expandIcon={<ExpandMoreIcon />}
  aria-controls="panel1bh-content"
  id="panel1bh-header"
>
  <Typography sx={{ width: '33%', flexShrink: 0 }}>
    {testCase.name}
  </Typography>
  <Typography sx={{ color: 'text.secondary' }}>{testCase.description}</Typography>
</AccordionSummary>
<AccordionDetails sx={{display: 'flex', justifyContent: 'space-between'}}>
  <TestCaseAccordionContent testCase={testCase}/>
</AccordionDetails>
</Accordion>
);
};

export default TestCaseAccordion;

```

Файл src/App.js

```

import * as React from "react";
import TestResultApp from "./apps/TestResultApp";
import AuthApp from "./apps/AuthApp";

function App() {
  const [isAuthenticated, setIsAuthenticated] = React.useState(true);
  const logoutUser = () => {setIsAuthenticated(false)}
  const loginUser = () => {setIsAuthenticated(true)}
  return (
    <div className="App">
      {isAuthenticated ? <TestResultApp logoutFunc={logoutUser}/> :
<AuthApp loginFunc={loginUser}/>}
    </div>
  );
}

export default App;

```

Файл src/components/accordions/TestCaseAccordionList.jsx

```

import React, {Fragment} from 'react';
import TestCaseAccordion from "./TestCaseAccordion";
import Button from "@mui/material/Button";
import EditIcon from "@mui/icons-material/Edit";
import DeleteForeverIcon from "@mui/icons-material/DeleteForever";
import Table from "@mui/material/Table";
import TableBody from '@mui/material/TableBody';
import TableRow from '@mui/material/TableRow';
import TableCell from '@mui/material/TableCell';

const TestCaseAccordionList = ({testCases}) => {
  const [open, setOpen] = React.useState(false);

  const accordionChange = (card) => (e, isOpen) => {
    setOpen(isOpen ? card : false);
  };

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        return (
            <Fragment>
                <Table>
                    <TableBody>
                        {testCases.map((testCase) => (
                            <TableRow>
                                <TableCell style={{width: '90%'}}>
                                    <TestCaseAccordion testCase={testCase} open={open}>
                                        onChange={accordionChange}/>
                                </TableCell>
                                <TableCell>
                                    <Button><EditIcon/></Button>
                                    <Button><DeleteForeverIcon/></Button>
                                </TableCell>
                            </TableRow>
                        ))}
                    </TableBody>
                </Table>
            </Fragment>
        );
    };

export default TestCaseAccordionList;

```

Файл src/components/accordions/TestRunAccordion.jsx

```

import * as React from 'react';
import Accordion from '@mui/material/Accordion';
import AccordionDetails from '@mui/material/AccordionDetails';
import AccordionSummary from '@mui/material/AccordionSummary';
import Typography from '@mui/material/Typography';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';
import {Fragment} from "react";
import Box from "@mui/material/Box";
import TestRunContentTable from "../tables/TestRunContentTable";

const TestRunAccordion = ({testRun, open, onChange}) => {
    return (
        <Accordion expanded={open === testRun.id}
            onChange={onChange(testRun.id)}>
            <AccordionSummary
                expandIcon={<ExpandMoreIcon />}
                aria-controls="panel1bh-content"
                id="panel1bh-header"
            >
                <Typography sx={{ width: '67%', flexShrink: 0 }}>
                    Run for '{testRun.testSuite.name}' at {testRun.timestamp}
                </Typography>
                <Typography sx={{ color: 'text.secondary' }}>Result:
{testRun.result}</Typography>
            </AccordionSummary>
            <AccordionDetails>
                <TestRunContentTable testRun={testRun}/>
            </AccordionDetails>
        </Accordion>
    );
};

export default TestRunAccordion;

```

Файл src/components/accordions/TestRunAccordionList.jsx

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

import React, {Fragment} from 'react';
import TestRunAccordion from "./TestRunAccordion";
import Box from "@mui/material/Box";
import Button from "@mui/material/Button";
import PictureAsPdfIcon from '@mui/icons-material/PictureAsPdf';
import Table from "@mui/material/Table";
import TableBody from '@mui/material/TableBody';
import TableRow from '@mui/material/TableRow';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';

const TestRunAccordionList = ({testRuns}) => {
  const [open, setOpen] = React.useState(false);

  const accordionChange = (card) => (e, isOpen) => {
    setOpen(isOpen ? card : false);
  };

  return (
    <Fragment>
      <Table>
        <TableBody>
          {testRuns.map((testRun) => (
            <TableRow>
              <TableCell style={{width: '90%'}}>
                <TestRunAccordion testRun={testRun} open={open}
onChange={accordionChange}/>
              </TableCell>
              <TableCell>
                <Button><PictureAsPdfIcon/></Button>
              </TableCell>
            </TableRow>
          )));
        </TableBody>
      </Table>
    </Fragment>
  );
};

export default TestRunAccordionList;

```

Файл src/components/accordions/TestSuiteAccordion.jsx

```

import * as React from 'react';
import Accordion from '@mui/material/Accordion';
import AccordionDetails from '@mui/material/AccordionDetails';
import AccordionSummary from '@mui/material/AccordionSummary';
import Typography from '@mui/material/Typography';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

import Box from "@mui/material/Box";
import Grid from '@mui/material/Grid';

import {styled} from "@mui/material/styles";
import Paper from "@mui/material/Paper";

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: theme.palette.mode === 'dark' ? '#1A2027' : '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: theme.palette.text.secondary,
})

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        });

const TestCasesGrid = ({testCases}) => {
    return (
        <Box sx={{ flexGrow: 1 }}>
            <Grid container spacing={2}>
                {testCases.map((testCase) => (
                    <Grid item xs={6}>
                        <Item>
                            Case: {testCase.name}
                            <br/>
                            <br/>
                            About: {testCase.description}
                        </Item>

                    </Grid>
                )));
            </Grid>
        </Box>
    );
}

const TestSuiteAccordion = ({testSuite, open, onChange}) => {
    return (
        <Accordion expanded={open === testSuite.id}
        onChange={onChange(testSuite.id)}>
            <AccordionSummary
                expandIcon={<ExpandMoreIcon />}
                aria-controls="panel1bh-content"
                id="panel1bh-header"
            >
                <Typography sx={{ width: '33%', flexShrink: 0 }}>
                    {testSuite.name}
                </Typography>
                <Typography sx={{ width: '40%', flexShrink: 0 }}>
                    {testSuite.description}
                </Typography>
            </AccordionSummary>
            <AccordionDetails>
                <TestCasesGrid testCases={testSuite.testCases}/>
            </AccordionDetails>
        </Accordion>
    );
};

export default TestSuiteAccordion;

```

Файл src/components/accordions/TestSuiteAccordionList.jsx

```

import React, {Fragment} from 'react';
import Button from "@mui/material/Button";
import EditIcon from "@mui/icons-material/Edit";
import DeleteForeverIcon from "@mui/icons-material/DeleteForever";
import PlayCircleOutlineIcon from '@mui/icons-material/PlayCircleOutline';
import Table from "@mui/material/Table";
import TableBody from '@mui/material/TableBody';
import TableRow from '@mui/material/TableRow';
import TableCell from "@mui/material/TableCell";
import TestSuiteAccordion from "./TestSuiteAccordion";

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

const TestSuiteAccordionList = ({testSuites}) => {
  const [open, setOpen] = React.useState(false);

  const accordionChange = (card) => (e, isOpen) => {
    setOpen(isOpen ? card : false);
  };

  return (
    <Fragment>
      <Table>
        <TableBody>
          {testSuites.map((testSuite) => (
            <TableRow>
              <TableCell style={{width: '90%'}}>
                <TestSuiteAccordion testSuite={testSuite} open={open}>
                  onChange={accordionChange}/>
              </TableCell>
              <TableCell>
                <Button>Case+</Button>
                <Button><PlayCircleOutlineIcon/></Button>
              </TableCell>
              <TableCell>
                <Button><EditIcon/></Button>
                <Button><DeleteForeverIcon/></Button>
              </TableCell>
            </TableRow>
          )))
        </TableBody>
      </Table>
    </Fragment>
  );
};

export default TestSuiteAccordionList;

```

Файл src/components/appbars/MainAppBar/MainAppBar.jsx

```

import * as React from 'react';

import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
import Toolbar from '@mui/material/Toolbar';
import IconButton from '@mui/material/IconButton';
import Typography from '@mui/material/Typography';
import Menu from '@mui/material/Menu';
import MenuItem from '@mui/icons-material/Menu';
import Container from '@mui/material/Container';
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import Tooltip from '@mui/material/Tooltip';
import MyLink from "./MyLink";
import MyLogo from "./MyLogo";

import {pages, settings} from '../../../../../utils/TestingItems'

import userIcon from '../../../../../assets/images/user-icon.png'
import MyLink from "./MyLink";
import MyLogo from "./MyLogo";

function MainAppBar({logoutFunc}) {
  const [anchorElNav, setAnchorElNav] = React.useState(null);

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        const [anchorElUser, setAnchorElUser] = React.useState(null);

        const handleOpenNavMenu = (event) => { setAnchorElNav(event.currentTarget) };
        const handleOpenUserMenu = (event) => {
            setAnchorElUser(event.currentTarget) };

        const handleCloseNavMenu = () => { setAnchorElNav(null) };
        const handleCloseUserMenu = () => { setAnchorElUser(null) };

        return (
            <AppBar position="static">
                <Container maxWidth="xl">
                    <Toolbar disableGutters>
                        <MyLogo xs={false}/>

                        <Box sx={{ flexGrow: 1, display: { xs: 'flex', md: 'none' } }}>
                            <IconButton
                                size="large"
                                aria-label="account of current user"
                                aria-controls="menu-appbar"
                                aria-haspopup="true"
                                onClick={handleOpenNavMenu}
                                color="inherit"
                            >
                                <MenuIcon />
                            </IconButton>
                            <Menu
                                id="menu-appbar"
                                anchorEl={anchorElNav}
                                anchorOrigin={{
                                    vertical: 'bottom',
                                    horizontal: 'left',
                                }}
                                keepMounted
                                transformOrigin={{
                                    vertical: 'top',
                                    horizontal: 'left',
                                }}
                                open={Boolean(anchorElNav)}
                                onClose={handleCloseNavMenu}
                                sx={{
                                    display: { xs: 'block', md: 'none' },
                                }}
                            >
                                {pages.map((page) => (
                                    <MenuItem key={page.name} onClick={handleCloseNavMenu}>
                                        <Typography textAlign="center">
                                            <MyLink color="black" to={page.path} name={page.name}/>
                                        </Typography>
                                    </MenuItem>
                                )));
                            </Menu>
                        </Box>

                        <MyLogo xs={true} flexGrow={1}/>
                        <Box sx={{ flexGrow: 1, display: { xs: 'none', md: 'flex' } }}>
                            {pages.map((page) => (
                                <Button
                                    key={page.name}
                                    onClick={handleCloseNavMenu}
                                    sx={{ my: 2, color: 'white', display: 'block' }}
                                >
                                    <MyLink color="white" to={page.path} name={page.name}/>
                                </Button>
                            )));
                        </Box>
                </Toolbar>
            </Container>
        );
    }

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        </Button>
    ))
</Box>

<Box sx={{ flexGrow: 0 }}>
    <Tooltip title="User Options">
        <IconButton onClick={handleOpenUserMenu} sx={{ p: 0 }}>
            <Avatar alt="User" src={userIcon} />
        </IconButton>
    </Tooltip>
    <Menu
        sx={{ mt: '45px' }}
        id="menu-appbar"
        anchorEl={anchorElUser}
        anchorOrigin={{{
            vertical: 'top',
            horizontal: 'right',
        }}}
        keepMounted
        transformOrigin={{{
            vertical: 'top',
            horizontal: 'right',
        }}}
        open={Boolean(anchorElUser)}
        onClose={handleCloseUserMenu}
    >
        {settings.map((setting) => (
            <MenuItem key={setting.name} onClick={setting.name === 'Log
Out' ? logoutFunc : handleCloseUserMenu}>
                <Typography textAlign="center">
                    <MyLink color="black" to={setting.path}
name={setting.name}/>
                </Typography>
            </MenuItem>
        )))
    </Menu>
</Box>
</Toolbar>
</Container>
</AppBar>
);
}
export default MainAppBar;

```

Файл src/components/appbars/MainAppBar/MyLink.jsx

```

import React from 'react';
import {Link} from "react-router-dom";

const MyLink = ({color, name, to}) => {
    return (
        <Link style={{textDecoration: "none", color: color}}
to={to}>{name}</Link>
    );
};

export default MyLink;

```

Файл src/components/appbars/MainAppBar/MyLogo.jsx

Вмін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.03.12	Арк.
						15

```

import React, {Fragment} from 'react';
import BugReportIcon from "@mui/icons-material/BugReport";
import Typography from "@mui/material/Typography";

const MyLogo = ({xs, flexGrow}) => {
  const display = xs ? {xs: 'flex', md: 'none'} : {xs: 'none', md: 'flex'};
  return (
    <Fragment>
      <BugReportIcon sx={{ display: display, mr: 1 }} />
      <Typography
        variant="h6"
        noWrap
        sx={{
          mr: 2,
          display: display,
          fontFamily: 'Helvetica',
          fontWeight: 700,
          flexGrow: flexGrow,
          letterSpacing: '.1rem',
          color: 'inherit',
          textDecoration: 'none',
        }}
      >
        TEST RESULT APP
      </Typography>
    </Fragment>
  );
};

export default MyLogo;

```

Файл src/components/appbars/ActiveProjectUserAppBar.jsx

```

import React from 'react';
import AppBar from "@mui/material/AppBar";
import Container from '@mui/material/Container';
import Toolbar from '@mui/material/Toolbar';
import Typography from "@mui/material/Typography";

const HeadingTypography = ({row1, row2}) => {
  return (
    <Typography
      noWrap
      sx={{
        mr: 2,
        fontFamily: 'Helvetica',
        fontWeight: 500,
        color: 'inherit',
        textDecoration: 'none',
      }}
    >
      {row1} <br/> {row2}
    </Typography>
  );
}

const ActiveProjectUserAppBar = ({projectName, userName}) => {
  return (
    <AppBar position="static" sx={{ bgcolor: "green" }}>
      <Container>
        <Toolbar disableGutters sx={{ justifyContent: "space-between" }}>

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        <HeadingTypography row1="Active project:" row2={projectName} />
        <HeadingTypography row1="Welcome," row2={`${userName} !`} />
    </Toolbar>
</Container>
</AppBar>
);
};

export default ActiveProjectUserAppBar;

```

Файл src/components/appbars/ PageHeadingAppBar.jsx

```

import React, {Fragment} from 'react';

import AppBar from "@mui/material/AppBar";
import Typography from "@mui/material/Typography";
import AddCircleIcon from '@mui/icons-material/AddCircle';
import IconButton from "@mui/material/IconButton";
import Container from "@mui/material/Container";
import Toolbar from "@mui/material/Toolbar";
import Tooltip from "@mui/material/Tooltip";

const PageHeadingAppBar = ({name}) => {
    return (
        <Fragment>
            <AppBar position="static" sx={{ bgcolor: "white", display: "flex",
justifyContent: 'space-between' }}>
                <Container>
                    <Toolbar disableGutters sx={{ justifyContent: "space-between" }}>
                        <Typography
                            variant="h6"
                            noWrap
                            sx={{
                                mr: 2,
                                fontFamily: 'Helvetica',
                                fontWeight: 500,
                                color: 'black',
                                textDecoration: 'none',
                            }}
                        >
                            {name}
                        </Typography>
                        <Tooltip title={`Add new ${name}`}>
                            <IconButton
                                size="large"
                                aria-label="add new item"
                                aria-controls="menu-appbar"
                                aria-haspopup="true"
                                color="black"
                            >
                                <AddCircleIcon/>
                            </IconButton>
                        </Tooltip>
                    </Toolbar>
                </Container>
            </AppBar>
            <br/>
        </Fragment>
    );
};

export default PageHeadingAppBar;

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Файл src/components/routers/AuthAppRouter.jsx

```
import React from 'react';
import {Route, Routes} from "react-router";
import SignInPage from "../../pages/SignInPage";
import SignUpPage from "../../pages/SignUpPage";

const AuthAppRouter = ({loginFunc}) => {
  return (
    <Routes>
      <Route path="login" element={<SignInPage loginFunc={loginFunc}/>} />
      <Route path="signup" element={<SignUpPage/>} />
    </Routes>
  );
};

export default AuthAppRouter;
```

Файл src/components/routers/TestResultAppRouter.jsx

```
import React from 'react';
import ProjectsPage from "../../pages/ProjectsPage";
import TestCasesPage from "../../pages/TestCasesPage";
import TestSuitesPage from "../../pages/TestSuitesPage";
import TestRunsPage from "../../pages/TestRunsPage";
import {Route, Routes} from "react-router";

const TestResultAppRouter = () => {
  return (
    <Routes>
      <Route path="/home" element={<div>Home</div>} />
      <Route path="projects" element={<ProjectsPage/>} />
      <Route path="test_cases" element={<TestCasesPage/>} />
      <Route path="test_suites" element={<TestSuitesPage/>} />
      <Route path="test_runs" element={<TestRunsPage/>} />

      <Route path="/settings" element={<div>Settings</div>} />
      <Route path="/logout" element={<div>Logout</div>} />
    </Routes>
  );
};

export default TestResultAppRouter;
```

Файл src/components/tables/ProjectsTable.jsx

```
import React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import SetActiveProjectButton from "../buttons/SetActiveProjectButton";
import Button from "@mui/material/Button";
import EditIcon from '@mui/icons-material/Edit';
import DeleteForeverIcon from '@mui/icons-material/DeleteForever';
import {StyledTableCell, StyledTableRow} from "./Common";
```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

const ProjectsTable = ({projects}) => {
  const [page, setPage] = React.useState(0);
  const [rowsPerPage, setRowsPerPage] = React.useState(5);

  return (
    <TableContainer component={Paper}>
      <Table sx={{ minWidth: 500 }} aria-label="projects table">
        <TableHead>
          <TableRow>
            <StyledTableCell>Project Name</StyledTableCell>
            <StyledTableCell align="center">Description</StyledTableCell>
            <StyledTableCell align="center">Is Active</StyledTableCell>
            <StyledTableCell align="center">Change state</StyledTableCell>
            <StyledTableCell align="center">Actions</StyledTableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {projects.map((project) => (
            <StyledTableRow key={project.id}>
              <StyledTableCell component="th" scope="row">
                {project.name}
              </StyledTableCell>
              <StyledTableCell align="center">{project.description}</StyledTableCell>
              <StyledTableCell align="center">{project.isActive ? "Yes" : "No"}</StyledTableCell>
              <StyledTableCell align="center">{!project.isActive ? <SetActiveProjectButton projectId={project.id} /> : null}</StyledTableCell>
              <StyledTableCell align="center">
                <Button><EditIcon/></Button>
                <Button><DeleteForeverIcon/></Button>
              </StyledTableCell>
            </StyledTableRow>
          )));
        </TableBody>
      </Table>
    </TableContainer>
  );
};

export default ProjectsTable;

```

Файл src/components/tables/TestRunContentTable.jsx

```

import React from 'react';
import { styled } from '@mui/material/styles';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import SetActiveProjectButton from "../buttons/SetActiveProjectButton";
import Button from "@mui/material/Button";
import EditIcon from '@mui/icons-material/Edit';
import DeleteForeverIcon from '@mui/icons-material/DeleteForever';
import { StyledTableCell, StyledTableRow } from "./Common";

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

const TestRunContentTable = ({testRun}) => {
  return (
    <TableContainer component={Paper}>
      <Table sx={{ minWidth: 500 }} aria-label="test runs table">
        <TableHead>
          <TableRow>
            <StyledTableCell align="center">Test Case Name</StyledTableCell>
            <StyledTableCell align="center">Test Case
Description</StyledTableCell>
            <StyledTableCell align="center">Status</StyledTableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {testRun.testCases.map((testCase) => (
            <StyledTableRow key={testCase.id}>
              <StyledTableCell
align="center">{testCase.name}</StyledTableCell>
              <StyledTableCell
align="center">{testCase.description}</StyledTableCell>
              <StyledTableCell
align="center">{testCase.status}</StyledTableCell>
            </StyledTableRow>
          )));
        </TableBody>
      </Table>
    </TableContainer>
  );
};

export default TestRunContentTable;

```

Серверна частина коду

Файл main.py

```
from src import create_app

app = create_app()

if __name__ == '__main__':
    app.run(port=5003)
```

Файл server.sh

```
#!/bin/sh
exec poetry run gunicorn -b :5003 --access-logfile - --error-logfile -
main:app
```

Файл docker/docker-compose.yml

```
version: '3.9'

# postgresql://postgres:postgres@localhost:5439/test_result_app_db

services:
  pg:
    image: postgres:15.3
    container_name: test_result_app_postgres
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
      POSTGRES_DB: "test_result_app_db"
    ports:
      - "5439:5432"
    volumes:
      - test_result_app_pg_data:/var/lib/postgresql/data
    restart: always

volumes:
  test_result_app_pg_data:
```

Файл src/models/user.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship

from src.models import Base

class Project(Base):
    __tablename__ = 'project'

    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('user.id', onupdate='CASCADE',
    ondelete='CASCADE'), index=True, nullable=False)
    name = Column(String)
    description = Column(String)
```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        user = relationship('User', uselist=False)

class User(Base):
    __tablename__ = 'user'

    id = Column(Integer, primary_key=True)
    username = Column(String)
    password_hash = Column(String)
    active_project_id = Column(Integer, nullable=True)
    first_name = Column(String)
    last_name = Column(String)

    @property
    def full_name(self):
        return f'{self.first_name} {self.last_name}'

```

Файл src/models/test_case.py

```

from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship

from src.models import Base

class TestCase(Base):
    __tablename__ = 'test_case'

    id = Column(Integer, primary_key=True)
    project_id = Column(Integer, ForeignKey('project.id', onupdate='CASCADE',
                                             ondelete='CASCADE'), index=True, nullable=False)
    name = Column(String)
    description = Column(String)
    precondition = Column(String)
    postcondition = Column(String)

    project = relationship('Project', uselist=False)
    test_steps = relationship('TestStep', back_populates='test_case',
                             uselist=True)

class TestStep(Base):
    __tablename__ = 'test_step'

    id = Column(Integer, primary_key=True)
    test_case_id = Column(Integer, ForeignKey('test_case.id',
                                              onupdate='CASCADE', ondelete='CASCADE'), index=True, nullable=False)
    content = Column(String)
    order = Column(Integer)

    test_case = relationship('TestCase', back_populates='test_steps',
                            uselist=False)

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Файл src/models/test_suite.py

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship

from src.models import Base


class TestSuite(Base):
    __tablename__ = 'test_suite'

    id = Column(Integer, primary_key=True)
    project_id = Column(Integer, ForeignKey('project.id', onupdate='CASCADE',
    ondelete='CASCADE'), index=True, nullable=False)
    name = Column(String)
    description = Column(String)

    project = relationship('Project', uselist=False)


class TestCaseTestSuite(Base):
    __tablename__ = 'test_case_test_suite'

    id = Column(Integer, primary_key=True)
    test_case_id = Column(Integer, ForeignKey('test_case.id',
    onupdate='CASCADE', ondelete='CASCADE'), index=True, nullable=False)
    test_suite_id = Column(Integer, ForeignKey('test_suite.id',
    onupdate='CASCADE', ondelete='CASCADE'), index=True, nullable=False)

    order = Column(Integer)


class TestRun(Base):
    __tablename__ = 'test_run'

    id = Column(Integer, primary_key=True)
    test_suite_id = Column(Integer, ForeignKey('test_suite.id',
    onupdate='CASCADE', ondelete='CASCADE'), index=True, nullable=False)
    result = Column(String)

    # add relationship
```

Файл src/models/__init__.py

```
from sqlalchemy.orm import declarative_base, scoped_session, sessionmaker

Base = declarative_base()
DBSession = scoped_session(sessionmaker())
```

Файл src/__init__.py

```
from flask import Flask

def create_app():
    app = Flask(__name__)

    # register all routers-blueprints here
    from .api.auth import auth_blueprint
```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

app.register_blueprint(auth_blueprint, url_prefix='/auth')

return app

```

Файл src/core/pdf/common.py

```

FONT_FAMILY = 'helvetica'

class Color:
    WHITE = (255, 255, 255)
    HEADING = (240, 240, 240)

class Font:
    MEDIUM_FONT_SIZE = 10
    GENERAL_FONT_SIZE = 14
    HEADING_FONT_SIZE = 16
    REPORT_HEADING = (FONT_FAMILY, 'B', HEADING_FONT_SIZE)
    GENERAL_HEADING = (FONT_FAMILY, 'B', GENERAL_FONT_SIZE)
    MEDIUM_HEADING = (FONT_FAMILY, 'B', MEDIUM_FONT_SIZE)
    GENERAL_TEXT = (FONT_FAMILY, '', GENERAL_FONT_SIZE)
    MEDIUM_TEXT = (FONT_FAMILY, '', MEDIUM_FONT_SIZE)

```

Файл src/core/pdf/mapping.py

```

STATUS_MAPPING = {
    "success": "Successful",
    "fail": "Failure",
    "retest": "Re-test again",
    "default": "Unknown",
}

```

Файл src/core/pdf/report.py

```

from fpdf import FPDF, YPos

from src.core.pdf.common import FONT_FAMILY, Font, Color
from src.core.pdf.mapping import STATUS_MAPPING

class ReportPDF:
    def __init__(self, file_name, data):
        self.file_name = file_name
        self.data = data
        self.pdf = FPDF()

    def prepare(self):
        self.pdf.set_line_width(1 / 10 ** 5)

    def build_pdf(self):
        pass

    def make(self):
        self.prepare()
        self.pdf.add_page()

```

Вмін.	Арк.	№ докум.	Підп.	Дата.

```

        self.build_pdf()
        self.pdf.output(self.file_name)

class TestRunReportPDF(ReportPDF):

    def build_report_heading(self):
        self.pdf.set_font(*Font.REPORT_HEADING)
        self.pdf.set_fill_color(*Color.WHITE)
        self.pdf.cell(self.pdf.epw, Font.HEADING_FONT_SIZE, 'Test Run
Report', align='C', fill=True)
        self.pdf.ln()
        self.pdf.ln(5)

    def build_test_suite_info(self):
        self.pdf.set_font(*Font.GENERAL_HEADING)
        self.pdf.set_fill_color(*Color.WHITE)

        row1 = {
            'Test Suite': self.data['test_suite']['name'],
            'Date': self.data['timestamp'],
        }
        row2 = {
            'About suite': self.data['test_suite']['description'],
            'Result of Run': STATUS_MAPPING.get(self.data['result'],
'UNKNOWN'),
        }
        rows = [row1, row2]

        for row in rows:
            for key, value in row.items():
                self.pdf.set_font(*Font.GENERAL_HEADING)
                self.pdf.set_fill_color(*Color.HEADING)
                self.pdf.cell(self.pdf.epw * 0.2, Font.GENERAL_FONT_SIZE,
key, align='L', fill=True)

                    self.pdf.set_font(*Font.GENERAL_TEXT)
                    self.pdf.set_fill_color(*Color.WHITE)
                    self.pdf.cell(self.pdf.epw * 0.3, Font.GENERAL_FONT_SIZE,
value, align='L', fill=True)

            self.pdf.ln()

        self.pdf.ln(3)

    def build_test_cases_list_heading(self):
        self.pdf.set_font(*Font.GENERAL_HEADING)
        self.pdf.set_fill_color(*Color.WHITE)
        self.pdf.cell(self.pdf.epw, Font.HEADING_FONT_SIZE, 'Test Cases
List', align='C', fill=True)
        self.pdf.ln()

    def build_test_case_index(self, idx):
        self.pdf.set_font(*Font.GENERAL_HEADING)
        self.pdf.set_fill_color(*Color.WHITE)
        self.pdf.cell(self.pdf.epw, Font.HEADING_FONT_SIZE, f'{idx}', align='C', border=1, fill=True)
        self.pdf.ln()
        self.pdf.ln(1)

    def build_test_case_name_status(self, test_case):
        proportions = [0.2, 0.6, 0.2]
        headings = ['Case Name', 'Case Description', 'Status']

```

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

```

        contents = [test_case[key] for key in ['name', 'description',
'status']]
        contents[-1] = STATUS_MAPPING.get(contents[-1], 'Unknown')

    def row(font, color, sentences):
        self.pdf.set_font(*font)
        self.pdf.set_fill_color(*color)
        for sentence in sentences:
            width = proportions[sentences.index(sentence)]
            self.pdf.cell(self.pdf.epw * width, Font.MEDIUM_FONT_SIZE,
sentence, align='C', fill=True)
            self.pdf.ln()

        row(Font.GENERAL_HEADING, Color.HEADING, headings)
        row(Font.MEDIUM_TEXT, Color.WHITE, contents)

    def build_test_case_conditions_steps(self, test_case):
        headings = ['Pre-Conditions', 'Test Steps', 'Post-Conditions']
        lines = []
        for test_step in test_case['test_steps']:
            line = f'{test_step["order"]}. {test_step["content"]}'
            lines.append(line)

        test_step_text = '\n'.join(lines)
        contents = [test_case['precondition'], test_step_text,
test_case['postcondition']]

        self.pdf.set_font(*Font.GENERAL_HEADING)
        self.pdf.set_fill_color(*Color.WHITE)
        for heading in headings:
            self.pdf.cell(self.pdf.epw / len(headings),
Font.MEDIUM_FONT_SIZE, heading, align='C', fill=True)
            self.pdf.ln()
            self.pdf.ln(1)

        self.pdf.set_font(*Font.MEDIUM_TEXT)
        for content in contents:
            self.pdf.multi_cell(self.pdf.epw / len(contents),
Font.MEDIUM_FONT_SIZE, content, align='C', fill=True, new_y=YPos.TOP)

        for _ in lines:
            self.pdf.ln()
            self.pdf.ln(5)

    def build_one_test_case(self, idx, test_case):
        self.build_test_case_index(idx)
        self.build_test_case_name_status(test_case)
        self.build_test_case_conditions_steps(test_case)

    def build_test_cases(self):
        for idx, test_case in enumerate(self.data['test_cases'], 1):
            self.build_one_test_case(idx, test_case)

    def build_pdf(self):
        self.build_report_heading()
        self.build_test_suite_info()
        self.build_test_cases_list_heading()
        self.build_test_cases()

```

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

Едуард ЖАРИКОВ

“ ____ ” _____ 2023 р.

**ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ
РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Програма та методика тестування

КПІ.ІП-9113.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Геннадій КОЧЕВ

Київ – 2023

ЗМІСТ

1	ОБ'ЄКТ ВИПРОБУВАНЬ.....	3
2	МЕТА ТЕСТУВАННЯ	4
3	МЕТОДИ ТЕСТУВАННЯ.....	5
4	ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ	6

Змін	Арк.	№ докум.	Підп.	Дата.

КП.ІП-9113.045440.04.51

Арк.
2

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є розроблений в процесі виконання дипломного проекту веб-застосунок. Додаток має клієнт-серверну архітектуру, що реалізована за допомогою Python та Flask на сервері та React.js на клієнті. Програмне забезпечення повинно працювати на операційних системах, що підтримують браузери останніх версій для перегляду веб-контенту (Firefox 110.0, Chrome 111.0).

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КП.ІП-9113.045440.04.51

Арк.

3

2 МЕТА ТЕСТУВАННЯ

Метою тестування є:

- 1) перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог, а саме:
 - можливість реєстрації користувача;
 - можливість авторизації користувача;
 - можливість створити новий проект
 - можливість створити тест-кейс;
 - можливість редагувати тест-кейс;
 - можливість видалити тест-кейс;
 - можливість створити тест-сьют з раніше створених тест-кейсів;
 - можливість редагувати тест-сьюти;
 - можливість видалити тест-сьюти;
 - можливість виконати тест-ран за створеним тест-сьютом;
 - можливість згенерувати звіт у форматі PDF зі змістом тест-рану.
- 2) перевірка сумісності веб-додатку з останніми версіями сучасних браузерів (Firefox 110.0, Chrome 111.0);
- 3) перевірка зручності графічного інтерфейсу.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КП.ІП-9113.045440.04.51

Арк.

4

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- інтеграційне тестування – перевірка коректності взаємодії між клієнською та серверною частинами веб-додатку;
- компонентне тестування – перевірка роботи серверної частини застосунку, роботи з базою даних;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

Змін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КП.ІП-9113.045440.04.51

Арк.

5

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Під час інтеграційного тестування створюється визначений набір запитів з клієнтської частини застосунку на серверну частину. Вихідними даними для перевірки є результати відповідей сервера на створені запити.

Під час компонентного тестування складаються набори параметрів, запитів до бази даних та отриманих результатів, що виконують роль вихідних даних такої перевірки.

Функціональне тестування було проведено мануально, без застосування автоматизації за описаними заздалегідь тест-кейсами. Детальний опис проведення функціонального тестування наведено в розділі 3 пояснівальної записки, документ КП.ІП-9113.045440.02.81.

За проведення тестів ставилась задача перевірки коректності роботи веб-застосунку та передчасне виявлення помилок для їх виправлення.

Змін.	Арк.	№ докум.	Підп.	Дата.	КП.ІП-9113.045440.04.51	Арк.
						6

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

Едуард ЖАРИКОВ

“ ____ ” _____ 2023 р.

**ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ
РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Керівництво користувача

КПІ.ІП-9113.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Геннадій КОЧЕВ

Київ – 2023

ЗМІСТ

1	ПРИЗНАЧЕННЯ ПРОГРАМИ	3
2	ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1	Системні вимоги для коректної роботи.....	4
2.2	Завантаження застосунку	4
2.3	Перевірка коректної роботи.....	4
3	ВИКОНАННЯ ПРОГРАМИ	5

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КП.ІП-9113.045440.05.34

Арк.

2

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«Test Result App» - це веб-застосунок для моніторингу та обробки результатів тестувань програмного забезпечення, що дозволяє створювати тест-кейси, групувати їх у тест-съоти, проводити тест-рани за створеними тест-съотами та автоматично генерувати звіт за результатами тест-рану.

Кінцева збірка програмного забезпечення включає два репозиторії – один для серверної частини коду (мовою програмування Python), інший для клієнтської частини коду (мовою програмування JavaScript).

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

КПІ.ІП-9113.045440.05.34

Арк.

3

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність комп'ютера з встановленою операційною системою (Windows, Linux, Mac OS);
- наявність доступу до Інтернету;
- наявність браузера однієї з останніх версій для перегляду веб-контенту (Chrome, Firefox, Safari, Edge).

2.2 Завантаження застосунку

На даний момент застосунок можна розгорнути власноруч, завантаживши репозиторії з кодом серверної та клієнтської частин застосунку та виконавши кроки, що описані у розділі 4 пояснювальної записки, документ КПІ.ІП-9113.045440.02.81.

Взаємодія із застосунком відбувається через встановлений в системі браузер, через графічний інтерфейс.

2.3 Перевірка коректної роботи

По завершенню розгортання застосунку, при переході за потрібним URL у браузері користувач повинен бачити сторінку логіну до веб-додатку. У разі, якщо застосунок не відкривається, потрібно перевірити, чи встановлені залежності коректно або повторити розгортання знову.

Вмін.	Арк.	№ докум.	Підп.	Дата.
-------	------	----------	-------	-------

3 ВИКОНАННЯ ПРОГРАМИ

При відкритті веб-застосунку у браузері, користувачу буде запропоновано увійти до існуючого аккаунту за допомогою пошти та паролю.

The screenshot shows a web browser window with the URL 'localhost:3000/login'. At the top, there's a navigation bar with icons for back, forward, search, and refresh. The main content area has a teal header with a lock icon and the text 'Sign in'. Below this are two input fields: one for 'Email Address *' containing 'myemail@mail.com' and another for 'Password *' containing a masked password. There's also a 'Remember me' checkbox which is checked. A large blue 'LOG IN' button is centered below the inputs. At the bottom of the form, there's a link 'DON'T HAVE AN ACCOUNT? SIGN UP'.

Рисунок 3.1 – Сторінка входу у додаток

Користувач має можливість перейти до сторінки реєстрації у випадку, якщо він не має створеного аккаунту.

Вмін.	Арк.	№ докум.	Підп.	Дата.	KPI.IP-9113.045440.05.34	Арк.
						5

The screenshot shows a web browser window with the URL `localhost:3000/signup`. At the top center is a purple circular icon with a white padlock symbol. Below it, the text "Create an account" is centered. The form consists of several input fields: "First Name *" containing "Hennadii", "Last Name *" containing "Kochev", "Email Address *" containing "myemail@mail.com", and "Password *". The password field contains a series of dots. Below the form is a checkbox labeled "I agree to the Rules of Test Result App usage." followed by a blue "SIGN UP" button. At the bottom of the page is a copyright notice: "Copyright © Test Result App 2023."

Рисунок 3.2 – Сторінка реєстрації у додатку

Користувач заповнює поля імені та прізвища, та логіну та паролю і підтверджує реєстрацію, після чого застосунок його перенаправляє на сторінку входу в аккаунт з рисунку 3.1.

На початку роботи після авторизації користувач має створити один або декілька проектів для продовження.

Вмін.	Арк.	№ докум.	Підп.	Дата.	KPI.IP-9113.045440.05.34	Арк.
						6

The screenshot shows a web browser window with the URL `localhost:3000/projects`. The page title is "TEST RESULT APP". The top navigation bar includes links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS", along with a user profile icon. A green header bar displays the message "Active project: Central Database" and "Welcome, Hennadii Kochev !". Below the header is a table titled "Projects" with columns: "Project Name", "Description", "Is Active", and "Change state". The table lists three projects: "Central Database" (Main application of company, Yes), "Affiliate Management Portal" (Secondary for influencers, No), and "Supplier Order App" (Secondary for suppliers, No). Each row has a "MAKE ACTIVE" button in the "Change state" column.

Рисунок 3.3 – Сторінка зі створеними проектами у додатку

Користувач обирає один з проектів як активний і натискає кнопку «Make Active» навпроти нього для цієї дії. Після цього користувач створює декілька тест-кейсів на відповідній сторінці.

Вмін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.05.34	Арк.
						7

The screenshot shows a web application titled "TEST RESULT APP" running on localhost:3000/test_cases. The top navigation bar includes links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS". A user profile icon is visible in the top right corner. The main content area is titled "Test Cases" and contains three entries:

- Creating New Item**: Description: "Need to create some item detail".
 - Pre-Conditions**: "User is logged in".
 - Test Steps**: "1: Step 1 C content", "2: Step 2 C content", "3: Step 3 C content".
 - Post-Conditions**: "Item is created".
- Reading New Item**: Description: "Need to Reading some item detail".
- Updating New Item**: Description: "Need to Updating some item detail".

Each entry has edit and delete icons on the right.

Рисунок 3.4 – Сторінка зі створеними тест-кейсами у додатку

Як можна побачити, користувач має можливість редагувати та видаляти потрібні тест-кейси, що було створено раніше.

Після створення тест-кейсів, користувач переходить до сторінки тест-сютів, створює новий, додає до нього існуючі тест-кейси.

Вмін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.05.34	Арк.
						8

The screenshot shows a web browser window with the URL `localhost:3000/test_suites`. The interface is divided into several sections:

- Header:** Shows "Active project: Central Database" on the left and "Welcome, Hennadii Kochev!" on the right.
- Navigation Bar:** Includes icons for back, forward, search, and refresh, followed by the URL. It also has links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS".
- User Profile:** A blue header bar with a user icon and a profile picture.
- Main Content:** A section titled "Test Suites" with a "+" button. Below it, there are two expandable sections:
 - Test suite 1 name:** Contains four items:
 - Case: Test case 1 name
 - About: Some1 Description
 - Test suite 2 name:** Contains four items:
 - Case: Test case 3 name
 - About: Some3 Description
 - Case: Test case 2 name
 - About: Some2 Description
 - Case: Test case 4 name
 - About: Some4 Description
- Action Buttons:** On the right side of each expandable section, there are buttons for "CASE+", a blue circular arrow, a pencil icon, and a trash bin icon.

Рисунок 3.5 – Сторінка зі створеними тест-съютами у додатку

Як можна побачити, користувач має можливість додати тест-кейс до існуючого тест-съяту, редагувати або видаляти тест-съяту, а також запустити тест-ран. Під час проведення тест-рану користувач зазначає статус результата для кожного окремого тест-кейсу і це відображається на сторінці тест-ранів у таблиці. відповідній проведенному тест-рану.

Вмін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.05.34	Арк.
						9

The screenshot shows a web application titled "TEST RESULT APP" running on localhost:3000/test_runs. The top navigation bar includes links for "PROJECTS", "TEST CASES", "TEST SUITES", and "TEST RUNS". A user profile icon is visible in the top right corner. The main content area displays two separate test run sections.

Test Run 1: Run for 'Test suite name' at 2023-05-01 12:34:56 | Result: Successful

Test Case Name	Test Case Description	Status
Test case 1 name	Some Description	Successful
Test case 2 name	Some 2 Description	Successful

Test Run 2: Run for 'Test suite 2 name' at 2023-05-02 22:34:56 | Result: Re-Test

Below the test runs, there is a footer table with columns for "Вмін.", "Арк.", "№ докум.", "Підп.", and "Дата.". The footer also contains the text "КПІ.ІП-9113.045440.05.34" and page numbers "Арк." and "10".

Рисунок 3.6 – Сторінка з проведеними тест-ранами у додатку

Як бачимо, додаток зберігає інформацію для кожного тест-кейсу у вигляді таблиці для зручності перегляду. Далі користувач натискає кнопку «PDF» для автоматичної генерації звіту. Користувачу пропонується завантажити звіт.

Вмін.	Арк.	№ докум.	Підп.	Дата.	КПІ.ІП-9113.045440.05.34	Арк.
						10

Test Run Report

Test Suite	Some suite name	Date	2023-05-01 12:34:56
About suite	Some description	Result of Run	Successful

Test Cases List

1

Case Name	Case Description	Status
Test case 1 name	Some Description	Successful
Pre-Conditions	Test Steps	Post-Conditions
Before status 1	1. Step 1 content 2. Step 2 content	After status 1

2

Case Name	Case Description	Status
Test case 2 name	Some 2 Description	Failure
Pre-Conditions	Test Steps	Post-Conditions
Before status 222	1. Step 1 content for 2 2. Step 2 content for 2	After status 22222

Рисунок 3.7 – Згенерований веб-додатком PDF-звіт

Таким чином, користувач завершує роботу із застосунком.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

Едуард ЖАРИКОВ

“ ____ ” _____ 2023 р.

**ВЕБ-ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ ТА ОБРОБКИ
РЕЗУЛЬТАТІВ ТЕСТУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Графічний матеріал

КПІ.ІП-9113.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

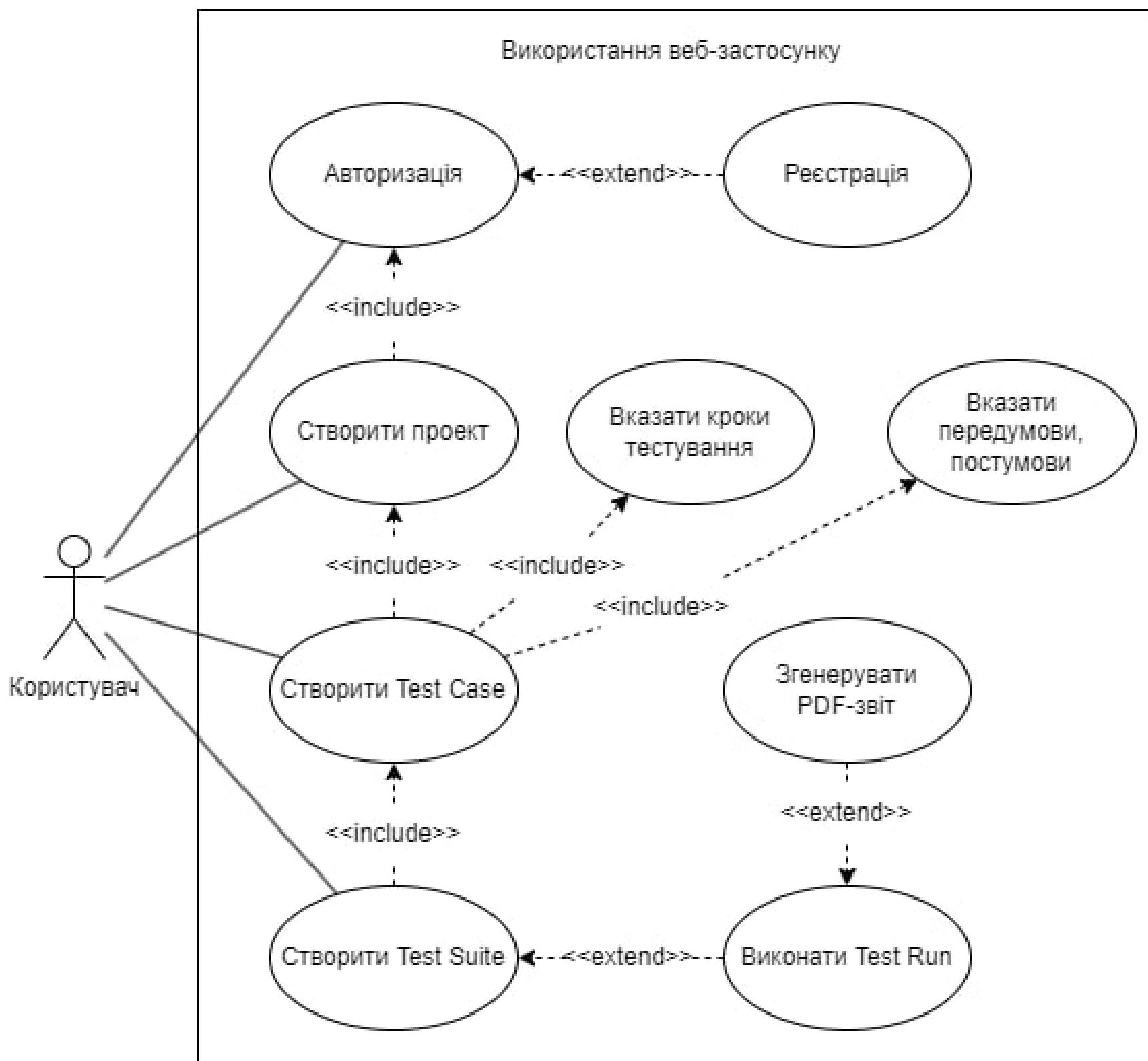
Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавець:

_____ Геннадій КОЧЕВ

Київ – 2023



Зм.	Арк.	№ докум.	Підп.	Дата

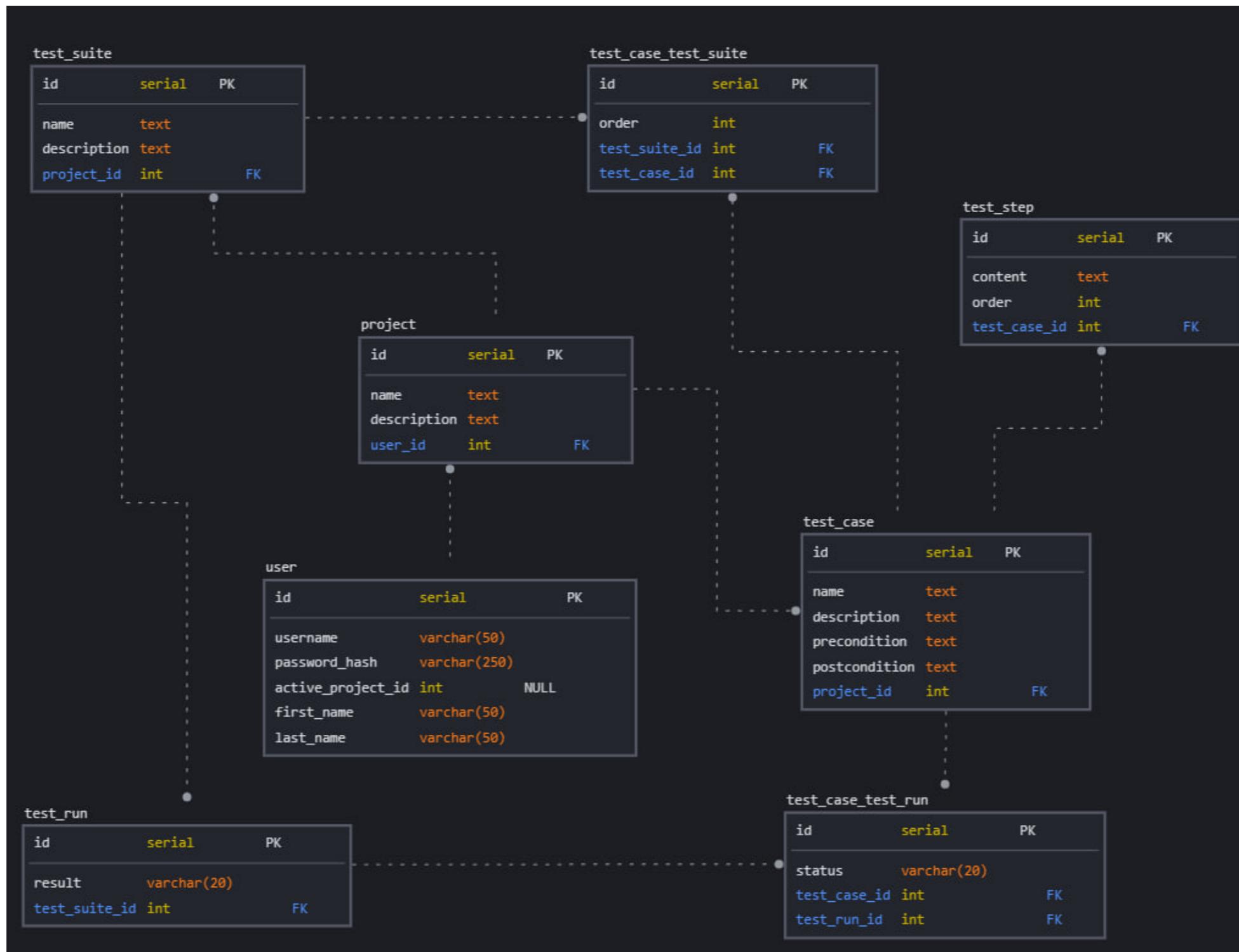
Розроб.	Кочев Г.Г.
Перев.	Ліщук К.І.
Т. Кон.	
Н. Кон.	Ліщук К.І.
Затв.	Жариков Е.В.

КПІ.ІП-9113.045440.06.99.ССВ

Схема структурна варіантів
використань

Лит.	Арк.	Аркушів
		1

Веб-застосунок для моніторингу та
обробки результатів тестувань
програмного забезпеченняКПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІП-91



Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Кочев Г.Г.					
Перевірила		Лішук К.І.					
Т. кон.							
Н. кон.		Лішук К.І.					
Затвердив		Жариков Е.В.					
Веб-застосунок для моніторингу та обробки результатів тестувань програмного забезпечення					КПІ ім. Ігоря Сікорського Кафедра ІПІ гр. ІП-91		

<localhost:3000/login>

Sign in

Email Address * myemail@mail.com

Password *

Remember me

LOG IN

[DON'T HAVE AN ACCOUNT? SIGN UP](#)

Copyright © Test Result App 2023.

<localhost:3000/signup>

Create an account

First Name * Hennadii Last Name * Kochev

Email Address * myemail@mail.com

Password *

I agree to the Rules of Test Result App usage.

SIGN UP

[ALREADY HAVE ACCOUNT? LOG IN](#)

Copyright © Test Result App 2023.

Active project: Central Database Welcome, Hennadii Kochev !

TEST RESULT APP PROJECTS TEST CASES TEST SUITES TEST RUNS

Projects

Project Name	Description	Is Active	Change state
Central Database	Main application of company	Yes	
Affiliate Management Portal	Secondary for influencers	No	MAKE ACTIVE
Supplier Order App	Secondary for suppliers	No	MAKE ACTIVE

localhost:3000/test_runs

Active project: Central Database Welcome, Hennadii Kochev !

TEST RESULT APP PROJECTS TEST CASES TEST SUITES TEST RUNS

Test Runs

Run for 'Test suite name' at 2023-05-01 12:34:56 Result: Successful

Test Case Name	Test Case Description	Status
Test case 1 name	Some Description	Successful
Test case 2 name	Some 2 Description	Successful

Run for 'Test suite 2 name' at 2023-05-02 22:34:56 Result: Re-Test

[test.pdf](#)

Test Run Report

Test Suite: Some suite name Date: 2023-05-01 12:34:56

About suite: Some description Result of Run: Successful

Test Cases List

Case Name	Case Description	Status
Test case 1 name	Some Description	Successful
Pre-Conditions	Test Steps	Post-Conditions
Before status 1	1. Step 1 content 2. Step 2 content	After status 1

Case Name	Case Description	Status
Test case 2 name	Some 2 Description	Failure
Pre-Conditions	Test Steps	Post-Conditions
Before status 222	1. Step 1 content for 2 2. Step 2 content for 2	After status 2222

Active project: Central Database Welcome, Hennadii Kochev !

TEST RESULT APP PROJECTS TEST CASES TEST SUITES TEST RUNS

Test Cases

Creating New Item Need to create some item detail

Pre-Conditions Test Steps Post-Conditions

User is logged in 1: Step 1 C content
2: Step 2 C content
3: Step 3 C content Item is created

Reading New Item Need to Reading some item detail

Updating New Item Need to Updating some item detail

localhost:3000/test_suites

Active project: Central Database Welcome, Hennadii Kochev !

TEST RESULT APP PROJECTS TEST CASES TEST SUITES TEST RUNS

Test Suites

Test suite 1 name Suite 1 description

Case: Test case 1 name	Case: Test case 2 name
About: Some1 Description	About: Some2 Description
Case: Test case 3 name	Case: Test case 4 name
About: Some3 Description	About: Some4 Description

CASE+

Test suite 2 name Suite 2 description

CASE+ <input type="button"/>	<input type="button"/>
------------------------------	------------------------

КПІ.ІП-9113.045440.06.99.KE

Креслення вигляду екранних форм

Зм.	Арк.	№ документа	Підпис	Дата
Розробив	Кочев Г.Г.			
Перевірила	Лішук К.І.			
Т. кон.				
Н. кон.	Лішук К.І.			
Затвердив	Жариков Е.В.			

Літера Маска Масштаб

Аркуш Аркушів

Веб-застосунок для моніторингу та обробки результатів тестувань програмного забезпечення

КПІ ім. Ігоря Сікорського Кафедра ІПІ гр. ІП-91