

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ
УПРАВЛІННЯ БАГ-РЕПОРТАМИ
DESIGN AND DEVELOPMENT OF A WEB APPLICATION
TO MANAGE BUG REPORTS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ТАРАСОВ Я.К.

(ПРИЗВИЩЕ та ініціали)

Керівник ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТИМЕНКО К.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Підпис **Сергій СУББОТІН**
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ТАРАСОВА Ярослава Костянтинівна
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Проектування та розробка вебзастосунку для управління баг-репортами. Design and Development of a Web Application to Manage Bug Reports

керівник проєкту (роботи) к.т.н., ШИТИКОВА Олена Вікторівна,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір засобів, середовища та інструментів розробки. 3. Проектування системи управління баг-репортами. 4. Реалізація та тестування системи управління баг-репортами.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ШИТІКОВА О.В., доцент	Підпис	Підпис
Нормоконтроль	КАЛІНІНА М.В., асистент	Підпис	Підпис

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки		
4	Розробка архітектури програми	2 тиждень	Розділ 3
5	Розробка програми	3-4 тижні	Розділ 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї		
8	Нормоконтроль та рецензування	7 тиждень	
9	Захист роботи	8 тиждень	

Студент(ка)

Підпис Ярослав ТАРАСОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

Підпис Олена ШИТІКОВА
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 106 с., 20 табл., 34 рис., 2 дод., 25 джерел.

БАГ-РЕПОРТИ, ВЕБЗАСТОСУНОК, УПРАВЛІННЯ, БАГИ, КОМАНДНА ВЗАЄМОДІЯ.

Об'єкт дослідження – процес створення, обробки та контролю баг-репортів у командній розробці програмного забезпечення.

Предмет дослідження – програмний засіб для управління баг-репортами.

Мета роботи – розробка вебзастосунок для управління баг-репортами в межах програмних проєктів, з підтримкою розподілу ролей учасників та організованою передачею баг-репортів між ними, що підвищує ефективність командної взаємодії та якість розробки програмного забезпечення.

Матеріали, методи та технічні засоби: методи структурного та об'єктно-орієнтованого програмування, архітектурний стиль REST, мови програмування C++ та TypeScript, серверний фреймворк ASP.NET Core, клієнтський фреймворк Angular, система управління базами даних Microsoft SQL Server, комп'ютер з операційною системою Windows 11.

Результати. Створено вебзастосунок для управління баг-репортами в межах програмних проєктів із підтримкою розподілу ролей і організованою передачею баг-репортів між учасниками команди.

Висновки. Розроблено вебзастосунок для управління баг-репортами в межах програмних проєктів, який забезпечує розподіл ролей учасників і ефективну організацію передачі баг-репортів між ними, що покращує командну взаємодію та якість розробки програмного забезпечення.

Галузь використання – тестування програмного забезпечення та управління дефектами.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 106 pages, 20 tables, 34 figures, 2 appendixes, 25 sources.

BUG REPORTS, WEB APPLICATION, MANAGEMENT, BUGS, TEAMWORK.

Object of study is software for bug report management.

Subject of study are web applications for organizing the process of creating, processing, and controlling bug reports in developer and tester teams.

The purpose of the work is to develop a web application for managing bug reports within software projects, supporting role-based user assignments and organized bug report transfers among team members, aimed at improving teamwork efficiency and software development quality.

Materials, methods, and tools: methods of structured and object-oriented programming, REST architectural style, programming languages C++ and TypeScript, server-side framework ASP.NET Core, client-side framework Angular, Microsoft SQL Server database management system, computer running Windows 11 operating system.

Results. The web application for managing bug reports within software projects was created, supporting role distribution and organized bug report transfers among team members.

Conclusions. A web application for managing bug reports within software projects was developed, providing role-based participant distribution and effective bug report transfer organization, which enhances teamwork and improves software development quality.

The field of use is software testing and defect management.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Опис предметної області	10
1.2 Огляд існуючих вебзастосунків.....	11
1.3 Висновки за розділом 1	16
2 Вибір засобів, середовища та інструментів розробки	17
2.1 Вибір інструментів серверної частини програмного забезпечення.....	17
2.2 Вибір інструментів клієнтської частини програмного забезпечення....	25
2.3 Огляд СУБД.....	31
2.4 Висновки за розділом 2	33
3 Проєктування системи управління баг-репортами.....	35
3.1 Архітектура застосунку	35
3.2 Вимоги до системи.....	37
3.3 Визначення сценаріїв роботи користувачів з вебзастосунком.....	39
3.4 Моделювання поведінки системи	44
3.5 Проєктування бази даних	46
3.6 Варфрейми інтерфейсу користувача	49
3.7 Висновки за розділом 3	55
4 Реалізація та тестування системи управління баг-репортами.....	57
4.1 Опис модулів реалізованої системи	57
4.2 Сценарії взаємодії користувачів із застосунком.....	63
4.3 Тестування системи	73
4.4 Висновки за розділом 4	85
Висновки	86
Перелік джерел посилання	88
Додаток А Технічне завдання	91
Додаток Б Слайди презентації	100

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ASP	– Active Server Pages;
CLI	– Command Line Interface;
CSRF	– Cross-Site Request Forgery;
GUI	– Graphical User Interface;
HTTP	– HyperText Transfer Protocol;
IDE	– Integrated Development Environment;
JSON	– JavaScript Object Notation;
LINQ	– Language Integrated Query;
ORM	– Object-Relational Mapping;
REST	– Representational State Transfer;
RxJS	– Reactive Extensions for JavaScript;
SPA	– Single Page Application;
SQL	– Structured Query Language;
UML	– Unified Modeling Language;
БД	– база даних;
СУБД	– система управління базами даних.

ВСТУП

Сучасна розробка програмного забезпечення охоплює все більше командних проєктів, у яких взаємодія між розробниками, тестувальниками та керівниками набуває вирішального значення. У таких умовах надзвичайно важливою є організація процесу фіксації та усунення помилок, які можуть виникати на різних етапах життєвого циклу програмного продукту. Будь-яка затримка чи непорозуміння в обробці інформації про виявлені помилки може призвести до затягування розробки або до зниження якості продукту.

Особливу увагу в командній роботі привертає управління баг-репортами. Такі звіти часто слугують відправною точкою для усунення проблем, тому важливо, щоб вони надходили до відповідальних осіб швидко, містили всю необхідну інформацію та не губилися серед інших завдань. Залежно від розміру команди, кількості учасників і складності проєкту, процес обробки баг-репортів може бути як простим, так і дуже заплутаним.

У таких умовах виникає потреба в системах, які дозволяли б не лише фіксувати баги, але й ефективно передавати їх між учасниками команди, контролювати статус виконання, а також надавати можливість зручно сортувати, пріоритезувати й відстежувати історію змін. При цьому важливо, щоб взаємодія між користувачами в межах одного проєкту була чітко організована, з урахуванням їхніх ролей та функціональних обов'язків.

Актуальність теми зумовлена потребою в створенні інструменту, який би полегшував процес управління баг-репортами, робив його більш прозорим, доступним і зручним для всіх учасників проєкту. Особливо це стосується тих команд, які працюють над кількома проєктами одночасно або мають обмежені ресурси для використання складних сторонніх систем.

Об'єктом дослідження є процес створення, обробки та контролю баг-репортів у командній розробці програмного забезпечення.

Предметом дослідження є програмний засіб для управління баг-репортами.

Враховуючи ці фактори, метою роботи стала розробка вебзастосунку для управління баг-репортами в межах програмних проєктів, з підтримкою розподілу ролей учасників та організованою передачею баг-репортів між ними, що підвищує ефективність командної взаємодії та якість розробки програмного забезпечення.

Для досягнення цієї мети необхідно вирішити ряд задач:

- проаналізувати предметну область та існуючі аналоги, на базі чого сформулювати функціональні вимоги до застосунку;
- розробити структуру проєкту та бази даних для збереження інформації про користувачів, баг-репорти, зміни статусів, завдання тестеру тощо;
- створити вебінтерфейс для взаємодії з системою;
- реалізувати функціональність для взаємодії з системою;
- забезпечити базову адаптивність та привабливий дизайн інтерфейсу;
- протестувати застосунок на коректність роботи та зручність використання.

Результатом цієї роботи буде створення функціонального вебзастосунку, який дозволить організувати процес взаємодії з баг-репортами більш структуровано, забезпечивши ефективність командної роботи в межах програмного проєкту. Такий застосунок може мати широкі практичні застосування у сфері забезпечення якості, тестування програмних продуктів та управління дефектами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі описано предметну область, проаналізовано вже існуючі аналоги та сформульовано постановку завдання.

1.1 Опис предметної області

У сучасному світі програмного забезпечення особливу увагу приділяють забезпеченню якості продуктів. Незалежно від складності чи масштабу програмного забезпечення, у процесі розробки, тестування та використання неминуче виникають баги (англ. bugs) – розбіжності між очікуваним і фактичним результатом [1]. Ефективне управління такими помилками є важливою складовою забезпечення високої якості кінцевого продукту.

Баг-репорт (англ. bug report) – це документ, який описує та пріоритезує знайдений баг [1]. Він зазвичай включає опис помилки, кроки для її відтворення, очікувану та фактичну поведінку системи, пріоритет, важливість, прикріплені файли тощо.

Для реєстрації та відслідковування ходу вирішення кожного бага використовуються системи відстеження помилок або баг-трекінгові системи (bug tracking systems) [2]. Вони дозволяють централізовано реєструвати, класифікувати, відстежувати хід вирішення та аналізувати помилки, що значно полегшує процес розробки програмного забезпечення і взаємодію між учасниками команди.

Таким чином, розробка вебзастосунку для управління баг-репортами є актуальною задачею, яка спрямована на автоматизацію процесів реєстрації та усунення помилок у програмному забезпеченні. Такий застосунок може стати ефективним інструментом для покращення продуктивності команди та підвищення надійності програмного продукту.

1.2 Огляд існуючих вебзастосунків

На сьогоднішній день існує багато інструментів для управління баг-репортами. Розглянемо чотири популярні системи для управління баг-репортами: Jira, Trello, Asana та Monday. Це найбільш поширені системи, які використовуються для управління проєктами та завданнями, зокрема в контексті управління багами. Кожен інструмент має свої переваги та недоліки, які можуть вплинути на вибір для конкретного проєкту.

1.2.1 Jira

Jira [3] – це популярний інструмент керування проєктами, який включає модуль баг-трекінгу. Інтерфейс системи Jira наведено на рисунку 1.1.

Переваги:

- потужний функціонал для управління багами та задачами;
- підтримка Scrum і Kanban [4];
- інтеграція з іншими інструментами Atlassian (наприклад, Confluence [4]);
- можливість створення звітів на основі Agile [4];
- підтримка кастомізації робочих процесів;
- є безкоштовна обмежена версія.

Недоліки:

- висока вартість;
- складність в освоєнні та налаштуванні;
- може бути надмірним для невеликих команд [4];
- поле для вказання серйозності (Severity) відсутнє у базовій конфігурації, проте може бути додане вручну шляхом створення кастомного поля.

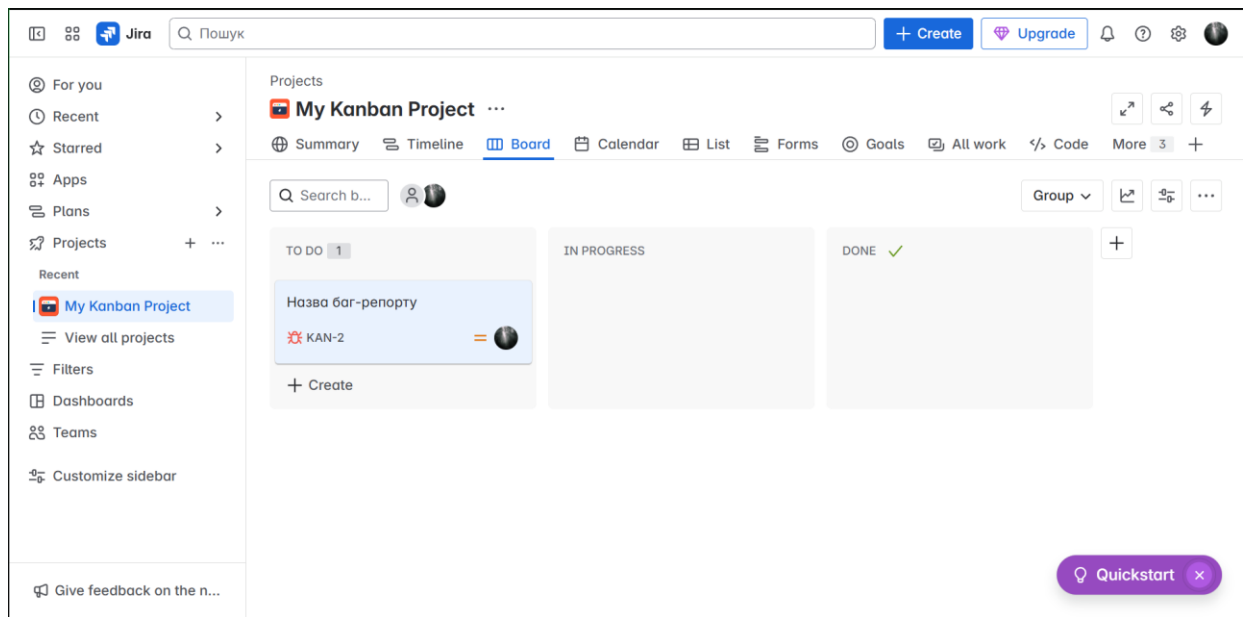


Рисунок 1.1 – Jira [3]

1.2.2 Trello

Trello [5] – це програмне забезпечення для спільної роботи, яке підходить для невеликих або короткотривалих проєктів, а також для команд без досвіду роботи з рішеннями для управління проєктами [4]. Інтерфейс системи Trello наведено на рисунку 1.2.

Переваги:

- є безкоштовна обмежена версія;
- легкий у використанні та безкоштовний для малих команд;
- гнучка система карток та списків;
- підтримка інтеграцій з іншими сервісами;
- простий візуальний інтерфейс.

Недоліки:

- обмежений функціонал для великомасштабних проєктів;
- обмежена підтримка складних робочих процесів;
- відсутність розширених можливостей для звітності та аналізу;
- відсутність повноцінної україномовної локалізації інтерфейсу;

— поля для вказання серйозності (Severity) та пріоритету (Priority) відсутні, проте можлива реалізація подібного механізму за допомогою кольорових міток (labels), використаних у заздалегідь визначеному порядку.

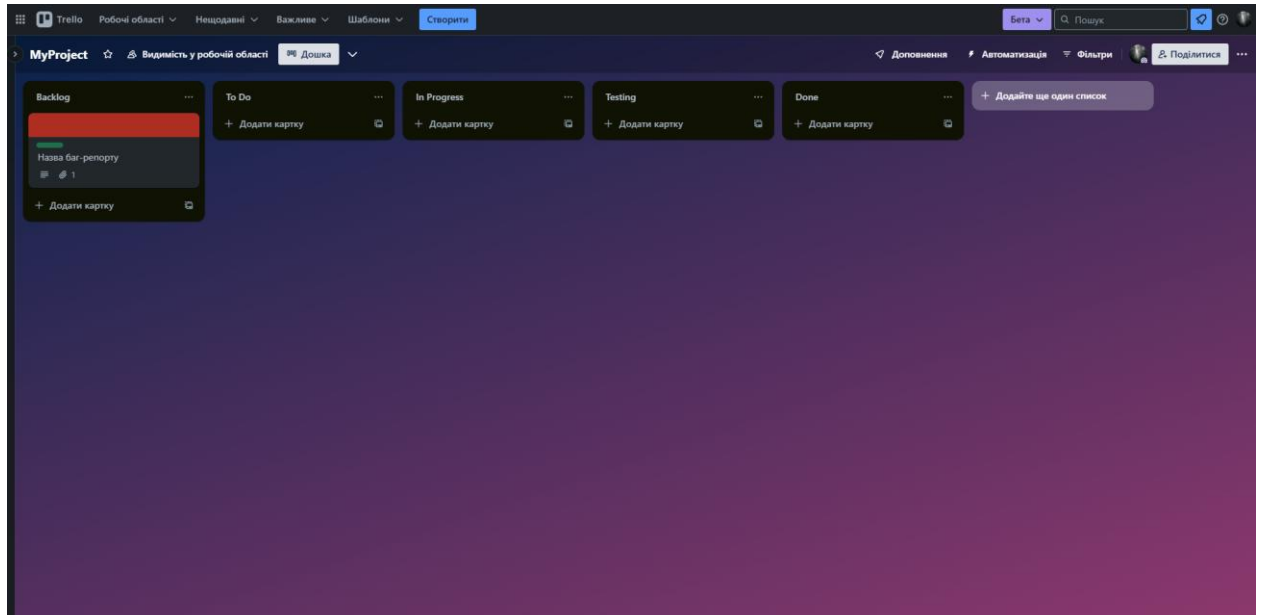


Рисунок 1.2 – Trello [5]

1.2.3 Asana

Asana [6] – це програмне забезпечення для керування проєктами з широким набором функцій для відстеження різноманітних проєктів і спрощення командної співпраці [4]. Asana популярна завдяки своєму зручному інтерфейсу та можливості роботи з командою. Інтерфейс системи Asana наведено на рисунку 1.3.

Переваги:

- інтуїтивно зрозумілий інтерфейс;
- можливість роботи з багатьма проєктами одночасно;
- підтримка інтеграцій з іншими сервісами;
- гарна організація задач та робочих процесів.

Недоліки:

- немає безкоштовної версії для невеликих команд;

- може бути надто спрощеним для складних проєктів;
- відсутність повноцінної україномовної локалізації інтерфейсу;
- поле для вказання серйозності (Severity) відсутнє у базовій конфігурації, проте може бути додане вручну шляхом створення кастомного поля.

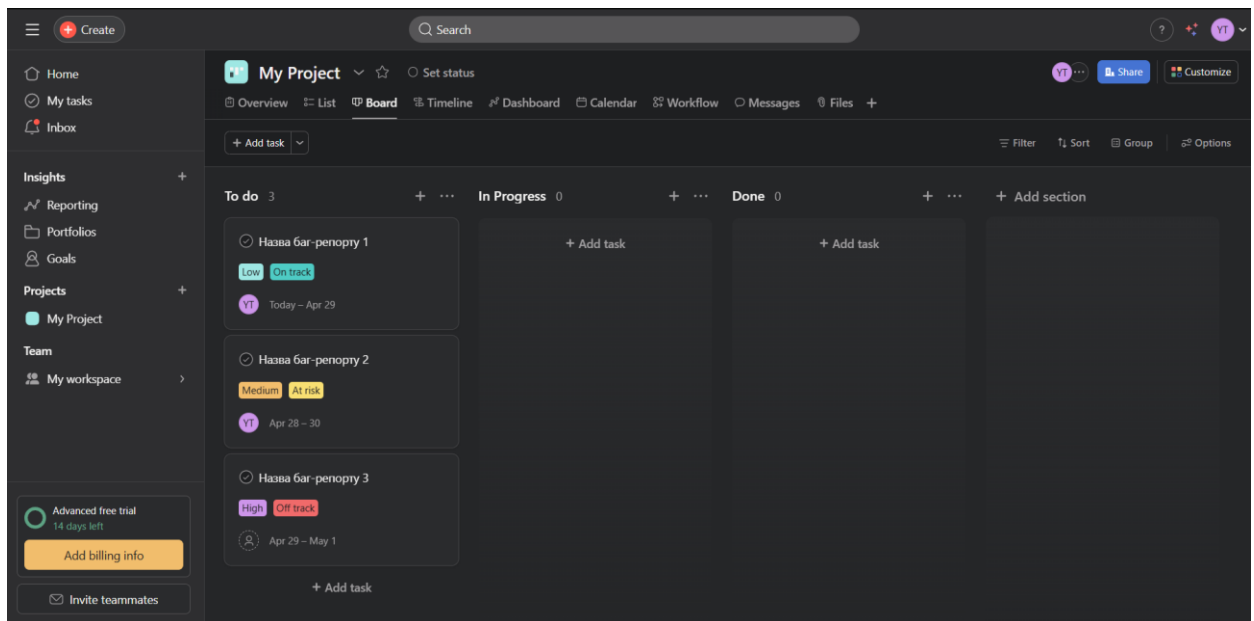


Рисунок 1.3 – Asana [6]

1.2.4 Monday

Monday [7] – це хмарна платформа, що дозволяє користувачам керувати проєктами, дотримуватись жорстких термінів і співпрацювати з командою віддалено [8]. Інтерфейс системи Monday наведено на рисунку 1.4.

Переваги:

- кастомізовані дошки та шаблони для різних типів завдань;
- простий та приємний інтерфейс;
- можливість інтеграції з різними інструментами;
- потужні можливості для звітності та аналітики.

Недоліки:

- немає безкоштовної версії для невеликих команд;

- може бути надто спрощеним для складних проєктів;
- відсутність повноцінної україномовної локалізації інтерфейсу;
- поле для вказання серйозності (Severity) відсутнє у базовій конфігурації, проте може бути додане вручну шляхом створення кастомного поля.

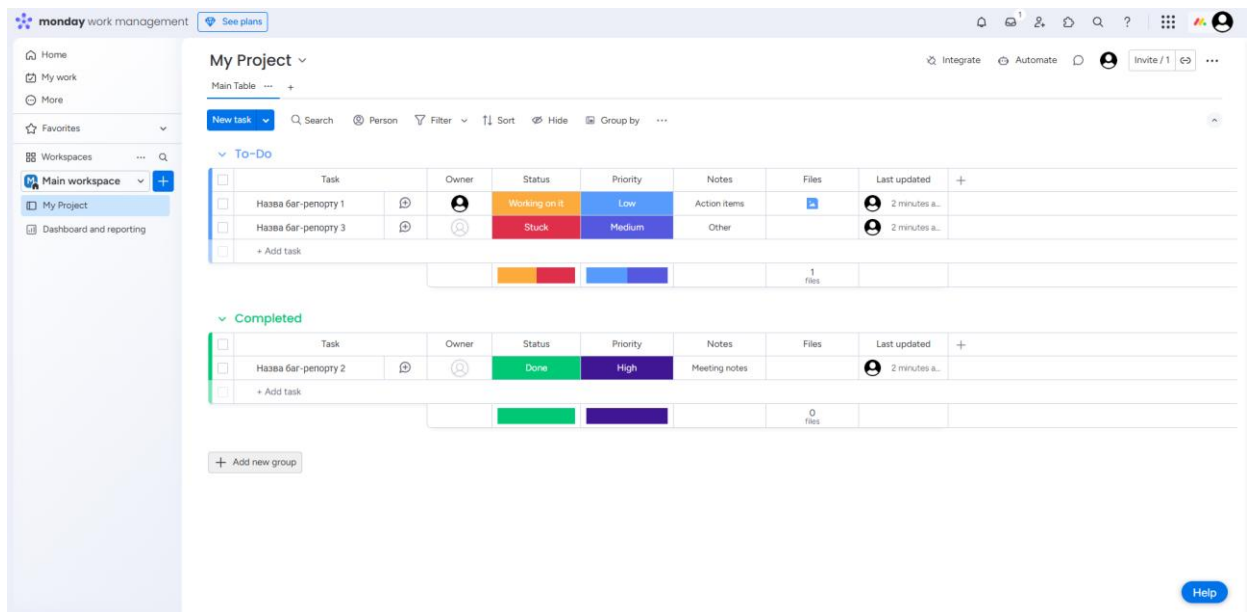


Рисунок 1.4 – Monday Work Management [7]

1.2.5 Порівняння досліджених систем

В таблиці 1.1 представлено порівняння досліджених систем.

Таблиця 1.1 – Порівняння досліджених систем

Критерій	Jira	Trello	Asana	Monday
Безкоштовна версія	+	+	-	-
Простота у використанні	-	+	+	+
Підтримка кастомізації	+	-	+	+
Підтримка Scrum/Kanban	+	-	+	+
Потужні можливості для аналітики/звітування	+	-	+	+
Підходить для великих команд/проєктів	+	-	+	+
Безкоштовна базова версія для невеликих команд	+	+	-	-

Загальні недоліки існуючих рішень:

- жоден із розглянутих інструментів не є спеціалізованим рішенням для управління баг-репортами, через що вони або містять забагато функцій, які не потрібні для невеликих команд, або навпаки, не забезпечують необхідного функціоналу;
- відсутність повноцінної україномовної локалізації інтерфейсу;
- відсутність вбудованого поля “Severity”, яке є обов’язковим для створення баг-репортів.

Це обґрунтовує необхідність створення власного вебзастосунку, орієнтованого на ефективне управління баг-репортами з урахуванням простоти використання та підтримки української мови.

1.3 Висновки за розділом 1

В розділі 1 висвітлено важливість ефективного управління баг-репортами для забезпечення якості програмного забезпечення. Аналіз існуючих інструментів, таких як Jira, Trello, Asana і Monday, показав, що хоча ці системи мають різні переваги, вони мають і значні недоліки, зокрема відсутність повноцінної локалізації, надмірний функціонал для малих команд та відсутність деяких важливих полів, таких як серйозність багів, які можна повернути тільки через додавання їх вручну.

Ці проблеми обґрунтовують необхідність розробки власного вебзастосунку для управління баг-репортами, який забезпечить зручність у використанні, підтримку української мови та гнучкість налаштувань, відповідаючи потребам малих команд і специфічним вимогам. Таким чином, створення такого інструменту є актуальним та доцільним кроком для оптимізації процесу реєстрації та вирішення багів у програмному забезпеченні.

2 ВИБІР ЗАСОБІВ, СЕРЕДОВИЩА ТА ІНСТРУМЕНТІВ РОЗРОБКИ

У цьому розділі розглядаються фреймворки, мови програмування та середовища розробки, що були обрані для реалізації як серверної, так і клієнтської частин програмного забезпечення. Вибір здійснювався на основі аналізу вимог до проєкту, специфіки реалізації функціоналу, а також переваг і недоліків існуючих технологічних рішень. Особливу увагу приділено порівнянню популярних фреймворків, що дозволило прийняти обґрунтовані рішення для побудови стабільного та зручного у підтримці вебзастосунку. Окремо визначено систему керування базами даних, яка використовуватиметься у проєкті.

2.1 Вибір інструментів серверної частини програмного забезпечення

Під час створення архітектури серверної частини майбутнього проєкту було обрано весь необхідний інструментарій, що найкраще підійде всім поставленим вимогам.

2.1.1 Огляд існуючих фреймворків для вирішення завдання

Перш за все треба визначити фреймворк, бо саме від нього залежить мова програмування. Для створення серверної частини вебзастосунків розробникам доступна велика кількість фреймворків, кожен із яких має свої особливості, переваги та сферу застосування. Розглянемо три найпопулярніші з них, які найбільш підходять для реалізації REST API та інтеграції з клієнтською частиною застосунку: Express.js, ASP.Net Core та Django. Їх детальний аналіз дозволить визначити найбільш оптимальне рішення для реалізації поставлених завдань.

2.1.1.1 Express.js

Express.js – це мінімалістичний та гнучкий фреймворк для вебзастосунків, побудованих на Node.js, що надає широкий набір функціональності [9]. Завдяки своїй простоті та гнучкості, він є найпоширенішим фреймворком у середовищі JavaScript.

Переваги:

- швидкий старт розробки;
- величезна екосистема сторонніх модулів та інструментів [10];
- висока продуктивність [10];
- повна кросплатформеність [10].

Недоліки:

- мінімальна структура за замовчуванням і розробник має самостійно обирати бібліотеки для більшості завдань;
- потребує уваги до безпеки та обробки помилок.

2.1.1.2 ASP.NET Core

ASP.NET Core – це кросплатформенний фреймворк з відкритим кодом для створення сучасних вебзастосунків і веб-API, орієнтованих на роботу в хмарному середовищі. Він здебільшого використовується з мовою програмування C# [11].

Переваги:

- кросплатформеність і висока продуктивність [12];
- потужна система залежностей [12];
- підтримує автентифікацію, що спрощує процес захисту вебзастосунків [12];
- активна підтримка та розвиток з боку Microsoft [10].

Недоліки:

- вимагає знання мови, що підтримується .NET (C#, Visual Basic або F#) і платформи .NET [12];
- висока гнучкість може ускладнити старт для новачків.

2.1.1.3 Django

Django – це широко відомий фреймворк в екосистемі Python. Він значно спрощує та прискорює розробку вебзастосунків та програм, надаючи програмісту вже готові інструменти [13].

Переваги:

- вбудована ORM, система аутентифікації [12];
- строга структура і мінімум кода, що повторюється;
- високий рівень безпеки за замовчуванням (захист від SQL-ін'єкцій, клікджекінгу і міжсайтового скриптування) [12];
- відмінна документація та активна спільнота [12].

Недоліки:

- складніший за інші фреймворки [12];
- не завжди оптимальний вибір для високонавантажених застосунків без додаткової оптимізації;
- має відносно повільний час запуску порівняно з іншими фреймворками [12].

2.1.1.4 Порівняння фреймворків

Узагальнюючи розгляд Express.js, ASP.NET Core та Django, можна зробити висновок, що кожен із фреймворків має свої сильні сторони та обмеження. В таблиці 2.1 представлено порівняння досліджених фреймворків для серверної частини. Це дозволяє обрати найкращий варіант, виходячи з вимог проєкту та кваліфікації розробника.

Таблиця 2.1 – Порівняння фреймворків

Критерій	Express.js	ASP.NET Core	Django
Швидкий старт розробки	+	-	+
Висока продуктивність	+	+	-
Вбудована безпека (CSRF, аутентифікація)	-	+	+
Структура проєкту з коробки	-	+	+
Вбудована ORM	-	-	+
Підходить для мікросервісної архітектури	+	+	-
Простота для новачків	+	-	-

Після аналізу наявних фреймворків для створення серверної частини вебзастосунку для управління баг-репортами, було прийнято рішення на користь ASP.NET Core, оскільки саме він найкраще відповідає вимогам цього проєкту. Цей фреймворк забезпечує високу продуктивність і кросплатформенність, дозволяючи розгортати застосунок як на Windows, так і на Linux [12]. Його архітектура дає змогу гнучко керувати обробкою запитів і відповідей, що особливо важливо для обліку подій, логування та обробки помилок.

Крім того, ASP.NET Core має потужну підтримку авторизації та аутентифікації, що дозволяє реалізувати чіткий розподіл ролей між користувачами (тестери, розробники тощо) і забезпечити високий рівень безпеки доступу до даних [12]. Фреймворк також відмінно інтегрується з Entity Framework, який спрощує роботу з реляційними базами даних, дозволяючи швидко будувати модель даних.

Усе це робить ASP.NET Core найбільш придатним для реалізації надійної, масштабованої та зручно керованої серверної частини вебзастосунку.

2.1.2 Огляд особливостей мови програмування

Оскільки для реалізації серверної частини системи було зупинено вибір на фреймворку ASP.NET Core, який працює на платформі .NET, відповідно

для написання логіки застосунку була обрана мова програмування C#. C# – це об'єктно-орієнтована мова, яка поєднує високий рівень виразності з суворого типізацією та широкими можливостями для створення масштабованих веб-сервісів. Вона підтримується компанією Microsoft і має активну спільноту, регулярні оновлення та велику кількість бібліотек, що значно прискорює розробку.

Однією з ключових переваг C# є повна підтримка принципів об'єктно-орієнтованого програмування, таких як абстракція, інкапсуляція, спадкування та поліморфізм, що сприяє створенню гнучкої архітектури програмного забезпечення [14]. Крім того, C# включає функціональні елементи програмування та засоби для паралельного виконання, що розширює її застосування в задачах підвищеної складності [15].

Значним фактором є також наявність стандартної бібліотеки .NET, яка охоплює велику кількість інструментів для роботи з мережею, базами даних та файлами. Це дозволяє значно прискорити розробку та зменшити обсяг низькорівневого коду.

Безпека в C# реалізована через автоматичне управління пам'яттю, перевірку типів під час компіляції та виконання, а також механізми виявлення помилок, що дозволяє знизити кількість вразливостей у застосунку [16].

Крім технічних переваг, мова вирізняється лаконічним і зрозумілим синтаксисом, що робить її зручною для програмістів з різним рівнем підготовки. Важливо й те, що C# постійно оновлюється, отримуючи нові можливості у відповідь на розвиток індустрії програмного забезпечення та потреби сучасного бізнесу.

2.1.3 Огляд середовищ розробки, що найкраще підходять для створення серверної частини проєкту

Для створення серверної частини вебзастосунку на фреймворку ASP.NET Core існує чимало середовищ розробки, кожне з яких має свої

особливості, переваги та сферу застосування. Розглянемо три найпопулярніші з них, які найбільш підходять для роботи з ASP.NET Core: Visual Studio, Visual Studio Code та Rider.

2.1.3.1 Visual Studio

Visual Studio – це всеохоплююче інтегроване середовище розробки, створене Microsoft [17], яке пропонує повну підтримку для ASP.NET Core, C#, .NET, а також різноманітні засоби для тестування, дебагу, роботи з базами даних тощо.

Переваги:

- глибока інтеграція з ASP.NET Core;
- інтелектуальне автодоповнення (IntelliSense) та рефакторинг;
- зручний візуальний дебаггер;
- інтегрована підтримка Git [17];
- вбудовані інструменти для роботи з SQL Server, Azure;
- розширення через Visual Studio Marketplace;
- професійна підтримка від Microsoft;
- є безкоштовна версія з можливістю комерційного використання.

Недоліки:

- високе споживання ресурсів;
- безкоштовна версія обмежена;
- тривалий запуск проєктів, особливо великих;
- працює лише на Windows [17].

2.1.3.2 Visual Studio Code

Visual Studio Code – це легкий редактор коду з розширеннями [17].

Переваги:

- кросплатформеність (Windows, macOS, Linux) [17];

- висока швидкість роботи, легкий інтерфейс;
- широка екосистема розширень [17];
- повністю безкоштовне використання, навіть для комерційних цілей [17];
- інтеграція з терміналом, Git;
- підтримка IntelliSense, дебагу та форматування коду (через розширення);
- відкритий вихідний код.

Недоліки:

- менш зручний для великих проєктів без додаткового налаштування;
- обмежений візуальний інтерфейс порівняно з Visual Studio;
- ASP.NET Core не вбудований безпосередньо в середовище;
- деякі функції (наприклад, профілювання) реалізуються вручну або через CLI.

2.1.3.3 Rider

Rider – це кросплатформне середовище розробки від JetBrains, в першу чергу орієнтоване на розробку .NET [18].

Переваги:

- висока швидкість роботи порівняно з Visual Studio;
- кросплатформеність (Windows, macOS, Linux) [18];
- потужні засоби аналізу коду [18];
- інтеграція з базами даних, Git, Azure DevOps;
- багато розумних інструментів для рефакторингу;
- зручна навігація по коду;
- є безкоштовна версія для некомерційного використання [18].

Недоліки:

- безкоштовна версія не дозволяється для жодного виду комерційної діяльності;
- потребує Java Virtual Machine [19].

2.1.3.4 Порівняння середовищ розробки для серверної частини

В таблиці 2.2 представлено порівняння середовищ розробки для серверної частини.

Таблиця 2.2 – Порівняння середовищ розробки для серверної частини

Критерій	Visual Studio	Visual Studio Code	Rider
Повна підтримка ASP.NET Core	+	-	+
Кросплатформеність	-	+	+
Висока швидкість роботи	-	+	+
Низьке споживання ресурсів	-	+	+
Професійна підтримка	+	-	+
Безкоштовне комерційне використання	+	+	-
Зручний GUI	+	-	+
Вбудовані засоби роботи з БД	+	-	+
Вбудовані засоби тестування	+	-	+
Не потребує Java Virtual Machine	+	+	-
Повна підтримка розгортання в Azure	+	-	-
Вбудовані інструменти для дебагінгу	+	-	+

Для створення серверної частини вебзастосунку на ASP.NET Core було обрано Visual Studio через її глибоку інтеграцію з фреймворком та вбудовані інструменти для тестування та дебагінгу. Важливим фактором є те, що багато необхідних функцій вже вбудовано, а безкоштовна версія доступна як для некомерційних, так і для обмежених комерційних проєктів, що робить її зручним вибором для розробки.

2.2 Вибір інструментів клієнтської частини програмного забезпечення

Під час створення архітектури клієнтської частини майбутнього проєкту було обрано весь необхідний інструментарій, що найкраще підійде всім поставленим вимогам.

2.2.1 Огляд існуючих фреймворків для вирішення завдання

Для реалізації клієнтської частини вебзастосунку важливо обрати фреймворк, який забезпечує зручну інтеграцію з REST API, підтримує модульність та компонентну архітектуру, а також дозволяє ефективно розробляти та масштабувати інтерфейс користувача. Розглянемо два найпопулярніші фреймворки, що активно використовуються у сучасній веб-розробці: Angular та Vue.js.

2.2.1.1 Angular

Angular – це фреймворк для розробки односторінкових вебзастосунків, розроблений компанією Google [20]. Він використовує TypeScript як основну мову програмування і пропонує розробнику комплексне середовище зі вбудованими інструментами та бібліотеками, які не обмежують розмір чи швидкість роботи додатка.

Переваги:

- все необхідне вбудовано у фреймворк [20];
- висока масштабованість, добре підходить для корпоративних проєктів;
- підтримує двосторонню прив'язку даних [20];
- типізація через TypeScript забезпечує надійність коду [20].

Недоліки:

- висока складність для вивчення [20];
- великий розмір початкового пакету застосунку;
- надмірна структура може бути обмежуючою у невеликих проєктах.

2.2.1.2 Vue.js

Vue.js – це фреймворк, що часто обирають для швидкої розробки сучасних вебінтерфейсів.

Переваги:

- простий та швидкий у вивченні;
- реактивне зв'язування даних [20];
- гнучка архітектура – підходить як для малих, так і для середніх проєктів;
- чітко написана та зручна офіційна документація [20].

Недоліки:

- менше корпоративного використання у порівнянні з Angular;
- може бути занадто простим для складних бізнес-застосунків без додаткових інструментів.

2.2.1.3 Порівняння фреймворків

У таблиці 2.3 представлено порівняння двох популярних фреймворків для створення клієнтської частини вебзастосунків – Angular і Vue.js. Обидва інструменти активно використовуються в сучасній веброзробці та мають потужну екосистему. Вони забезпечують високу продуктивність, підтримку компонентного підходу та масштабованість застосунків.

У порівняльній таблиці враховано ключові характеристики, які допомагають прийняти обґрунтоване рішення щодо вибору технології.

Таблиця 2.3 – Порівняння клієнтських фреймворків

Критерій	Angular	Vue.js
Простота освоєння	-	+
Вбудована повна архітектура	+	-
Корпоративне використання	+	-
Низький розмір початкового пакету	-	+
Гнучкість архітектури	-	+
Вбудована типізація через TypeScript	+	-

Після аналізу доступних фреймворків для клієнтської частини вебзастосунку для управління баг-репортами було прийнято рішення на користь Angular. Цей фреймворк надає повноцінний інструментарій для побудови масштабованих SPA із чіткою структурою та високою модульністю. Завдяки вбудованим сервісам, маршрутизації, реактивному підходу (RxJS) і підтримці TypeScript, Angular дозволяє реалізувати комплексну логіку інтерфейсу користувача з високим рівнем повторного використання компонентів.

Крім того, Angular активно підтримується компанією Google, має регулярні оновлення та широку спільноту, що забезпечує швидке знаходження рішень типових проблем. Строга архітектура Angular також полегшує командну розробку, забезпечуючи узгодженість коду в проєкті.

Усе це робить Angular найбільш придатним вибором для реалізації стабільного, масштабованого та підтримуваного клієнтського інтерфейсу вебзастосунку.

2.2.2 Огляд особливостей мови програмування

Мова програмування TypeScript є надмножиною JavaScript, яка додає статичну типізацію [21]. TypeScript активно використовується у розробці великих вебзастосунків, особливо у поєднанні з фреймворком Angular. Нижче наведено ключові особливості цієї мови:

- статична типізація дозволяє виявляти помилки ще на етапі компіляції, що значно покращує якість та надійність коду [21];
- будь-який валідний JavaScript-код є також валідним TypeScript-кодом, що забезпечує легке входження для розробників і поступову міграцію проєктів [21];
- підтримка об'єктно-орієнтованого програмування, включаючи класи, інтерфейси, наслідування та модулі, дозволяє створювати масштабовані та підтримувані архітектури застосунків [21];
- можливість визначення власних типів, інтерфейсів та узагальнень дозволяє створювати гнучкі та багаторазово використовувані компоненти [21];
- розвинена система автодоповнення, навігації та рефакторингу у сучасних IDE значно підвищує продуктивність розробника;
- інтеграція з популярними фреймворками та бібліотеками, зокрема Angular, RxJS, React, забезпечує розробникам широкі можливості для створення SPA;
- висока гнучкість у конфігурації компілятора дозволяє адаптувати поведінку TypeScript до потреб конкретного проєкту;
- активна підтримка та оновлення гарантує розвиток мови, сумісність з новими технологіями та стабільну екосистему.

Таким чином, TypeScript є сучасною, надійною та масштабованою мовою програмування, яка значно спрощує підтримку та розвиток великих фронтенд-проєктів.

2.2.3 Огляд середовищ розробки, що найкраще підходять для створення клієнтської частини проєкту

Для створення клієнтської частини вебзастосунку на фреймворку Angular існує чимало середовищ розробки, кожне з яких має свої особливості, переваги та сферу застосування. Розглянемо два найпопулярніші з них, які найбільш підходять для роботи з Angular – Visual Studio Code та WebStorm.

2.2.3.1 Visual Studio Code

Visual Studio Code – це легкий, кросплатформений, безкоштовний редактор коду, який є одним із найпопулярніших інструментів для розробки Angular-застосунків [17].

Переваги:

- кросплатформеність (Windows, macOS, Linux) [17];
- висока швидкість роботи та мінімальне споживання ресурсів;
- підтримка Angular через офіційні розширення;
- інтеграція з терміналом, що дозволяє зручно працювати з Angular CLI;
- повністю безкоштовне використання, включно з комерційними проєктами [17];
- інтеграція з Git, можливість дебагінгу браузерних застосунків;
- підтримка IntelliSense для TypeScript, HTML та CSS через розширення [17];
- відкритий вихідний код, активна спільнота, регулярні оновлення.

Недоліки:

- відсутні вбудовані інструменти для візуального дизайну інтерфейсу;
- інструменти тестування не вбудовані у середовище.

2.2.3.2 WebStorm

WebStorm – це інтегроване середовище розробки від JetBrains, спеціально створене для фронтенд-розробки, з повною підтримкою Angular, React, Vue тощо.

Переваги:

- глибока інтеграція з Angular CLI та TypeScript.
- інтелектуальні підказки, рефакторинг і навігація по проєкту [22];

- інтеграція з Git [22];
- вбудована перевірка помилок, unit-тести;
- немає потреби в додаткових розширеннях, бо усе потрібне вже вбудовано [22];

- є безкоштовна версія для некомерційного використання.

Недоліки:

- безкоштовна версія не дозволяється для жодного виду комерційної діяльності;
- відсутні вбудовані інструменти для візуального дизайну інтерфейсу;
- вимагає більше ресурсів, ніж Visual Studio Code.

2.2.3.3 Порівняння середовищ розробки для клієнтської частини

В таблиці 2.4 представлено порівняння середовищ розробки для клієнтської частини.

Таблиця 2.4 – Порівняння середовищ розробки для клієнтської частини

Критерій	Visual Studio Code	WebStorm
Безкоштовне комерційне використання	+	-
Вбудована підтримка тестування	-	+
Низьке споживання ресурсів	+	-

Для створення клієнтської частини вебзастосунку на Angular було обрано Visual Studio Code через його високу швидкість роботи, мінімальне споживання ресурсів і повну безкоштовність, включаючи комерційне використання. Це середовище підтримує всі необхідні інструменти для розробки Angular-застосунків через розширення, зокрема Angular CLI, та має інтеграцію з Git. Відкритий вихідний код і активна спільнота також роблять Visual Studio Code ідеальним вибором для швидкої розробки і підтримки проєктів різного масштабу.

2.3 Огляд СУБД

Для зберігання та обробки даних у вебзастосунках, створених із використанням ASP.NET Core та Angular, використовуються різні системи управління базами даних (СУБД). Серед них найпоширенішими є Microsoft SQL Server (MSSQL) та MySQL. Вибір конкретної СУБД залежить від вимог проєкту, архітектури застосунку та можливостей інтеграції з іншими компонентами.

2.3.1 Microsoft SQL Server

Microsoft SQL Server (MSSQL) – це система управління реляційними базами даних, розроблена компанією Microsoft [23].

Переваги:

- вбудована підтримка LINQ через Entity Framework [23];
- можливість масштабування для великих проєктів [24];
- найкраще працює з .NET-проєктами [23];
- забезпечує вищий рівень безпеки, ніж MySQL [23];
- існує безкоштовна версія для малих і середніх проєктів;
- можливість зупинити запит без завершення процесу [23];
- вбудований транзакційний механізм для підтримки консистентності даних [23];
- легко інтегрується з іншими продуктами Microsoft [24];
- автоматичне резервне копіювання [24].

Недоліки:

- переважно орієнтована на Windows-середовище і працює з обмеженнями на macOS і Linux [23];
- вимагає більше ресурсів у порівнянні з легшими СУБД;
- деякі функції недоступні у безкоштовній версії [25].

2.3.2 MySQL

MySQL – це популярна система управління реляційними базами даних з відкритим вихідним кодом, яка широко використовується для керування та зберігання структурованих даних [23].

Переваги:

- безкоштовна, з відкритим кодом [25];
- повна кросплатформність [23];
- для використання MySQL з Entity Framework та LINQ потрібно завантажувати сторонні бібліотеки [23];
- можливість масштабування для великих проєктів [24];
- найкраще працює з проєктами на PHP [23].

Недоліки:

- менша продуктивність при складних транзакціях порівняно з MSSQL [23];
- не дозволяє перервати запит під час виконання без зупинки всього процесу [23];
- деякі функції недоступні у безкоштовній версії.

2.3.3 Порівняння СУБД

В таблиці 2.5 представлено порівняння СУБД.

Таблиця 2.5 – Порівняння СУБД

Критерій	MSSQL	MySQL
Вбудована підтримка LINQ через Entity Framework	+	-
Інтеграція з .NET	+	-
Автоматичне резервне копіювання	+	-
Підтримка повної кросплатформності	-	+
Продуктивність при складних транзакціях	+	-
Безкоштовність комерційного використання	-	+

Microsoft SQL Server був обраний для вебзастосунку через його відмінну інтеграцію з .NET, вбудовану підтримку LINQ через Entity Framework і можливість автоматичного резервного копіювання [23]. MSSQL також підтримує масштабування для великих проєктів і надає високий рівень безпеки. Завдяки безкоштовній версії для малих проєктів і потужним можливостям для обробки складних транзакцій [23], він є ідеальним вибором для цього проєкту.

2.4 Висновки за розділом 2

У розділі 2 розглянуто вибір СУБД, середовищ та фреймворків для створення серверної та клієнтської частин вебзастосунку.

У серверній частині було обрано ASP.NET Core через його високу продуктивність, кросплатформеність і інтеграцію з базами даних. Для реалізації серверної частини було вирішено використовувати мову програмування C#, яка забезпечує об'єктно-орієнтоване програмування та високу ефективність. Вибір середовища розробки припав на Microsoft Visual Studio, що надає потужні інструменти для тестування і налагодження коду.

Щодо клієнтської частини, було обрано фреймворк Angular, що дозволяє створювати масштабовані односторінкові вебзастосунки, а для програмування обрано TypeScript, який дозволяє працювати з статичною типізацією та зручними можливостями для створення складних інтерфейсів. Вибір середовища розробки припав на Visual Studio Code, завдяки його легкості, потужним плагінам та інтеграції з Angular, що значно спрощує процес розробки.

У якості системи управління базами даних було обрано Microsoft SQL Server, оскільки вона має вбудовану підтримку LINQ через Entity Framework, дозволяє автоматизувати процес резервного копіювання та підтримує складні транзакції з високою продуктивністю. MSSQL також надає безкоштовну

версію для невеликих проєктів, що робить її оптимальним вибором для побудови стабільної та масштабованої серверної частини вебзастосунку.

Вибір цих інструментів обґрунтований необхідністю забезпечення високої продуктивності, зручності розробки та масштабованості застосунку, що є важливими факторами для успішної реалізації проєкту.

3 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ БАГ-РЕПОРТАМИ

У цьому розділі розглядається проєктування системи управління баг-репортами, зокрема архітектура вебзастосунку, вимоги до системи, сценарії роботи користувачів та проєктування бази даних. Описано взаємодію клієнтської та серверної частини через HTTP-запити у форматі JSON. Також наведено функціональні та нефункціональні вимоги, визначені ролі користувачів та їх сценарії роботи, а також концептуальну модель бази даних для збереження інформації про проєкти, баг-репорти та користувачів. Додатково представлено модель поведінки системи у вигляді UML-діаграми станів для баг-репорту, який проходить через різні етапи життєвого циклу. Завершальним етапом проєктування стало створення варфреймів інтерфейсу користувача, які візуально відображають основні сторінки майбутнього застосунку та допомагають сформувати уявлення про взаємодію користувача з системою.

3.1 Архітектура застосунку

Система управління баг-репортами проєктується як клієнт-серверний вебзастосунок, у якому клієнтська частина взаємодіє із серверною за допомогою HTTP-запитів. Обмін даними між клієнтом і сервером здійснюватиметься у форматі JSON.

Клієнтська частина створюється з використанням Angular і відповідатиме за інтерфейс користувача, відображення даних, навігацію між сторінками та відправлення запитів до серверної частини. Вона функціонуватиме як односторінковий застосунок (SPA), що забезпечуватиме швидку та безперебійну роботу користувача без постійного перезавантаження сторінок.

Серверна частина проєктується на базі ASP.NET Core. Вона відповідатиме за обробку запитів, бізнес-логіку, керуванням правами доступу

до даних та працюватиме з базою даних. Усі дані користувачів, проєктів, ролей та баг-репортів зберігатимуться в реляційній базі даних Microsoft SQL Server.

Вибір клієнт-серверної архітектури з чітким розділенням на фронтенд і бекенд зумовлений її масштабованістю, зручністю в розробці та супроводі, а також можливістю незалежного оновлення клієнтської та серверної частини. Angular забезпечує динамічний і зручний інтерфейс користувача, тоді як ASP.NET Core дозволяє створити продуктивний, безпечний і легко розширюваний серверний застосунок. Формат обміну JSON є легким і широко підтримується, що також сприяє ефективній взаємодії між клієнтом і сервером.

Архітектуру вебзастосунку зображено на рисунку 3.1. Діаграму компонентів зображено на рисунку 3.2, де представлено основні модулі системи, такі як інтерфейс користувача, логіка застосунку, доступ до бази даних та файлового сховища.

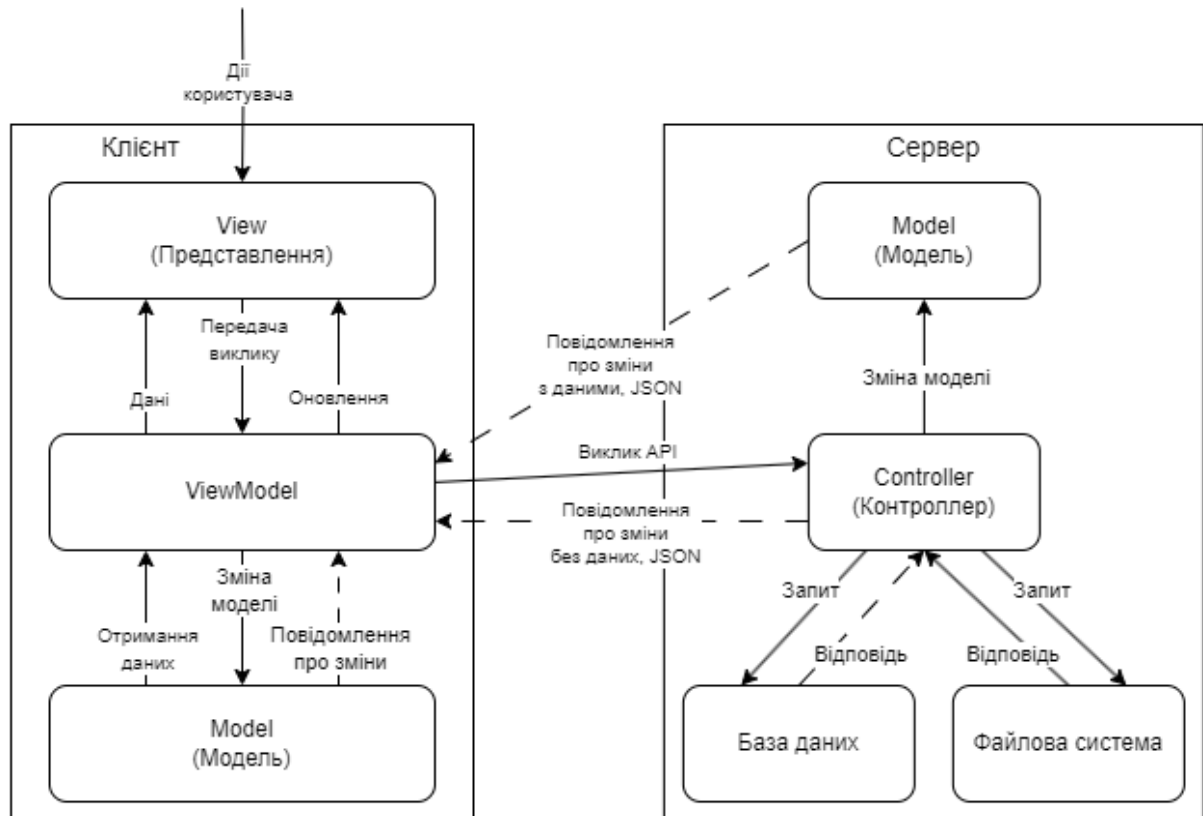


Рисунок 3.1 – Архітектура вебзастосунку

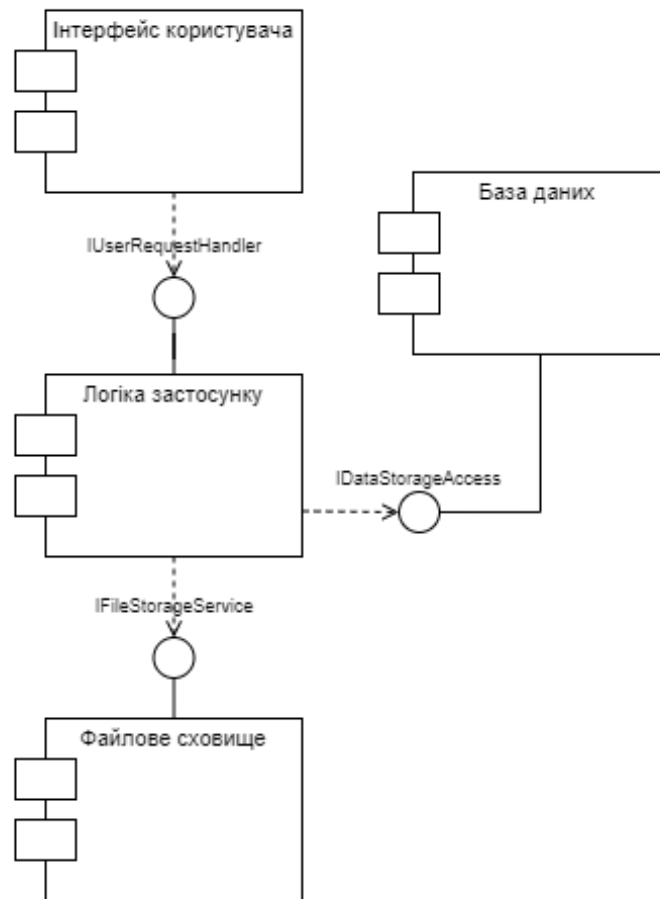


Рисунок 3.2 – Діаграма компонентів вебзастосунку

3.2 Вимоги до системи

У цьому підрозділі наведено основні функціональні та нефункціональні вимоги до системи управління баг-репортами, які визначають її поведінку, можливості та технічні характеристики.

3.2.1 Функціональні вимоги

Система повинна забезпечувати автентифікацію користувачів шляхом введення логіна та пароля. Паролі мають зберігатися у зашифрованому вигляді в базі даних. Після успішного входу користувача система повинна підтримувати механізм авторизації за допомогою cookie.

Система повинна забезпечувати контроль доступу до функціональності на основі ролі користувача у конкретному проєкті. Один і той самий користувач може мати різні ролі у різних проєктах. Доступ до операцій (створення баг-репорту, призначення відповідального, закриття баг-репорту) обмежується згідно з його правами в поточному проєкті.

Користувачі повинні мати можливість створювати нові проєкти, додавати інших користувачів до проєкту, призначаючи їм ролі.

Тестувальники повинні мати можливість створювати баг-репорти із зазначенням назви, опису, кроків для відтворення, рівня серйозності та пріоритету. Також тестувальники повинні мати можливість додавати файл до баг-репорту для візуалізації проблеми. Повідомлення про зміну статусу баг-репорту мають бути доступні учасникам проєкту з доступом до баг-репорту.

Застосунок повинен дозволяти перегляд статистики за проєктами.

3.2.2 Нефункціональні вимоги

Система має підтримувати одночасну роботу кількох користувачів без втрати даних, блокувань або конфліктів при виконанні паралельних запитів.

Архітектура застосунку повинна забезпечувати стабільну роботу навіть при значному збільшенні кількості користувачів, проєктів та баг-репортів.

Середній час відповіді на типові запити не повинен перевищувати 1 секунди за нормального навантаження.

Уся взаємодія клієнтської частини з серверною повинна відбуватися через REST API. Структура запитів і відповідей – у форматі JSON. API повинен бути логічно організований, що забезпечить зручність розширення та підтримки системи.

Система повинна обмежувати доступ до запитів до серверного API відповідно до ролей користувача.

3.3 Визначення сценаріїв роботи користувачів з вебзастосунком

Користувачі не мають статичної категоризації на ролі. Категоризація відбувається тільки на рівні конкретного проєкту. У проєкті користувачі категоризовані на наступні ролі:

- головний розробник;
- головний тестер;
- розробник;
- тестер.

3.3.1 Користувач

Користувач має такі основні сценарії:

- авторизація в системі;
- відображення списку доступних проєктів, розділених на ролі;
- вихід з системи.

Альтернативні сценарії для користувача:

- реєстрація в системі.

Діаграму прецедентів для користувача представлено на рисунку 3.3.



Рисунок 3.3 – Діаграма прецедентів для користувача

3.3.2 Головний розробник

Діаграму прецедентів для головного розробника представлено на рисунку 3.4.

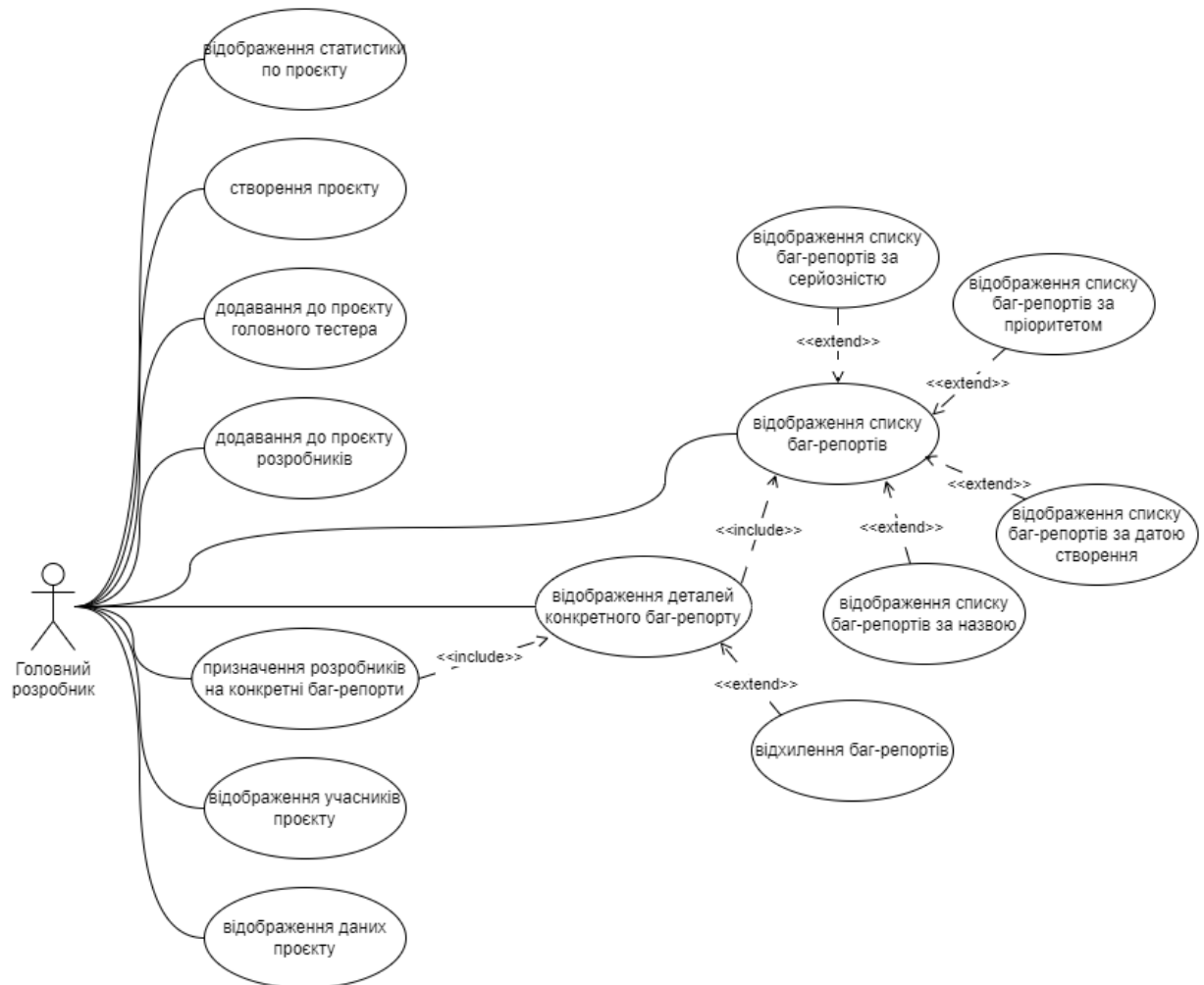


Рисунок 3.4 – Діаграма прецедентів для головного розробника

Головний розробник має такі основні сценарії:

- створення проєкту;
- додавання до проєкту головного тестера;
- додавання до проєкту розробників;
- відображення списку баг-репортів за серйозністю;
- відображення списку баг-репортів за пріоритетом;
- відображення списку баг-репортів за датою створення;

- відображення списку баг-репортів за назвою;
- відображення деталей конкретного баг-репорту;
- відображення статистики по проєкту;
- відображення даних проєкту;
- відображення учасників проєкту;
- призначення розробників на конкретні баг-репорти.

Альтернативні сценарії для головного розробника:

- відхилення баг-репорту, якщо повідомлення містить недостатньо інформації, є дублюванням вже існуючого багу або не відтворюється.

3.3.3 Головний тестер

Головний тестер має такі основні сценарії:

- додавання до проєкту тестувальників;
- видача тестерам завдання на тестування;
- відображення списку баг-репортів за серйозністю;
- відображення списку баг-репортів за пріоритетом;
- відображення списку баг-репортів за датою створення;
- відображення списку баг-репортів за назвою;
- передача баг-репорту головному розробнику;
- відображення деталей конкретного баг-репорту;
- відображення статистики по проєкту;
- відображення даних проєкту;
- відображення учасників проєкту;
- підтвердження результатів виправлення багу;
- закриття баг-репорту.

Альтернативні сценарії для головного тестера:

- відхилення баг-репорту, якщо він не відповідає вимогам для тестування, наприклад, якщо баг не відтворюється або містить недостатньо деталей;

– відхилення результатів виправлення багу, якщо виправлення не задовольняє вимогам тестування або не виправляє помилку.

Діаграму прецедентів для головного тестера представлено на рисунку 3.5.

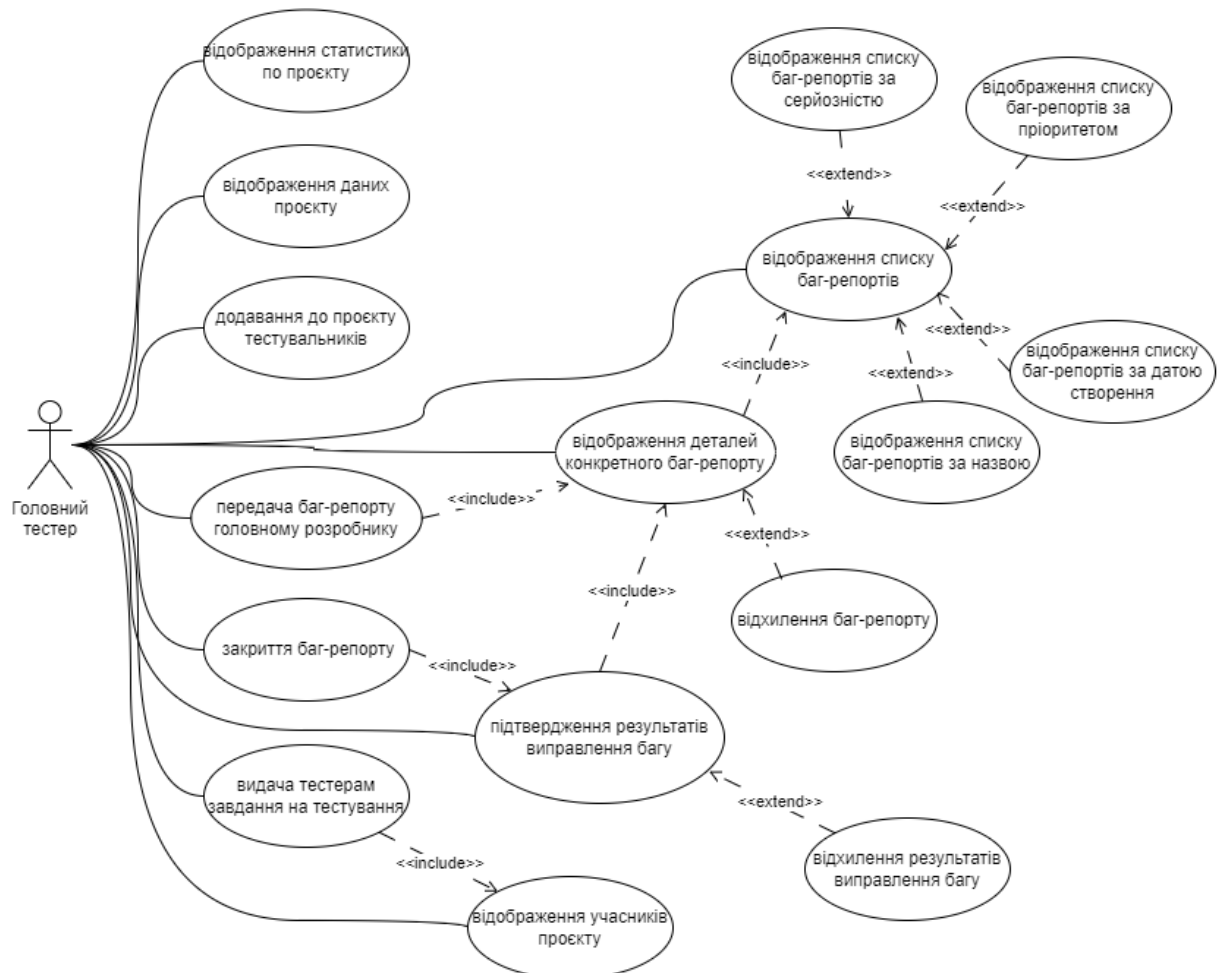


Рисунок 3.5 – Діаграма прецедентів для головного тестера

3.3.4 Розробник

Розробник має такі основні сценарії:

- відображення списку призначених баг-репортів за серйозністю;
- відображення списку призначених баг-репортів за пріоритетом;
- відображення списку призначених баг-репортів за датою створення;

- відображення списку призначених баг-репортів за назвою;
- відображення деталей конкретного баг-репорту;
- відображення статистики по проєкту;
- відображення даних проєкту;
- відображення учасників проєкту;
- надсилання повідомлення тестувальнику про виправлення багу.

Альтернативні сценарії для розробника:

- відхилення баг-репортів, якщо повідомлення містить недостатньо інформації, є дублюванням вже існуючого багу або не відтворюється.

Діаграму прецедентів для розробника представлено на рисунку 3.6.

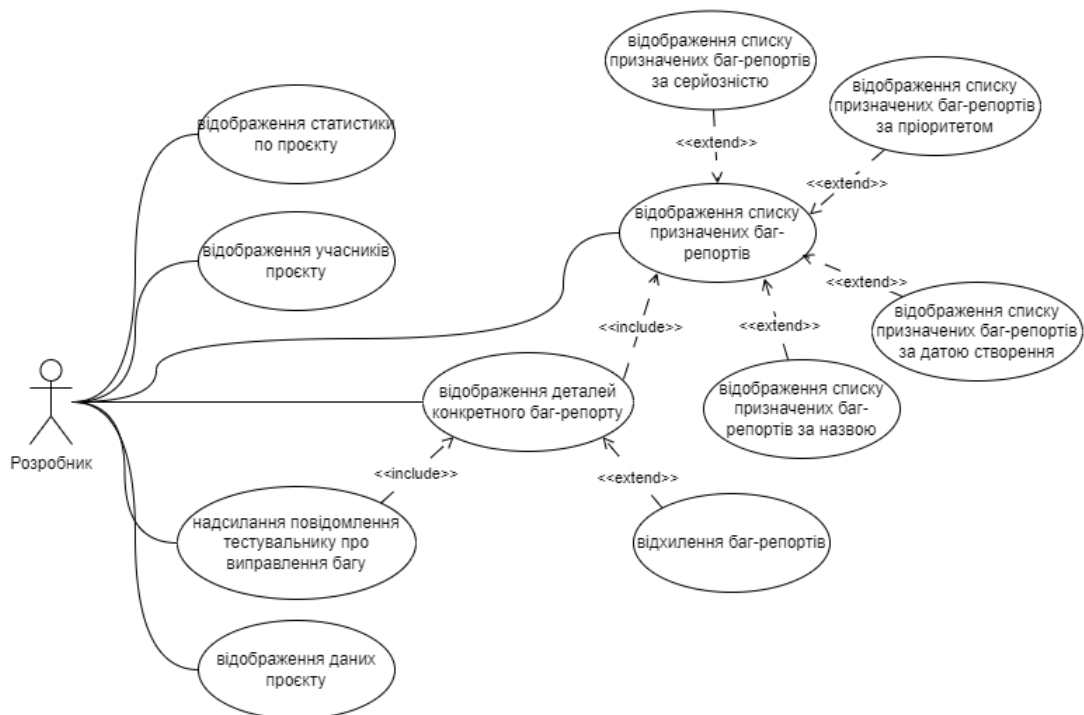


Рисунок 3.6 – Діаграма прецедентів для розробника

3.3.5 Тестер

Тестер має такі основні сценарії:

- відображення списку створених баг-репортів за серйозністю;
- відображення списку створених баг-репортів за пріоритетом;
- відображення списку створених баг-репортів за датою створення;

- відображення списку створених баг-репортів за назвою;
- відображення деталей конкретного баг-репорту;
- відображення статистики по проєкту;
- перегляд даних проєкту;
- відображення учасників проєкту;
- створення баг-репорту;
- надсилання запиту на закриття баг-репорту.

Альтернативні сценарії для тестера:

- надсилання запиту на повторне відкриття баг-репорту, якщо після виправлення баг все ще існує.

Діаграму прецедентів для тестера представлено на рисунку 3.7.

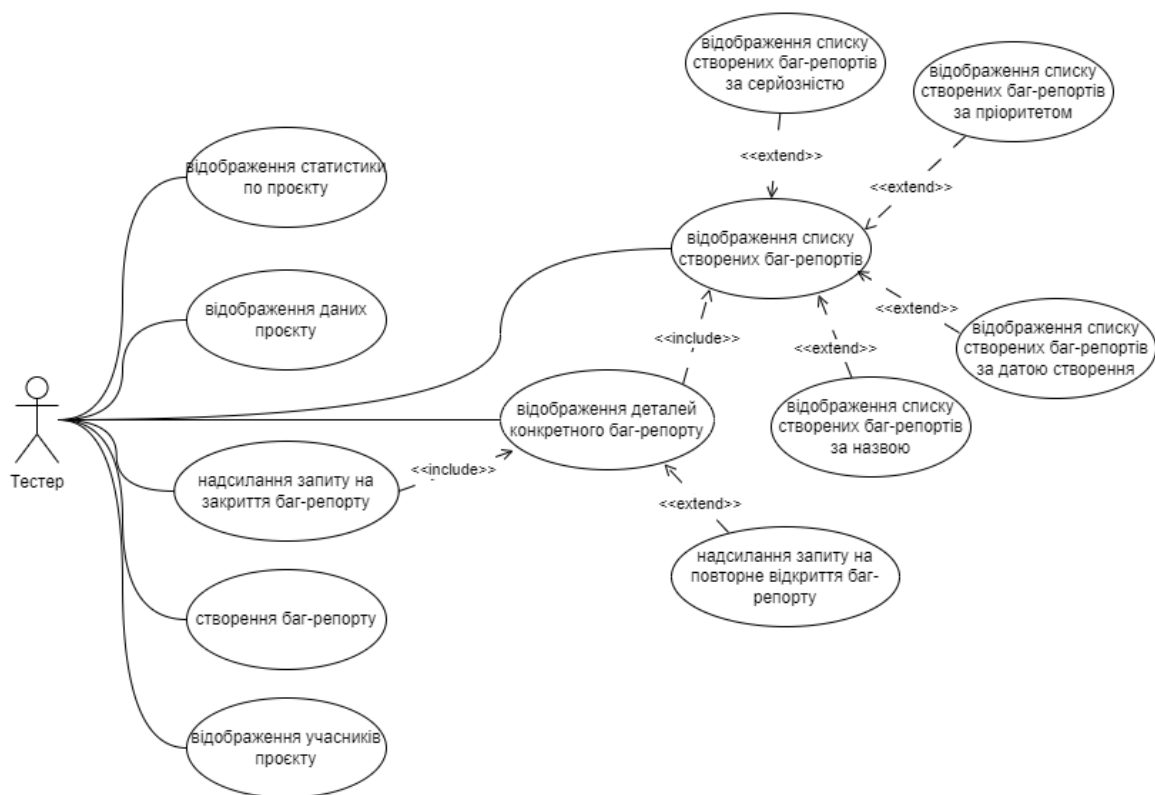


Рисунок 3.7 – Діаграма прецедентів для тестера

3.4 Моделювання поведінки системи

Для кращого розуміння логіки функціонування системи та взаємодії користувачів із баг-репортами було побудовано UML-діаграму станів, що

представлена на рисунку 3.8. Баг-репорт проходить через різні етапи життєвого циклу – від створення до закриття або повторного відкриття і вже потім закриття – залежно від дій тестувальників і розробників.

Діаграма відображає всі можливі стани баг-репорту та умови переходу між ними. Це дає змогу чітко структурувати поведінку системи при обробці помилок і гарантувати коректне реагування на типові сценарії у процесі розробки та тестування.

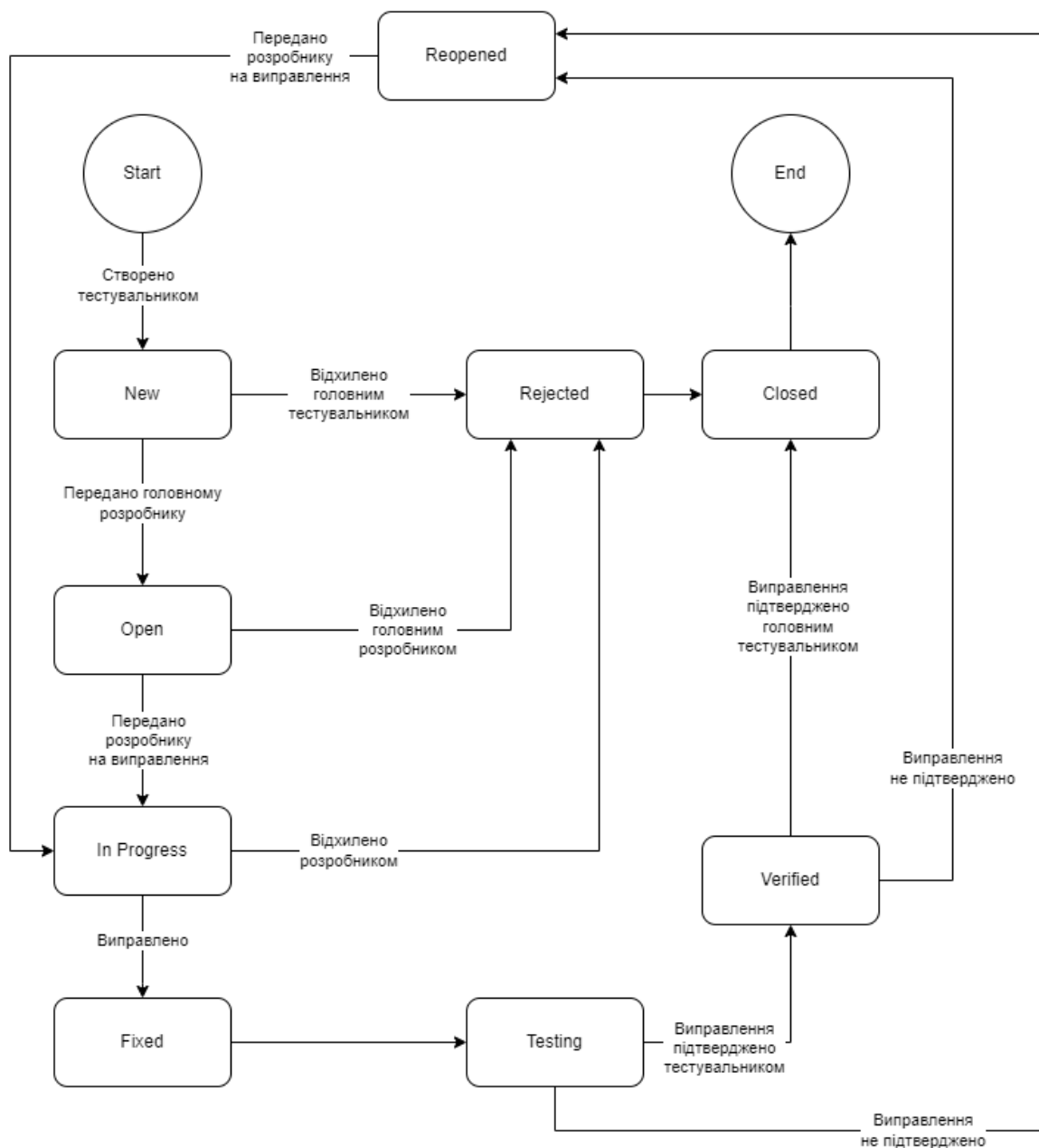


Рисунок 3.8 – Діаграма станів баг-репорту

3.5 Проектування бази даних

Сутність bugReport (id, title, description, stepsToReproduce, severity, priority, createdAt, projectId, testerId, developerId, fileId) – зберігає інформацію про баг-репорти. Опис полів сутності та їх типу наведено у таблиці 3.1.

Таблиця 3.1 – Опис полів сутності bugReport

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	title	nvarchar	100
3	description	nvarchar	-
4	stepsToReproduce	nvarchar	-
5	severity	nvarchar	20
6	priority	nvarchar	20
7	createdAt	datetime	-
8	projectId	int	-
9	testerId	int	-
10	developerId	int	-
11	fileId	int	-

Сутність bugReportUpdateHistory (id, bugId, changedBy, newValue, changedAt) – зберігає інформацію про зміну статусів баг-репортів. Опис полів сутності та їх типу наведено у таблиці 3.2.

Таблиця 3.2 – Опис полів сутності bugReportUpdateHistory

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	bugId	int	-
3	changedBy	int	-
4	newValue	nvarchar	-
5	changedAt	datetime	-

Сутність fileData (id, name, relativePath, mimeType, fileSize, uploadDate) – зберігає інформацію про файли, що прикріплюються к баг-репортам. Ця сутність дозволяє зберігати метадані файлів, що необхідні для їх ідентифікації, завантаження та обробки у системі. Опис полів сутності та їх типу наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності fileData

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	name	nvarchar	255
3	relativePath	nvarchar	500
4	mimeType	nvarchar	100
5	fileSize	int	-
6	uploadDate	datetime	-

Сутність userData (id, name, patronymic, surname, phoneNumber, email, passwordHash) – зберігає інформацію про користувачів, що зареєстровані у системі. Опис полів сутності та їх типу наведено у таблиці 3.4.

Таблиця 3.4 – Опис полів сутності userData

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	name	nvarchar	50
3	patronymic	nvarchar	50
4	surname	nvarchar	50
5	phoneNumber	nvarchar	50
6	email	nvarchar	50
7	passwordHash	nvarchar	255

Сутність project (id, title, description, dateNow) – зберігає інформацію про проекти. Опис полів сутності та їх типу наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів сутності project

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	title	nvarchar	50
3	description	nvarchar	-
4	dateNow	datetime	-

Сутність testersTask (id, TesterID, ProjectID, taskText, createdAt) – зберігає інформацію про завдання для тестерів від головного тестера. Опис полів сутності та їх типу наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів сутності testersTask

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	TesterID	int	-
3	ProjectID	int	-
4	taskText	nvarchar	-
5	createdAt	datetime	-

Сутність userProject (id, userId, projectId, role) – забезпечує зв'язок між користувачами та проектами, дозволяючи одному користувачу брати участь у кількох проектах, а одному проекту включати кількох користувачів. Опис полів сутності та їх типу наведено у таблиці 3.7.

Таблиця 3.7 – Опис полів сутності userProject

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	userId	int	-
3	projectId	int	-
4	role	nvarchar	20

За допомогою описаних вище сутностей була побудована ER-модель бази даних, що представлена на рисунку 3.9.

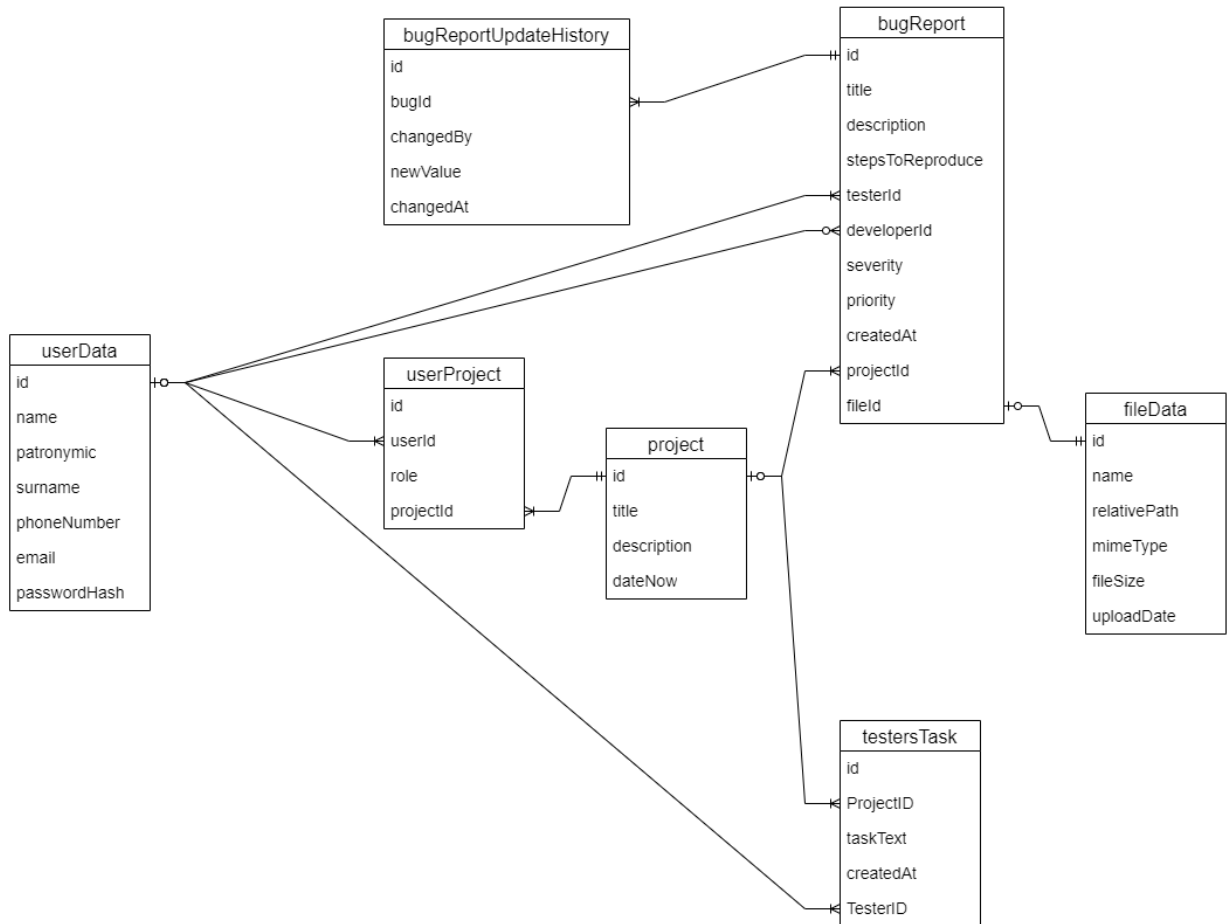


Рисунок 3.9 – ER-модель бази даних

3.6 Варфрейми інтерфейсу користувача

На етапі проєктування програмного забезпечення важливим кроком стало створення варфреймів основних сторінок майбутньої системи. Варфрейми допомагають наочно представити структуру інтерфейсу користувача. Це дозволяє ще до початку реалізації провести первинне оцінювання зручності користування системою та своєчасно внести зміни у випадку виявлення недоліків.

На першому етапі взаємодії користувача із системою передбачено відображення вступної сторінки, яка виконує інформативну та навігаційну

функцію. На ній розміщено вітальний текст із загальним представленням призначення системи, а також кнопки переходу до сторінки авторизації або реєстрації. Такий підхід дозволяє створити позитивне перше враження та зорієнтувати користувача щодо подальших дій. Варфрейм цієї сторінки представлено на рисунку 3.10.

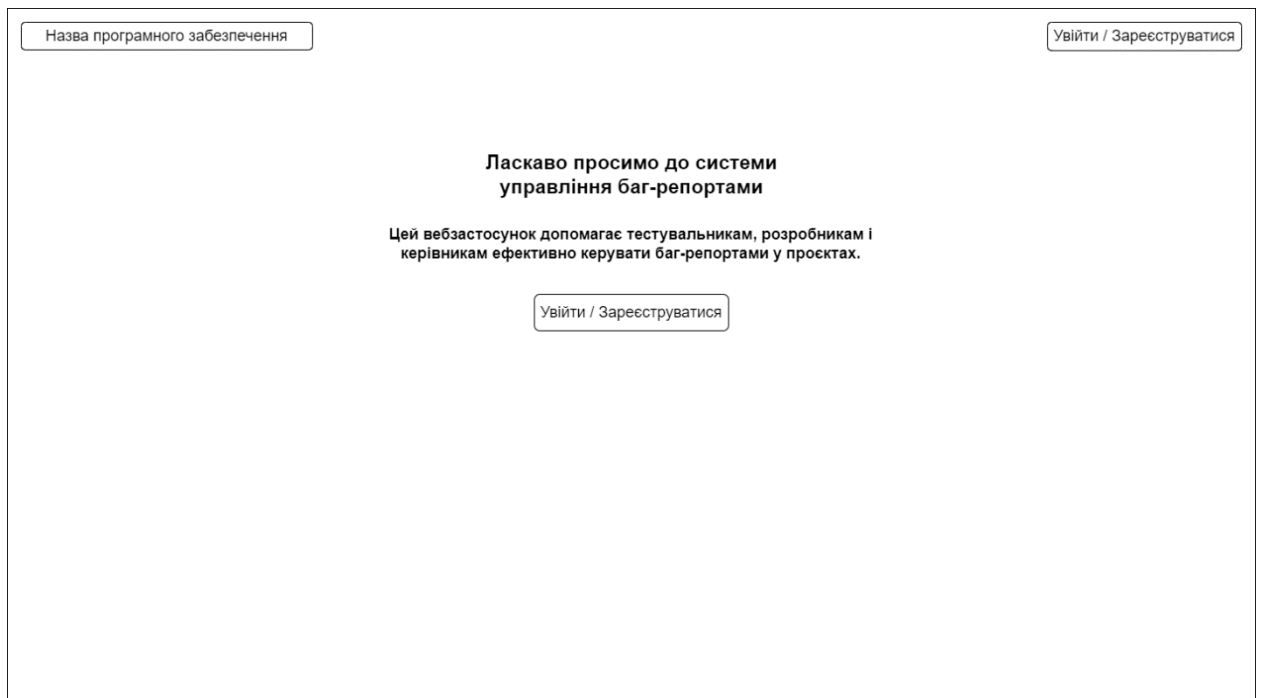


Рисунок 3.10 – Варфрейм вступної сторінки

Наступними кроками є сторінки реєстрації та входу. Вони забезпечують базову автентифікацію користувача, що є необхідною умовою для захисту даних та персоніфікації функціоналу. Сторінка входу містить поля для введення логіна та пароля. Також на сторінці входу повинна бути кнопка, що дозволяє перемикати режим відображення пароля, показуючи або приховуючи символи при введенні. Варфрейм цієї сторінки зображено на рисунку 3.11.

Сторінка реєстрації дозволяє створити новий обліковий запис із зазначенням імені, по-батькові, прізвища, номера телефону, електронної пошти та пароля. Варфрейм демонструє інтуїтивно зрозумілу форму з базовою валідацією. Варфрейм сторінки реєстрації наведено на рисунку 3.12.

Авторизація

Вхід

Реєстрація

Вхід

Email

Email

Пароль

Пароль

Показати
пароль

Увійти

Рисунок 3.11 – Варфрейм сторінки входу до системи

Авторизація

Вхід

Реєстрація

Реєстрація

Ім'я

Ім'я

По-батькові

По-батькові

Прізвище

Прізвище

Email

Email

Номер телефону

Номер телефону

Пароль

Пароль

Підтвердження паролю

Підтвердження паролю

Зареєструватися

Рисунок 3.12 – Варфрейм сторінки реєстрації нового користувача

Після входу користувач потрапляє на сторінку вибору проєкту. Вона є центральною точкою початку роботи над конкретними задачами в системі. Варфрейм демонструє перелік доступних користувачеві проєктів з можливістю вибору або створення нового проєкту. Такий підхід дозволяє гнучко організувати роботу різних команд у межах однієї системи. Проєкти згруповано відповідно до ролі, яку користувач займає в кожному з них. Це спрощує орієнтацію в списку проєктів та дозволяє швидко знаходити потрібний. Варфрейм цієї сторінки подано на рисунку 3.13.

Назва програмного забезпечення

Вийти

Виберіть проєкт:

Головний розробник	Створити проєкт	Назва проєкту	Назва проєкту	Назва проєкту
	Назва проєкту			
Головний тестер	Назва проєкту	Назва проєкту	Назва проєкту	Назва проєкту
Розробник	Назва проєкту	Назва проєкту	Назва проєкту	Назва проєкту
Тестер	Назва проєкту	Назва проєкту	Назва проєкту	Назва проєкту

Рисунок 3.13 – Варфрейм сторінки вибору проєкту

У разі створення нового проєкту користувач переходить до сторінки створення проєкту. Варфрейм передбачає поля для введення назви та опису. Це дозволяє сформулювати основні дані про проєкт перед початком роботи. Таке попереднє введення інформації є необхідним для подальшої ідентифікації та управління проєктами в системі. Варфрейм представлено на рисунку 3.14.

Закрити

Створити проєкт

Назва

Назва проєкту

Опис

Опис баг-репорту

Створити

Рисунок 3.14 – Варфрейм сторінки створення проєкту

Основна функціональність системи полягає у створенні та керуванні баг-репортами. Варфрейм сторінки списку баг-репортів показує перелік існуючих баг-репортів із зазначенням основних даних, таких як назва, пріоритет, серйозність, статус, дата створення. Передбачена можливість сортування, що полегшує навігацію великими обсягами інформації. Варфрейм цієї сторінки наведено на рисунку 3.15.

Назва програмного забезпечення

Вийти

Назад до списку проєктів

Про цей проєкт	Баг-репорти	Завершені баг-репорти	Статистика
----------------	-------------	-----------------------	------------

Список баг-репортів

ID	Назва	Пріоритет	Серйозність	Статус	Дата	Більше інформації
[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	Відкрити
[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	Відкрити
[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	Відкрити
[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	[[Значення]]	Відкрити

Рисунок 3.15 – Варфрейм сторінки списку баг-репортів

Створення нового баг-репорту реалізується через відповідну форму, яка включає поля для назви, опису, кроків відтворення, пріоритету,

серйозності та прикріплення файлу. Варфрейм такої сторінки дозволяє візуалізувати логічну послідовність введення інформації, сприяючи її повноті та точності. Варфрейм представлено на рисунку 3.16.

Закрити

Створити баг-репорт

Тестер

Ім'я та прізвище тестера

Проект

Назва проекту

Назва

Назва баг-репорту

Опис

Опис баг-репорту

Кроки для відтворення

Кроки для відтворення баг-репорту

Пріоритет

Пріоритет баг-репорту

Серйозність

Серйозність баг-репорту

Скріншот (тільки зображення)

Обрати файл

Назва файлу

Завантажити

Створити

Рисунок 3.16 – Варфрейм сторінки створення баг-репорту

Окрему увагу приділено функціоналу, доступному головному тестеру. Варфрейм сторінки призначення завдань дозволяє йому переглядати наявних

тестерів, бачити поточні завдання та призначати нові. Це забезпечує гнучке управління командою та координацію тестувальних активностей. Варфрейм цієї сторінки наведено на рисунку 3.17.

Рисунок 3.17 – Варфрейм сторінки призначення завдань тестерам

Усі описані варфрейми сформовані з урахуванням принципів зручності, логічності та модульності. Вони стануть основою для подальшої реалізації інтерфейсу у вигляді Angular-компонентів із поступовим впровадженням функціоналу відповідно до визначених вимог.

3.7 Висновки за розділом 3

У розділі 3 здійснено детальне проєктування системи управління баг-репортами. Визначено архітектуру застосунку як клієнт-серверну, де клієнт реалізований за допомогою Angular, а серверна частина – на базі ASP.NET Core з використанням Microsoft SQL Server для зберігання даних. Наведено функціональні та нефункціональні вимоги, які гарантують безпечну, надійну та зручну роботу системи. Розглянуто основні ролі користувачів у проєкті та відповідні сценарії їхньої взаємодії із системою. Також спроектовано базу

даних, яка забезпечує ефективне зберігання й обробку інформації про користувачів, проекти, баг-репорти та історію змін.

Крім того, було виконано моделювання поведінки системи за допомогою UML-діаграми станів, що дозволило описати життєвий цикл баг-репорту та формалізувати переходи між його можливими станами.

На завершення розділу представлено варфрейми інтерфейсу користувача, що демонструють структуру й логіку майбутніх вебсторінок системи, а також візуалізують взаємодію користувача з основним функціоналом.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ БАГ-РЕПОРТАМИ

У цьому розділі подано реалізацію вебзастосунку для управління баг-репортами, створеного на основі вимог, сформульованих під час етапу проєктування. Представлено структурований опис модулів серверної та клієнтської частин системи, згрупованих за їх функціональним призначенням: робота з базою даних, обробка запитів, керування процесами, представлення інтерфейсу та передача даних.

Також у цьому розділі наведено типові сценарії взаємодії користувачів із системою, що ілюструють її функціональні можливості. На завершення подано результати тестування, яке проводилося з метою перевірки коректності реалізації функцій вебзастосунку.

4.1 Опис модулів реалізованої системи

У таблицях 4.1 – 4.4 представлено структурований опис модулів серверної частини розробленого програмного забезпечення. Таблиця 4.1 містить опис модулів, що відповідають за взаємодію з базою даних. Таблиця 4.2 містить опис модулів, які використовуються для передачі та обробки даних, зокрема у взаємодії з клієнтською частиною.

Таблиця 4.1 – Опис модулів взаємодії з базою даних

№	Модуль	Опис
1	BugReport	Представляє звіт про баг, включаючи заголовок, опис, кроки для відтворення, пріоритет, серйозність та пов'язаних користувачів
2	BugReport-UpdateHistory	Містить історію змін статусів по кожному багу
3	FileData	Зберігає інформацію про файли, пов'язані з багами
4	Project	Представляє проєкт: назва, опис
5	TestersTask	Містить завдання для тестерів
6	UserData	Зберігає загальні дані про користувача
7	UserProject	Зв'язує користувача з проєктом, описує його роль у проєкті

Таблиця 4.2 – Опис модулів для обміну та обробки даних

№	Модуль	Опис
1	UserLoginData	Дані, які користувач вводить для авторизації: логін та пароль
2	UserRegistration-Data	Дані, які подаються користувачем при реєстрації
3	BugFixConfirmationRequest	Дані для підтвердження того, що помилка виправлена та протестована
4	BugReportDeveloperAssignment-Request	Дані для призначення розробника до конкретного баг-репорту
5	BugReportPassOn-Request	Дані для передачі баг-репорта наступному користувачу, згідно життєвому циклу баг-репорту
6	BugReportRejectionRequest	Дані для відхилення баг-репорта
7	BugIdRequest	Дані, що містять ідентифікатор баг-репорта
8	BugsByPriority	Містить список багів, відсортованих за пріоритетом
9	BugsBySeverity	Містить список багів, відсортованих за критичністю
10	ProjectIdRequest	Дані, що містять ідентифікатор проєкту
11	ProjectStatistics-Developer	Статистика проєкту для розробника
12	ProjectStatistics-Head	Статистика проєкту для головного розробника або головного тестера
13	ProjectStatistics-Tester	Статистика проєкту для тестера
14	RoleAndProject-Request	Дані для отримання статистики проєкту для користувача
15	UserLoginData	Дані, які користувач вводить для авторизації: логін та пароль
16	UserRegistration-Data	Дані, які подаються користувачем при реєстрації
17	BugFixConfirmationRequest	Дані для підтвердження того, що помилка виправлена та протестована
18	BugReportDeveloperAssignment-Request	Дані для призначення розробника до конкретного баг-репорту
19	BugReportPassOn-Request	Дані для передачі баг-репорта наступному користувачу, згідно життєвому циклу баг-репорту
20	BugReport-RejectionRequest	Дані для відхилення баг-репорта
21	BugIdRequest	Дані, що містять ідентифікатор баг-репорта
22	BugsByPriority	Містить список багів, відсортованих за пріоритетом
23	BugsBySeverity	Містить список багів, відсортованих за критичністю
24	ProjectIdRequest	Дані, що містять ідентифікатор проєкту

Продовження таблиці 4.2

№	Модуль	Опис
25	ProjectStatistics-Developer	Статистика проєкту для розробника
26	ProjectStatisticsHead	Статистика проєкту для головного розробника або головного тестера
27	ProjectStatistics-Tester	Статистика проєкту для тестера
28	RoleAndProject-Request	Дані для отримання статистики проєкту для користувача
29	TaskAssignment	Дані для призначення тестеру завдання
30	TesterWithTasks	Інформація про тестера разом зі списком призначених йому завдань
31	BugReportCreateDto	Дані, необхідні для створення нового баг-репорта
32	ProjectCreateDto	Дані, які передаються для створення нового проєкту
33	ProjectIdRequestDto	Дані, що містять ідентифікатор проєкту і використовуються для отримання назви проєкту
34	UserIdRequestDto	Дані, що містять ідентифікатор користувача і використовуються для отримання імя та прізвища користувача
35	AddTesterToProject	Дані для додавання тестера до вказаного проєкту
36	AddWorkerTo-ProjectByHead-Developer	Дані для додавання користувача (головного тестера або розробника) в проєкт головним розробником
37	DataForBug-ReportsRequest	Дані для отримання всіх баг-репортів у проєкті
38	ProjectData	Структура, що містить повну інформацію про проєкт та користувачів у ньому
39	ProjectRequest	Дані для отримання всієї інформації про проєкт
40	TesterAndProjectDto	Дані для отримання завдань для тестера
41	UserShortData	Стисла інформація про користувача
42	UserIdRequest	Дані для отримання всіх проєктів для користувача

Таблиця 4.3 містить опис модулів, що реалізують логіку обробки запитів. Таблиця 4.4 містить опис сервісних модулів, а також інтерфейсів, що визначають їх функціональність.

Такий поділ дозволяє чітко структурувати кодову базу та спростити її підтримку. Кожен модуль виконує окреме завдання в межах архітектури серверної частини програмного забезпечення, що відповідає принципам

розділення відповідальностей. Усі частини системи взаємодіють через добре визначені інтерфейси, що забезпечує модульність і гнучкість.

Таблиця 4.3 – Опис модулів керування

№	Модуль	Опис
1	Authentication-Controller	Обробка запитів на вхід та реєстрацію користувачів
2	BugReportFlow-Controller	Управління життєвим циклом баг-репорта
3	CommonController	Обробка запитів від допоміжних сторінок, що є частинами головної сторінки
4	FileController	Обробка створення та завантаження файлів
5	FormController	Робота з запитами від форм
6	MainController	Обробка запитів від головних сторінок застосунку
7	ProjectController	Обробка запитів, пов'язаних з проектами

Таблиця 4.4 – Опис сервісних модулів та інтерфейсів

№	Модуль	Опис
1	IAuthentication-Service	Інтерфейс сервісу для обробки авторизації та реєстрації користувачів
2	Authentication-Service	Реалізація логіки входу та реєстрації
3	IBugReportFlow-Service	Інтерфейс сервісу для керування життєвим циклом баг-репортів
4	BugReportFlow-Service	Реалізація дій: передача між ролями, підтвердження, відмова від обробки багу
5	ICommonService	Інтерфейс сервісу для обробки допоміжних запитів на головній сторінці
6	CommonService	Реалізація функцій для наповнення головної сторінки додатковими даними
7	IFileService	Інтерфейс сервісу для роботи з файлами
8	FileService	Реалізація логіки завантаження, збереження та обробки файлів
9	IFormService	Інтерфейс сервісу для роботи з формами
10	FormService	Реалізація функцій створення проекту, баг-репорта, та формування даних для форм
11	IMainService	Інтерфейс сервісу для основних сторінок
12	MainService	Реалізація логіки отримання основних даних для головних сторінок
13	IProjectService	Інтерфейс сервісу для обробки проектів
14	ProjectService	Реалізація логіки отримання даних проектів користувача

У таблицях 4.5 – 4.6 представлено структурований опис модулів клієнтської частини розробленого програмного забезпечення. Таблиця 4.5 містить опис модулів, що відповідають за представлення. Таблиця 4.6 містить опис модулів, які використовуються для передачі та обробки даних, зокрема у взаємодії з серверною частиною.

Такий поділ клієнтської частини на логічні шари забезпечує чітку структуру та спрощує налагодження. Завдяки цьому досягається гнучкість і масштабованість застосунку при збереженні високої продуктивності. Крім того, це полегшує командну розробку, оскільки різні фахівці можуть працювати над окремими модулями без виникнення конфліктів.

Таблиця 4.5 – Опис модулів представлення

№	Модуль	Опис
1	Login	Сторінка авторизації користувача в системі
2	Register	Сторінка реєстрації нового користувача
3	Auth	Компонент-обгортка для модулів авторизації та реєстрації
4	NotFound	Сторінка, що відображається при спробі переходу на неіснуючий маршрут
5	AddBugReportForm	Форма створення нового баг-репорту
6	AddProjectForm	Форма створення нового проєкту
7	Form	Компонент-обгортка для форм
8	GiveTask	Компонент для видачі завдань тестувальникам
9	InfoAboutProject	Докладна інформація про вибраний проєкт
10	ListBugReports	Компонент для перегляду списку баг-репортів проєкту
11	OpenBugReport	Відображення детальної інформації про конкретний баг-репорт
12	ProjectStatistics	Статистика у межах проєкту
13	MainDeveloper	Головна сторінка розробника
14	MainHeadDeveloper	Головна сторінка головного розробника
15	MainHeadTester	Головна сторінка головного тестера
16	MainNoUser	Початкова сторінка до авторизації, з описом системи та переходами до входу чи реєстрації
17	MainTester	Головна сторінка тестера
18	Main	Компонент-обгортка для головних сторінок різних ролей
19	ProjectSelector	Компонент вибору проєкту зі списку доступних
20	Toast	Компонент для коротких повідомлень

Таблиця 4.6 – Опис модулів для передачі та обробки даних

№	Модуль	Опис
1	LoginRequest	Модель для формування запиту на авторизацію користувача
2	LoginResponse	Модель відповіді сервера при виконанні авторизації
3	RegisterRequest	Модель для формування запиту на реєстрацію нового користувача
4	AssignByHead-DeveloperRequest	Модель для надсилання запиту на додавання головного тестувальника або розробника до проєкту
5	AssignByHead-TesterRequest	Модель для надсилання запиту на додавання тестувальника до проєкту
6	AssignDeveloper-Request	Модель для надсилання запиту на призначення баг-репорту розробнику
7	AssignTaskRequest	Модель для надсилання запиту на призначення завдання тестувальнику
8	BugDataResponse	Модель відповіді сервера з детальною інформацією про баг-репорт
9	BugHistoryItem	Модель, що описує один запис в історії змін статусів баг-репорту
10	BugIdRequest	Модель запиту для отримання детальної інформації про конкретний баг-репорт та історію його змін
11	BugIdUserIdRequest	Модель запиту для переходу баг-репорту між станами або дочасного закриття
12	BugsByPriority	Модель для представлення статистичних даних про баг-репорти за пріоритетом
13	BugsBySeverity	Модель для представлення статистичних даних про баг-репорти за рівнем серйозності
14	ConfirmBugFix-Request	Модель для запиту на підтвердження виправлення баг-репорту тестером або головним тестером після його усунення розробником
15	CreateProject-Request	Модель для надсилання запиту на створення нового проєкту
16	Developer-ResponseModel	Модель відповіді сервера з даними про розробників, доступних у проєкті
17	DeveloperStats-Response	Модель відповіді сервера зі статистикою роботи розробника
18	DeveloperStats	Модель, що описує статистичні показники роботи розробника
19	Developer	Модель, що описує основні дані про розробника в межах проєкту
20	GetTesters-Response	Модель відповіді сервера з даними про доступних тестувальників у проєкті
21	HeadStatsResponse	Модель відповіді сервера зі статистикою головного розробника та головного тестера

Продовження таблиці 4.6

№	Модуль	Опис
22	HeadStats	Модель, що описує статистику головного розробника і головного тестера
23	ProjectData-Response	Модель відповіді сервера з повною інформацією про проєкт
24	ProjectData	Модель, що зберігає детальну інформацію про проєкт
25	ProjectidRequest	Модель запиту для отримання повної інформації про конкретний проєкт
26	StatsRequest	Модель запиту для отримання статистичних даних про діяльність користувачів у проєкті
27	TesterStatsResponse	Модель відповіді сервера зі статистикою тестувальника
28	TesterStats	Модель, що описує статистику роботи тестувальника
29	Testers	Модель, що містить інформацію про доступних у проєкті тестувальників
30	UserShortData	Модель, що описує стислі дані про користувача
31	BugReportRequest	Модель запиту для отримання списку баг-репортів у межах проєкту
32	BugReportResponse	Модель відповіді сервера з даними про баг-репорти проєкту
33	BugReport	Модель, що описує основні поля баг-репорту
34	TaskRequest	Модель запиту на отримання списку завдань тестувальника
35	TaskResponse	Модель відповіді сервера з переліком завдань тестера
36	Task	Модель, що описує дані одного завдання тестера

4.2 Сценарії взаємодії користувачів із застосунком

Після запуску застосунку користувач потрапляє на привітальне вікно. У верхній частині розміщено хедер із назвою системи та кнопкою “Увійти / Зареєструватися”, яка веде на сторінку авторизації. Інтерфейс побудований таким чином, щоб забезпечити інтуїтивну навігацію навіть для нових користувачів. Основна частина вікна містить короткий вступний текст, що пояснює призначення системи та її головні функції. Також у центрі розміщено ще одну таку ж кнопку для входу або реєстрації, щоб користувач міг легко перейти до наступного кроку. Інтерфейс привітального вікна представлено на рисунку 4.1.

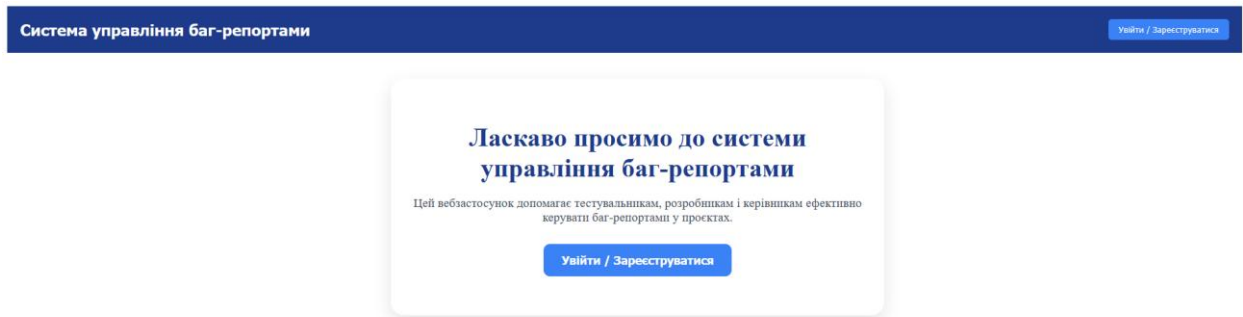


Рисунок 4.1 – Інтерфейс привітального вікна

Далі користувач переходить до форми авторизації. Спочатку відкривається вікно входу, яке містить два поля для введення електронної пошти та пароля. Для зручності реалізовано можливість перегляду введеного пароля за допомогою спеціальної кнопки. Після заповнення полів користувач натискає кнопку “Увійти”, щоб здійснити вхід до системи. Інтерфейс форми входу показано на рисунку 4.2.

Якщо у користувача ще немає облікового запису, він може скористатися формою реєстрації. У цій формі необхідно заповнити поля з особистими даними: ім’я, по батькові, прізвище, електронна пошта та номер телефону. Також користувач вводить пароль і підтвердження паролю, які повинні збігатися. Після заповнення всіх полів користувач натискає кнопку “Зареєструватися”, щоб створити обліковий запис. Інтерфейс форми реєстрації представлено на рисунку 4.3.

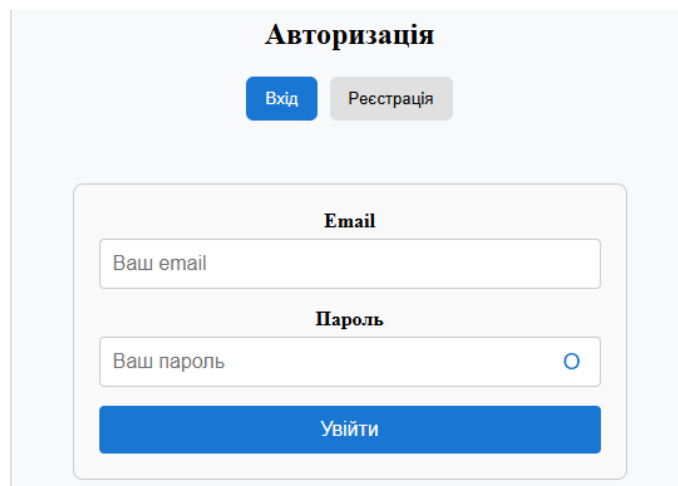


Рисунок 4.2 – Інтерфейс форми входу

Авторизація

Вхід Реєстрація

Реєстрація

Ім'я

Введіть ваше ім'я

По-батькові

Введіть ваше по-батькові

Прізвище

Введіть ваше прізвище

Email

Введіть ваш email

Номер телефону

Введіть ваш номер телефону

Пароль

Введіть пароль

Підтвердження паролю

Підтвердіть пароль

Зареєструватися

Рисунок 4.3 – Інтерфейс форми реєстрації

Після входу до облікового запису користувач потрапляє на сторінку зі списком проєктів. Якщо це не новий користувач, то на сторінці відображаються всі проєкти, у яких він бере участь, згруповані за його роллю: як тестувальник, розробник, головний тестувальник або головний розробник. Якщо ж обліковий запис тільки-но створено і користувач ще не доданий до жодного проєкту, у всіх категоріях виводиться повідомлення “Проектів немає”. Незалежно від наявності проєктів, у секції, де користувач є головним розробником, завжди присутня кнопка “Створити проєкт”, яка дозволяє ініціювати створення нового проєкту. Інтерфейс цієї сторінки представлено на рисунку 4.4.

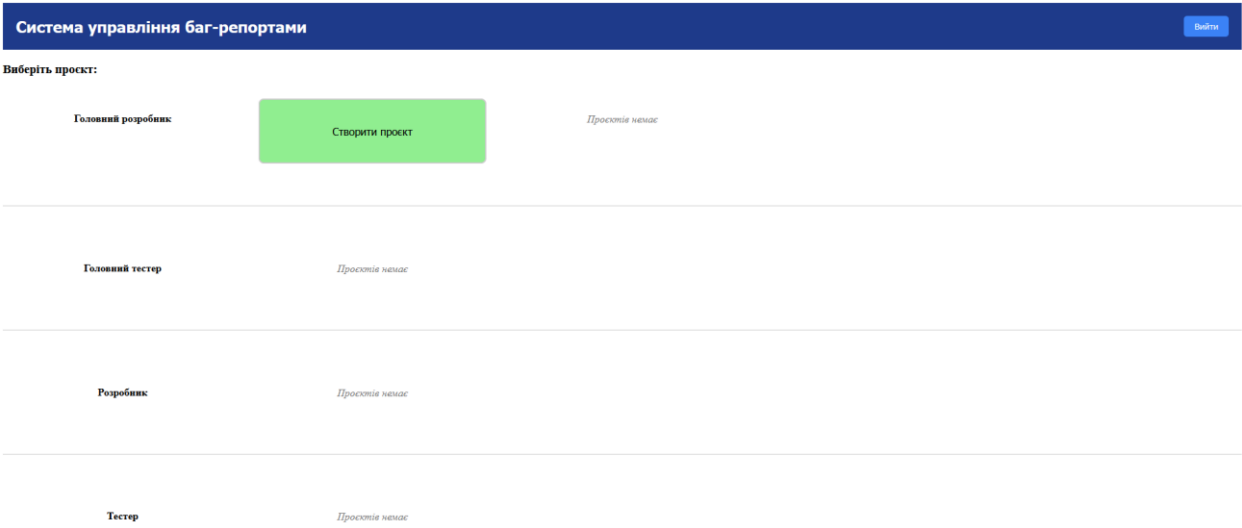


Рисунок 4.4 – Інтерфейс сторінки вибору проєкту

Якщо натиснути кнопку “Створити проєкт”, поверх основного інтерфейсу з’являється форма створення проєкту. У цій формі користувач повинен ввести назву проєкту та його опис, а потім натиснути кнопку “Створити” для підтвердження створення нового проєкту. Такий інтерфейс дозволяє швидко та зручно додати новий проєкт, не залишаючи поточного контексту. На рисунку 4.5 показано, як виглядає ця форма.

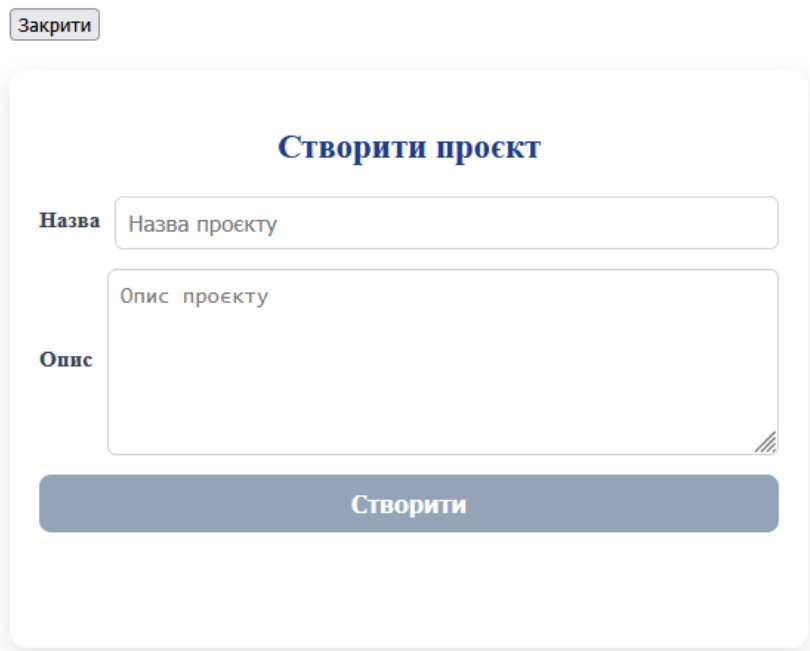


Рисунок 4.5 – Інтерфейс форми створення проєкту

Після успішного створення проєкту форма автоматично закривається, і кнопка з назвою нового проєкту з’являється у списку проєктів користувача. Таким чином, користувач одразу бачить доданий проєкт у відповідній категорії. Це забезпечує зручність та швидкий доступ до щойно створеного проєкту без необхідності перезавантажувати сторінку. На рисунку 4.6 наведено приклад оновленого списку проєктів.

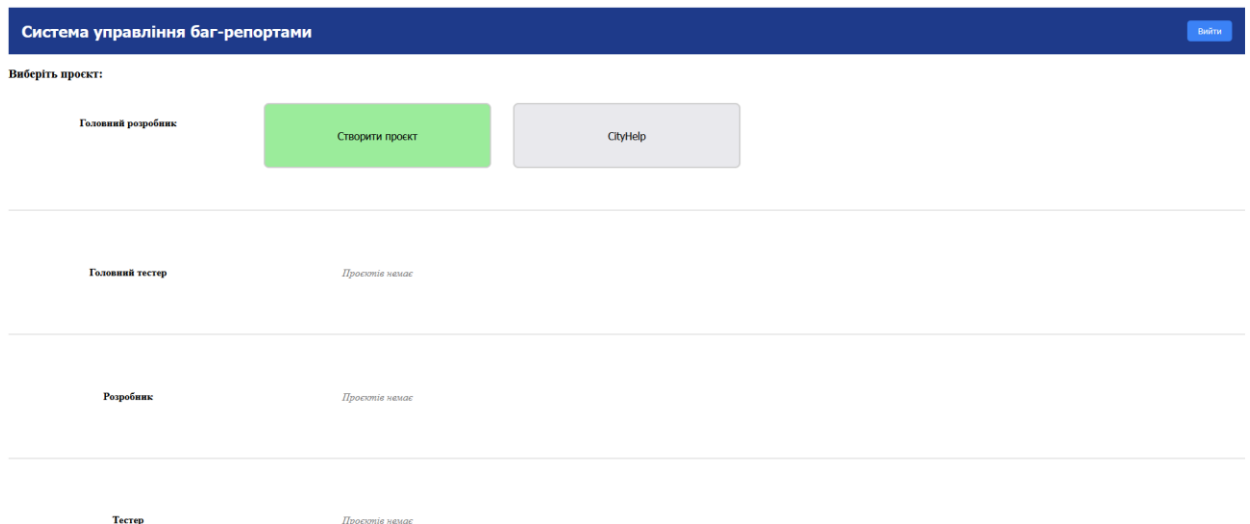


Рисунок 4.6 – Інтерфейс сторінки вибору проєкту після створення проєкту

Якщо користувач відкриває проєкт, інтерфейс адаптується залежно від його ролі в цьому проєкті. Для головного розробника та розробника доступні сторінки: “Про цей проєкт”, “Баг-репорти”, “Завершені баг-репорти” та “Статистика”. Головний тестер та тестер має доступ до: “Про цей проєкт”, “Завдання”, “Баг-репорти”, “Завершені баг-репорти” і “Статистика”. Такий розподіл дозволяє кожному користувачу отримувати доступ лише до функціоналу, необхідного для виконання його ролі в проєкті.

Розглянемо сторінку “Про цей проєкт”. Вона показує всю необхідну інформацію про проєкт та перелік учасників із їх ролями. Залежно від ролі користувача, можливості на цій сторінці відрізняються. Якщо користувач – тестер або розробник, він бачить лише інформацію про проєкт і хто в ньому бере участь, без можливості щось додавати.

Головний розробник бачить ту ж інформацію, але може додати головного тестера лише один раз, а також додавати будь-яку кількість розробників.

Головний тестер має змогу додавати необмежену кількість тестерів, при цьому теж бачить усю інформацію про проєкт та учасників.

Приклад інтерфейсу сторінки “Про цей проєкт” представлено на рисунку 4.7.

Рисунок 4.7 – Приклад інтерфейсу сторінки “Про цей проєкт”

Сторінка “Статистика” демонструє користувачу узагальнені або персоналізовані дані, що стосуються ефективності роботи над проєктом або участі конкретного користувача в ньому.

Для головного розробника та головного тестера виводиться загальна статистика по всьому проєкту. Тут користувач бачить кількість знайдених і закритих баг-репортів, а також розподіл виявлених багів за пріоритетом і серйозністю. Це дозволяє оцінити загальний стан проєкту та ефективність процесу тестування і виправлення помилок.

Для розробника статистика персоналізована. Вона містить інформацію про кількість отриманих ним багів, успішно виправлених, середній час, витрачений на виправлення, а також відсоток повернення багів на

доопрацювання. Окремо також відображається розподіл багів за пріоритетом і серйозністю, що дає змогу оцінити складність і критичність завдань, з якими працює розробник.

Для тестера також доступна персоналізована статистика. Вона включає кількість створених багів та їх розподіл за пріоритетом і серйозністю. Це дозволяє оцінити активність тестера та важливість знайдених ним помилок.

Приклад інтерфейсу сторінки “Статистика” представлено на рисунку 4.8.

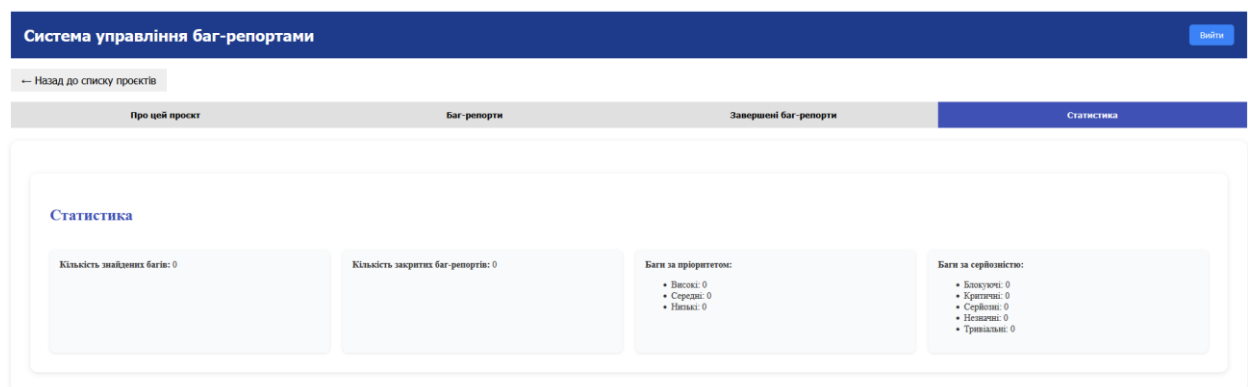


Рисунок 4.8 – Приклад інтерфейсу сторінки “Статистика”

Сторінка “Завдання” представлена у двох варіаціях, залежно від ролі користувача в проєкті.

Для головного тестера ця сторінка дає змогу керувати роботою тестерів, які залучені до проєкту. Вона містить список усіх тестерів, що беруть участь у тестуванні, а також для кожного з них відображається історія вже надісланих завдань. Головний тестер має можливість створити нове завдання для будь-якого тестера через текстове поле, що дозволяє оперативно розподіляти обов’язки й координувати процес перевірки проєкту. Приклад інтерфейсу сторінки “Завдання” для головного тестера представлено на рисунку 4.9.

Для тестера ця ж сторінка містить лише список завдань, що були надіслані особисто йому. Для кожного завдання вказується дата його

створення, що допомагає орієнтуватися у черговості та термінах виконання. Таким чином, тестер бачить лише релевантну для себе інформацію та може зосередитися на виконанні поставлених задач. Приклад інтерфейсу сторінки “Завдання” для тестера представлено на рисунку 4.10.

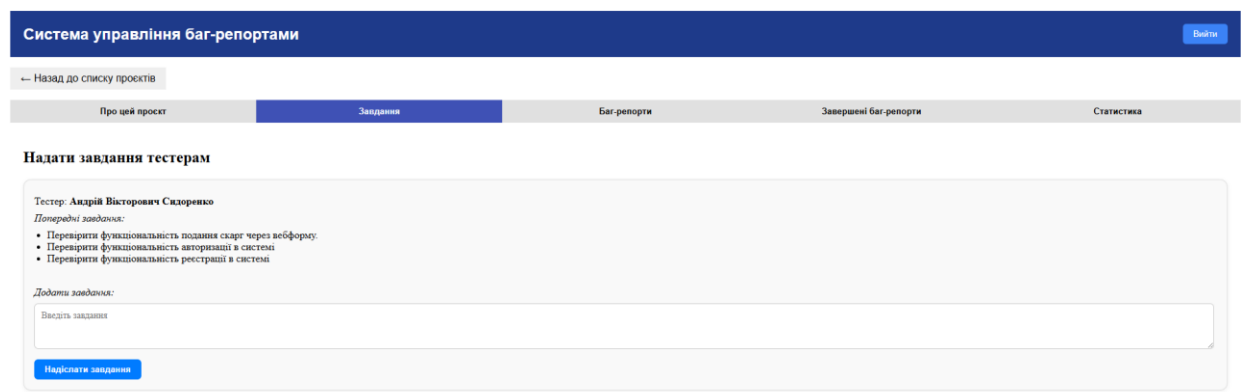


Рисунок 4.9 – Приклад інтерфейсу сторінки “Завдання” для головного тестера

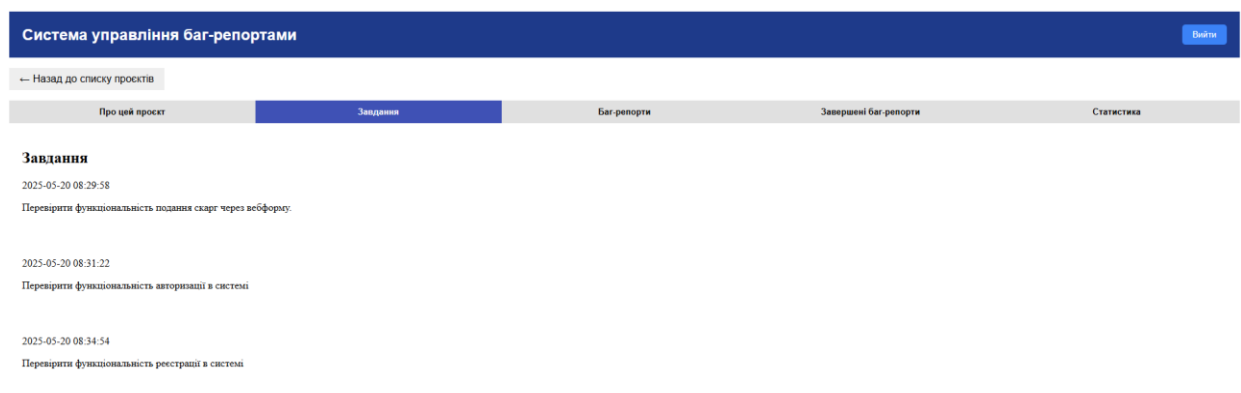


Рисунок 4.10 – Приклад інтерфейсу сторінки “Завдання” для тестера

Сторінки “Баг-репорти” та “Завершені баг-репорти” візуально практично ідентичні. Основна відмінність полягає лише в назві сторінки, яка вказує на тип відображених записів. На сторінці “Баг-репорти”, у тестера додатково присутня кнопка “Новий баг”, яка відкриває відповідну форму для створення нового баг-репорту.

Обидві сторінки містять таблицю, що відображає ключову інформацію про баг-репорти. У таблиці представлені такі стовпці: ідентифікатор баг-репорту, назва, пріоритет, серйозність, статус, дата створення, а також кнопка “Відкрити”, яка відкриває деталі відповідного запису. Для зручності перегляду користувач може натискати на заголовки стовпців, щоб здійснити сортування даних за вибраним параметром. Інтерфейс сторінок “Баг-репорти” та “Завершені баг-репорти” представлено на рисунку 4.11.

Система управління баг-репортами

← Назад до списку проєктів

Про цей проєкт

Завдання

Баг-репорти

Завершені баг-репорти

Статистика

+ Новий баг

Список баг-репортів

ID	Назва	Пріоритет	Серйозність	Статус	Дата	Більше інформації
13	Відсутня перевірка email	Середній	Серйозний	New	20 Травня 2025 року о 10:54:00 (GMT +2)	Відкрити
14	Некоректне відображення даних	Середній	Серйозний	New	20 Травня 2025 року о 11:31:00 (GMT +2)	Відкрити
15	Помилка входу	Високий	Критичний	New	20 Травня 2025 року о 11:34:00 (GMT +2)	Відкрити
16	Відсутність злітну символів	Низький	Серйозний	New	20 Травня 2025 року о 11:42:00 (GMT +2)	Відкрити

Рисунок 4.11 – Інтерфейсу сторінок “Баг-репорти” та “Завершені баг-репорти”

У тестера на сторінці "Баг-репорти" розташована кнопка "Новий баг", натискання на яку викликає форму створення баг-репорту, що відкривається поверх решти вмісту. Форма дозволяє внести всю необхідну інформацію для створення нового баг-репорту.

У верхній частині форми знаходяться два поля – “Тестер” та “Проект”, які є лише для перегляду і не підлягають редагуванню. Вони автоматично заповнюються системою відповідно до поточного користувача та проєкту, в якому створюється репорт, щоб допомогти уникнути помилок і переконатися, що баг-репорт дійсно стосується потрібного проєкту.

Далі користувач повинен заповнити поля “Назва”, “Опис”, а також “Кроки для відтворення”, де детально описується, як відтворити помилку.

Після цього необхідно обрати значення з двох випадających списків – “Пріоритет” та “Серйозність”.

Форма також містить можливість додати файл – кнопка “Оберіть файл” відкриває провідник системи, де можна вибрати потрібний файл. Якщо файл обрано, поруч з’являється його назва. У разі, якщо файл не вибрано, відображається повідомлення “Файл не вибрано”. Для завантаження вибраного файлу у систему передбачено кнопку “Завантажити”.

У нижній частині форми розміщується кнопка “Створити”, яка після натискання надсилає всі заповнені дані, створює новий баг-репорт у системі та автоматично закриває вікно форми. Інтерфейс форми створення нового баг-репорту представлено на рисунку 4.12.

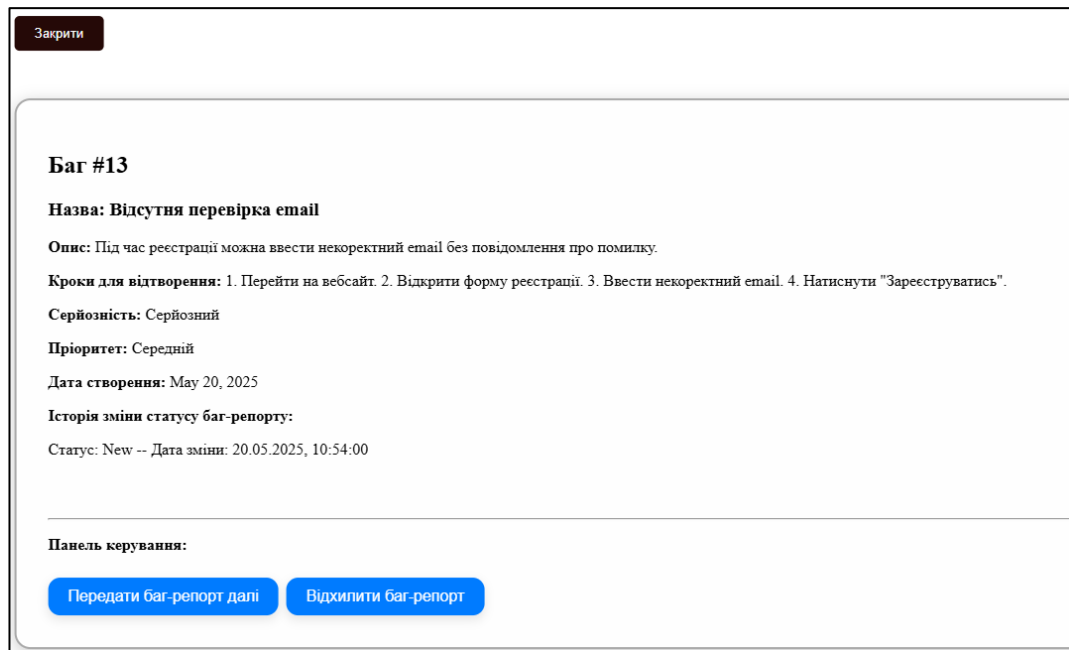
The image shows a web form titled "Створити баг-репорт" (Create bug report). At the top left is a "Закрити" (Close) button. The form contains several input fields and dropdown menus: "Тестер" (Tester) with the value "Андрій Сидоренко", "Проект" (Project) with "CityHelp", "Назва" (Name) with "Короткий заголовок", "Опис" (Description) with a placeholder "Опиши помилку...", "Кроки для відтворення" (Steps to reproduce) with a placeholder "Опиши покроково, як отримати цю помилку...", "Пріоритет" (Priority) with a dropdown showing "Середній", and "Серйозність" (Severity) with a dropdown showing "Незначний". Below these is a section for "Скріншот (тільки зображення)" (Screenshot (images only)) containing an "Оберіть файл" (Select file) button, the text "Файл не вибрано" (File not selected), and a "Завантажити" (Upload) button. At the bottom is a large blue "Створити" (Create) button.

Рисунок 4.12 – Інтерфейс форми створення нового баг-репорту

Остання сторінка відображає детальну інформацію про обраний баг-репорт. На ній виводяться всі характеристики: ідентифікатор, назва, опис, кроки для відтворення помилки, серйозність, пріоритет, дата створення, а також історія зміни статусів – тобто, перелік змін, які відбувалися з цим баг-репортом у процесі його обробки. Користувач може переглянути, коли саме йому було призначено той чи інший статус. Якщо до баг-репорту прикріплено файл, з’являється можливість його завантажити.

У випадку, якщо баг-репорт ще не закрито, під основною інформацією з’являється панель керування. Вона залежить від поточного статусу баг-репорту та ролі користувача в проєкті, надаючи лише ті дії, які дозволені саме цьому користувачу в конкретному випадку.

Інтерфейс форми з докладним описом баг-репорту та можливістю керувати ним представлено на рисунку 4.13.



Закрити

Баг #13

Назва: Відсутня перевірка email

Опис: Під час реєстрації можна ввести некоректний email без повідомлення про помилку.

Кроки для відтворення: 1. Перейти на вебсайт. 2. Відкрити форму реєстрації. 3. Ввести некоректний email. 4. Натиснути "Зареєструватись".

Серйозність: Серйозний

Пріоритет: Середній

Дата створення: Мау 20, 2025

Історія зміни статусу баг-репорту:

Статус: New -- Дата зміни: 20.05.2025, 10:54:00

Панель керування:

Передати баг-репорт далі Відхилити баг-репорт

Рисунок 4.13 – Інтерфейс форми з докладним описом баг-репорту

4.3 Тестування системи

У якості методу тестування розробленого програмного забезпечення використано мануальне тестування. Тестування повинно проводитись у різних

браузерах (Chrome, Firefox, Opera GX, Edge). Було перевірено наступний функціонал системи:

а) реєстрація нового користувача:

- 1) відкрити сторінку реєстрації;
- 2) заповнити всі обов'язкові поля: ім'я, по батькові, прізвище, електронну пошту, телефон, пароль та підтвердження паролю;
- 3) перевірити валідацію полів (збіг пароля та підтвердження, наявність всіх полів);
- 4) натиснути кнопку “Зареєструватися” та переконатися, що новий акаунт створено;

б) авторизація користувача:

- 1) відкрити сторінку входу;
- 2) ввести правильну електронну пошту та пароль;
- 3) натиснути кнопку “Увійти” та переконатися, що користувач потрапляє на сторінку з проєктами;

в) створення нового проєкту:

- 1) увійти в обліковий запис;
- 2) у розділі “Головний розробник” натиснути кнопку “Створити проєкт”;
- 3) у формі, що відкриється, ввести назву проєкту та опис;
- 4) натиснути кнопку “Створити” та переконатися, що проєкт з'явився у списку;

г) додавання учасників до проєкту головним розробником:

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є головним розробником;
- 3) перейти на сторінку “Про цей проєкт”;
- 4) додати головного тестера до проєкту, вписавши його електронну пошту у відповідне поле та натиснувши кнопку “Додати головного тестера”;

5) додати розробника до проєкту, вписавши його електронну пошту у відповідне поле та натиснувши кнопку “Додати розробника”;

6) перевірити, що додані учасники з’явилися у списку із правильними ролями;

7) перевірити, що після додання головного тестера можливість додати ще одного зникає;

г) додавання тестерів до проєкту головним тестером:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є головним тестером;

3) перейти на сторінку “Про цей проєкт”;

4) додати тестера до проєкту, вписавши його електронну пошту у відповідне поле та натиснувши кнопку “Додати тестера”;

5) перевірити, що доданий учасник з’явився у списку із правильною роллю;

д) призначення завдань тестерам:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є головним тестером;

3) перейти на сторінку “Завдання”;

4) для кожного тестера написати завдання та надіслати його;

5) перевірити, що завдання з’явилися у відповідному списку;

е) перевірка перегляду завдань тестером:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є тестером;

3) перейти на сторінку “Завдання”;

4) перевірити, що відображаються лише призначені цьому тестеру завдання;

є) створення баг-репорту:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є тестером;

3) перейти на сторінку “Баг-репорти”;

- 4) натиснути кнопку “Новий баг”;
- 5) заповнити поля: назва, опис, кроки для відтворення, пріоритет, серйозність;
- 6) натиснути кнопку “Створити”;
- 7) перевірити, що баг-репорт з’явився у таблиці з баг-репортами;
- ж) прикріплення файлу до баг-репорту:
 - 1) увійти в обліковий запис;
 - 2) обрати проєкт, де користувач є тестером;
 - 3) перейти на сторінку “Баг-репорти”;
 - 4) натиснути кнопку “Новий баг”;
 - 5) заповнити поля: назва, опис, кроки для відтворення, пріоритет, серйозність;
 - 6) натиснути кнопку “Оберіть файл” та вибрати файл зі свого комп’ютера;
 - 7) натиснути кнопку “Завантажити”;
 - 8) перевірити, що назва вибраного файлу відображається поруч із кнопкою;
 - 9) натиснути кнопку “Створити”;
 - 10) знайти баг-репорт в таблиці баг-репортів та відкрити докладну інформацію про нього;
 - 11) переконатися, що баг-репорт створений із прикріпленим файлом;
- з) перевірка автозаповнення та незмінності полів “Тестер” і “Проект”:
 - 1) увійти в обліковий запис;
 - 2) обрати проєкт, де користувач є тестером;
 - 3) перейти на сторінку “Баг-репорти”;
 - 4) натиснути кнопку “Новий баг”;

- 5) перевірити, що поля “Тестер” і “Проект” автоматично заповнені;
- 6) перевірити, що ці поля не можна редагувати;
- и) перевірка сортування у таблиці баг-репортів:
 - 1) увійти в обліковий запис;
 - 2) обрати будь-який проєкт, де вже створено декілька баг-репортів;
 - 3) перейти на сторінку “Баг-репорти” або “Завершені баг-репорти”;
 - 4) натискати заголовки стовпців та перевірити, що елементи таблиці сортуються відповідно;
- і) перегляд інформації про баг-репорт:
 - 1) увійти в обліковий запис;
 - 2) обрати будь-який проєкт, де вже створено хоча б один баг-репорт;
 - 3) перейти на сторінку “Баг-репорти” або “Завершені баг-репорти”;
 - 4) натиснути кнопку “Більше інформації” біля будь-якого баг-репорту;
 - 5) перевірити відображення наступної інформації: ідентифікатор, назва, опис, кроки для відтворення, серйозність, пріоритет, дата створення, історія статусів;
- ї) перевірка прикріпленого файлу у баг-репорті
 - 1) увійти в обліковий запис;
 - 2) відкрити будья-який проєкт, де є створені баг-репорти;
 - 3) відкрити будь-який баг-репорт, у якому був прикріплений файл;
 - 4) перевірити наявність посилання для завантаження файлу;
 - 5) завантажити файл;

б) переконатися, що файл зберігся локально і відкривається без помилок;

й) перевірка сторінки “Статистика” (головний розробник або головний тестер):

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є головним розробником або головним тестером;

- 3) перейти на сторінку “Статистика”;

- 4) перевірити, що на сторінці відображається загальнопроєктна статистика: кількість знайдених багів, кількість закритих баг-репортів, а також розподіл багів за пріоритетами та серйозністю;

к) перевірка сторінки “Статистика” (розробник):

- 1) увійти в обліковий запис;

- 2) обрати проєкт, де користувач є розробником;

- 3) перейти на сторінку “Статистика”;

- 4) перевірити, що на сторінці відображається персональна статистика: кількість отриманих баг-репортів, кількість успішно виправлених багів, середній час виправлення, відсоток повернення баг-репортів, а також розподіл багів за пріоритетами та серйозністю;

л) перевірка сторінки “Статистика” (тестер):

- 1) увійти в обліковий запис;

- 2) обрати проєкт, де користувач є тестером;

- 3) перейти на сторінку “Статистика”;

- 4) перевірити, що на сторінці відображається персональна статистика: кількість створених баг-репортів, а також розподіл багів за пріоритетами та серйозністю;

м) відхилення баг-репорту головним тестером:

- 1) увійти в обліковий запис;

- 2) обрати проєкт, де користувач є головним тестером;

- 3) перейти на сторінку “Баг-репорти”;

- 4) знайти баг-репорт зі статусом “New”;
- 5) відкрити докладну інформацію про баг-репорт;
- 6) натиснути кнопку “Відхилити баг-репорт”;
- 7) перейти на сторінку “Завершені баг-репорти”;
- 8) знайти відхилений баг-репорт та відкрити його докладну інформацію;

9) перевірити, що в історії зміни статусів зафіксовано послідовність: “New”, “Reject”, “Closed”;

н) відхилення баг-репорту головним розробником:

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є головним розробником;
- 3) перейти на сторінку “Баг-репорти”;
- 4) знайти баг-репорт зі статусом “Open”;
- 5) відкрити докладну інформацію про баг-репорт;
- 6) натиснути кнопку “Відхилити баг-репорт”;
- 7) перейти на сторінку “Завершені баг-репорти”;
- 8) знайти відхилений баг-репорт та відкрити його докладну інформацію;

9) перевірити, що в історії зміни статусів зафіксовано послідовність: “New”, “Open”, “Reject”, “Closed”;

о) відхилення баг-репорту розробником:

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є розробником;
- 3) перейти на сторінку “Баг-репорти”;
- 4) знайти баг-репорт зі статусом “In Progress”;
- 5) відкрити докладну інформацію про баг-репорт;
- 6) натиснути кнопку “Відхилити баг-репорт”;
- 7) перейти на сторінку “Завершені баг-репорти”;
- 8) знайти відхилений баг-репорт та відкрити його докладну інформацію;

9) перевірити, що в історії зміни статусів зафіксовано послідовність: “New”, “Open”, “In Progress”, “Reject”, “Closed”;

п) передача баг-репорту від головного розробника до розробника:

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є головним розробником;
- 3) перейти на сторінку “Баг-репорти”;
- 4) знайти баг-репорт зі статусом “Open”;
- 5) відкрити докладну інформацію про баг-репорт;
- 6) відкрити випадаючий список з електронними поштами усіх розробників проєкту;
- 7) обрати розробника зі списку;
- 8) натиснути кнопку “Призначити розробника”;
- 9) перевірити, що баг-репорт зник зі списку баг-репортів головного розробника і не з’явився у списку завершених баг-репортів;
- 10) перевірити, що баг-репорт з’явився у списку баг-репортів розробника;

р) вихід із системи:

- 1) увійти в обліковий запис;
- 2) натиснути кнопку “Вийти”;
- 3) перевірити, що користувач перенаправлений на сторінку авторизації;

с) передача баг-репорту від головного тестера до головного розробника:

- 1) увійти в обліковий запис;
- 2) обрати проєкт, де користувач є головним тестером;
- 3) перейти на сторінку “Баг-репорти”;
- 4) знайти баг-репорт зі статусом “New”;
- 5) відкрити докладну інформацію про баг-репорт;
- 6) натиснути кнопку “Передати баг-репорт далі”;

7) перевірити, що баг-репорт зник зі списку баг-репортів головного тестера і не з'явився у списку завершених баг-репортів;

8) перевірити, що баг-репорт з'явився у списку баг-репортів головного розробника;

т) передача баг-репорту від розробника до тестера:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є розробником;

3) перейти на сторінку “Баг-репорти”;

4) знайти баг-репорт зі статусом “In Progress”;

5) відкрити докладну інформацію про баг-репорт;

6) натиснути кнопку “Передати баг-репорт далі”;

7) перевірити, що баг-репорт зник зі списку баг-репортів розробника і не з'явився у списку завершених баг-репортів;

8) перевірити, що баг-репорт з'явився у списку баг-репортів тестера;

у) підтвердження тестером виправлення багу:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є тестером;

3) перейти на сторінку “Баг-репорти”;

4) знайти баг-репорт зі статусом “Testing”;

5) відкрити докладну інформацію про баг-репорт;

6) натиснути кнопку “Підтвердити виправлення багу”;

7) перевірити, що баг-репорт зник зі списку баг-репортів тестера і не з'явився у списку завершених баг-репортів;

8) перевірити, що баг-репорт з'явився у списку баг-репортів головного тестера;

ф) підтвердження головним тестером виправлення багу:

1) увійти в обліковий запис;

2) обрати проєкт, де користувач є головним тестером;

3) перейти на сторінку “Баг-репорти”;

- 4) знайти баг-репорт зі статусом “Verified”;
 - 5) відкрити докладну інформацію про баг-репорт;
 - 6) натиснути кнопку “Підтвердити виправлення багу”;
 - 7) перевірити, що баг-репорт зник зі списку баг-репортів головного тестера і з’явився у списку завершених баг-репортів;
- х) повернення баг-репорту тестером на доопрацювання:
- 1) увійти в обліковий запис;
 - 2) обрати проєкт, де користувач є тестером;
 - 3) перейти на сторінку “Баг-репорти”;
 - 4) знайти баг-репорт зі статусом “Testing”;
 - 5) відкрити докладну інформацію про баг-репорт;
 - 6) натиснути кнопку “Не підтвердити виправлення багу”;
 - 7) перевірити, що баг-репорт зник зі списку баг-репортів тестера і не з’явився у списку завершених баг-репортів;
 - 8) перевірити, що баг-репорт з’явився у списку баг-репортів розробника і в історії статусів баг-репорту передостаннім статусом є “Reopened”;
- ц) повернення баг-репорту головним тестером на доопрацювання:
- 1) увійти в обліковий запис;
 - 2) обрати проєкт, де користувач є головним тестером;
 - 3) перейти на сторінку “Баг-репорти”;
 - 4) знайти баг-репорт зі статусом “Verified”;
 - 5) відкрити докладну інформацію про баг-репорт;
 - 6) натиснути кнопку “Не підтвердити виправлення багу”;
 - 7) перевірити, що баг-репорт зник зі списку баг-репортів головного тестера і не з’явився у списку завершених баг-репортів;
 - 8) перевірити, що баг-репорт з’явився у списку баг-репортів розробника і в історії статусів баг-репорту передостаннім статусом є “Reopened”.

Чек-ліст тестування розробленого програмного забезпечення наведено у таблиці 4.7.

Таблиця 4.7 – Чек-ліст тестування

№	Тест	Браузер			
		Google Chrome 136.0.7103.114	Opera GX 118.0.5461.133	Mozilla Firefox 138.0.4	Microsoft Edge 136.0.3240.92
1	2	3	4	5	6
1	Реєстрація нового користувача	Passed	Passed	Passed	Passed
2	Авторизація користувача	Passed	Passed	Passed	Passed
3	Створення нового проєкту	Passed	Passed	Passed	Passed
4	Додавання учасників до проєкту головним розробником	Passed	Passed	Passed	Passed
5	Додавання тестерів до проєкту головним тестером	Passed	Passed	Passed	Passed
6	Призначення завдань тестерам	Passed	Passed	Passed	Passed
7	Перевірка перегляду завдань тестером	Passed	Passed	Passed	Passed
8	Створення баг-репорту	Passed	Passed	Passed	Passed
9	Прикріплення файлу до баг-репорту	Passed	Passed	Passed	Passed
10	Перевірка автозаповнення та незмінності полів “Тестер” і “Проєкт”	Passed	Passed	Passed	Passed
11	Перевірка сортування у таблиці баг-репортів	Passed	Passed	Passed	Passed
12	Перегляд інформації про баг-репорт	Passed	Passed	Passed	Passed

Продовження таблиці 4.7

1	2	3	4	5	6
13	Перевірка прикріпленого файлу у баг-репорті	Passed	Passed	Passed	Passed
14	Перевірка сторінки “Статистика” (головний розробник або головний тестер)	Passed	Passed	Passed	Passed
15	Перевірка сторінки “Статистика” (розробник)	Passed	Passed	Passed	Passed
16	Перевірка сторінки “Статистика” (тестер)	Passed	Passed	Passed	Passed
17	Відхилення баг-репорту головним тестером	Passed	Passed	Passed	Passed
18	Відхилення баг-репорту головним розробником	Passed	Passed	Passed	Passed
19	Відхилення баг-репорту розробником	Passed	Passed	Passed	Passed
20	Передача баг-репорту від головного розробника до розробника	Passed	Passed	Passed	Passed
21	Вихід із системи	Passed	Passed	Passed	Passed
22	Передача баг-репорту від головного тестера до головного розробника	Passed	Passed	Passed	Passed
23	Передача баг-репорту від розробника до тестера	Passed	Passed	Passed	Passed
24	Підтвердження тестером виправлення багу	Passed	Passed	Passed	Passed
25	Підтвердження головним тестером виправлення багу	Passed	Passed	Passed	Passed
26	Повернення баг-репорту тестером на доопрацювання	Passed	Passed	Passed	Passed
27	Повернення баг-репорту головним тестером на доопрацювання	Passed	Passed	Passed	Passed

4.4 Висновки за розділом 4

У розділі 4 здійснено реалізацію та тестування вебзастосунку для управління баг-репортами. Надано детальний опис модулів серверної та клієнтської частин системи, згрупованих за їх функціональним призначенням, що дозволяє чітко уявити архітектуру застосунку та взаємодію його компонентів.

Показано, як різні ролі користувачів (тестер, розробник, головний розробник, головний тестер) взаємодіють із системою у межах життєвого циклу баг-репортів.

Проведене тестування підтвердило працездатність функцій системи та стабільність обробки даних. Таким чином, реалізований застосунок відповідає поставленим завданням та може бути використаний як інструмент для ефективної організації керування баг-репортами у командній розробці програмного забезпечення.

ВИСНОВКИ

У дипломній роботі бакалавра було успішно спроектовано та реалізовано вебзастосунок для управління баг-репортами, що відповідає сучасним вимогам до організації процесу фіксації, обробки та контролю помилок у програмному забезпеченні.

Проведений аналіз предметної області показав актуальність проблематики, пов'язаної з ефективною взаємодією між учасниками команд розробки, а також виявив потребу в створенні інструменту, який би забезпечував прозорий, зручний та адаптивний підхід до управління баг-репортами.

На основі вивчення типових сценаріїв взаємодії між тестувальниками, розробниками та керівниками проєктів було розроблено концепцію системи, що враховує потреби всіх категорій користувачів. В основі вебзастосунку – структурована архітектура з чітким розподілом функціональних обов'язків згідно з ролями користувачів у межах кожного проєкту.

Розроблений інтерфейс є інтуїтивно зрозумілим та функціонально насиченим: реалізовано механізми створення, фільтрації, сортування та зміни статусу баг-репортів. Застосунок забезпечує структуровану взаємодію між учасниками команд тестувальників та розробників під час роботи з баг-репортами, що дозволяє зменшити час реагування на знайдені помилки та покращити якість кінцевого продукту.

Оцінка розробленого програмного забезпечення з погляду зручності використання, функціональності та організації командної роботи підтвердила його ефективність. Створена система здатна підтримувати декілька паралельних проєктів, масштабуватися та зберігати стабільність під час активної роботи кількох користувачів.

Перспективи подальшого розвитку вебзастосунку включають розширення функціональності за рахунок інтеграції з системами контролю

версій, удосконалення засобів аналітики та візуалізації звітів, а також оптимізацію інтерфейсу для кращого користувацького досвіду.

Таким чином, створений вебзастосунок для управління баг-репортами є корисним програмним рішенням для команд розробників, яке сприяє підвищенню ефективності процесів тестування та вдосконалення якості програмного забезпечення в цілому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке Bug і Bug Report в тестуванні? | стаття EPAM Campus [Електрон. ресурс]. – Режим доступу: <https://campus.epam.ua/ua/blog/460>.
2. Огляд популярних баг-трекінгових систем [Електрон. ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/helpful-materials/overview-of-popular-bug-tracking-systems/>.
3. Jira | Issue & Project Tracking Software | Atlassian [Electronic resource]. – Access mode: <https://www.atlassian.com/software/jira>.
4. Jira vs Trello vs Asana: Best for PM in 2025 [Electronic resource]. – Access mode: <https://productive.io/blog/jira-vs-trello-vs-asana/>.
5. Домашня сторінка | Trello [Електрон. ресурс]. – Режим доступу: <https://trello.com>.
6. Home – Asana [Electronic resource]. – Access mode: <https://app.asana.com>.
7. monday – Home [Electronic resource]. – Access mode: <https://monday.com>.
8. What is monday.com and Why Do You Need It? | enable.services [Electronic resource]. – Access mode: <https://www.enable.services/2024/05/16/what-is-monday-com-and-why-do-you-need-it/>.
9. Express - це фреймворк для веб-застосунків, побудованих на Node.js [Електрон. ресурс]. – Режим доступу: <https://expressjs.com/uk/>.
10. ASP.NET vs ExpressJS - A Detailed Comparison | Flexiple – Flexiple [Electronic resource]. – Access mode: <https://flexiple.com/compare/asp-dot-net-vs-express-js>.
11. Xiaodi Yan. Web API Development with ASP.NET Core 8. Бірмінгем, Велика Британія: Вид-во Packt Publishing, 2024. 804 с.
12. ASP.NET vs Django - A Detailed Comparison | Flexiple – Flexiple [Electronic resource]. – Access mode: <https://flexiple.com/compare/asp-dot-net-vs->

django.

13. Що таке Django і як його застосовувати в Python [Електрон. ресурс]. – Режим доступу: <https://highload.tech/uk/shho-take-django-i-yak-z-nym-pratsyuvaty-v-python/>.

14. Object Oriented Programming Concepts in C# (2023) [Electronic resource]. – Access mode: <https://www.c-sharpcorner.com/UploadFile/mkagrahari/introduction-to-object-oriented-programming-concepts-in-C-Sharp/>.

15. Parallel programming in C# - DEV Community [Electronic resource]. – Access mode: <https://dev.to/kamilbugnokrk/parallel-programming-in-c-53eg>.

16. Exploring C#'s Garbage Collector: How It Works | Medium [Electronic resource]. – Access mode: <https://medium.com/%40kmorpex/garbage-collector-in-c-how-does-it-work-bbcc88ef45c2>.

17. Visual Studio Code vs Visual Studio – Are They The Same? - Techvify [Electronic resource]. – Access mode: <https://techvify.com/visual-studio-code-vs-visual-studio/>.

18. Free (Non-Commercial) JetBrains Rider vs. Visual Studio 2022 Community Edition -- Visual Studio Magazine [Electronic resource]. – Access mode: <https://visualstudiomagazine.com/Articles/2024/10/28/Compare-New-Non-Commercial-JetBrains-Rider-with-Visual-Studio-Community-Edition.aspx>.

19. Advanced configuration | JetBrains Rider Documentation [Electronic resource]. – Access mode: https://www.jetbrains.com/help/rider/Tuning_the_IDE.html.

20. Angular VS VueJS - DEV Community [Electronic resource]. – Access mode: <https://dev.to/lenormor/angular-vs-vuejs-2ea6>.

21. TypeScript: Documentation - TypeScript for JavaScript Programmers [Electronic resource]. – Access mode: <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>.

22. WebStorm vs VS Code: Key Differences, Features & Performance [Electronic resource]. – Access mode:

<https://www.techrepublic.com/article/webstorm-vs-vscode/>.

23. MySQL vs MSSQL: What are the Key Differences? [Electronic resource]. – Access mode: <https://www.plesk.com/blog/various/mysql-vs-mssql/>.

24. MySQL vs MSSQL: Key Differences You Need to Know [Electronic resource]. – Access mode: https://www.znetlive.com/blog/mysql-vs-mssql-key-differences/?srsltid=AfmBOoqnpUSDW0lkrFIIUAaJwctvN_JPG9YINcpDFPFfqO8FTYIBPCV.

25. Microsoft SQL Server vs. MySQL vs. SQLite Comparison [Electronic resource]. – Access mode: <https://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BSQLite>.

ДОДАТОК А
Технічне завдання

Вступ

У сучасному світі при розробці програмного забезпечення надзвичайно важливою є ефективна взаємодія між членами команди, зокрема у процесах фіксації, обробки та усунення програмних помилок. Баг-репорти є ключовим елементом комунікації між тестувальниками та розробниками, тому потребують чіткої організації та контролю.

Даний технічний проєкт передбачає розробку вебзастосунку для управління баг-репортами, який дозволить автоматизувати процеси реєстрації, призначення відповідальних осіб, зміни статусів та закриття баг-репортів. Особливістю запропонованого рішення є підтримка ролей користувачів, зручний інтерфейс українською мовою та відображення статистики.

Розробка спрямована на створення інтуїтивно зрозумілого вебінтерфейсу відповідно до сучасних стандартів користувацького досвіду.

Вебзастосунок буде доступним через браузер та орієнтованим на використання в командах розробників і тестувальників. Рішення буде адаптоване під потреби команд, яким важливо мати простий, але потужний інструмент для взаємодії у процесі тестування програмного забезпечення.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему: “Проєктування та розробка вебзастосунку для управління баг-репортами”, затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Вебзастосунок призначений для автоматизації процесів фіксації, обробки та контролю баг-репортів у рамках командної розробки програмного

забезпечення. Його використання забезпечує централізоване управління баг-репортами, розподіл обов'язків між учасниками проєкту та покращення взаємодії між тестувальниками, розробниками та їх керівниками.

Основні цілі розробки:

а) забезпечення ефективного керування баг-репортами протягом усього їхнього життєвого циклу:

- 1) реєстрація, опрацювання та закриття баг-репортів;
- 2) відхилення баг-репортів та підтвердження усунення помилки;
- 3) призначення відповідальних осіб;

б) реалізація чіткої ролевої моделі доступу:

- 1) підтримка ролей: тестувальник, розробник, головний розробник, головний тестувальник;
- 2) контроль дій користувачів відповідно до їх ролей у проєкті;

в) оптимізація процесу усунення помилок:

- 1) фільтрація за пріоритетами, серйозністю, датою створення;
- 2) виведення статистики;

г) підтримка сучасних вимог до інтерфейсу та доступності:

- 1) адаптивний інтерфейс користувача;
- 2) повна локалізація українською мовою;
- 3) зручна навігація й зрозумілий дизайн.

А.3 Вимоги до програми чи програмного виробу

Розробка вебзастосунку для управління баг-репортами базується на забезпеченні чітко визначених вимог, які гарантують його функціональність, сумісність, продуктивність та зручність використання.

Вимоги до програмного забезпечення поділяються на три основні категорії:

- функціональні вимоги, які визначають основні завдання та функції, що виконує додаток. Вони забезпечують можливість користувачам виконувати всі необхідні дії, включаючи авторизацію та реєстрацію, керування баг-репортами та проєктами;
- нефункціональні вимоги, що регламентують якість роботи програми. Ці вимоги враховують продуктивність, стабільність та безпеку даних;
- обмеження, які окреслюють технічні рамки розробки, такі як необхідність інтернет-з'єднання.

A.3.1 Функціональні вимоги

Функціональні вимоги до програмного забезпечення:

- а) робота з користувачами:
 - 1) реєстрація та авторизація користувачів;
 - 2) призначення ролей;
- б) робота з баг-репортами:
 - 1) створення, редагування, перегляд, фільтрація та закриття багів;
 - 2) призначення відповідального виконавця;
 - 3) встановлення пріоритету та серйозності (Severity);
 - 4) прикріплення файлів;
- в) управління проєктами:
 - 1) створення та редагування проєктів;
 - 2) призначення користувачів до проєктів;
 - 3) виведення статистики за обраним проєктом або користувачем;
- г) робота з завданнями на тестування:
 - 1) призначення завдання головним тестувальником;
 - 2) перегляд завдань тестувальником.

А.3.2 Нефункціональні вимоги

Нефункціональні вимоги до програмного забезпечення:

а) продуктивність:

1) час відгуку застосунку на стандартні операції – не більше 2 секунд;

б) сумісність:

1) підтримка браузерів Google Chrome, Mozilla Firefox, Microsoft Edge та Opera GX;

в) безпека:

1) захист паролей користувачів хешуванням;

2) контроль доступу до функціоналу відповідно до ролей;

г) інтерфейс:

1) інтуїтивно зрозумілий адаптивний дизайн;

2) повна підтримка української мови;

г) масштабованість:

1) можливість додавання нових модулів у майбутньому.

А.3.3 Обмеження

Обмеження до програмного забезпечення:

— для коректної роботи застосунку необхідне постійне підключення до мережі.

А.4 Вимоги до програмної документації

У межах розробки вебзастосунку для управління баг-репортами програмна документація представлена двома основними документами: технічним завданням і пояснювальною запискою до дипломної роботи. Ці документи повинні містити повну, структуровану та зрозумілу інформацію

для забезпечення ефективної розробки, тестування, впровадження та оцінки якості системи.

Документація має відповідати таким вимогам:

а) повнота та структурованість:

1) вся інформація повинна бути логічно впорядкована відповідно до загальноприйнятої структури дипломної роботи;

2) пояснювальна записка охоплює всі етапи життєвого циклу програмного продукту: від аналізу предметної області до реалізації та тестування;

3) технічне завдання має містити чітко визначені вимоги до функціональності, інтерфейсу, продуктивності та обмежень системи;

б) актуальність:

1) уся документація повинна відповідати поточному стану реалізації системи;

2) зміни в реалізації повинні бути синхронізовані з відповідними розділами пояснювальної записки;

в) відповідність стандартам:

1) документація оформлюється відповідно до вимог ДСТУ 3008:2015 (для структури пояснювальної записки);

г) формати подання:

1) документи повинні бути доступними в цифровому форматі.

A.5 Техніко-економічні показники

Розробка вебзастосунку для управління баг-репортами має на меті досягнення оптимального співвідношення між функціональністю, зручністю використання та витратами на реалізацію. Система орієнтована на невеликі та середні команди розробників, що не мають змоги використовувати складні та дорогі корпоративні рішення. Основні техніко-економічні показники:

а) ефективність розробки:

1) використання сучасних фреймворків дозволяє скоротити час розробки завдяки модульному підходу та наявності готових бібліотек;

б) економічна доцільність:

1) вебзастосунок не потребує закупівлі платних ліцензій або серверного ПЗ, бо достатньо стандартного вебсервера та бази даних;

2) рішення може бути адаптоване під різні проєкти без значних фінансових витрат;

3) скорочення витрат на ручне ведення звітності та зменшення кількості повторних помилок у проєкті за рахунок централізованого контролю;

в) технічні показники:

1) час відгуку застосунку на типові запити – до 500 мс;

2) максимальна кількість одночасних користувачів – до 200 за базової конфігурації;

3) архітектура підтримує масштабування при зростанні команди або кількості проєктів;

г) зручність та доступність:

1) адаптивний інтерфейс для роботи з десктопів і ноутбуків;

2) повна локалізація інтерфейсу українською мовою;

3) інтуїтивна навігація дозволяє швидко навчити нових користувачів системі;

д) перспективи розвитку:

1) додавання модуля звітності у форматі PDF або Excel.

Таким чином, розробка додатку забезпечує баланс між технічними характеристиками, зручністю та економічною вигодою для користувачів.

A.6 Порядок контролю та приймання

Порядок контролю та приймання вебзастосунку для управління баг-репортами визначає етапи перевірки відповідності розробленого програмного

продукту встановленим функціональним і нефункціональним вимогам, а також процедуру офіційного приймання результатів роботи.

A.6.1 Етапи контролю

Основні етапи контролю:

а) первинне тестування:

1) перевірка працездатності основного функціоналу (створення, фільтрація, зміна статусів та закриття баг-репортів);

2) тестування реєстрації, авторизації, розподілу ролей та завдань, створення проєкту;

б) функціональне тестування:

1) верифікація реалізації всіх заявлених функціональних вимог;

2) перевірка доступності функцій згідно з ролями: тестувальник, розробник, головний розробник, головний тестувальник;

в) кросбраузерне тестування:

1) перевірка стабільності роботи застосунку у популярних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Opera GX);

г) фінальне тестування:

1) демонстрація всіх основних функцій системи перед замовником (керівником проєкту);

2) заповнення чек-ліста приймання з верифікацією відповідності вимогам.

A.6.2 Критерії приймання

Основні критерії приймання:

— повна відповідність функціональним та нефункціональним вимогам, зазначеним у розділі A.4;

- відсутність критичних помилок, що впливають на базовий функціонал;
- стабільна робота під навантаженням;
- коректна реалізація ролевої моделі;
- позитивний результат проходження контрольного тестування.

A.6.3 Документація для приймання

Під час здачі проєкту замовнику передається наступна документація:

- файли проєкту (вихідні коди);
- дампи бази даних;
- пояснювальна записка до дипломної роботи;
- технічне завдання (цей документ).

A.6.4 Процедура приймання

Замовник проводить незалежну перевірку роботи додатку відповідно до наданої документації. У разі успішного проходження всіх тестів і відсутності зауважень здійснюється підписання акту приймання.

Цей порядок контролю та приймання дозволяє забезпечити високу якість програмного продукту, відповідність його функціоналу заявленим вимогам та готовність до використання кінцевими користувачами.

ДОДАТОК Б
Слайди презентації

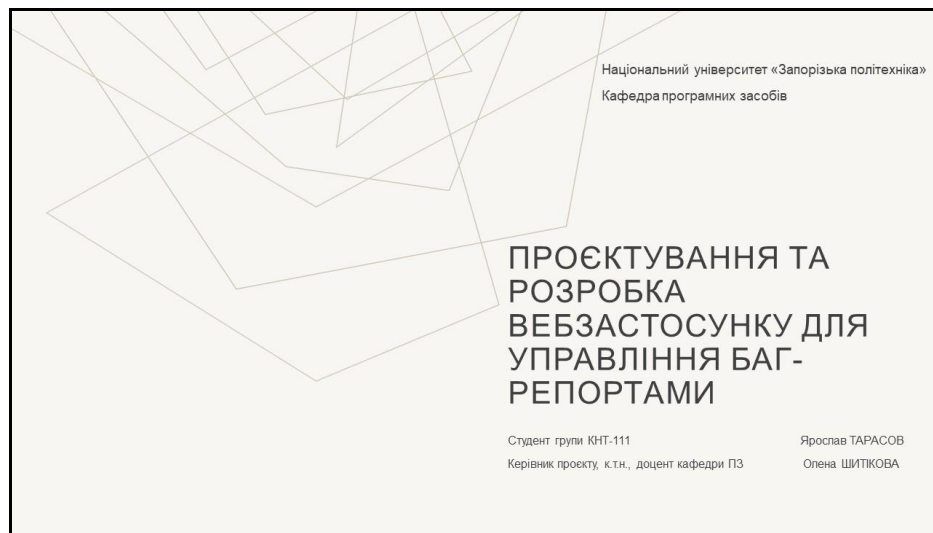


Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2



Рисунок Б.3 – Слайд 3

ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ

Критерій	Express.js	ASP.NET Core	Django
Швидкий старт розробки	+	-	+
Висока продуктивність	+	+	-
Вбудована безпека (CSRF, аутентифікація)	-	+	+
Вбудована структура проєкту	-	+	+
Вбудована ORM	-	-	+
Підходить для мікросервісної архітектури	+	+	-
Простота для новачків	+	-	-

4

Рисунок Б.4 – Слайд 4

ПОРІВНЯННЯ IDE ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ

Критерій	Visual Studio	Visual Studio Code	Rider
Повна підтримка ASP.NET Core	+	-	+
Кросплатформеність	-	+	+
Висока швидкість роботи	-	+	+
Низьке споживання ресурсів	-	+	+
Професійна підтримка	+	-	+
Безкоштовне комерційне використання	+	+	-
Зручний GUI	+	-	+
Вбудовані засоби роботи з БД	+	-	+
Вбудовані засоби тестування	+	-	+
Не потребує Java Virtual Machine	+	+	-
Повна підтримка розгортання в Azure	+	-	-
Вбудовані інструменти для дебагінгу	+	-	+

5

Рисунок Б.5 – Слайд 5

ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Критерій	Angular	Vue.js
Простота освоєння	-	+
Вбудована повна архітектура	+	-
Корпоративне використання	+	-
Низький розмір початкового пакету	-	+
Гнучкість архітектури	-	+
Вбудована типізація через TypeScript	+	-

6

Рисунок Б.6 – Слайд 6

ПОРІВНЯННЯ IDE ДЛЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Критерій	Visual Studio Code	WebStorm
Безкоштовне комерційне використання	+	-
Вбудована підтримка тестування	-	+
Низьке споживання ресурсів	+	-

7

Рисунок Б.7 – Слайд 7

ПОРІВНЯННЯ СУБД

Критерій	MSSQL	MySQL
Вбудована підтримка LINQ через Entity Framework	+	-
Інтеграція з .NET	+	-
Автоматичне резервне копіювання	+	-
Підтримка повної кросплатформеності	-	+
Продуктивність при складних транзакціях	+	-
Безкоштовність комерційного використання	-	+

8

Рисунок Б.8 – Слайд 8

ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

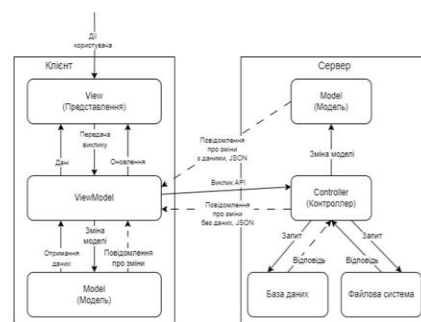


Рисунок 1 – Архітектура вебзастосунку

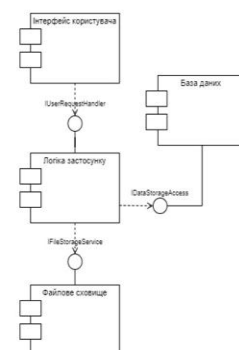


Рисунок 2 – Діаграма компонентів вебзастосунку

9

Рисунок Б.9 – Слайд 9

ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

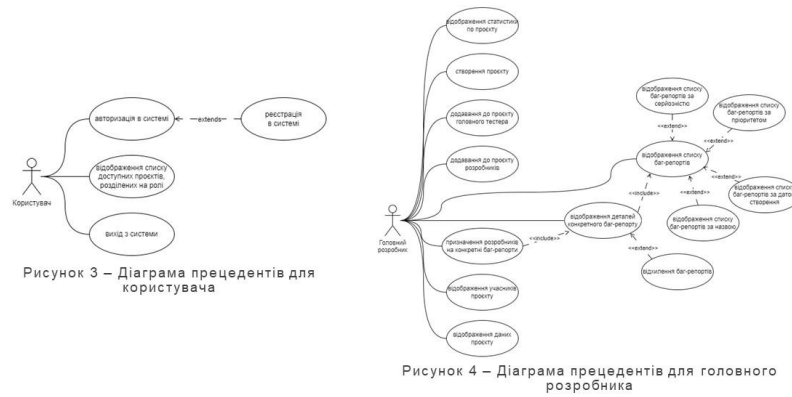


Рисунок Б.10 – Слайд 10

ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ



Рисунок Б.11 – Слайд 11

ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ



Рисунок Б.12 – Слайд 12



Рисунок Б.13 – Слайд 13



Рисунок Б.14 – Слайд 14

Система управління баг-репортами

Вибрати проект:

Вибрати проєкт: Створити проєкт: Очистити:

Вибрати тип:	Детальніше:
Репортаж:	Детальніше:
Репортаж:	Детальніше:
Репортаж:	Детальніше:
Блок:	Детальніше:

Рисунок 10 – Інтерфейс сторінки вибору проєкту

Рисунок Б.15 – Слайд 15

ВИСНОВКИ

У межах дипломної роботи було створено вебзастосунок для управління баг-репортами, який відповідає сучасним вимогам до організації процесів тестування та взаємодії в команді. Розробка базується на попередньому аналізі предметної області, що підтвердив потребу у спеціалізованому інструменті для зручної роботи з помилками програмного забезпечення. Вебзастосунок має чітку архітектуру, адаптовану під різні ролі користувачів, та інтерфейс, що забезпечує простоту навігації й повний набір функцій для керування баг-репортами.

Система демонструє стабільність, можливість масштабування та підтримку одночасної роботи кількох користувачів у межах кількох проєктів. Оцінювання зручності, функціональності та командної взаємодії засвідчило практичну цінність створеного рішення.

17

Рисунок Б.17 – Слайд 17