Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка ІТ-сервісу «Створення та управління тест-кейсами» інформаційної системи ІТ-компанії

(тема)

Виконав:
здобувач 4 року навчання,
групи ІТУ-21-3
Анастасія БІЛОУС
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні технології управління
(повна назва освітньої програми)

Керівник: доц. каф. ІУС Іван ЮР'ЄВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ІУС _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні технології управління
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

" 19 " травня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Білоус Анастасії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка IT-сервісу «Створення та управління тест-кейсами» інформаційної системи IT-компанії

затверджена наказом по університету від " 19 " травня 2025 р. № 3700Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 " червня 2025 р.

3. Вихідні дані до роботи розробити IT-сервіс «Створення та управління тест-кейсами» інформаційної системи IT-компанії. Сервіс повинен являти собою клієнт-серверний браузерний сервіс

4. Перелік питань, що потрібно опрацювати у роботі Змістовний опис та аналіз структурних і функціональних особливостей предметної області IT-компанії. Огляд і аналіз існуючих методів і засобів створення та управління тест кейсами. Формування вимог до IT-сервісу . Опис та графічне представлення архітектури IT-сервісу. Опис та графічне представлення бази даних IT-сервісу. Опис та графічне представлення елементів математичної забезпечуючої системи IT-сервісу. Опис та графічне представлення елементів програмної забезпечуючої системи IT-сервісу. Опис та графічне представлення елементів технічної забезпечуючої системи IT-сервісу. Опис та графічне представлення елементів рішень з User Experience (UX) та User Interface (UI)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд і аналіз сучасного стану розглянутої проблеми та існуючих методів і засобів вирішення задач кваліфікаційної роботи	19.05.2025 - 21.05.2025	Виконано
2	Огляд існуючих методів і засобів створення та управління тест кейсами	22.05.2025 - 24.05.2025	Виконано
3	Формування вимог до IT-сервісу «Створення та управління тест-кейсами»	25.05.2025 - 27.05.2025	Виконано
4	Опис та графічне представлення архітектури IT-сервісу	28.05.2025 - 30.05.2025	Виконано
5	Опис та графічне представлення бази даних IT-сервісу	01.06.2025 - 03.06.2025	Виконано
6	Опис та графічне представлення елементів математичної забезпечуючої системи IT-сервісу	04.06.2025 - 06.06.2025	Виконано
7	Опис та графічне представлення елементів програмної забезпечуючої системи IT-сервісу	07.06.2025 - 09.06.2025	Виконано
8	Опис та графічне представлення елементів технічної забезпечуючої системи IT-сервісу	10.06.2025 - 11.06.2025	Виконано
9	Опис та графічне представлення елементів рішень з User Experience (UX) та User Interface (UI) IT-сервісу	12.06.2025 - 13.06.2025	Виконано
10	Оформлення пояснювальної записки	14.06.2025 - 15.06.2025	Виконано
11	Захист кваліфікаційної роботи	16.06.2025	Виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач  (підпис)

Керівник роботи  (підпис) доц. каф. ІУС Іван ЮР’ЄВ (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 139 с., 45 рис., 3 табл., 1 дод., 19 джерел.

БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕСТ-КЕЙС, ТЕСТУВАННЯ, УПРАВЛІННЯ, ФУНКЦІОНАЛЬНА МОДЕЛЬ, PYTHON.

Об’єктом дослідження є процес обліку та організації тест-кейсів у межах життєвого циклу розробки програмного забезпечення в IT-компанії, створення тестових сценаріїв, формування тест-сьютів та підготовки звітів про результати тестування.

Метою дослідження є розробка сервісу «Створення та управління тест кейсами» для інформаційної системи IT-компанії Danaviero.

Предметом дослідження є інформаційні технології та програмні методи створення веборієнтованого сервісу з клієнт-серверною архітектурою, що дозволяє автоматизувати процеси ведення тест-кейсів, їх актуалізації, групування та контролю.

У ході роботи проведено аналіз предметної області, особливостей організації процесу тестування в IT-компанії, визначено об’єкт розробки – сервіс створення, обліку та супроводу тест-кейсів. Під час формування вимог до сервісу було проведено функціональне моделювання за стандартом IDEF0, розроблено структуру бази даних для зберігання тестової документації, спроєктовано архітектуру клієнтської та серверної частини сервісу.

Галузь застосування – підтримка процесів створення, управління та контролю якості тестової документації в межах розробки програмного забезпечення IT-компанії.

ABSTRACT

Bachelor`s thesis: 139 pages, 45 figures, 3 tables, 19 appendices, 19 sources.

DATABASE, FUNCTIONAL MODEL, INFORMATION SYSTEM, MANAGEMENT, PYTHON, SOFTWARE, TEST CASE, TESTING.

The object of research is the process of accounting and organizing test cases within the software development life cycle in an IT company, creating test scenarios, forming test suites and preparing reports on test results.

The aim of the study is to develop a service “Creating and managing test cases” for the information system of the IT company Danaviero.The goal of the research i is to develop a service «Creation and management of test cases» for the information system of the IT company Danaviero.

The research subject is information technologies and software methods for creating a web-oriented service with a client-server architecture, which allows you to automate the processes of maintaining test cases, their updating, grouping and control.

In the course of the work, we analyzed the subject area, the peculiarities of organizing the testing process in an IT company, and defined the object of development - a service for creating, accounting and maintaining test cases. During the formation of requirements for the service, functional modeling was carried out according to the IDEF0 standard, a database structure for storing test documentation was developed, and the architecture of the client and server parts of the service was designed.

Application area - support for the processes of creating, managing and quality control of test documentation within the framework of software development in an IT company.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ДБЖ – джерело безперебійного живлення

ІС – Інформаційна система

ПЗ – програмне забезпечення

СУБД – система управління базами даних

API – application programming interface

DFD – data flow diagram

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

IDEF0 – integration definition for function modeling

MVC – Model-View-Controller

ORM – Object-Relational Mapping

SQL – Structured Query Language

UML – Unified Modeling Language

ВСТУП

Інтеграція спеціалізованих сервісів у робочі процеси IT-компаній сприяє підвищенню ефективності роботи персоналу та забезпеченню якості програмних продуктів. У сфері розробки програмного забезпечення особливе місце посідає організація процесу тестування, яка потребує впорядкованого зберігання, актуалізації та контролю тест-кейсів. Відсутність єдиного інструменту для централізованого управління тестовою документацією призводить до ускладнення процесів тестування, дублювання сценаріїв, втрати важливої інформації та зниження якості кінцевого продукту.

Використання спеціалізованого сервісу для управління тест-кейсами дозволяє мінімізувати ймовірність помилок, прискорити процес підготовки до релізів, оптимізувати взаємодію між різними частинами команди, а також підвищити рівень контролю за якістю продуктів. Централізований облік тест-кейсів спрощує ведення документації забезпечуючи зручний доступ до тестових сценаріїв, та дозволяє формувати різні звіти про результати тестування та активність користувача.

Проблематика полягає у використанні фрагментованих засобів обліку тест-кейсів, що ускладнює процеси тестування, затягує підготовку до релізів і підвищує ризик помилок. Впровадження окремого сервісу дозволить систематизувати роботу тестувальників і полегшити формування звітів про проходження тестування.

Метою кваліфікаційної роботи є розробка спеціалізованого IT-сервісу для створення, обліку, редагування, групування та супроводу тест-кейсів , що дозволить покращити якість тестування, скоротити час на підготовку релізів і підвищити загальну стабільність програмних продуктів компанії.

1 ЗМІСТОВИЙ ОПИС ТА АНАЛІЗ СТРУКТУРНИХ І ФУНКЦІОНАЛЬНИХ ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ІТ-КОМПАНІЇ

1.1 Аналіз ІТ-компанії

Об'єктом розробки є ІТ-сервіс для створення, організації та управління тест-кейсами в межах інформаційної системи ІТ-компанії Danaviero. Danaviero займається повним життєвим циклом створення програмного забезпечення(ПЗ) для бізнесу, освіти, охорони здоров’я, логістики та інших галузей. Основними принципами роботи компанії є гнучкість, якість, масштабованість і орієнтованість на результат.

Процеси, що реалізуються в компанії Danaviero, охоплюють усі етапи життєвого циклу та собливу увагу приділено якості, і саме тому важливою складовою є тестування. У компанії застосовуються як ручне, так і автоматизоване тестування. Проте за відсутності централізованого інструменту управління тест-кейсами виникають труднощі, пов’язані з дублюванням сценаріїв, недостатнім покриттям тестами нових функцій, відсутністю єдиного простору для зберігання, а також складнощами в формуванні звітів та статистики.

Розробка ПЗ виконується в межах кількох взаємопов’язаних підрозділів: відділ розробки відповідає за реалізацію функціоналу як з боку клієнта, так і з боку сервера, відділ тестування забезпечує контроль якості програмного продукту, а DevOps-команда відповідає за інфраструктуру, автоматизацію та підтримку безперервного циклу доставки. Окреме місце посідає управління продуктами, в межах якого працюють бізнес-аналітики, проєктні менеджери та дизайнери. Підтримувальні функції, такі як рекрутинг, HR та фінансове адміністрування, реалізуються у відповідних сервісних відділах. На рисунку 1.1 представлена схема організаційної стуктури ІТ-компанії.

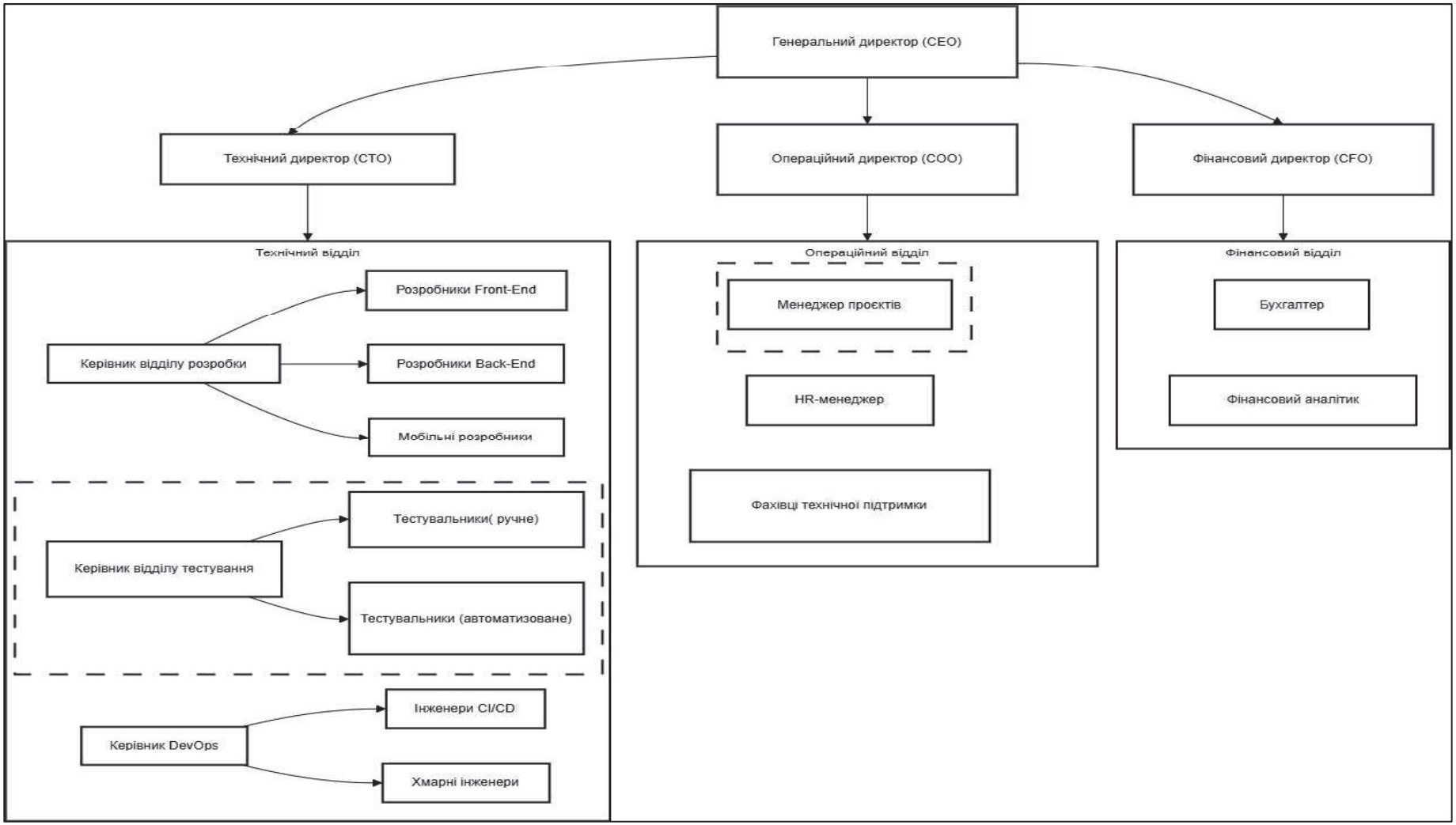

Рисунок 1.1 – Схема організаційної структури IT-компанії Danaviero

Управління тест-кейсами є важливим компонентом забезпечення якості ПЗ. Правильно організоване ведення тест-кейсів дозволяє підтримувати актуальність даних, зменшувати витрати часу на оновлення та підвищувати ефективність команди тестувальників. Завдяки системі управління тест-кейсами можна структурувати й категоризувати тести, що спрощує їх пошук і повторне використання, забезпечуючи послідовність і цілісність тестування. Це особливо важливо в умовах швидкої зміни вимог та динамічного розвитку програмних продуктів. Централізоване зберігання тест-кейсів забезпечує доступність всієї необхідної інформації для кожного члена команди та підтримку історії змін. Використання такої системи дозволяє забезпечити прозорість процесу тестування, що сприяє виявленню проблем на ранніх етапах розробки.

Система потрібна користувачам, котрі займаються тестуванням програмних продуктів, тобто тестувальникам, автоматизаторам, розробникам ПЗ, менеджерам проєкту та іншим. За допомогою цієї системи користувач може оцінити стан покриття тестами програмного продукту, оцінити стан проходження регресійного тестування та використовувати тест-сьюти як проєктну документацію.

1.2 Характеристика робочого місця користувача

В компанії Danaviero тестувальник виконує ключову роль у забезпеченні якості ПЗ, що розробляється. Це фахівець, який перевіряє відповідність реалізованого функціоналу початковим вимогам, виявляє дефекти, аналізує ризики та попереджає появу критичних помилок у продуктах, які впливають на користувачів. Його діяльність тісно пов’язана з усіма етапами життєвого циклу розробки ПЗ, а також із взаємодією з іншими учасниками команди: бізнес-аналітиками, від яких він отримує

специфікації та user stories, розробниками, яким передає баг-репорти та результати перевірок, менеджерами, яким надає інформацію про поточний статус стабільності продукту, а також інженерами DevOps. Тестувальник створює та підтримує тест-кейси у системі, оформлює баг-репорти у разі виявлення невідповідностей, проводить тестування нових функцій вручну або за допомогою автоматизованих сценаріїв, відстежує проходження тестів, перевіряє зв'язок кейсів із вимогами та формує звіти для менеджменту. У межах сервісу управління тест-кейсами тестувальник виступає не лише як користувач, що заповнює тестову документацію, але і як контролер якості та аналітик, який впливає на загальний підхід до перевірки продуктів. Його участь забезпечує прозорість тестового покриття, зменшує ризики помилок, що можуть вийти у продакшн, та формує об'єктивну картину технічної якості ПЗ.

Наразі у Danaviero тест-кейси зберігаються у вигляді неструктурованих документів та локальних файлів, що знижує ефективність тестування, ускладнює звітність і створює труднощі в масштабуванні процесів. Такий підхід створює низку труднощів, зокрема проблеми з актуальністю тестів, дублюванням сценаріїв, ускладненням пошуку потрібної інформації та складністю ведення звітності щодо проходження тестів. Крім того, відсутність єдиного централізованого середовища не дозволяє ефективно аналізувати покриття функціоналу, а також формувати історію змін тест-кейсів та контролювати якість тестування на рівні компанії. У таких умовах тестувальник витрачає значну частину часу на організацію власної роботи: пошук старих кейсів, створення копій, ведення звітів вручну. Відсутність автоматизації ускладнює співпрацю між тестувальниками та розробниками, а також уповільнює процес аналізу дефектів, що були виявлені під час тестування. У межах поточного підходу тестувальники фактично змушені дублювати одні й ті самі дії в різних документах, витрачаючи робочий час на задачі, які могли б бути автоматизовані.

Інформаційна система (ІС) компанії Danaviero має модульну архітектуру, що дозволяє легко інтегрувати нові компоненти без необхідності зміни основного ядра системи. Розробка сервісу для управління тест-кейсами не вимагає порушення існуючої IT-інфраструктури, а навпаки забезпечує додаткову гнучкість і прозорість у процесах тестування.

В результаті функціонального моделювання було створено графічне представлення процесів IT-сервісу Створення та управління тест-кейсами» методологією Integration Definition for Function Modeling (IDEF0). На рисунку 1.2 і 1.3 представлено контекстну діаграму та її декомпозицію першого рівня.

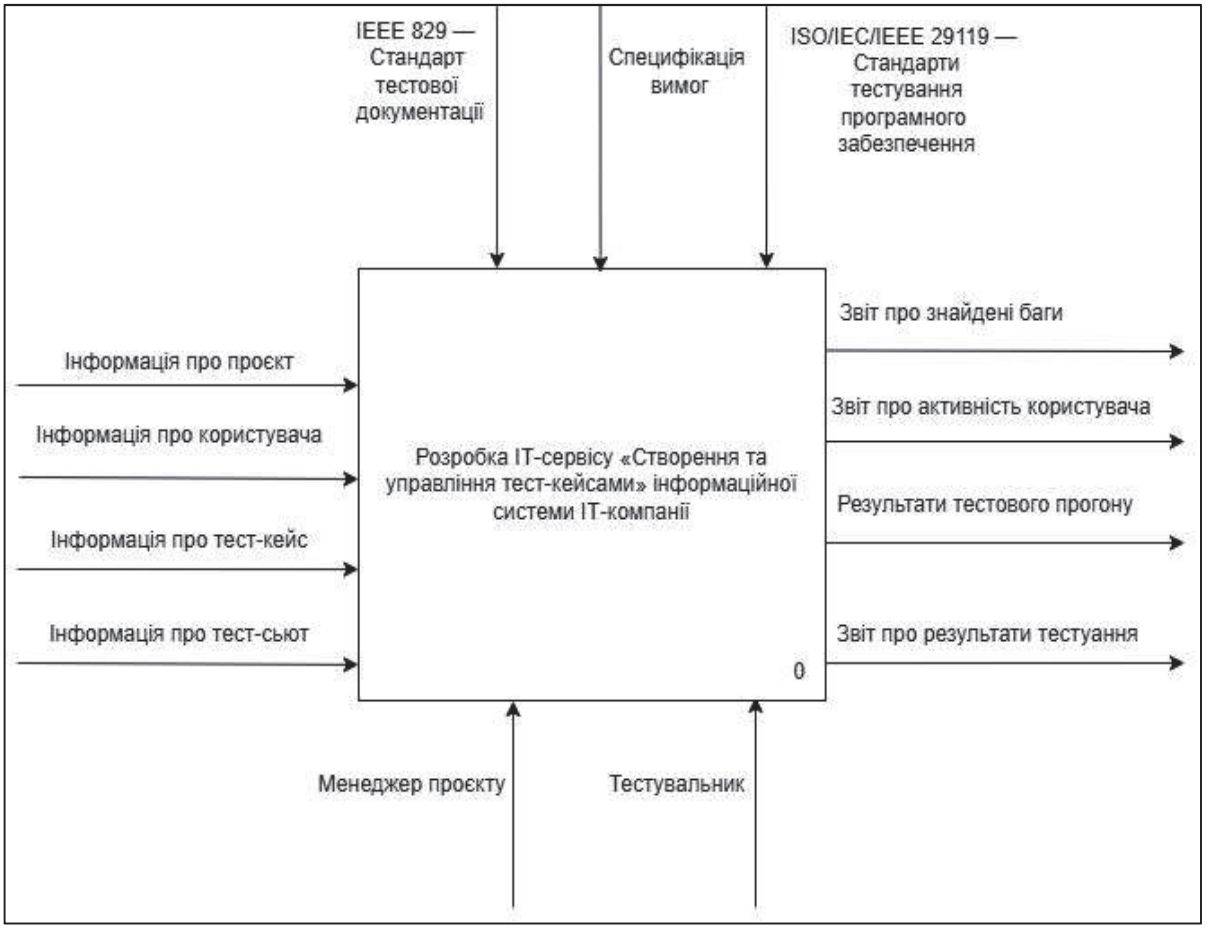

Рисунок 1.2 – Контекстна діаграма IDEF0 IT-сервісу «Створення та управління тест-кейсами

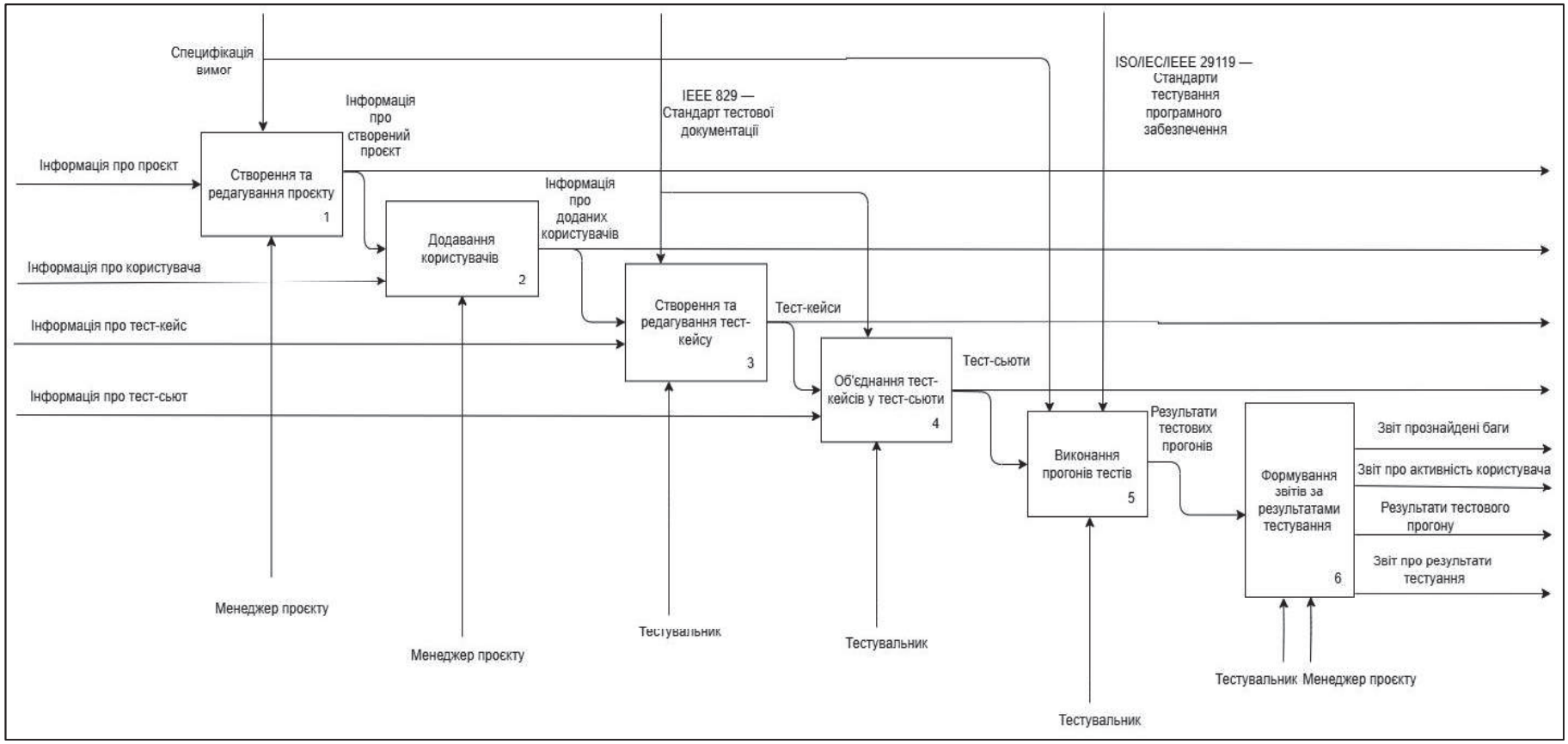

Рисунок 1.2 – Декомпозиція першого рівня IT-сервісу «Створення та управління тест кейсами»

Вхідними даними для розробки IT-сервісу «Створення та управління тест кейсами» інформаційної системи IT-компанії є інформація про проєкт , користувача , тест-кейси та тест-сьюти, які забезпечують вихідні вимоги для формування якісної бази тестової документації. Додатково на процес впливають методологічні засади, такі як Agile, стандарти якості тестування[1], а також політики тестування безпеки[2]. Ці джерела формують рамкові вимоги до процесу створення, валідації та підтримки тест-кейсів і тест-сьютів.

Механізмами роботи сервісу є тестувальники та менеджери , які безпосередньо взаємодіють із системою, спільно вони забезпечують ефективну роботу над створенням і редагуванням тест-кейсів відповідно до заданих технічних характеристик, їх об'єднанням у логічні структури тест-сьютів, організацією тестових прогонів та фіксацією отриманих результатів. Варто відзначити роль менеджера проєкту, який ініціює створення нового проєкту, призначає користувачів до нього та визначає початкову структуру перевірок. Реалізовані процеси передбачають послідовне накопичення інформації та можливість її подальшого аналізу. Усі дії користувачів автоматично реєструються в журналі активності, що забезпечує повну прозорість процесів у системі.

Результатами роботи сервісу є створені тест-кейси, ієрархічно організовані тест-сьюти, результати виконання тестових прогонів, а також сформовані звіти які поділяються за типами: звіт про активність користувача, звіт про знайдені баги, звіт про тестовий прогін і загальний звіт про результати тестування. Ці вихідні дані дозволяють не лише оцінити якість продукту на певному етапі розробки, а й забезпечити повноцінну підтримку процесів аналізу, контролю якості та планування подальших етапів життєвого циклу ПЗ.

В основі декомпозиції першого рівня лежить послідовна обробка даних через чотири ключові функції: створення та редагування тест-кейсів із

подальшою підготовкою тестової бази, об'єднання тест-кейсів у тест-сьюти для зручної організації і запуску тестів, виконання тестових прогонів для перевірки відповідності продукту вимогам та формування аналітичних звітів за результатами тестування, що служать основою для ухвалення подальших рішень щодо розвитку продукту. Перед цими функціями передують етапи створення самого проєкту та додавання до нього користувачів, що забезпечує структуровану основу для подальшої роботи.

Розробка сервісу «Створення та управління тест кейсами» сприяє суттєвому підвищенню продуктивності команди тестувальників. Завдяки впровадженню спеціалізованого сервісу, який дозволяє створювати, редагувати та групувати тест кейси, значно спрощується процес підтримки актуальної тестової документації. Це дозволяє не лише зменшити ризик людських помилок, але й оптимізувати взаємодію між розробниками, аналітиками та менеджерами проєктів.

2 ОГЛЯД І АНАЛІЗ СУЧАСНОГО СТАНУ СТВОРЕННЯ ТА УПРАВЛІННЯ ТЕСТ-КЕЙСАМИ, І ТАКОЖ ІСНУЮЧИХ МЕТОДІВ І ЗАСОБІВ ІЇ ВИРІШЕННЯ

2.1 Сучасний стан створення та управління тест-кейсами

В наш час забезпечення якості ПЗ є важливою складовою успішного розгортання та підтримки будь-якого ІТ-продукту. Управління тест-кейсами це невід'ємна частина цього процесу, оскільки забезпечує структурування, організацію та збереження тестових даних[3].

Сучасні системи управління тест-кейсами надають великий обсяг можливостей для тестувальників. Попри різноманіття доступних рішень, існуючі системи управління тест-кейсами мають певні недоліки, які можуть впливати на ефективність тестування та організацію процесів. Значною перешкодою для їх широкого впровадження є висока вартість ліцензій, що обмежує доступність таких систем для невеликих компаній і стартапів. Інтеграція з іншими інструментами, зокрема з системами управління проєктами на кшталт Jira, нерідко вимагає значних ресурсів та складного налаштування.

Крім цього, наявні продукти часто демонструють недостатню гнучкість у налаштуваннях, що ускладнює адаптацію систем до індивідуальних особливостей конкретного проєкту. Використання застарілих інтерфейсів у ряді рішень знижує зручність та інтуїтивність роботи для користувачів. Додатковим обмеженням виступає зниження продуктивності при обробці великих обсягів даних, що негативно впливає на швидкість і якість роботи в масштабних проєктах.

2.2 Огляд існуючих методів і засобів створення та управління тест-кейсами

На ринку ПЗ існують різні рішення для управління тест-кейсами, проте їх кількість обмежена. Це пояснюється тим, що такі системи мають вузьку спеціалізацію, і часто великі компанії створюють власні інструменти для управління тест-кейсами. Водночас існують і загальнодоступні продукти, кожен з яких має свої переваги та недоліки.

Через необхідність підтримки під час експлуатації багато таких програм є платними. Безкоштовні рішення або програми з відкритим вихідним кодом часто не мають офіційної підтримки та потребують значних зусиль для адаптації під конкретний проєкт.

Основним компонентом системи є база даних (БД), де зберігаються тест-кейси. Вона може розміщуватися на серверах розробника або на власних серверах користувача у разі можливості самостійного розгортання.

У реальних умовах експлуатації кількість тестів, тестових наборів та користувачів швидко зростає, особливо в умовах великих проєктів або розширення команд тестування. Без належної масштабованості система починає втрачати продуктивність: збільшується час відкриття тест-кейсів, сповільнюється генерація звітів та обробка запитів, що безпосередньо впливає на ефективність тестування [4].

Гнучкість налаштувань важлива для адаптації системи під специфіку конкретного проєкту або компанії. Без можливості налаштовувати структуру тест-кейсів, створювати власні типи зв'язків або модифікувати робочі процеси користувачі змушені пристосовуватися до стандартної моделі системи, що часто ускладнює роботу і знижує ефективність.

Таким чином, ми визначили критерії порівняння існуючих систем

менеджменту тест-кейсів: вартість, розміщення бекенду, масштабованість та гнучкість налаштувань.

Виділимо 3 найпопулярніших систем менеджменту тест-кейсів:TestRail, Zephyr, Xray.

TestRail є одним із найпопулярніших комерційних рішень для управління тест-кейсами[5]. Він пропонує централізоване управління процесами тестування, створення тест-кейсів, тест-планів та запусків тестів. В системі передбачено два варіанти розміщення: хмарний сервіс (TestRail Cloud) та локальну інсталяцію (TestRail Server), що дає базовий вибір для компаній різних масштабів. Масштабування TestRail здійснюється шляхом розширення обсягів проєктів та тестових наборів, однак при значному збільшенні даних або кількості одночасних запитів на великі обсяги інформації зростає навантаження на систему, що потребує додаткової оптимізації application programming interface (API) або інфраструктури сервера. TestRail має серйозні проблеми з гнучкістю, які відчутно впливають на повсякденну роботу. Велика проблема це обробка дефектів бо TestRail її не підтримує напряму. Якщо потрібно створити баг, доводиться інтегрувати зовнішній трекер на кшталт Jira, і навіть у цьому випадку зв’язок між тестом і дефектом не завжди зберігається належним чином. У налаштуваннях тест-кейсів можна змінювати лише шаблони полів, їхні назви або обов’язковість заповнення. Але не можна налаштувати, наприклад, логіку залежностей між полями, зміну полів залежно від обраного типу кейсу чи умовне відображення полів - усе фіксовано. Так само обмеженою є звітність: TestRail пропонує кілька стандартних шаблонів, але якщо потрібен звіт за нестандартними критеріями , наприклад, поєднати фільтри дати, статусу, секції й автора то це зробити через інтерфейс неможливо. Автоматизація в TestRail також не реалізована повноцінно: у системі немає жодних внутрішніх тригерів або сценаріїв, які могли б запускатися за подіями. Наприклад, не можна налаштувати, щоб

після завершення прогона автоматично надсилався звіт або змінювався статус пов'язаних кейсів. Єдиний спосіб – це написання скриптів через API, що потребує додаткових ресурсів та інженерної підтримки. Усе це робить TestRail дуже негнучким інструментом: він підходить лише для команд, які готові повністю адаптувати свої процеси під можливості системи, а не навпаки.

TestRail є одним із найдорожчих рішень на ринку. Ліцензія оплачується щороку за кількість користувачів, а також додатково — за розширене зберігання даних та інтеграції. Це робить систему менш привабливою для невеликих команд або стартапів, які не можуть дозволити собі великі постійні витрати. Приклад вигляду TestRail зображено на рисунку 2.1.

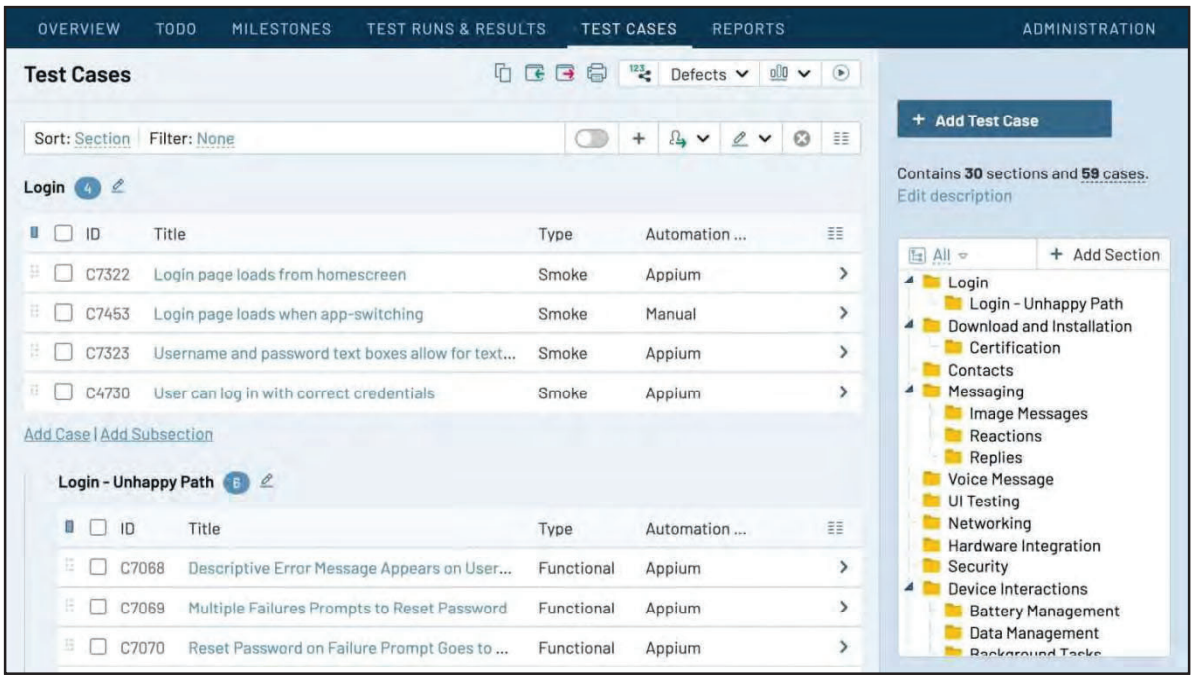

Рисунок 2.1 – Сторінка Test cases в TestRail

Zephyr є популярним комерційним інструментом для управління тест-кейсами, доступним у декількох варіантах: Zephyr Squad, Zephyr Scale та

Zephyr Enterprise [6]. Основною перевагою є глибока інтеграція з Atlassian Jira, що дозволяє безпосередньо працювати з тестами у середовищі задач проєкту. Масштабування Zephyr безпосередньо залежить від масштабування Jira: при зростанні кількості тестів та користувачів на великих проєктах може знадобитися оптимізація серверних ресурсів Jira або перехід на Data Center версію. При цьому час відгуку при роботі з великими тестовими базами поступово знижується. У питанні гнучкості налаштувань Zephyr також має певні обмеження. Хоча можливе додавання власних полів та атрибутів, більш глибока кастомізація життєвих циклів тест-кейсів обмежена наявними механізмами Jira Workflows. Побудова специфічних сценаріїв вимагає додаткової конфігурації або зовнішніх плагінів. Zephyr теж має регулярну оплату ліцензій залежно від кількості користувачів Jira. Оскільки розширення Zephyr базується на Jira, необхідно оплачувати як саму Jira, так і плагін Zephyr окремо. На рисунку 2.2 зображено сторінку з тест-кейсами в Zephyr.

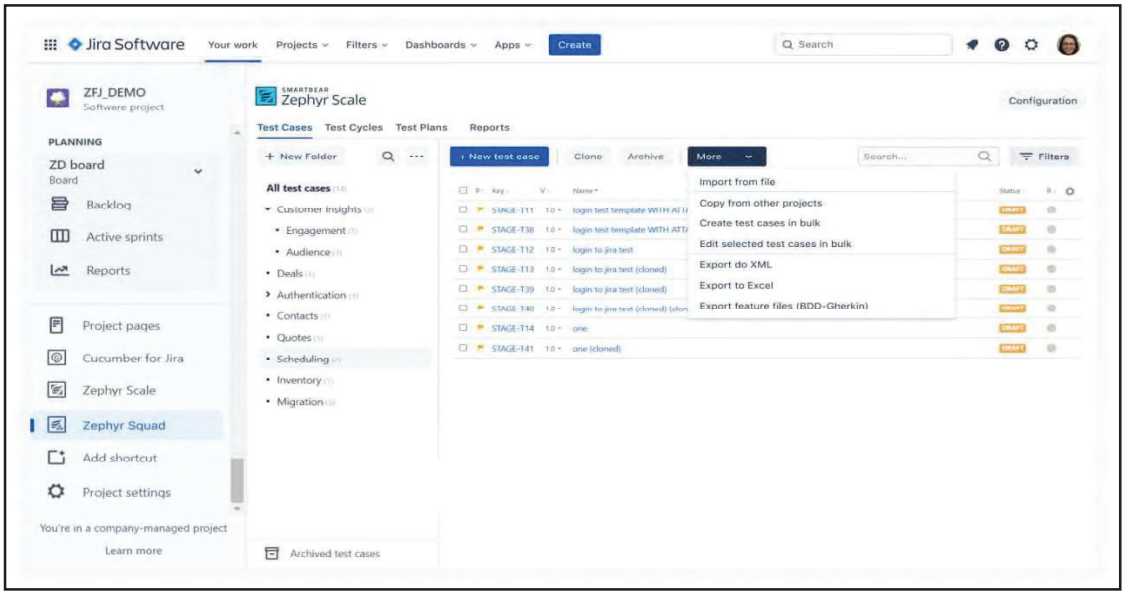

Рисунок 2.2 – Сторінка Test cases в Zephyr

Xray є комерційним розширенням для Jira, яке забезпечує управління тест-кейсами безпосередньо в інтерфейсі задач [7]. Архітектура Xray тісно інтегрована з Jira і використовує її об'єкти для створення тестів, тестових планів та прогонів. Масштабування Xray також напряму залежить від можливостей Jira: при великій кількості тестів виникають додаткові навантаження на сервер Jira, що може вимагати апгрейду інфраструктури або переходу на більш дорогі тарифні плани. Xray дозволяє змінювати деякі атрибути тестових об’єктів, але всі ключові процеси і життєві цикли тестування базуються на налаштуваннях Jira. Це створює складнощі у випадку потреби реалізації нетипових процесів тестування. Xray, подібно до Zephyr, вимагає оплати ліцензії окремо від Jira. Для великих компаній або організацій з активним тестуванням витрати на ліцензії Xray плюс необхідні ресурси для розширення Jira Data Center можуть стати суттєвими.

На рисунку 2.3 зображено сторінки Xray.

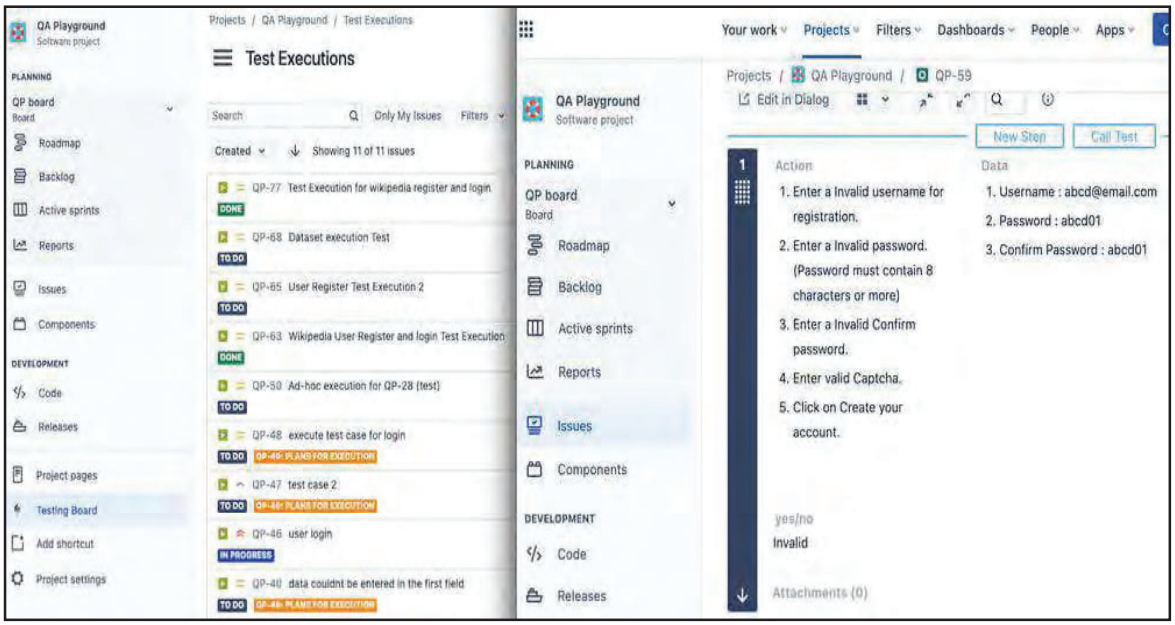

Рисунок 2.3 – Дошка тестування та типів проблем в Xray

На основі отриманих даних в ході аналізу систем : TestRail, Zephyr, Xray було створено порівняльну таблицю 2.1, яка дозволяє порівняти і детально проаналізувати можливості досліджувальних систем.

Таблиця 2.1 – Порівняльний аналіз систем управління тест-кейсів

Система	Вартість	Розміщення бекенду	Масштабованість	Гнучкість налаштувань
TestRail	Платна Cloud: $38/міс/кор. Server: $1,412/міс	Хмарна або локальна	Обмежена	Обмежена
Zephyr	Платна Squad: $10/міс/до 10 кор. Scale: $5.21/кор.	Хмарна або локальна (плагін Jira)	Обмежена	Обмежена
Xray	Платна Cloud: $10/міс/до 10 кор. Data Center: $1,740/рік/50 кор.	Хмарна або локальна через Jira	Обмежена	Обмежена

Систем управління тест-кейсами TestRail, Zephyr та Xray, є комерційними продуктами з високою вартістю ліцензій, що може створювати значні фінансові навантаження для компаній, які прагнуть оптимізувати витрати без втрати якості інструментів. Хоча ці системи пропонують гнучкі варіанти розміщення бекенду як хмарного, так і локального, їх масштабованість супроводжується зниженням продуктивності при великій кількості користувачів або даних, що критично для динамічних середовищ.

Гнучкість налаштувань у розглянутих системах є обмеженою: значна частина функціоналу жорстко інтегрована в загальну екосистему, що ускладнює індивідуальну адаптацію під специфічні процеси кожної окремої компанії. Особливо це стосується продуктів, тісно прив'язаних до Jira, таких як Zephyr та Xray, де залежність від сторонньої платформи обмежує незалежність системи та ускладнює масштабування під нестандартні потреби.

Таким чином, незважаючи на існуючий вибір, жодна з проаналізованих систем повністю не задовільняє вимоги нашої IT-компанії. Враховуючи виявлені недоліки такі як високу вартість, обмежену гнучкість кастомізації ,проблеми з масштабованістю прийнято рішення розробити власний IT-сервіс «Створення та управління тест-кейсами», який буде відповідати внутрішнім потребам компанії, а саме забезпечить адаптивність, масштабованість, незалежність та ефективну підтримку роботи команди тестувальників.

3 ЗФОРМУВАННЯ ВИМОГ ДО ІТ- СЕРВІСУ «СТВОРЕННЯ ТА УПРАВЛІННЯ ТЕСТ КЕЙСАМИ»

3.1 Функціональні вимоги до ІТ-сервісу

Формування вимог до ІТ-сервісу є не лише необхідним, а й одним із визначальних етапів у процесі його створення, незалежно від специфіки чи масштабу розробки. На цьому етапі особлива увага приділяється виявленню основних функціональних можливостей і характеристик сервісу, що надалі визначатимуть архітектуру, логіку роботи та напрями розвитку продукту.

У результаті ретельного аналізу та формулювання вимог формується чіткий перелік функціональних задач і очікуваних результатів, що має забезпечити сервіс. Також визначаються потенційні можливості, які необхідно надати користувачам для ефективного створення, організації та управління тест-кейсами. Правильно сформовані вимоги дозволяють досягти високої якості розробки, забезпечити зручність використання та підвищити продуктивність робочих процесів.

Функціональні вимоги до ІТ-сервісу «Створення та управління тест кейсами» визначають основні можливості, які мають бути реалізовані для підтримки повного циклу роботи із тестовою документацією в межах ІТ-компанії.

Система повинна забезпечувати реєстрацію нових користувачів та автентифікацію вже існуючих облікових записів [8]. Процес реєстрації має реалізовуватись через спеціальну форму введення, що включає обов'язкові поля для імені користувача, електронної пошти та пароля із перевіркою складності пароля відповідно до вимог безпеки. Автентифікація користувачів передбачає перевірку правильності введених облікових даних. Після успішного входу користувач отримує доступ до функціоналу системи, тоді як

для неавторизованих користувачів доступ має бути обмежений.

Функціонал сервісу повинен дозволяти створення, редагування та видалення тест-кейсів. Створення тест-кейсів має здійснюватися через зручну форму введення, яка передбачає заповнення таких обов'язкових полів, як назва тесту, короткий опис, кроки виконання, очікуваний результат та критичність. При редагуванні тест-кейсів користувач має мати змогу змінювати наявні дані через інтуїтивно зрозумілий інтерфейс із подальшим збереженням змін. Видалення тест-кейсу повинно супроводжуватися підтвердженням дії для уникнення випадкових втрат інформації.

У системі має бути реалізовано логіку для управління тест-сьютами. Очікується, що користувачі зможуть створювати тест-сьюти, надавати їм унікальні назви та описи, додавати або вилучати тест-кейси, а також редагувати наявну інформацію. Функціонал тест-сьютів має підтримувати гнучке структурування тестової бази з урахуванням специфіки проєктів.

Окремий модуль системи має забезпечувати запуск тестових прогонів на базі вже створеного тест-сьюта з доданими тест-кейсами. У ході прогону повинна зберігатися інформація про фактичний результат виконання кожного тесту. При виявленні відхилень від очікуваного результату має бути доступною опція створення звіту про помилку, яка відкриватиме модальне вікно для опису проблеми.

Передбачається, що окрема роль у системі буде відведена менеджеру проєкту, для якого необхідно реалізувати функціонал створення нових проєктів, внесення описової інформації, редагування наявних записів та їх видалення. Також має бути передбачений інтерфейс для керування користувачами, який включатиме функції створення облікових записів, генерації паролів, призначення проєктів і редагування доступів.

Формування звітності є обов’язковою частиною функціональних вимог. Система повинна надавати інструменти для генерації звітів про активність

користувачів, результати тестування, виконані прогони, а також знайдені баги. Кожен тип звіту має формуватись на основі заданих фільтрів і параметрів, а результати повинні виводитися у візуально структурованій формі з можливістю експорту в поширені формати.

Також має бути створена окремої аналітична сторінки, яка відображатиме статистику використання системи, що для менеджера що для тестувальника. Для користувачів така сторінка повинна демонструвати індивідуальні показники активності, загальну кількість стіворених тест-кейсів та тест-сьютів та опис поточного проєкту для якого користувач спворює тест-кейси і тест-сьюти. Менеджеру проєкту має бути наданий доступ до узагальненої аналітики, включаючи загальну кількість проєктів та користувачів в системі і також активність менеджера за останній місяць з детільним логуванням всіх дій .

Представимо функціональні вимоги до сервісу за допомогою Unified Modeling Language (UML) діаграми варіантів яка є зручним і корисним інструментом для формалізації вимог до майбутньої системи. Діаграма допомагає структуровано описати функціональні сценарії: наприклад, тестувальник повинен мати змогу створювати та запускати тест-кейси, фіксувати результати, формувати звіти. Менеджер, у свою чергу, має мати доступ до функцій створення проєктів, керування користувачами, призначення тестувальників до проєктів і перегляду аналітики.

Для точного опису взаємодії користувачів із системою було розроблено UML Use Case діаграми . Діаграми для ролі тестувальника зображена на рисунку 3.1, яка охоплює усі основні операції з тест-кейсами, тест-сьютами, прогоном та аналітикою, а також діаграми для менеджера проєкту зображену на рисунку 3.2, що ілюструватиме його можливості з керування користувачами, проєктами, генерації звітності та аналізу статистичних даних.

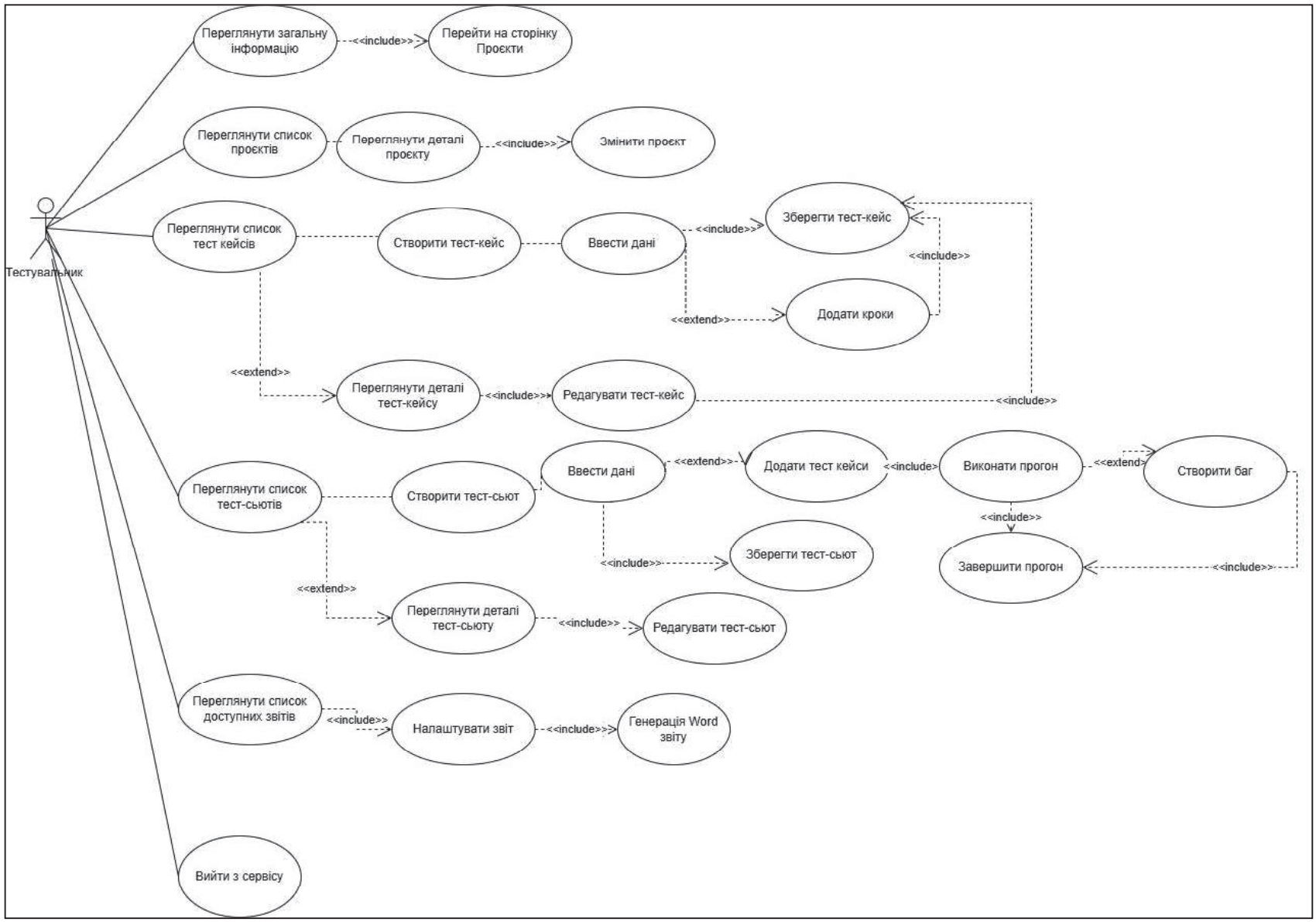

Рисунок 3.1 – UML-діаграма IT-сервісу «Створення та управління тест кейсами» для тестувальника

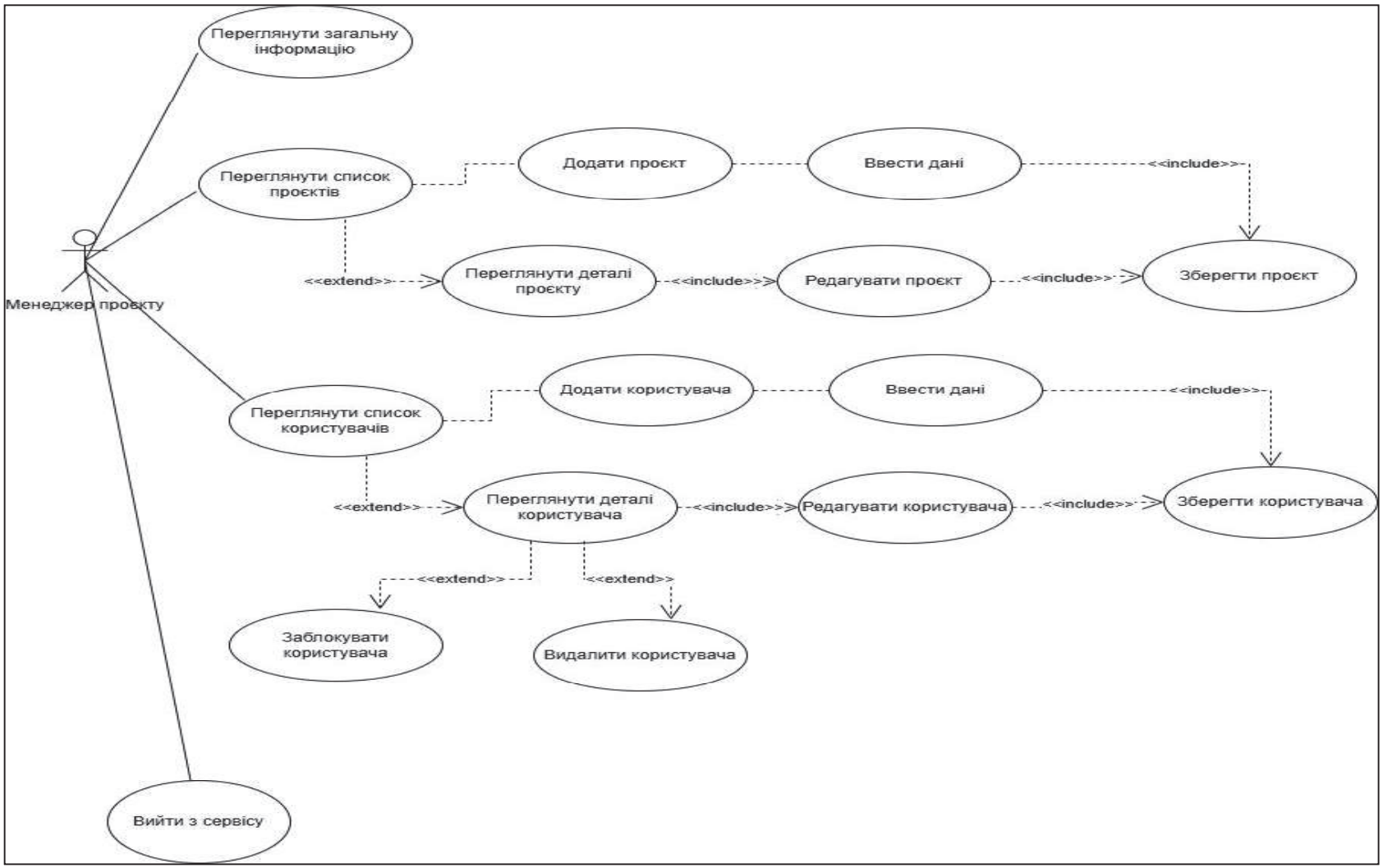

Рисунок 3.2 – UML-діаграма IT-сервісу «Створення та управління тест кейсами» для менеджера проєкт

3.2 Нефункціональні вимоги до IT-сервісу

У процесі створення IT-сервісу «Створення та управління тест кейсами» важливо враховувати не лише функціональні можливості системи, але й нефункціональні вимоги. Саме нефункціональні вимоги визначають загальну якість сервісу, його надійність, продуктивність, зручність використання, безпеку та можливість подальшого розвитку. Вони формують уявлення про те, наскільки ефективно сервіс буде інтегруватися в існуючі робочі процеси, забезпечувати комфортну взаємодію користувачів та підтримувати стабільну роботу в різних умовах експлуатації.

Надійність системи гарантується шляхом забезпечення точності збереження даних у БД. Для текстових даних, що містять інформацію тест-кейсів, необхідно використовувати відповідні типи даних, які зберігають текст повністю без втрат форматування. Окрему увагу потрібно приділити захисту даних користувачів, зокрема обов'язкове шифрування паролів та обмеження доступу до БД за допомогою складних паролів.

Вимоги безпеки також включають обробку всіх запитів через захищений протокол передачі даних для запобігання можливим атакам перехоплення даних. У разі спроби неавторизованого доступу до ресурсів системи, необхідно перенаправляти користувача на сторінку автентифікації або виводити відповідне повідомлення про помилку.

Важливою характеристикою системи є її підтримуваність та здатність до подальшого розвитку. Архітектура має бути побудована відповідно до шаблону Model-View-Controller (MVC), що дозволить ефективно розділити логіку роботи з даними, логіку управління і представлення даних користувачеві. Такий підхід сприятиме легкому оновленню функціональності, розширенню системи та внесенню змін у майбутньому без суттєвого впливу на вже реалізовані модулі.

Зручність використання системи передбачає реалізацію інтуїтивно зрозумілого інтерфейсу, що спрощує взаємодію для як досвідчених, так і недосвідчених користувачів. Система повинна забезпечувати візуальне виділення інтерактивних елементів, логічну структуру сторінок та швидкий доступ до основних функцій.

Система повинна функціонувати коректно у сучасних версіях настільних браузерів без необхідності встановлення додаткового ПЗ на стороні користувача. Клієнтська частина сервісу розробляється як веб-додаток, що забезпечує простоту доступу через браузер.

Також до нефункціональних вимог належать вимоги до атрибутів якості системи: супроводжуваність, тестопридатність, функціональність, коректність обробки даних, можливість розширення функціоналу та простота перенесення даних у разі інтеграції з іншими системами [9].

3.3 Обгрунтування мети і критеріїв ефективності ІТ-сервісу

Метою розробки ІТ-сервісу «Створення та управління тест кейсами» є реорганізація процесів документування тестування ПЗ шляхом автоматизації створення, організації, виконання та супроводу тест-кейсів. Запровадження даного сервісу має на меті усунути проблеми ручного ведення тестової документації, підвищити прозорість процесів тестування, скоротити час на підготовку тестових сценаріїв та полегшити командну взаємодію між тестувальниками, розробниками і менеджерами проєктів. Автоматизація дасть змогу швидко адаптувати тестову документацію до змін у вимогах до продукту, скоротити кількість помилок, пов'язаних із людським фактором, а також забезпечити своєчасне оновлення інформації про статуси тест-кейсів.

Інтеграція сервісу в робочі процеси компанії сприятиме підвищенню

ефективності тестування, пришвидшенню розробки, а також забезпечить більш якісне документування змін у продукті. Це дозволить тестувальникам більше часу приділяти аналізу складних сценаріїв і розробці нових тестових підходів замість рутинного ведення документації.

Критерії ефективності IT-сервісу визначаються такими показниками: підвищення продуктивності роботи тестувальників, скорочення часу на створення та оновлення тест-кейсів, зменшення кількості помилок у тестовій документації. Інтерфейс сервісу має бути інтуїтивно зрозумілим, логічно структурованим та забезпечувати швидке навчання користувачів. Універсальність використання передбачає доступ до сервісу з будь-якого пристрою, що підтримує сучасні веб-браузери: Google Chrome, Mozilla Firefox, Safari, Opera тощо. Також сервіс повинен підтримувати збереження даних у разі перебоїв мережі та забезпечувати доступ до раніше завантаженої інформації за допомогою механізмів кешування. Важливим елементом ефективності сервісу стане система аналітики, яка дозволить відстежувати прогрес виконання тестування і своєчасно реагувати на відхилення. Наявність автоматичних звітів про виконання тест-кейсів підвищить прозорість тестування і полегшить прийняття управлінських рішень.

4 ОПИС АРХІТЕКТУРИ ОБ'ЄКТА РОЗРОБКИ НА РІВНІ ФУНКЦІЙ

Розробка IT-сервісу вимагає не лише опису функціональних можливостей, а й формального представлення архітектури, що забезпечує реалізацію поставлених вимог. Одним із ключових етапів розробки є побудова функціональної моделі, яка дозволяє візуалізувати основні компоненти системи, їхню взаємодію та логіку обробки інформації. Для цього використаємо діаграми потоків даних (Data Flow Diagram, DFD), які надають можливість чітко, структуровано й послідовно подати логіку роботи IT-сервісу «Створення та управління тест-кейсами».

DFD-моделювання дає змогу представити систему через потоки даних, що передаються між процесами, зовнішніми сутностями та сховищами інформації[10]. Цей підхід дозволяє не лише продемонструвати, як відбувається обробка даних на різних етапах, але й визначити межі відповідальності кожного компонента системи, що вкрай важливо з точки зору підтримки, супроводу та масштабування ПЗ. DFD-діаграми були обрані через їхню гнучкість, простоту візуального сприйняття та широке використання в структурному аналізі.

З метою комплексного відображення архітектури IT-сервісу, було побудовано дві взаємопов'язані моделі: контекстну DFD-діаграму, яка демонструє взаємодію системи з зовнішнім користувачем на найвищому рівні узагальнення, та DFD-діаграму першого рівня, що деталізує внутрішню функціональну структуру модуля та показує послідовність обробки даних у межах підпроцесів. Обидві діаграми логічно пов'язані між собою, забезпечуючи повноцінне уявлення про принципи роботи сервісу. Для загального відображення логіки взаємодії користувача із системою управління тест-кейсами було побудовано контекстну діаграму потоків даних, що представлена на рисунку 4.1

Рисунок 4.1 – Контекстна DFD ІТ-сервісу «Створення та управління тест-кейсами»

У центрі діаграми розташовано головний функціональний блок — ІТ-сервіс «Створення та управління тест-кейсами», який взаємодіє із зовнішньою сутністю «Тестувальник» та «Менеджер проєкту» . На вхід до системи надходить низка даних, зокрема інформація про тест-кейс, тест-сьют та інформація про проєкт та користувача. У відповідь сервіс формує результати своєї роботи у вигляді звітів, зокрема про виявлені баги, результати тестового прогона, загальні підсумки тестування, а також активність користувачів у системі. Діаграма дозволяє візуально охопити повну картину інформаційного обміну між користувачем і системою, встановити функціональні межі сервісу та проаналізувати типи даних, що використовуються на рівні взаємодії між компонентами системи. Для деталізації внутрішньої структури ІТ-сервісу «Створення та управління тест-кейсами» виконано декомпозицію головного процесу, що дозволяє представити функціональну архітектуру на рівні підпроцесів.

Декомпозиція першого рівня зображена на рисунку 4.2.

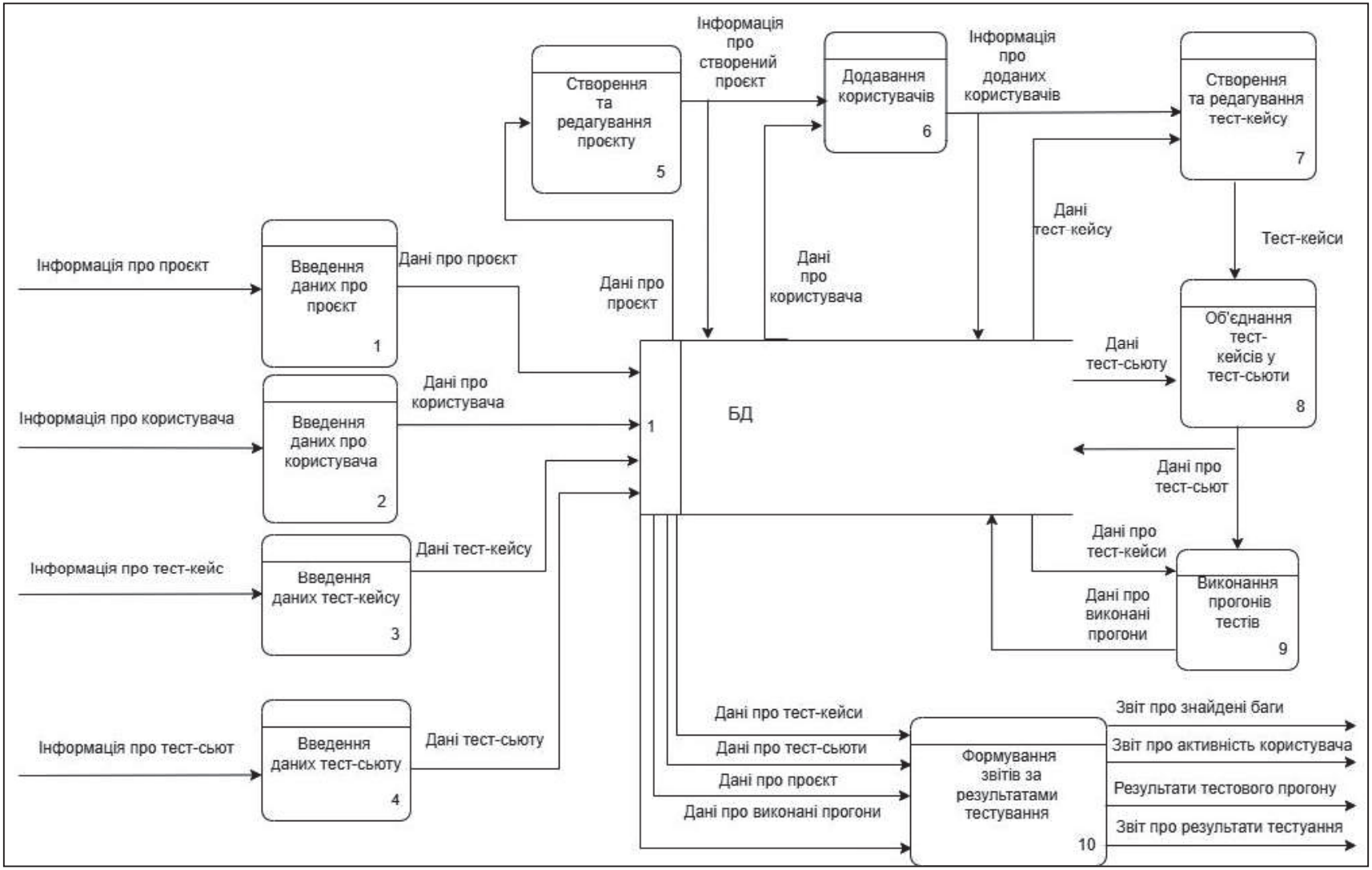

Рисунок 4.2 – DFD IT-сервісу «Створення та управління тест-кейсами», декомпозиція першого рівня

Декомпозиція першого рівня подає систему як сукупність взаємопов’язаних логічних операцій, кожна з яких виконує окрему функцію: від введення первинних даних і створення проєкту до формування тест-кейсів, запуску тестових прогонів та генерації звітної інформації. Такий підхід дає змогу чітко простежити шлях обробки даних у межах системи, починаючи з моменту, коли користувач взаємодіє з інтерфейсом введення, і завершуючи формуванням підсумкових результатів, які повертаються користувачу. Цей рівень деталізації дозволяє більш глибоко проаналізувати логіку реалізації основних функцій, що виконуються в межах сервісу.

На початкових етапах відбувається введення даних про проєкти, користувачів, тест-кейси та тест-сьюти. Ці дії супроводжуються паралельним створенням і редагуванням проєктів, а також додаванням користувачів, які долучаються до роботи в системі. Усі відповідні дані передаються до централізованої бази даних, де вони зберігаються у структурованому вигляді. Далі виконується створення та редагування тест-кейсів, які можуть бути оновлені або переглянуті на основі наданих шаблонів.

Після цього система переходить до етапу об’єднання тест-кейсів у тест-сьюти, підготовлені тест-сьюти передаються до підсистеми виконання тестових прогонів, де відбувається запуск автоматичних або ручних перевірок. Після завершення виконання результати фіксуються та передаються до блоку формування звітів. Тут система генерує різні типи звітів — про знайдені баги, активність користувачів, результати окремих прогонів та узагальнені результати тестування.

Уся інформація в межах діаграми переміщується між підпроцесами у вигляді чітко окреслених потоків даних, що забезпечує прозорість і послідовність обробки інформації. Завдяки такій побудові стає можливим відстеження повного життєвого циклу даних у системі — від моменту їх створення до формування кінцевого результату[11].

5 РОЗРОБКА Й ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ІНФОРМАЦІЙНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

Моделювання БД є одним з основних етапів розробки IT-сервісу, що дозволяє структуровано представити об'єкти предметної області, їх атрибути та взаємозв’язки. У межах розробки IT-сервісу «Створення та управління тест-кейсами» побудова моделі БД дозволяє забезпечити логічну цілісність, коректність обробки інформації, а також масштабованість архітектури системи.

Модель БД дозволяє не лише сформувати уявлення про логіку зберігання інформації, але й слугує основою для реалізації більшості функціональних можливостей сервісу таких як створення проєктів, формування тест-кейсів, об’єднання їх у сьюти, запуск тестових прогонів, збереження результатів, ведення журналу дій та обробка знайдених помилок. Наявність чітко визначених структур сутностей і зв’язків між ними дає змогу уникнути дублювання даних, забезпечити їх цілісність і актуальність, а також полегшує подальшу розробку інтерфейсів, API та звітності.

Процес моделювання починається зі створення логічної моделі, яка описує основні інформаційні об’єкти системи та їхню взаємодію на концептуальному рівні[12]. Такий підхід дозволяє відокремити логіку від технологічної реалізації, сконцентрувавшись на змістовному представленні даних і їхніх відносинах у межах предметної області. На етапі логічного моделювання формується уявлення про структуру даних без прив’язки до конкретної системи керування БД. Це дозволяє розробникам, бізнес-аналітикам та іншим учасникам процесу узгодити бачення того, як саме має виглядати інформаційна основа системи. У логічній моделі для IT-сервісу виділено такі сутності, як користувачі, проєкти, тест-кейси, тестові сьюти, прогони тестів, звіти, баги та дії

користувачів. Для кожної сутності визначено ключові атрибути, а також встановлено типи взаємозв'язків: один до одного, один до багатьох або багато до багатьох. Логічна модель даних представлена на рисунку 5.1.

Надалі ця модель трансформується у фізичну, де вже враховуються технічні вимоги, особливості обраної системи управління БД, продуктивність, цілісність і масштабованість. Фізичне моделювання уточнює логічну модель, трансформуючи її у специфікацію, що враховує всі технічні аспекти реалізації. На цьому етапі визначаються точні типи даних, обмеження цілісності, ключі, індекси, формати збереження вкладених даних, особливості роботи з масивами або документами, а також вимоги до продуктивності й безпеки. Це дозволяє досягти високої узгодженості даних, ефективного доступу до інформації, а також забезпечити збереження й контроль змін. Фізична модель даних для IT-сервісу «Створення та управління тест-кейсами» представлена на рисунку 5.2.

Логічна та фізична моделі взаємодоповнюють одна одну, утворюючи єдиний підхід до проєктування. Логічна модель гарантує коректність і зрозумілість структури даних з погляду бізнес-логіки, тоді як фізична модель транслює цю логіку у технічну реалізацію, оптимізовану під конкретні умови розгортання. Це дозволяє системі не лише ефективно виконувати свої функції у поточних умовах, а й бути гнучкою до майбутніх змін і розширень. Доцільно буде описати сутності і атрибути це покращить розуміння структури нашої бази даних. Узагальнена характеристика сутностей та особливості їх застосування наведені в таблиці 5.1. Деталізуємо інформацію табличкою 5.2 з даними про всі атрибути .

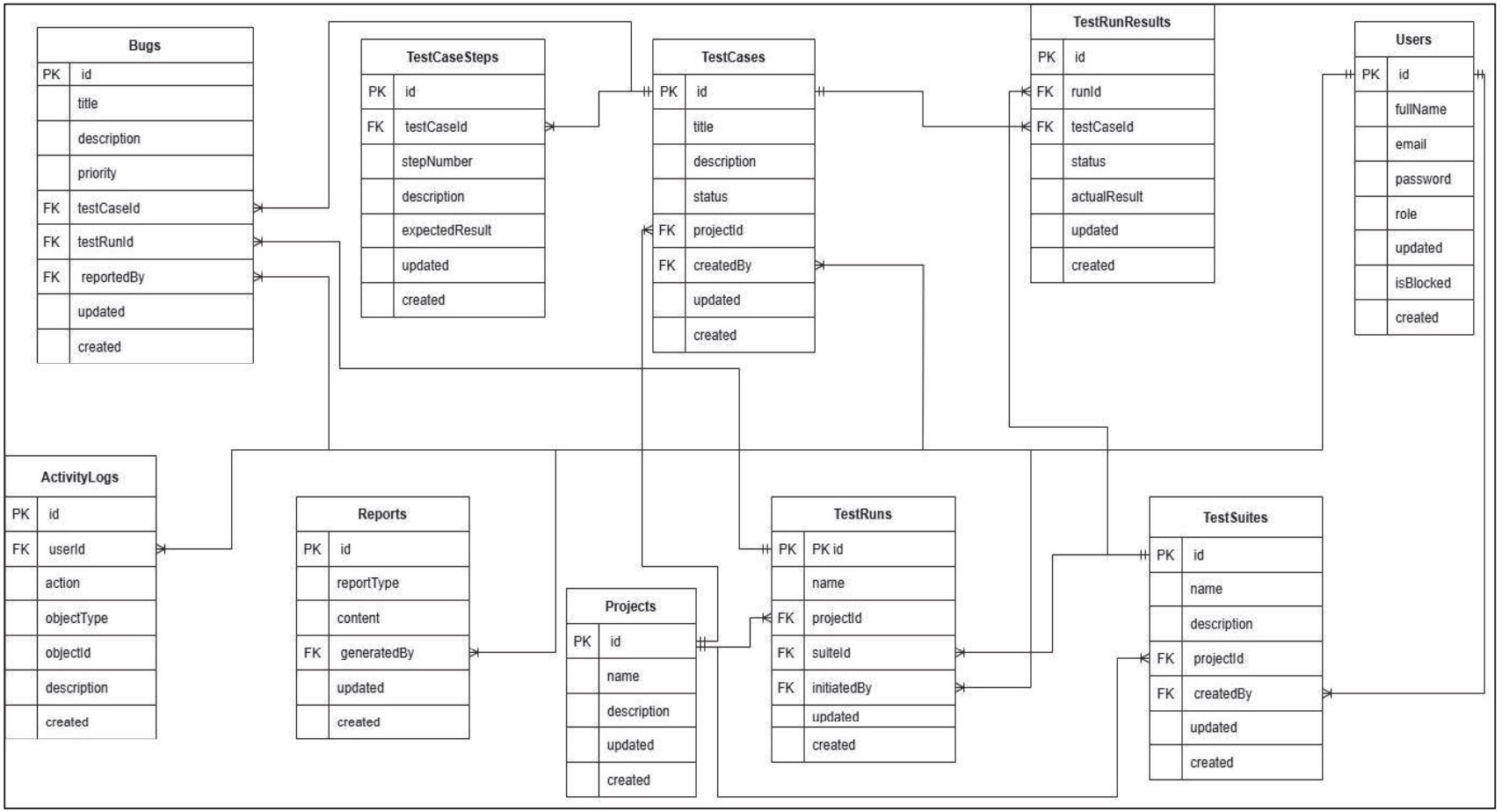

Рисунок 5.1 – Логічна модель даних IT-сервісу «Створення та управління тест-кейсами»

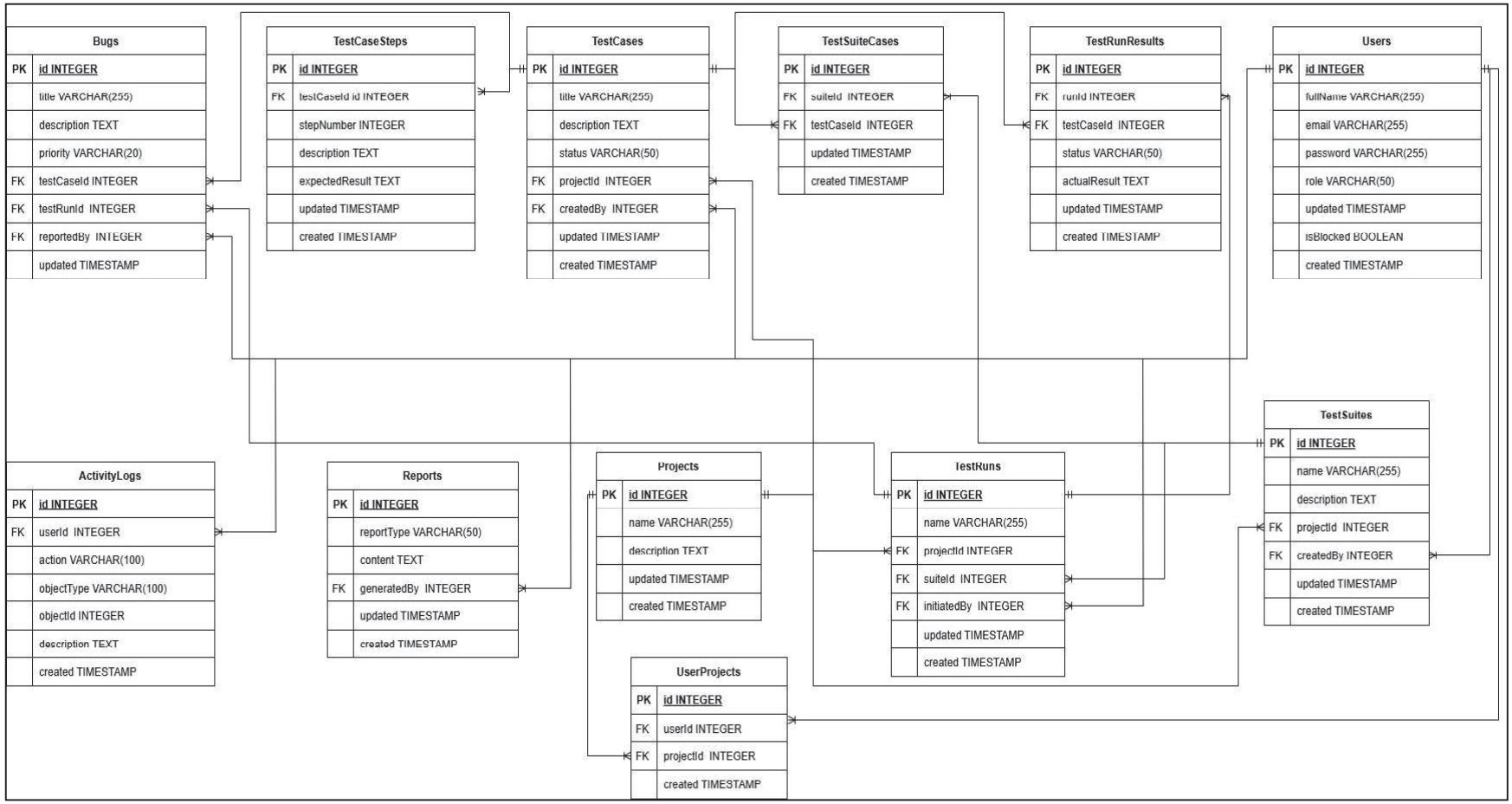

Рисунок 5.2 – Фізична модель даних IT-сервісу «Створення та управління тест-кейс

Таблиця 5.1 – Відомості про типи сутностей

Ім’я типу сутності	Опис	Особливості використання
Користувач (User)	Інформація про користувачів	Містить дані для ідентифікації користувачів
Проєкт (Project)	Інформація про проєкти	Кожен проєкт має свого автора та включає унікальний набір тест-кейсів і сьютів
Тест-кейс (TestCase)	Сценарій перевірки функціоналу	Може створюватися незалежно від сьюта, підтримує збереження очікуваних результатів, даних і файлів
Тестовий сьют (TestSuite)	Об’єднання тест-кейсів для групового запуску	Має логічний зв’язок із переліком ID тест-кейсів, прив’язується до проєкту
Прогін тестів (TestRun)	Виконання групи тестів	Пов’язаний із тестовим сьютом та зберігає статистику виконання
Статус тесту (TestStatus)	Результати виконання конкретного тест-кейсу	Включає інформацію про статус, час запуску, нотатки, прикріплені файли та посилання на баги
Баг (Bug)	Інформація про знайдені помилки	Зберігає опис, пріоритет, статус, прив’язку до кейсів/сьютів/прогонів і автора помилки
Журнал активності (ActivityLog)	Історія дій користувачів	Фіксує події, дату, час, користувача та об’єкт дії
Звіт (Report)	Збережені дані про згенеровані звіти	Містить тип звіту, дату генерації, автора
Призначення до проєкту (ProjectUsers)	Зв’язок між користувачами та проєктами	Визначає, хто з користувачів має доступ до конкретного проєкту і коли його призначено

Таблиця 5.2 – Відомості про атрибути

Назва таблиці	Атрибут	Опис	Обмеження	Допустимість Null
Users	id	Ідентифікатор користувача	Первинний ключ	Ні
	fullName	Ім’я користувача		Ні
	email	Електронна адреса користувача		Ні
	password	Пароль користувача		Ні
	role	Роль користувача		Ні
	isBlocked	Ознака блокування облікового запису		Ні
	created	Дата створення		Ні
	updated	Дата оновлення		Ні
Projects	id	Ідентифікатор проєкту	Первинний ключ	Ні
	name	Назва проєкту		Ні
	description	Опис проєкту		Так
	updated	Дата оновлення		Ні
	created	Дата створення		Ні
UserProjects	id	Ідентифікатор зв’язку	Первинний ключ	Ні
	userId	Ідентифікатор користувача	Зовнішній ключ	Ні
	projectId	Ідентифікатор проєкту	Зовнішній ключ	Ні
	created	Дата створення		Ні

Продовження таблиці 5.2

Назва таблиці	Атрибут	Опис	Обмеження	Допустимість Null
TestCaseSteps	id	Ідентифікатор кроку	Первинний ключ	Ні
	testCaseId	Ідентифікатор тест-кейсу	Зовнішній ключ	Ні
	stepNumber	Номер кроку		Ні
	description	Опис кроку		Так
	expectedResult	Очікуваний результат		Ні
	updated	Дата оновлення		Ні
	created	Дата створення		Ні
TestRuns	id	Ідентифікатор прогона	Первинний ключ	Ні
	name	Назва прогона		Ні
	projectId	Ідентифікатор проєкту	Зовнішній ключ	Ні
	suiteId	Ідентифікатор сьюта	Зовнішній ключ	Ні
	initiatedBy	Ініціатор прогона	Зовнішній ключ	Ні
	updated	Дата оновлення		Ні
	created	Дата створення		Ні
TestSuiteCases	id	Ідентифікатор зв’язку	Первинний ключ	Ні
	suiteId	Ідентифікатор тест-сьюта	Зовнішній ключ	Ні
	testCaseId	Ідентифікатор тест-кейсу	Зовнішній ключ	Ні
	updated	Дата оновлення		Ні
	created	Дата створення		Ні

Продовження таблиці 5.2

Назва таблиці	Атрибут	Опис	Обмеження	Допустимість Null
TestSuites	id	Ідентифікатор сьюта	Первинний ключ	Ні
	name	Назва сьюта		Ні
	description	Опис сьюта		Так
	projectId	Ідентифікатор проєкту	Зовнішній ключ	Ні
	updated	Дата оновлення		Ні
	created	Дата створення		Ні
	createdBy	Автор сьюта	Зовнішній ключ	Ні
TestCases	id	Ідентифікатор тест-кейсу	Первинний ключ	Ні
	description	Опис тест-кейсу		Так
	status	Статус тест-кейсу		Ні
	updated	Дата оновлення		Ні
	created	Дата створення		Ні
	projectId	Ідентифікатор проєкту	Зовнішній ключ	Ні
	createdBy	Автор тест-кейсу	Зовнішній ключ	Ні
TestRunResults	id	Ідентифікатор результату	Первинний ключ	Ні
	runId	Ідентифікатор прогона	Зовнішній ключ	Ні
	testCaseId	Ідентифікатор тест-кейсу	Зовнішній ключ	Ні
	status	Статус виконання		Ні
	actualResult	Фактичний результат		Так
	updated	Дата оновлення		Ні
	created	Дата створення		Ні

Кінець таблиці 5.2

Назва таблиці	Атрибут	Опис	Обмеження	Допустимість Null
Bugs	id	Ідентифікатор бага	Первинний ключ	Ні
	title	Заголовок бага		Ні
	description	Опис бага		Так
	priority	Пріоритет		Ні
	testCaseId	Ідентифікатор тест-кейсу	Зовнішній ключ	Ні
	testRunId	Ідентифікатор прогона	Зовнішній ключ	Ні
	reportedBy	Автор	Зовнішній ключ	Ні
	updated	Дата оновлення		Ні
Reports	id	Ідентифікатор звіту	Первинний ключ	Ні
	reportType	Тип звіту		Ні
	content	Вміст звіту		Так
	generatedBy	Хто згенерував	Зовнішній ключ	Ні
	created	Дата створення		Ні
	updated	Дата оновлення		Ні
ActivityLogs	id	Ідентифікатор запису	Первинний ключ	Ні
	userId	Ідентифікатор користувача	Зовнішній ключ	Ні
	action	Тип дії		Ні
	objectType	Тип об’єкта		Ні
	objectId	Ідентифікатор об’єкта		Ні
	description	Опис дії		Так
	created	Дата створення		Ні

Було здійснено логічне та фізичне моделювання БД для IT-сервісу «Створення та управління тест-кейсами». Результатом логічного моделювання стало виявлення ключових сутностей системи, таких як користувачі, проєкти, тест-кейси, тестові сьюти, прогони, статуси тестів, баги та активність користувачів, а також встановлення між ними основних зв’язків відповідно до логіки роботи сервісу.

Модель побудована з урахуванням потреб сучасної гнучкої IT-архітектури, що допускає використання зв’язків типу «багато до багатьох» без суворої нормалізації[13]. Такий підхід забезпечує баланс між структурованістю даних і гнучкістю системи, дозволяє спростити реалізацію повторного використання об’єктів (наприклад, тест-кейсів у різних сьютах) та підвищує продуктивність при виконанні типових запитів.

У процесі фізичного моделювання були конкретизовані типи полів, визначено ключові залежності, уточнено структуру таблиць, включаючи підтримку вкладених даних і масивів. Це дозволяє ефективно реалізувати багаторівневу логіку взаємодії між об’єктами, швидко масштабувати систему, а також забезпечити зручне зберігання та обробку інформації на рівні БД.

Побудована модель є повноцінною основою для реалізації функціональних можливостей сервісу, підтримки складної логіки тестування, генерації звітів і аудиту користувацьких дій.

6 РОЗРОБКА Й ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ МАТЕМАТИЧНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

Успішне функціонування інформаційної системи залежить від чітко визначеної послідовності дій, яка забезпечує логіку взаємодії між користувачами, даними та основними компонентами системи. Алгоритм роботи ІТ-сервісу дозволяє узагальнити та формалізувати ці взаємозв'язки, описуючи порядок виконання ключових операцій відповідно до ролей користувачів, етапів роботи та обробки інформації. Такий формат візуалізації є універсальним способом передавання технічної інформації, що підтверджується сучасними підходами до інженерії програмного забезпечення, де візуальні моделі активно використовуються в якості документації й супроводу систем[14]. Блок-схеми можуть бути використані як елемент технічної документації, зокрема для аналізу, оптимізації та супроводу системи в майбутньому.

Менеджер розпочинає взаємодію з сервісом із форми логіну. Після успішної авторизації система перевіряє роль користувача і надає доступ до навігаційного меню з пунктами «Головна», «Проєкти», «Користувачі», «Звіти». Після входу менеджер потрапляє на головну сторінку, де завантажується загальна інформація: кількість усіх створених проєктів, кількість усіх створених користувачів, таблиця з даними про активність менеджера, наприклад, які операції здійснено, у який час. Дані завантажуються з бази даних і відображаються у вигляді інформативних блоків.

Менеджер переходить до розділу «Користувачі», де відображається список усіх створених користувачів де може додати нового користувача, переглянути або редагувати існуючого, заблокувати або видалити обраного

користувача. Додавання нового користувача відбувається через модальне вікно, у якому вводяться базові дані: повне ім’я, електронна пошта, пароль і список проєктів. Після підтвердження система зберігає користувача до бази даних і створює розсилку на електронну пошту з логін-даними. Якщо менеджер бажає переглянути або відредагувати користувача, відкривається форма з поточними даними. Тут можна змінити прізвищета ім’я, email, список проєктів або пароль. Зміни зберігаються лише після підтвердження. У разі потреби користувача можна заблокувати або видалити. Для цього система відкриває модальне вікно з підтвердженням.

У розділі «Проєкти» менеджер може керувати всіма проєктами, створеними в системі. Додавання нового проєкту також виконується через модальне вікно, де вводиться назва та опис проєкту. Після підтвердження дані записуються в базу. При перегляді або редагуванні проєкту відкривається форма з існуючими даними. Менеджер може змінити назву чи опис і зберегти ці зміни. Якщо потрібно, проєкт можна видалити після підтвердження у модальному вікні.

Менеджер також має можливість створення звітів за різними критеріями. Система пропонує декілька типів звітів: про знайдені баги, про результати тестового прогона, про загальні результати тестування та про активність користувачів. В залежності від вибору, користувач здійснює налаштування параметрів звіту, таких як дата, тест-сьют, пріоритет багів, статус тест-кейсів або тип активності. Після введення усіх необхідних параметрів система формує файл звіту відповідного типу, який можна зберегти або експортувати.

Схема алгоритму роботи ІТ-сервісу «Створення та управління тест-кейсами» для менеджера проєкту наведено на рисунках 6.1.

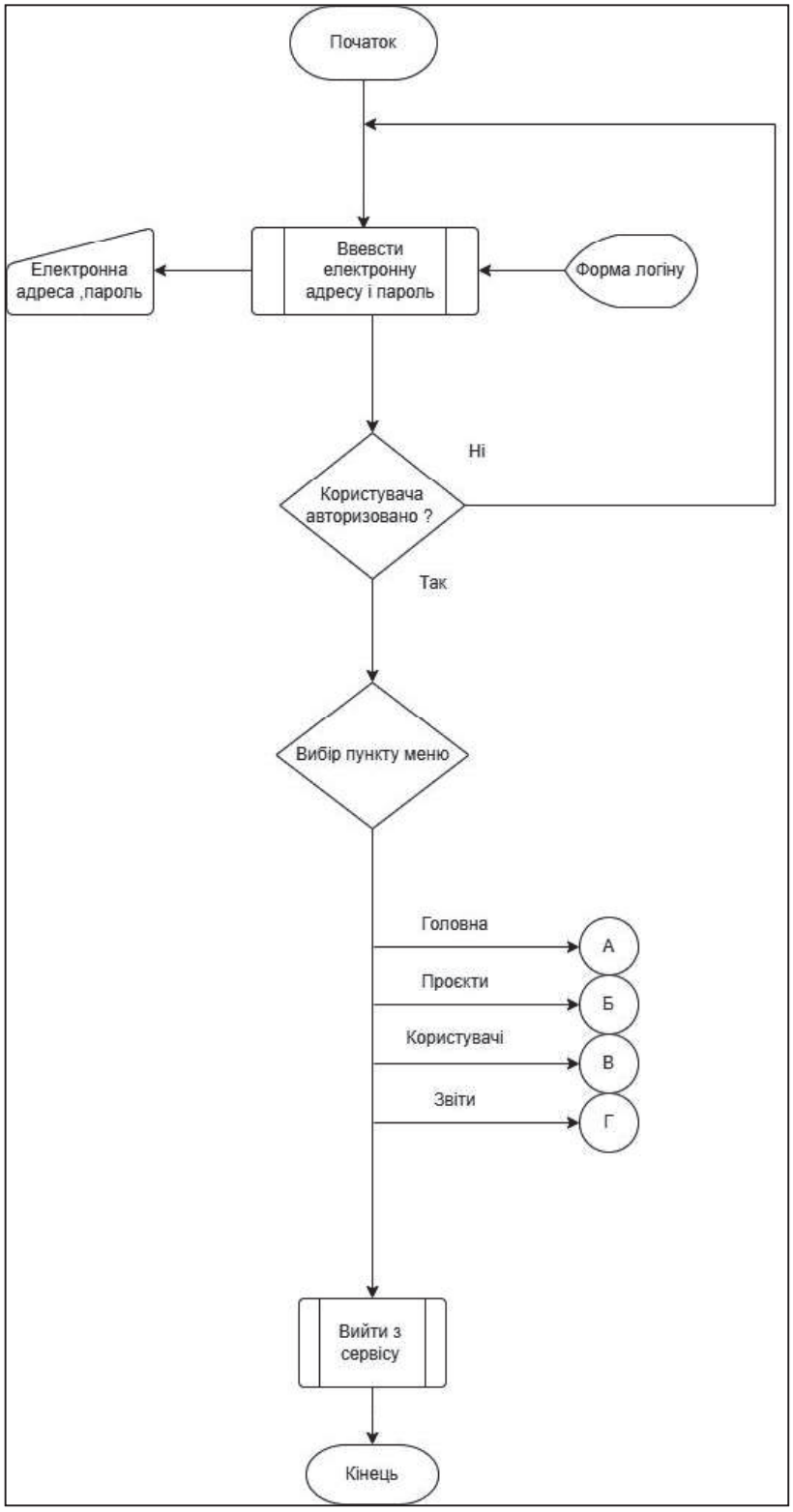

Рисунок 6.1 – Схема алгоритму роботи IT-сервісу «Створення та управління тест-кейсами» для менеджера проєкту

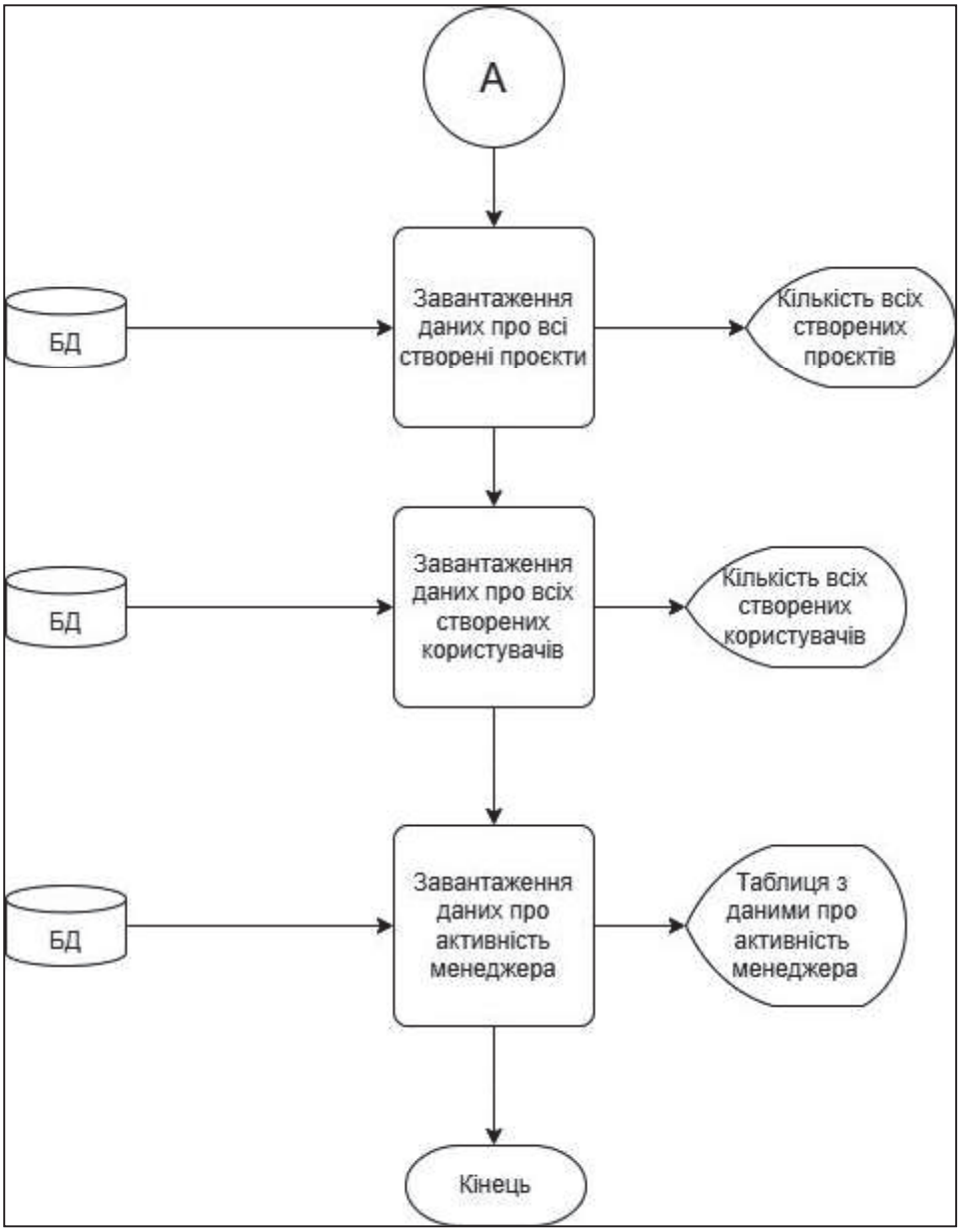

Рисунок 6.1, аркуш 2

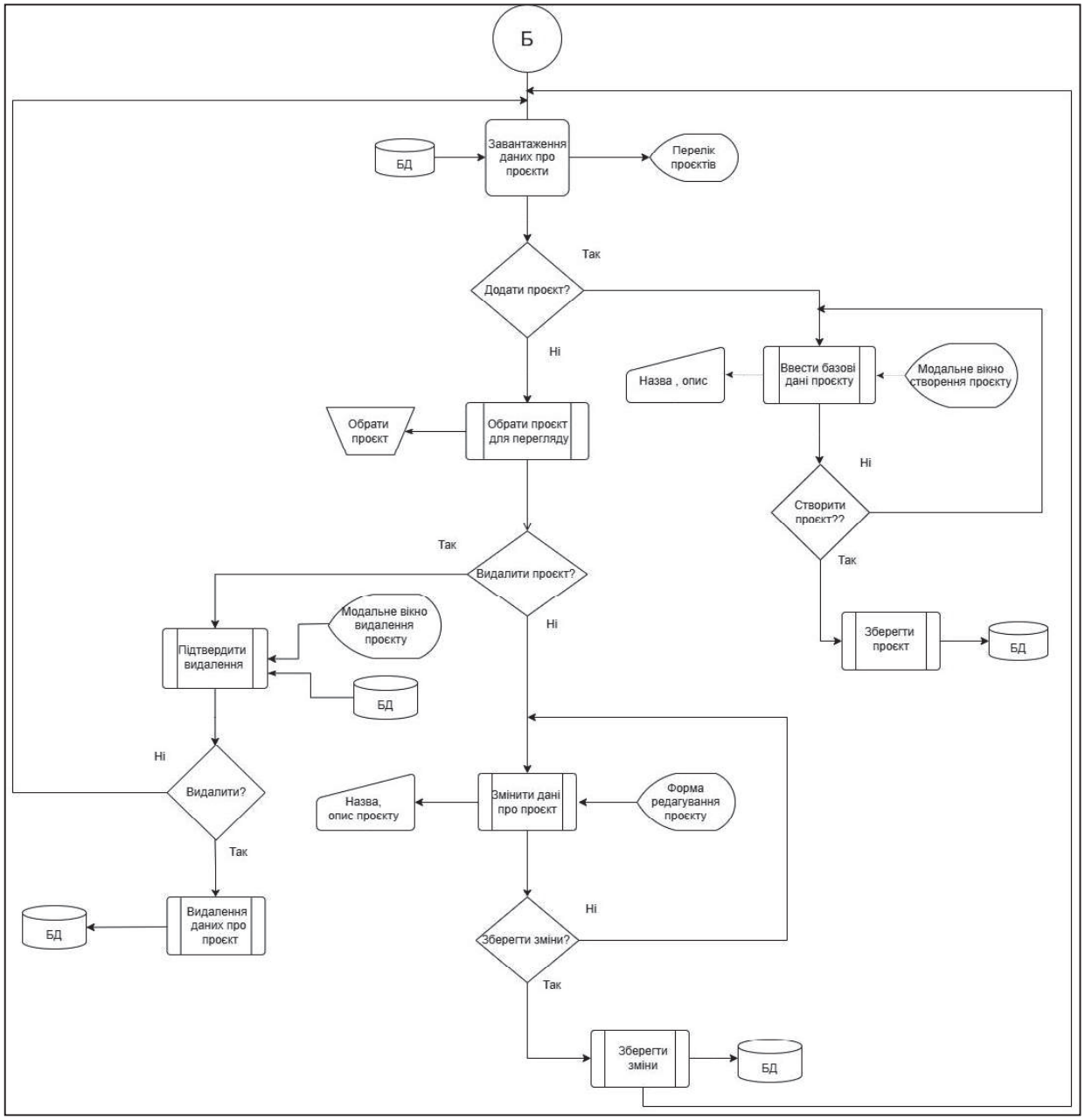

Рисунок 6.1, аркуш 3

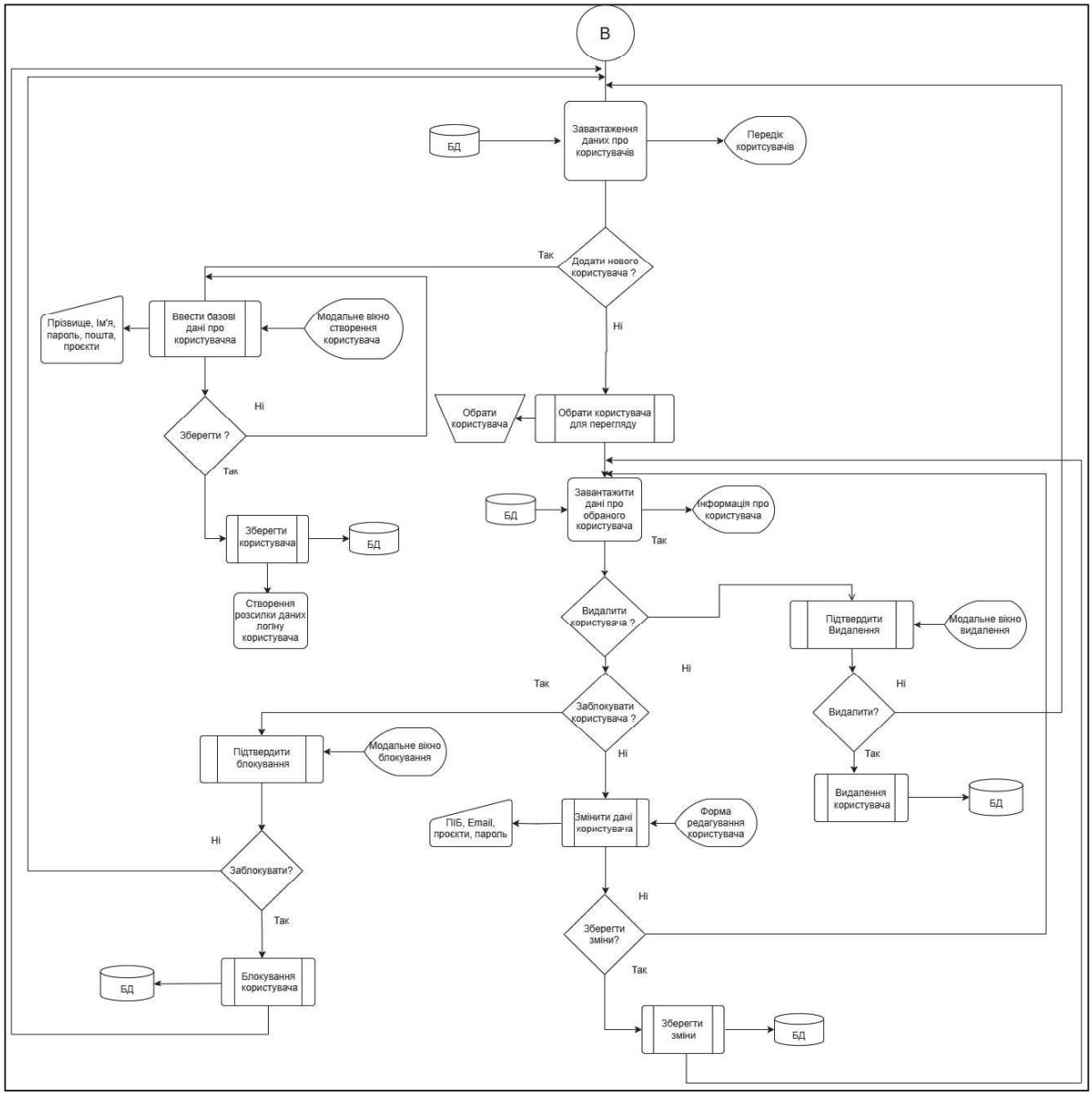

Рисунок 6.1, аркуш 4

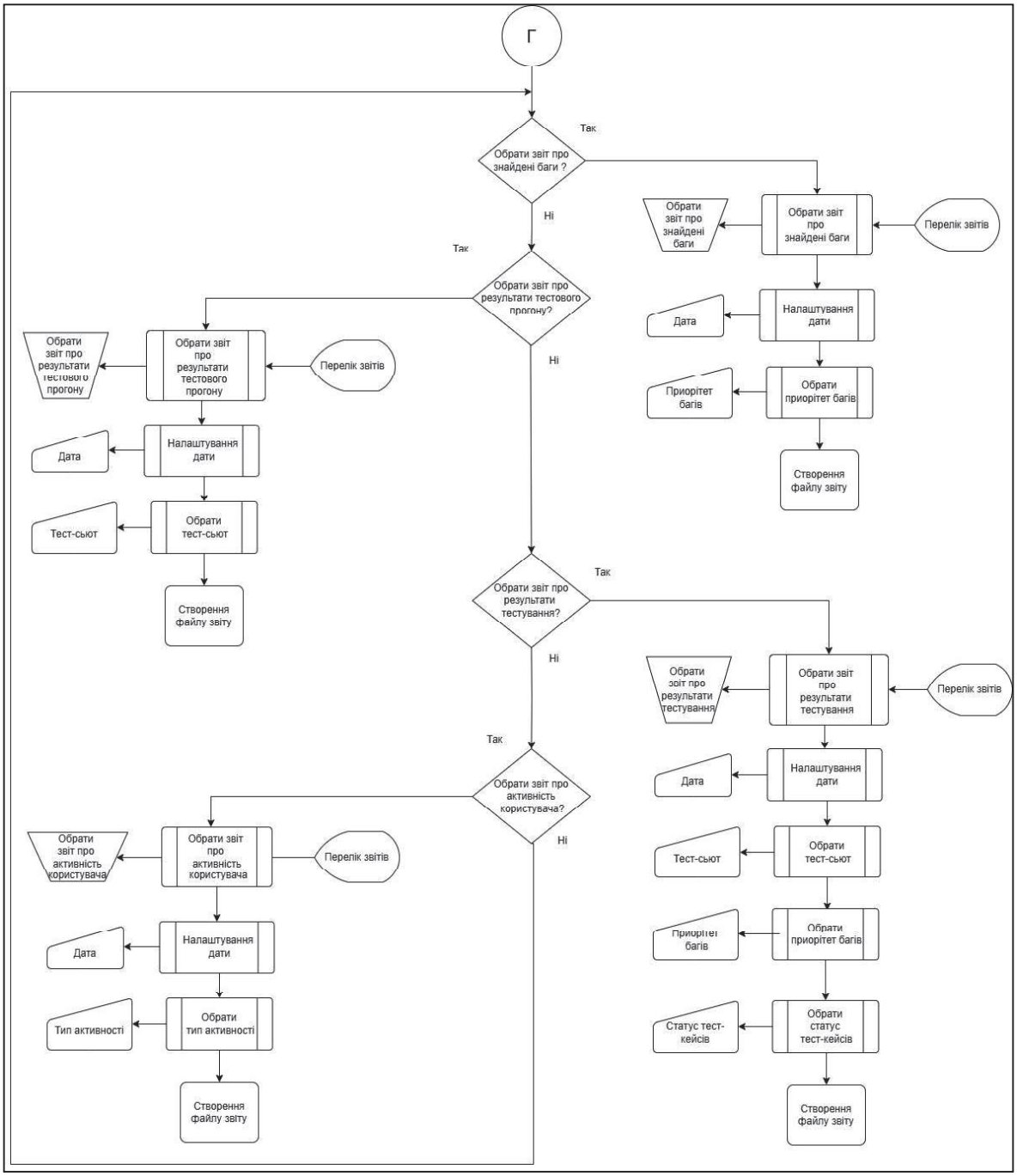

Рисунок 6.1, аркуш 5

Робота користувача починається зі сторінки авторизації, де вводиться електронна адреса та пароль. Після успішного входу користувач отримує доступ до основного меню, яке включає такі розділи, як «Головна», «Проєкти», «Тест-кейси», «Тест-сьюти» та «Звіти». Вибір розділу визначає подальший алгоритм взаємодії із сервісом. У розділі «Проєкти» реалізовано створення, редагування та видалення інформації про проєкти.

Розділ «Тест-кейси» дозволяє створювати, переглядати, редагувати або видаляти окремі тест-кейси. Процес створення передбачає введення базової інформації, а також додавання етапів тестування, таких як кроки, тестові дані, очікуваний і фактичний результат. Усі дані зберігаються після завершення редагування.

У розділі «Тест-сьюти» користувач має змогу створювати та формувати набори тест-кейсів, обирати кейси з бази, запускати прогін тестів, проставляючи статус кожному кейсу. Після завершення усіх дій тест-сьют зберігається разом з інформацією про результати.

Система також підтримує перегляд і редагування вже створених тест-сьютів. Користувач може завантажити дані про конкретний тест-сьют, ознайомитися з повною інформацією, або за необхідності видалити його через модальне підтвердження.

Окрему роль у системі відіграє модуль створення звітів, який дозволяє генерувати звіти чотирьох типів: про знайдені баги, про результати тестового прогону, результати тестування та активність користувача. Алгоритм генерації звіту передбачає послідовний вибір параметрів: діапазону дат, пріоритетів, статусів тест-кейсів або типу активності. На основі обраних критеріїв формується відповідна вибірка даних, яка автоматично експортується у формат звіту, доступного для збереження на клієнтському пристрої. Схема алгоритму роботи ІТ-сервісу «Створення та управління тест-кейсами»для тестувальника наведено на рисунках 6.2.

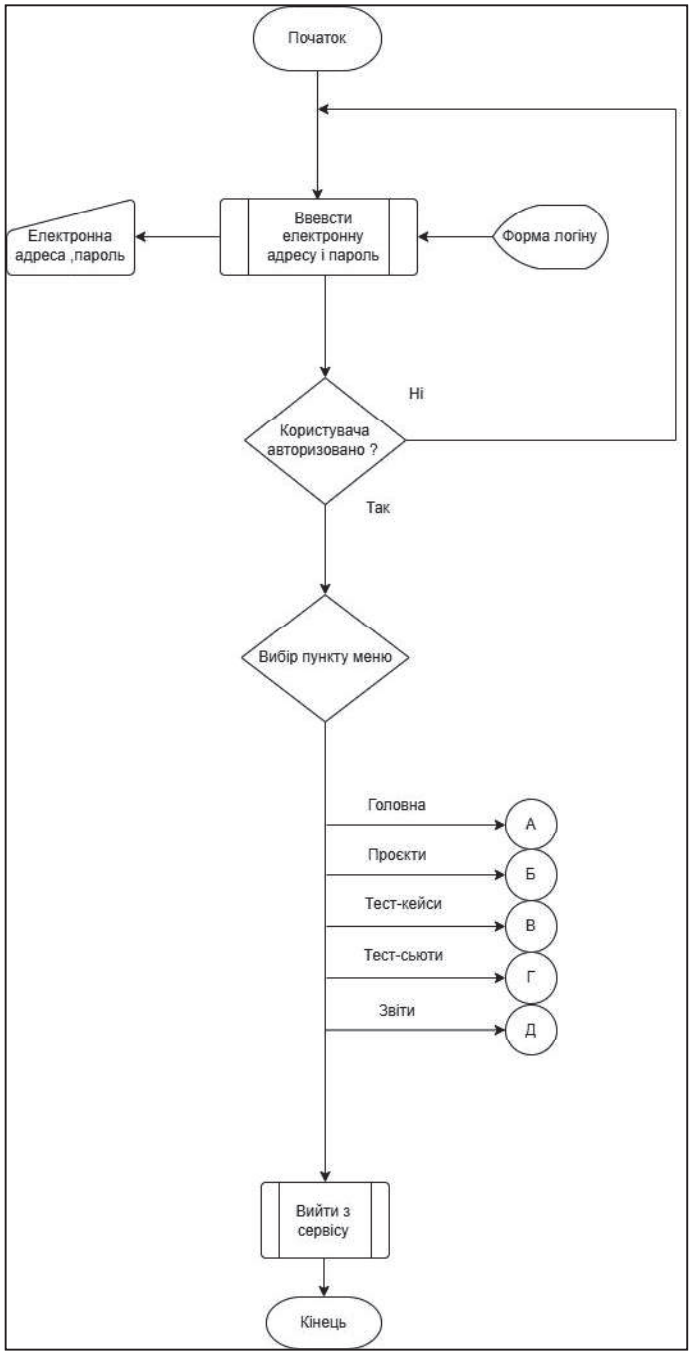

Рисунок 6.2 – Схема алгоритму роботи IT-сервісу «Створення та управління тест-кейсами» для тестувальника

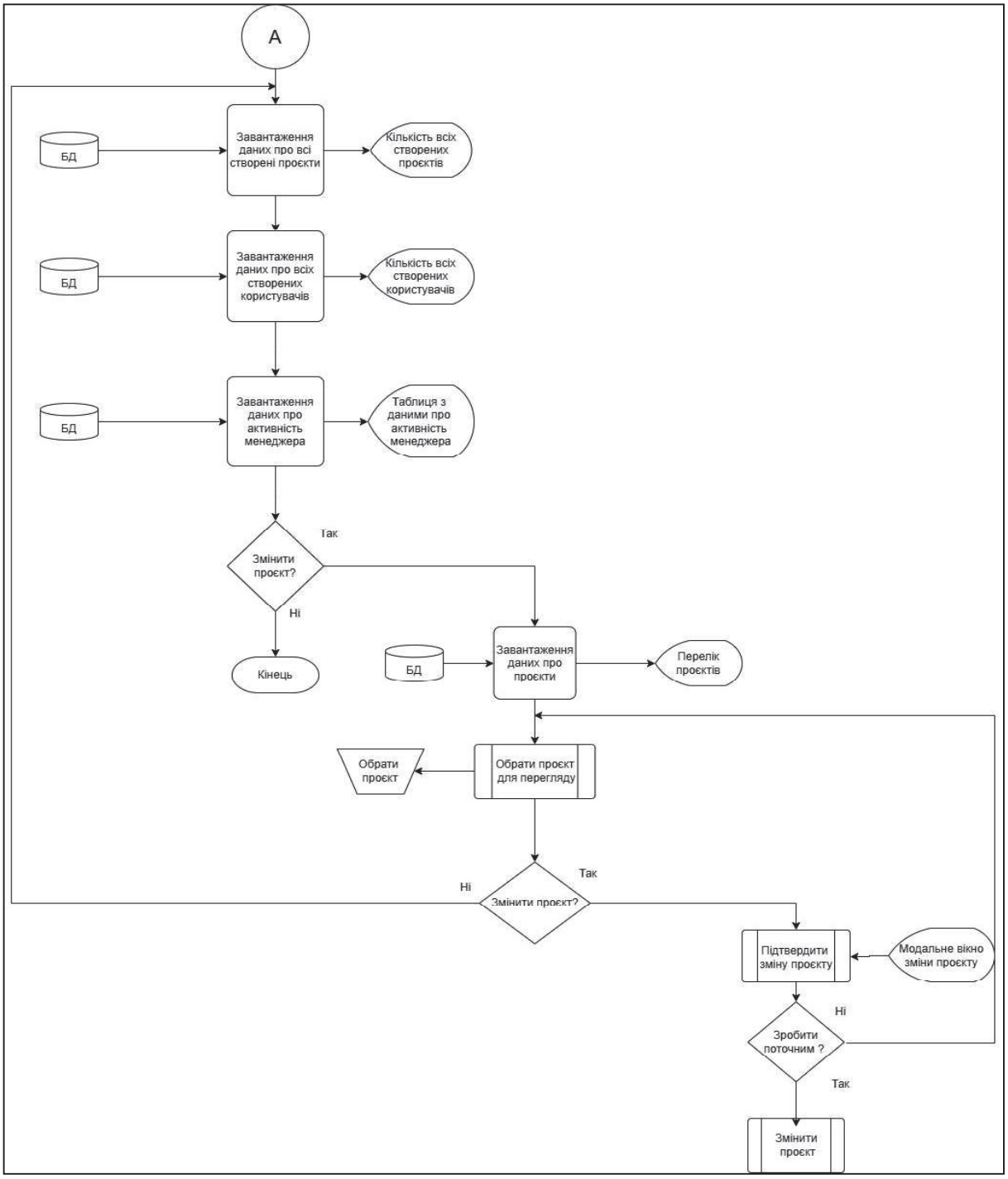

Рисунок 6.2, аркуш 2

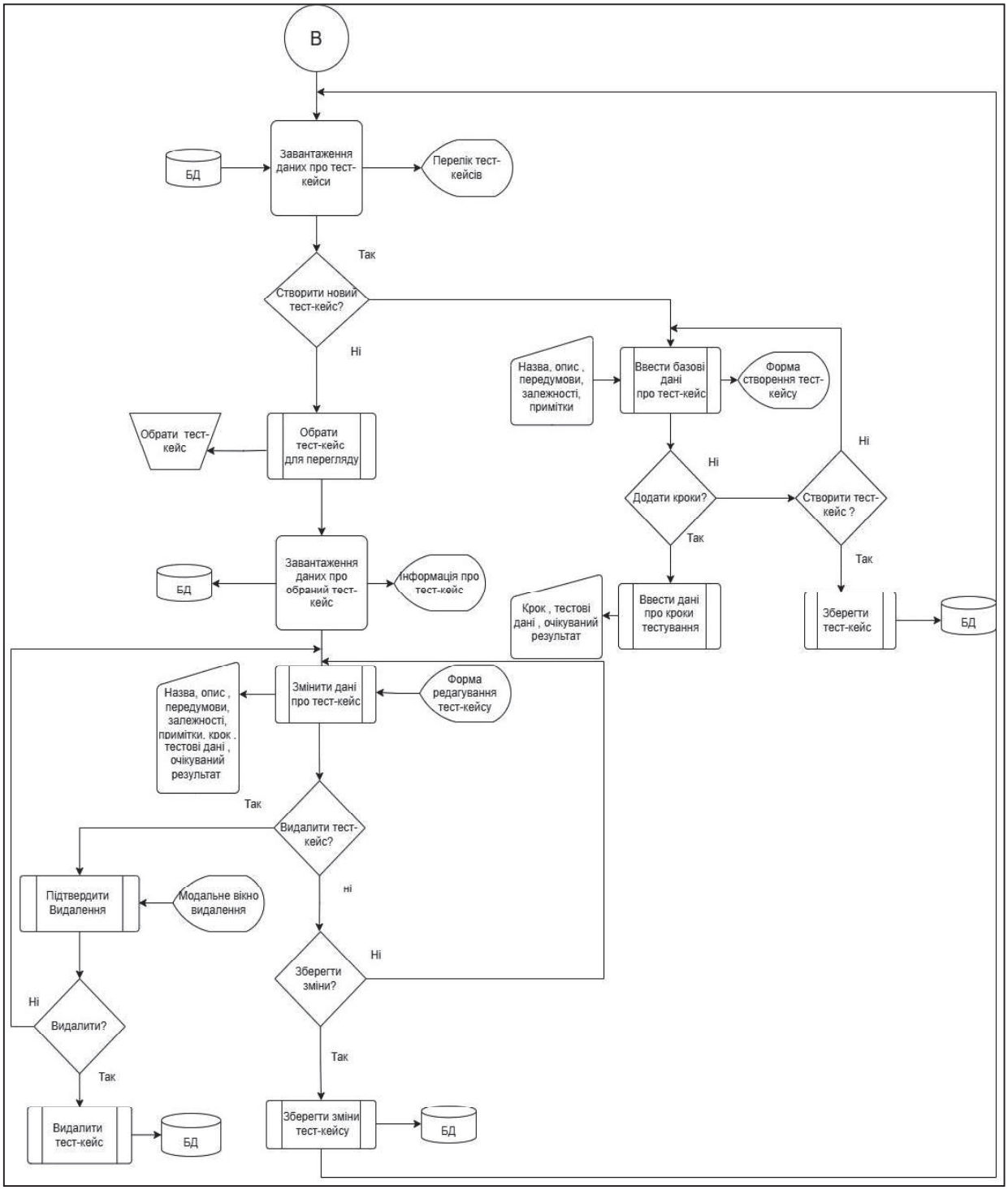

Рисунок 6.2, аркуш 3

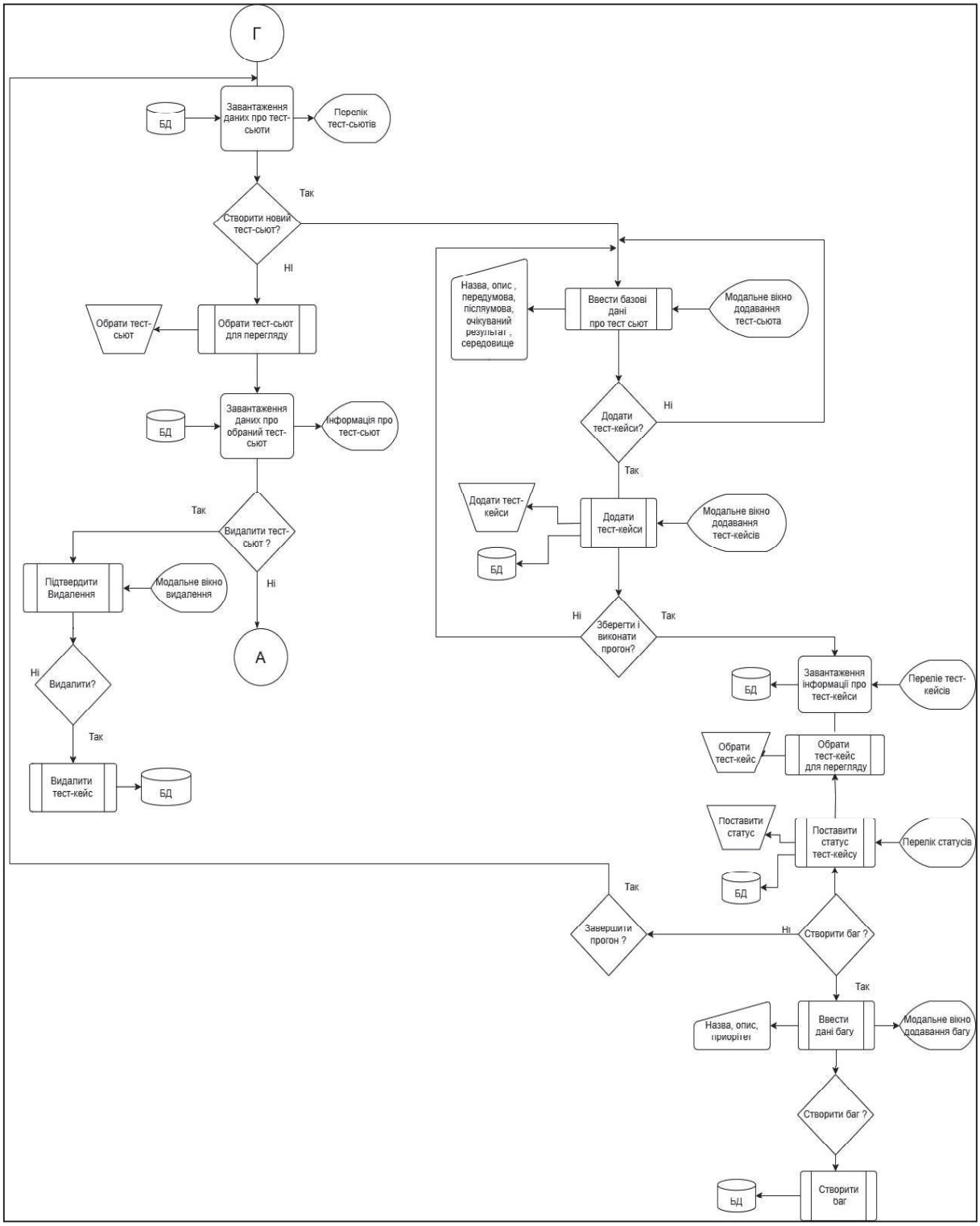

Рисунок 6.2, аркуш 4

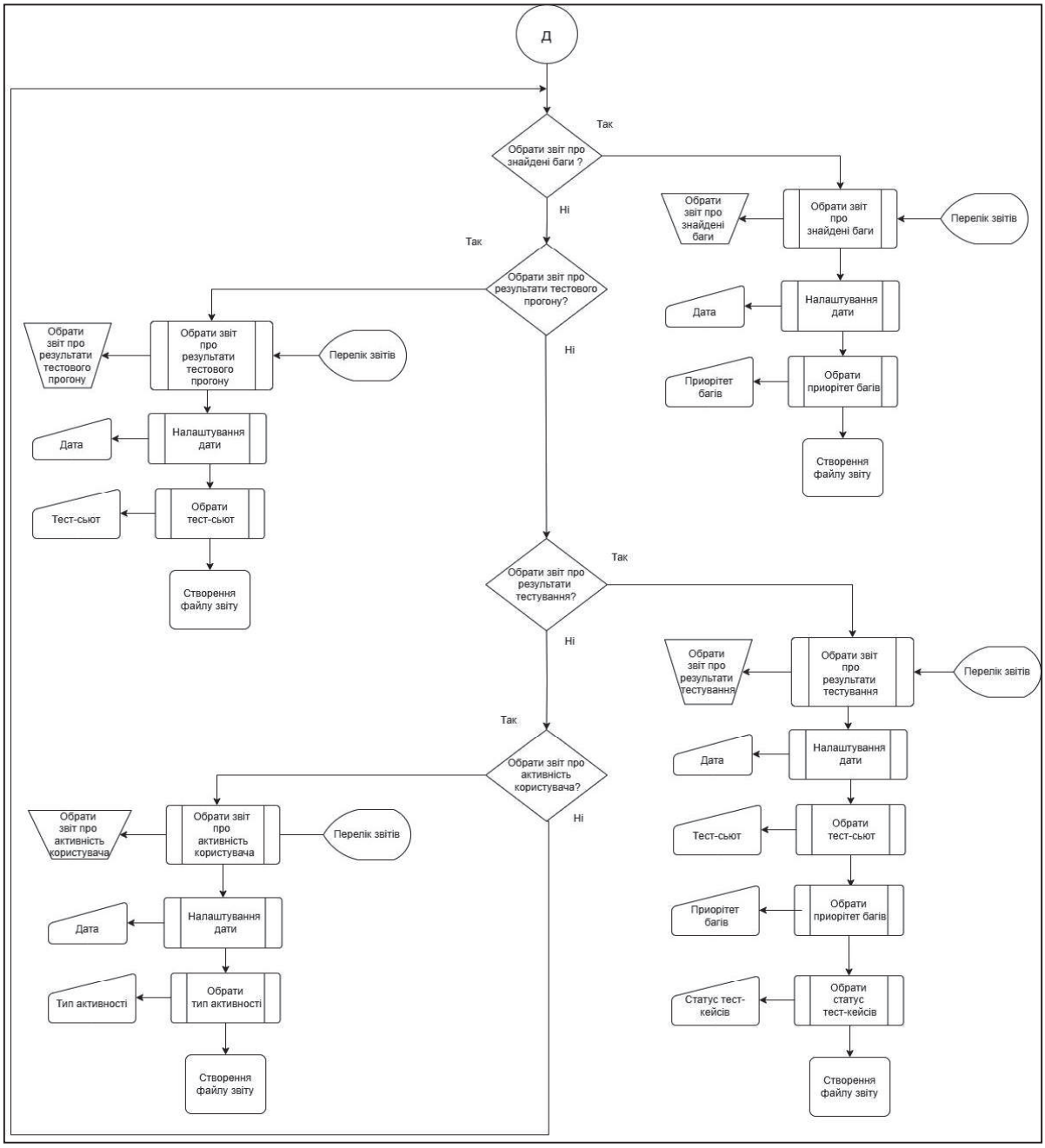

Рисунок 6.2, аркуш 5

7 РОЗРОБКА Й ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ПРОГРАМНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

Технічна архітектура ІТ-сервісу «Створення та управління тест-кейсами» побудована за класичною моделлю клієнт-серверної взаємодії, у якій серверна частина виконує як логіку обробки запитів, так і генерацію користувацького інтерфейсу. Такий підхід дозволяє знизити навантаження на клієнтську частину, централізувати контроль над даними та забезпечити ефективне управління доступом на рівні окремих функціональних модулів. Завдяки цьому забезпечується стабільність, контрольованість і розширюваність усіх процесів у системі.

Серверна частина реалізована за допомогою високорівневого фреймворку Django, що працює у зв’язці з мовою програмування Python. Django забезпечує повний цикл розробки починаючи від маршрутизації й обробки Hypertext Transfer Protocol (HTTP)-запитів до взаємодії з базою даних через Object-Relational Mapping (ORM), автентифікації користувачів, генерації HyperText Markup Language (HTML)-шаблонів і реалізації валідації введених даних[15]. Всі запити надходять через Uniform Resource Locator-конфігурації, обробляються відповідними представленнями та викликають необхідні моделі. У структурі проєкту чітко розмежовано модулі, які відповідають за Create, Read, Update та Delete-операції, авторизацію, перевірку прав доступу, генерацію звітів і роботу з шаблонами, що підвищує підтримуваність, модульність і масштабованість розв’язку.

Використання Python як основної мови реалізації дозволяє зосередитись на логіці проєкту, мова має чистий синтаксис і широку екосистему бібліотек, що забезпечує гнучкість у розробці, а також відкриває можливості для подальшої інтеграції з аналітичними, автоматизованими та ML-модулями.

Django, у свою чергу, забезпечує високу швидкість розробки та має вбудовані механізми захисту, що є критично важливим для корпоративних інформаційних систем.

Таке поєднання дозволяє побудувати надійний серверний додаток із розділенням відповідальностей, централізованою логікою обробки запитів та прозорою інтеграцією з базою даних, що зменшує кількість помилок та спрощує підтримку системи.

Клієнтська частина представлена HTML-шаблонами, які динамічно рендеряться сервером із використанням шаблонізатора Django. Оформлення інтерфейсу здійснюється за допомогою Cascading Style Sheets, що дозволяє підтримувати єдиний стиль відображення всіх сторінок. Усі дії користувача — авторизація, створення тест-кейсів, запуск тестових прогонів, редагування проєктів і перегляд звітів — ініціюються через браузер, однак обробляються виключно на сервері. Така централізація дозволяє підвищити безпеку системи, гарантувати узгодженість стану й забезпечити повний контроль над усіма змінами.

Система управління БД відіграє ключову роль у забезпеченні цілісності, доступності та консистентності інформації в межах сервісу. У даному проєкті використовується потужна реляційна система управління базами даних(СУБД) PostgreSQL, яка підтримує всі сучасні механізми збереження даних: транзакційність, обмеження цілісності, зовнішні ключі, індексацію, збережені процедури та розширену роботу з типами даних. PostgreSQL забезпечує стабільну роботу навіть при інтенсивних навантаженнях і є оптимальним вибором для систем, що передбачають активне розширення та розвиток[16].

Незважаючи на складність функціональності самої СУБД, взаємодія з PostgreSQL реалізована через високорівневу обгортку — ORM Django. Це дозволяє працювати з базою даних не через прямі Structured Query Language(SQL)-запити, а за допомогою Python-об’єктів і класів моделей, це

значно спрощує розробку, покращує читаність коду, унеможливлює SQL-ін’єкції та зменшує ймовірність логічних помилок при взаємодії з базою. ORM самостійно трансформує Python-операції у відповідні SQL-запити, автоматично керує транзакціями та відстежує зміни даних.

Таким чином, ми отримуємо зручний інтерфейс взаємодії з даними на мові Python, водночас користуючись усіма перевагами масштабованої серверної бази даних. Це поєднання дозволяє легко адаптувати систему до складніших сценаріїв використання без зміни архітектури взаємодії з базою.

Ще однією ключовою особливістю архітектури проєкту є підтримка RESTful API. Усі основні функції сервісу, включно зі створенням, редагуванням і видаленням сутностей (користувачів, тест-кейсів, прогонів, звітів тощо), реалізовано через HTTP-інтерфейси з використанням методів GET, POST, PUT і DELETE. API забезпечує стандартизовану взаємодію як із внутрішніми модулями, так і з потенційними зовнішніми системами, зокрема мобільними клієнтами або інтеграційними рішеннями. Дана архітектура дозволяє масштабувати систему, інтегрувати нові інтерфейси й забезпечувати взаємодію з будь-якими клієнтськими платформами без змін у бізнес-логіці.

Для глибшого розуміння структури внутрішньої логіки системи була побудована діаграма класів зображена на рисунку 7.1, що відображає основні об’єкти предметної області, їх властивості та взаємозв’язки. Кожен з класів відповідає реальній сутності, яка бере участь у функціонуванні сервісу — зокрема користувач, проєкт, тест-кейс, сьют, прогін, баг, звіт тощо. Модель класів дозволяє не лише логічно описати взаємодію компонентів, але й слугує основою для реалізації бази даних, API і всіх серверних механізмів.

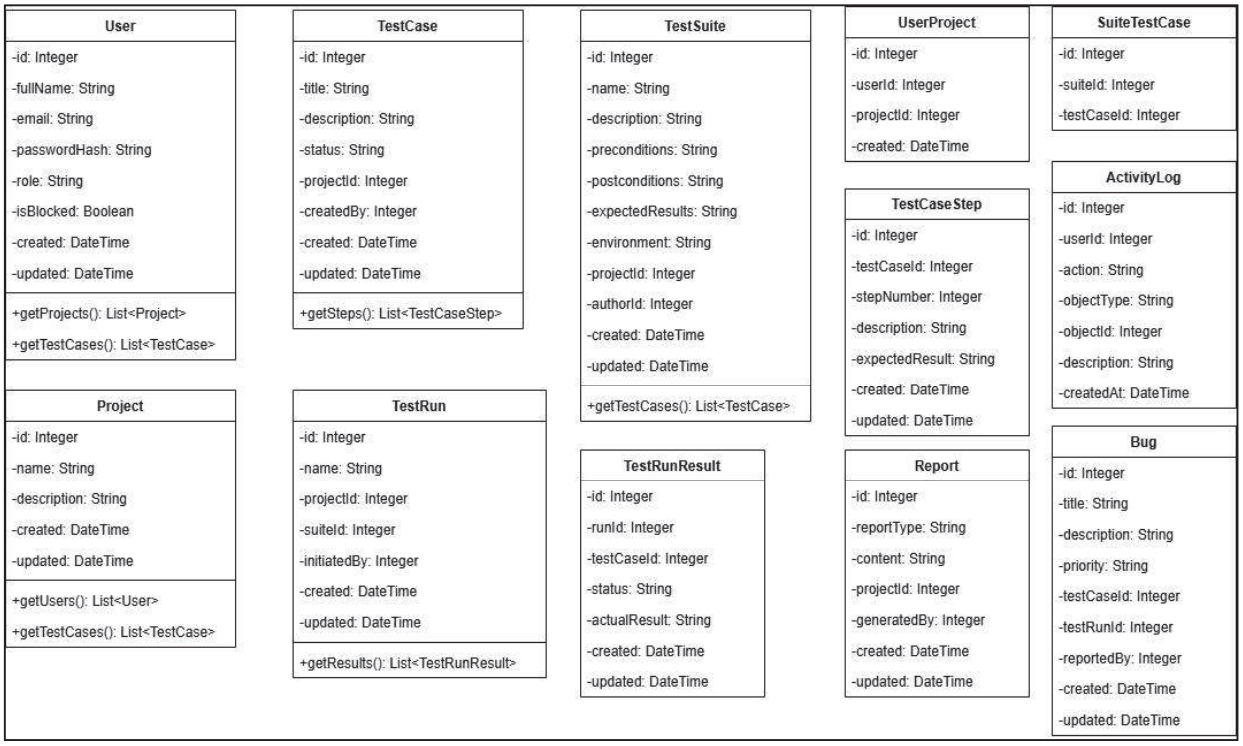

Рисунок 7.1 – Діаграма класів IT-сервісу «Створення та управління тест-кейсами»

Обрана технологічна база дозволяє досягти високого рівня стабільності системи в умовах реального навантаження. Вона підтримує масштабованість, дозволяє легко додавати нові модулі або адаптувати існуючі компоненти під змінні потреби.

8 РОЗРОБКА Й ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТЕХНІЧНОЇ ЗАБЕЗПЕЧУЮЧОЇ СИСТЕМИ

Для забезпечення надійного функціонування ІТ-сервісу «Створення та управління тест-кейсами» необхідно створити відповідну технічну інфраструктуру. Важливо правильно організувати взаємодію між клієнтськими пристроями, сервером та мережевим обладнанням, а також передбачити заходи на випадок перебоїв з електроживленням.

Сервіс реалізовано за моделлю клієнт–серверної архітектури, що забезпечує чітке розмежування між обробкою даних та відображенням інформації для кінцевого користувача[17]. Серверна частина функціонує на базі одного фізичного сервера, який виконує роль як застосункового, так і серверу бази даних. На ньому розгорнуто вебзастосунок, що забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних, яка зберігає тест-кейси, прогони, звіти, облікові записи користувачів та іншу інформацію, необхідну для роботи системи.

Для запобігання втраті даних та забезпечення стабільної роботи у разі перебоїв з електропостачанням, сервер підключено до джерела безперебійного живлення (ДБЖ) типу C13/C14. Це дозволяє автоматично підтримувати живлення на критичному обладнанні у разі аварійного вимкнення струму, тим самим зберігаючи цілісність усіх збережених даних.

Користувачі взаємодіють із сервісом через клієнтські пристрої (персональні комп’ютери або ноутбуки), які підключені до локальної мережі організації. Для повноцінної роботи з системою достатньо наявності сучасного веббраузера та стабільного з'єднання з локальним сервером. Усі компоненти системи це клієнтські пристрої, сервер, ДБЖ та інтернет-з’єднання пов’язані через мережевий комутатор, який забезпечує обмін

даними в межах локальної мережі. Комутатор також з'єднаний із зовнішньою мережею через Ethernet-інтерфейс, що дозволяє за потреби здійснювати адміністрування, оновлення або резервне копіювання з віддалених точок доступу.

Загальна схема технічної архітектури системи наведена на рисунку 8.1. Вона демонструє взаємозв’язки між ключовими компонентами: робочим місцем користувача, мережевим комутатором, сервером, ДБЖ та каналом зовнішнього підключення до мережі. Така конфігурація повністю відповідає вимогам для внутрішнього корпоративного застосування у сфері тестування програмного забезпечення.

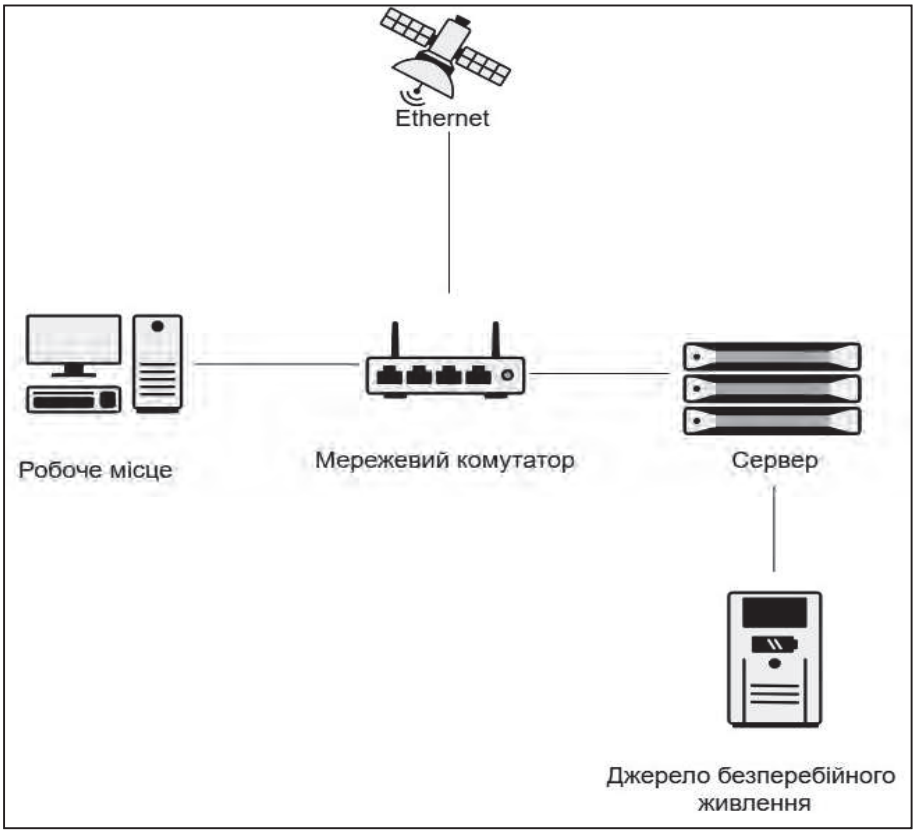

Рисунок 8.1 – Схема комплексу технічних засобів для впровадження IT-сервісу «Створення та управління тест-кейсами»

Завдяки такій інфраструктурі користувачеві не потрібно встановлювати додаткове ПЗ, а доступ до сервісу здійснюється через браузер. Це значно спрощує експлуатацію, зменшує витрати на впровадження та підвищує зручність роботи.

Розміщення технічного обладнання в приміщенні, де розгортається система, повинно відповідати вимогам умов безпечної експлуатації комп'ютерної техніки в організаціях. Встановлення, підключення та розташування серверів, клієнтських пристроїв, мережевого обладнання та джерел безперебійного живлення має здійснюватися з урахуванням нормативних документів у сфері охорони праці та пожежної безпеки.

Зокрема, необхідно дотримуватись вимог щодо безпечних і нешкідливих умов праці, що включає вимоги до мікроклімату приміщення, освітлення, рівня шуму та ергономіки робочого місця. Крім того, варто дотримуватись правила розміщення електронного обладнання з метою запобігання виникненню пожеж, забезпечення вільного доступу до засобів пожежогасіння, недопущення перевантаження електромережі та забезпечення належного стану кабельних трас. Також при облаштуванні приміщення рекомендовано враховувати санітарні норми щодо організації роботи з використанням ПК, зокрема щодо безпечної відстані від очей до монітора, положення обладнання відносно джерел природного і штучного освітлення, а також зручності доступу до портів і перемикачів. Усі технічні засоби повинні бути встановлені на стабільних поверхнях, мати заземлення (при необхідності) та не створювати перешкод для евакуації у разі надзвичайної ситуації.

9 РОЗРОБКА USER EXPERIENCE (UX) ТА USER INTERFACE (UI) РІШЕНЬ

Успішна реалізація ІТ-сервісу «Створення та управління тест-кейсами» неможлива без ретельно продуманого інтерфейсу користувача, який забезпечує ефективну взаємодію з усіма функціональними модулями системи. Для даного сервісу інтерфейс відіграє важливу роль у щоденній роботі, оскільки дозволяє користувачам швидко й інтуїтивно виконувати основні дії — створення, перегляд, редагування та обробку тестової документації. Під час проєктування інтерфейсу було враховано реальні сценарії використання сервісу, а також типові дії, які виконують учасники проєктів. Це дало змогу сформувати логічну структуру елементів керування, мінімізувати кількість кроків до ключових функцій та забезпечити зручність виконання повсякденних задач.

Особливістю проєкту є підтримка двох типів користувачів із різними рівнями доступу: менеджера та тестувальника. Менеджер виконує адміністративну та організаційну функцію бо він створює проєкти, додає тестувальників, формує завдання, а також має змогу переглядати та аналізувати результати тестування через відповідні звіти. Тестувальник, у свою чергу, працює безпосередньо з тест-кейсами, виконує тестові прогони, реєструє знайдені баги та взаємодіє з призначеними йому задачами. Такий розподіл ролей дозволяє розмежувати обов’язки в команді, підвищити безпеку системи й забезпечити цілеспрямований функціонал для кожної категорії користувачів.

У зв’язку з цим структура інтерфейсу була побудована з урахуванням ролі користувача, який увійшов у систему. При вході на платформу відбувається ідентифікація ролі, після чого кожному користувачу надається

доступ лише до тих функцій, які передбачені його повноваженнями. Це дозволяє уникнути перевантаження інтерфейсу, зменшити ризики помилкових дій та покращити загальну зручність користування сервісом.

Після запуску системи менеджер бачить інтерфейс авторизації який зображено на рисунку 9.1, що містить форму введення особистих облікових даних.

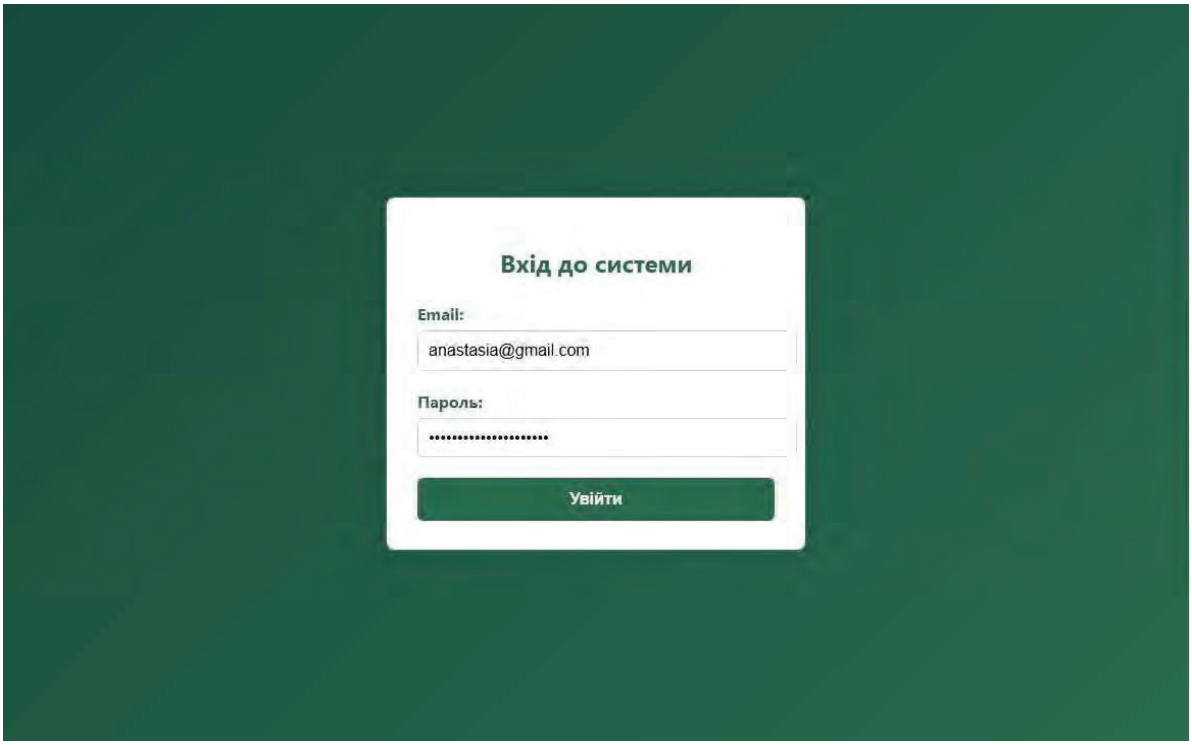

Рисунок 9.1 – Екранна форма входу у додаток

Інтерфейс включає два поля: для введення електронної пошти та пароля. Поле для пароля приховує введені символи, що забезпечує додаткову конфіденційність та захист. Після заповнення форми менеджер натискає кнопку Увійти, яка ініціює перевірку введених даних а також автоматично перевіряє роль користувача для відображення відповідних даних і, у разі успішної аутентифікації, перенаправляє на головну сторінку яку зображено

на рисунку 9.2.

У верхній частині головної сторінки розташовано два інформаційні блоки, які відображають статистику про загальну кількість проєктів та кількість зареєстрованих користувачів у системі.

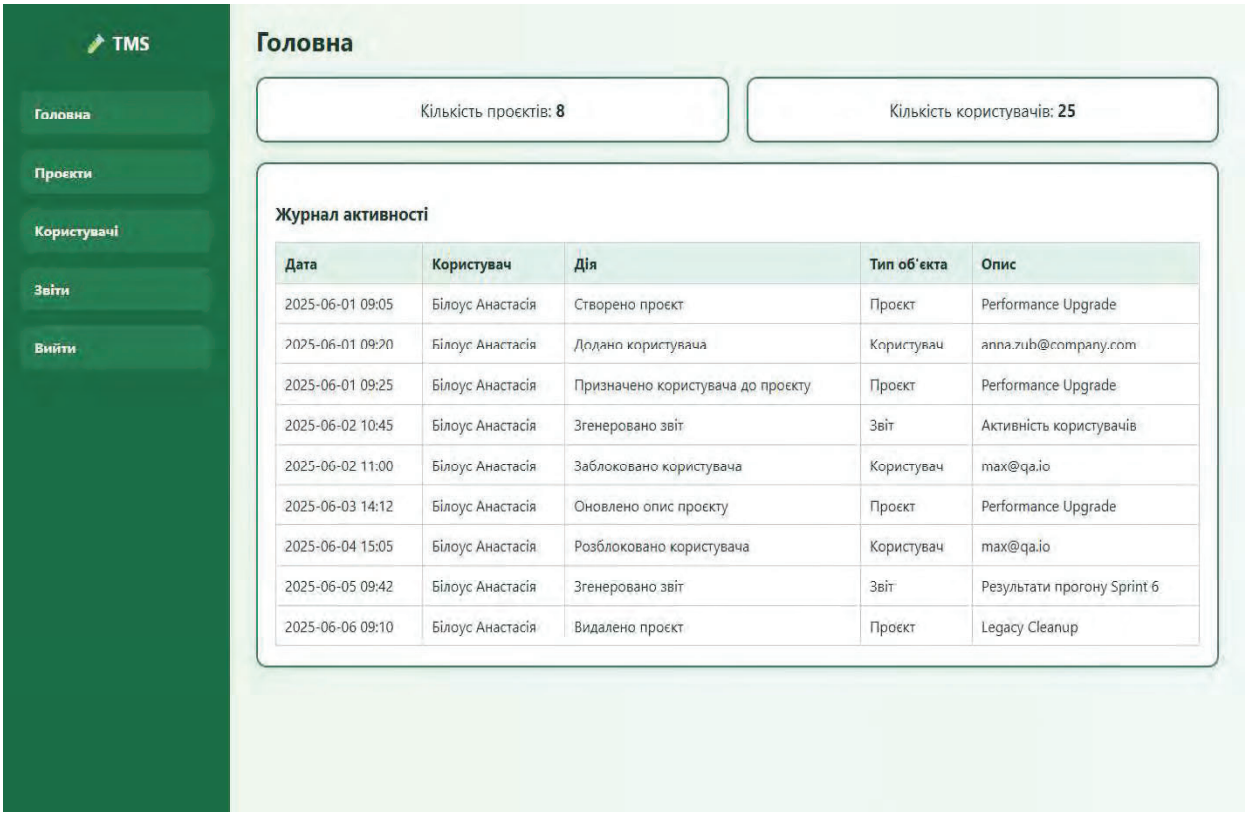

Рисунок 9.2 – Екранна форма головної сторінки

Далі розміщено журнал активності, який представлений у вигляді таблиці та містить перелік усіх ключових дій, що відбулися в системі за останній період. Для кожної дії вказано точну дату та час виконання, опис дії, а також об’єкт, до якого вона відноситься. Це дозволяє оперативно відстежити події, що відбувалися в системі: створення та оновлення проєктів, генерацію звітів, додавання або блокування користувачів, а також інші важливі зміни.

Зліва розташовано вертикальне навігаційне меню, яке забезпечує швидкий перехід між основними розділами сервісу: головна, проєкти, користувачі, звіти та вихід із системи. Далі перейдемо до сторінки проєктів натиснувши відповідну кнопку в навігаційному меню і побачимо інтерфейс сторінки зображений на рисунку 9.3.

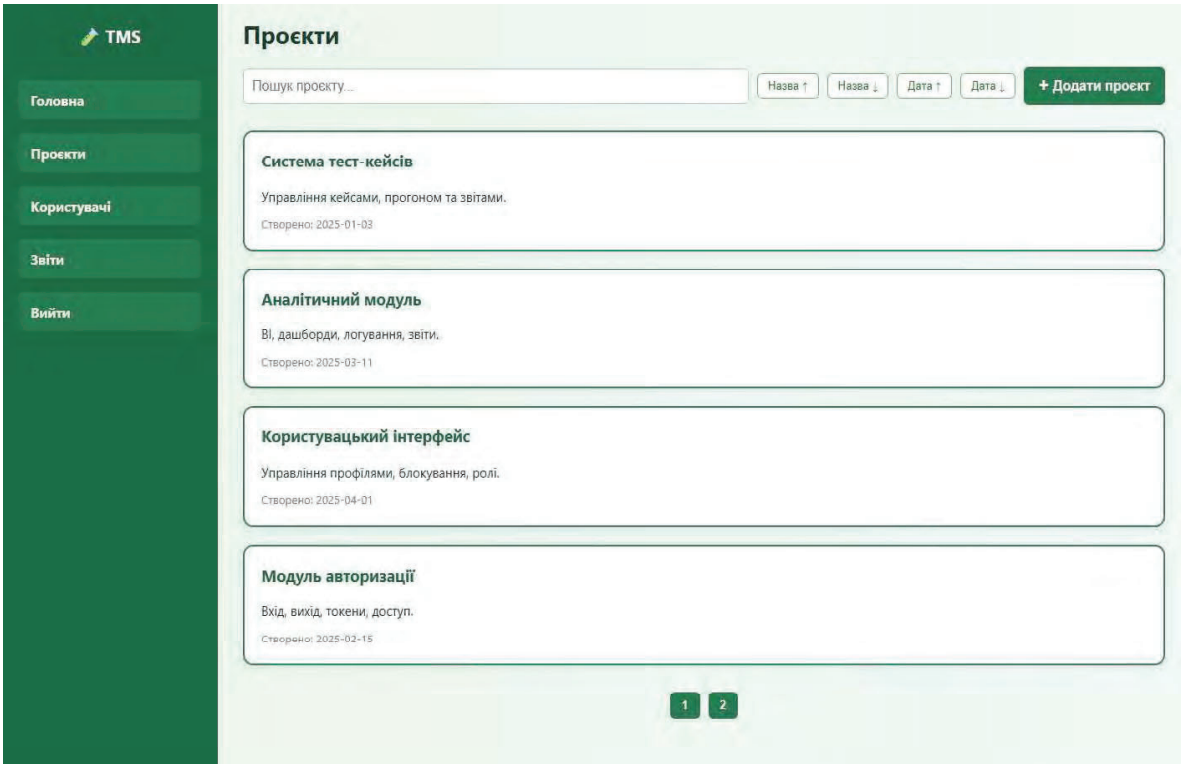

Рисунок 9.3 – Екранна форма сторінки Проєкти

Сторінка «Проєкти» відображає перелік усіх наявних проєктів, з якими працює менеджер. У верхній частині розташоване текстове поле пошуку, яке дозволяє швидко знайти проєкт за назвою, а також панель сортування, що надає змогу впорядковувати елементи за назвою чи датою створення у зростаючому або спадному порядку. Справа розміщено кнопку «Додати проєкт», що відкриває модальне вікно для створення нового проєкту яке ми бачимо на рисунку 9.4.

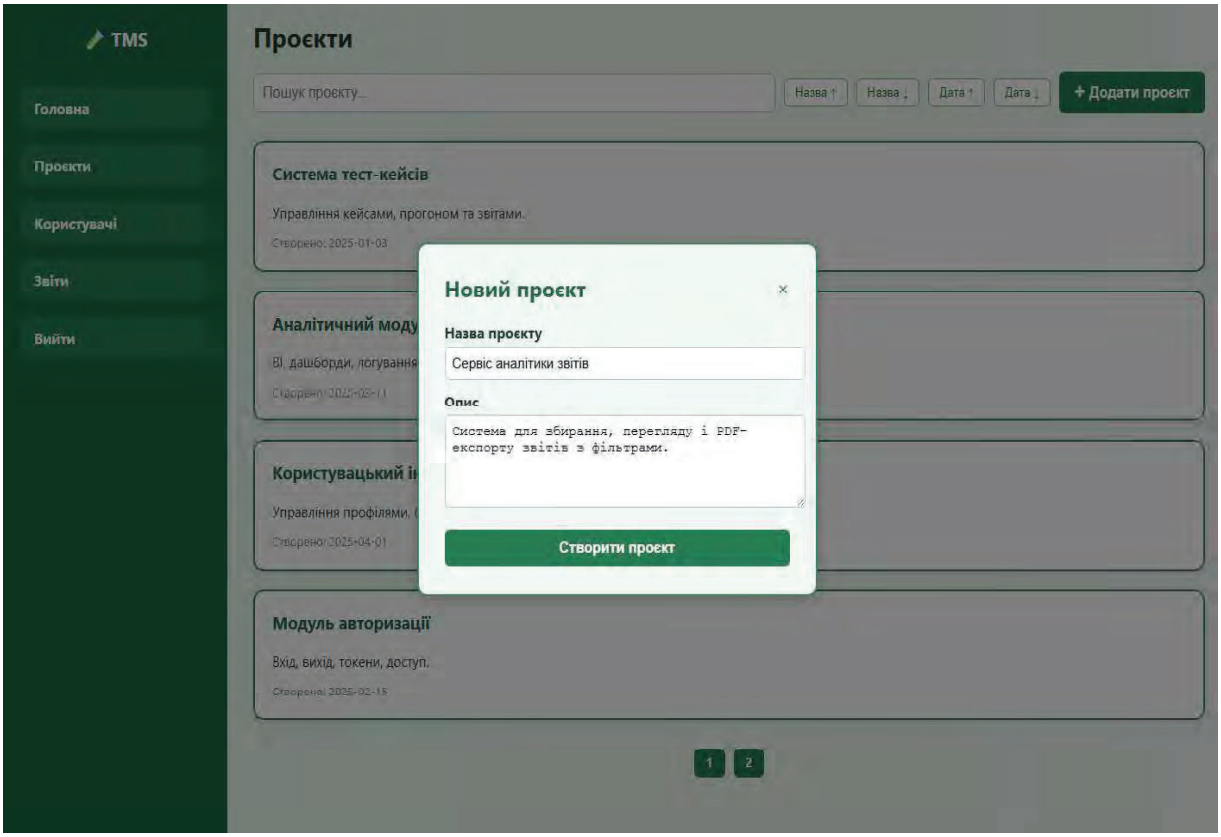

Рисунок 9.4 – Екранна форма додавання нового проєкту

У цьому для створення проєкту користувач може ввести назву проєкту та короткий опис його призначення або функціональності. У полі «Назва проєкту» вводиться унікальний заголовок, за яким надалі відображатиметься цей проєкт у загальному переліку, а в полі «Опис» вводимо текстове пояснення, яке може містити інформацію про основне призначення, функціональні модулі або особливості роботи проєкту. Форма завершується кнопкою «Створити проєкт», натискання на яку ініціює передачу даних на сервер для збереження нового об’єкта. Після успішного створення проєкту він автоматично додається до списку і відображається серед інших проєктів на відповідній сторінці. Перейдемо до сторінки користувачів натиснувши на відповідну кнопку навігаційного меню зовнішній вигляд сторінки зображено на рисунку 9.5.

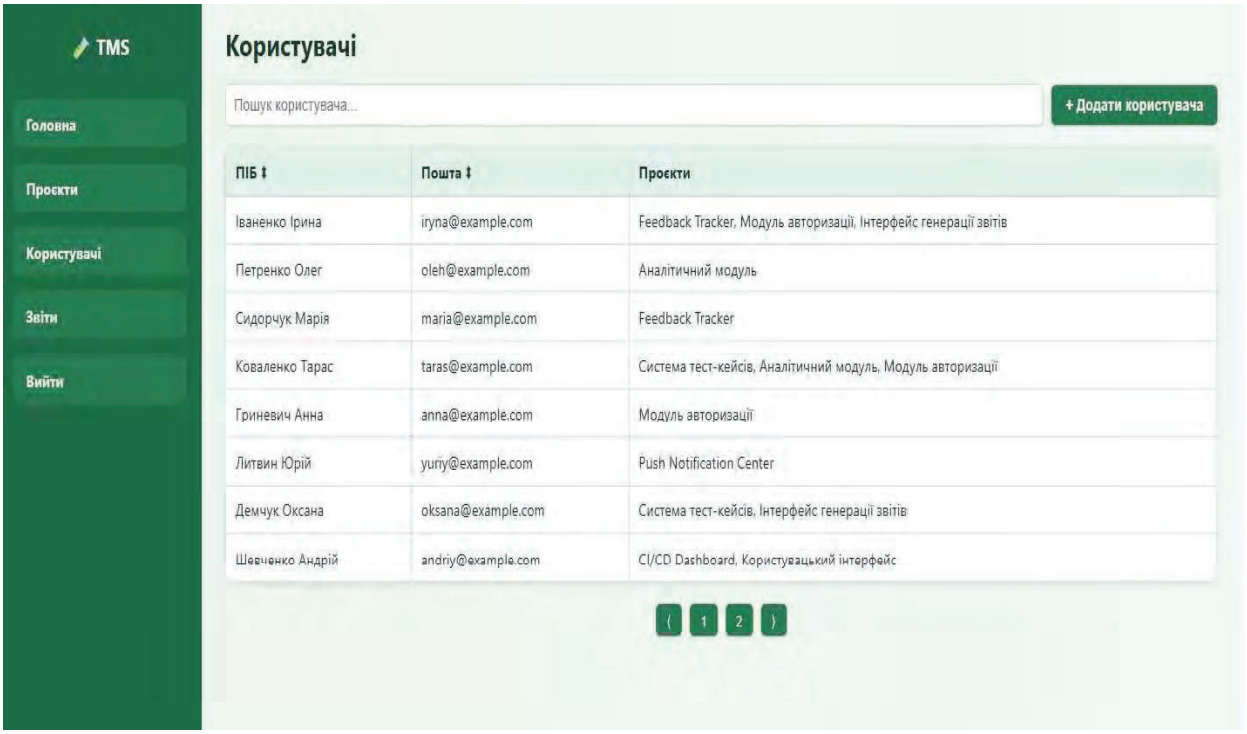

Рисунок 9.5 – Екранна форма сторінки Користувачі

Сторінка «Користувачі» призначена для перегляду всіх зареєстрованих у системі облікових записів та управління ними. У верхній частині розміщено поле пошуку, яке дозволяє фільтрувати користувачів за прізвищем або ім’ям, що полегшує навігацію при великій кількості записів. Поруч розташована кнопка «Додати користувача», яка відкриває форму для створення нового облікового запису відображену на рисунку 9.6.

Основна частина сторінки представлена у вигляді табличного подання, яке містить три основні колонки: повне ім’я користувача, електронну пошту та список проєктів, у яких він бере участь. Колонки підтримують сортування за зростанням або спаданням, що дозволяє впорядковувати дані залежно від потреб користувача.

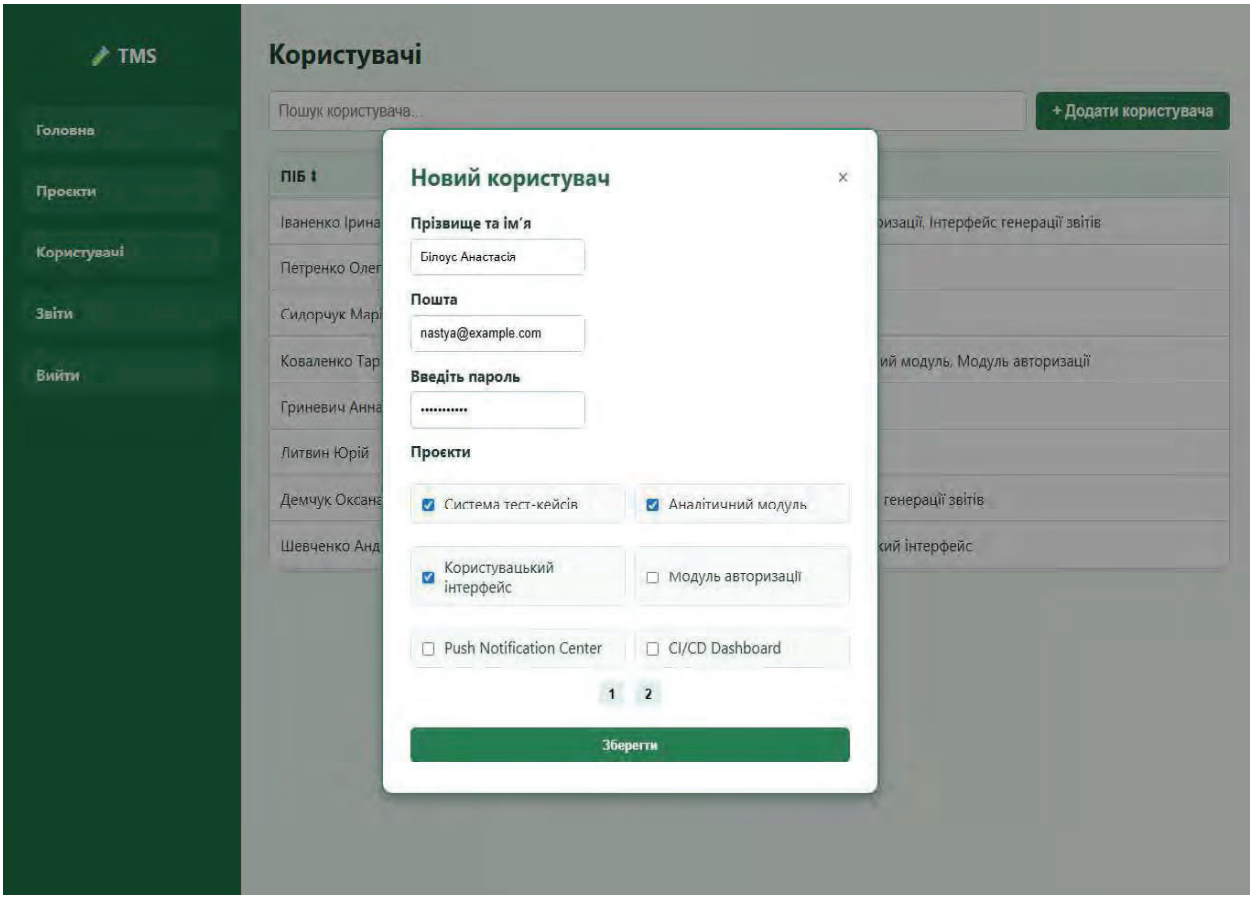

Рисунок 9.6 – Екранна форма додавання користувача

Після натискання кнопки «Додати користувача» на сторінці управління обліковими записами відкривається модальне вікно, призначене для введення даних про нового користувача системи. У цьому вікні менеджер має змогу вказати необхідну інформацію для реєстрації нового облікового запису. Форма складається з чотирьох основних полів: «Прізвище та ім’я», «Пошта», «Пароль» та «Проєкти».

Поле для проєктів дозволяє одразу призначити нового користувача до одного або кількох проєктів, що значно пришвидшує процес первинного налаштування прав доступу. Всі дані, введені у форму, передаються на сервер після натискання кнопки «Зберегти», яка розташована в нижній частині вікна , також після натискання кнопки збереження нвому користувачу на вказану

пошту надходить лист з даними для входу в систему. Успішне створення облікового запису призводить до автоматичного оновлення таблиці користувачів на головній сторінці відповідного розділу. Всі пункти сторінки таблиці всіх користувачів є клікабельними і після натискання на них відкривається сторінка редагування користувачів інтерфейс якої зображено на рисунку 9.7

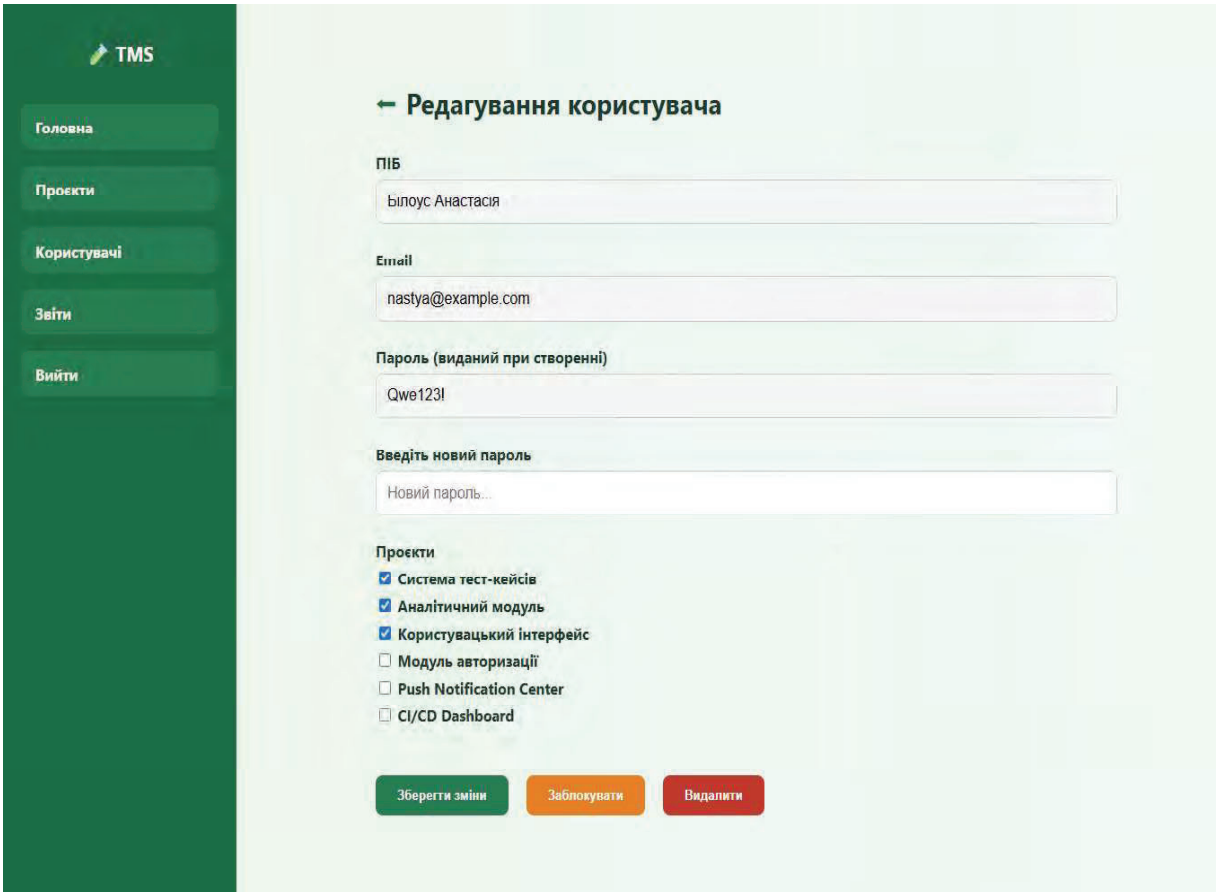

Рисунок 9.7 – Екранна форма редагування користувача

Основна форма представлена полями для перегляду та оновлення персональних даних обраного користувача. Відображаються прізвище, ім’я та по батькові, електронна адреса, а також пароль, виданий при первинному створенні облікового запису. Нижче розташоване поле для введення нового

пароля, що дозволяє за потреби швидко оновити облікові дані користувача без повторної реєстрації.

Окремим блоком представлено список прикріплених проєктів, у яких бере участь відповідний користувач. У нижній частині форми розміщено три функціональні кнопки: «Зберегти зміни» потрібна для підтвердження редагування а «Заблокувати» для тимчасового призупинення доступу користувача до системи без втрати даних, модальне вікно блокування користувача зображене на рисунку 9.8. Кнопка «Видалити» потрібна для повного видалення облікового запису з бази даних. Усі ці дії передбачають серверну обробку та фіксацію змін у базі, що забезпечує безпеку та відстежуваність усіх модифікацій.

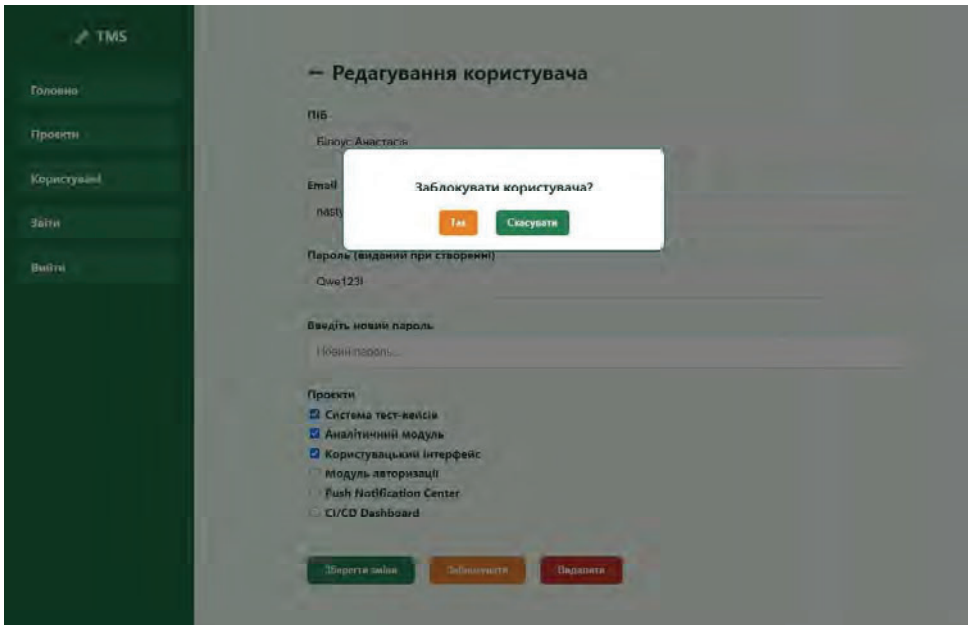

Рисунок 9.8 – Екранна форма модального вікна блокування користувача

У разі натискання кнопки «Заблокувати» на сторінці редагування користувача, в інтерфейсі з’являється модальне вікно підтвердження дії. Це вікно слугує механізмом додаткової перевірки намірів менеджера, що

дозволяє запобігти випадковому блокуванню активного облікового запису. У центрі вікна відображається повідомлення з прямим запитом «Заблокувати користувача?», під яким розміщено дві кнопки це «Так» та «Скасувати». Кнопка «Так» підтверджує дію, після чого користувач позначається як неактивний у системі, що унеможливлює його авторизацію до моменту розблокування. Кнопка «Скасувати» закриває вікно без змін у статусі облікового запису. Після натискання на кнопку видалення користувача ми бачимо модальне вікно яке зображено на рисунку 9.9

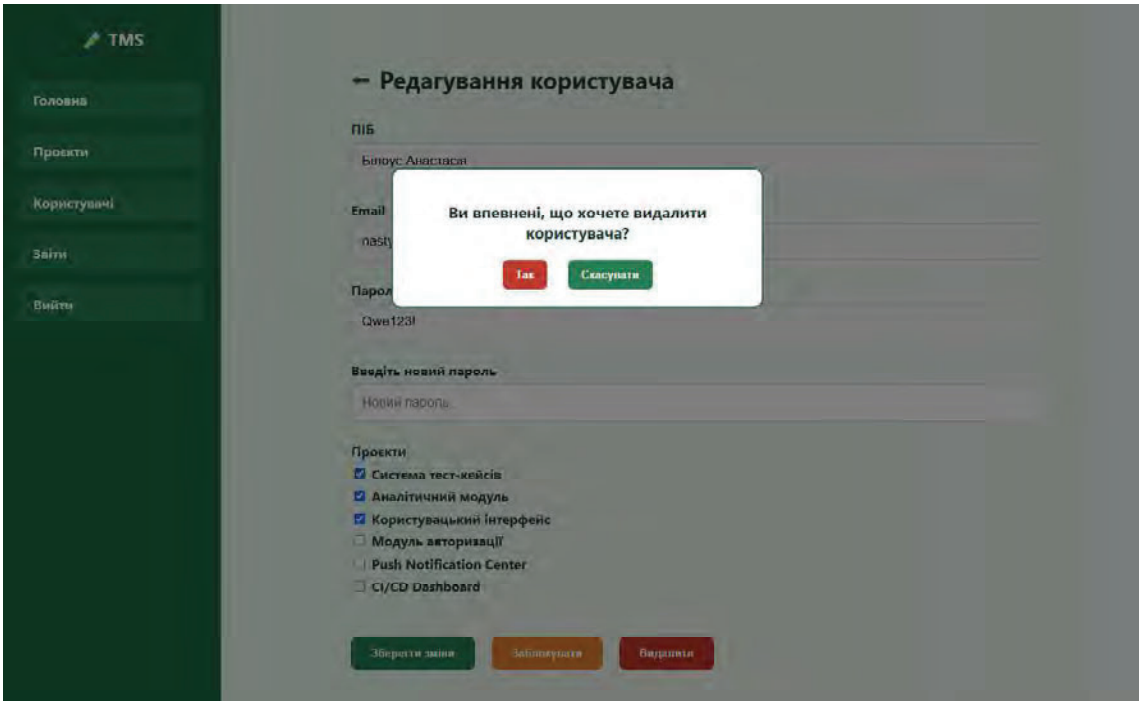

Рисунок 9.9 – Екранна форма модального вікна видалення користувача

Це вікно є обов’язковим етапом перевірки наміру менеджера, що запобігає випадковому або необдуманому видаленню облікового запису з системи. У центрі модального діалогу розміщено чітке формулювання запиту: «Ви впевнені, що хочете видалити користувача?», під яким знаходяться дві кнопки для прийняття рішення «Так» для підтвердження та «Скасувати» для

відмови від дії. У разі підтвердження запиту обліковий запис видаляється з бази даних, а користувача виключають із усіх пов’язаних проєктів. Скасування операції просто закриває модальне вікно без змін у системі.

У менеджера також є можливість генерації звітів функціонал та графічна частина повністю ідентичні до того що має тестувальний тому генеріція звітів буде преставлена далі на прикладі акауна тестувальника.

Після розгляду функціональності, доступної менеджеру, доцільно перейти до опису можливостей, які реалізовані для тестувальника, що безпосередньо взаємодіє з тестовою документацією, виконує тести та фіксує результати. У межах системи тестувальники авторизуються у такий самий спосіб, як і менеджери: через форму входу з використанням електронної пошти та пароля. Після успішної авторизації користувач бачить Головну сторінку зображену на рисунку 9.10.

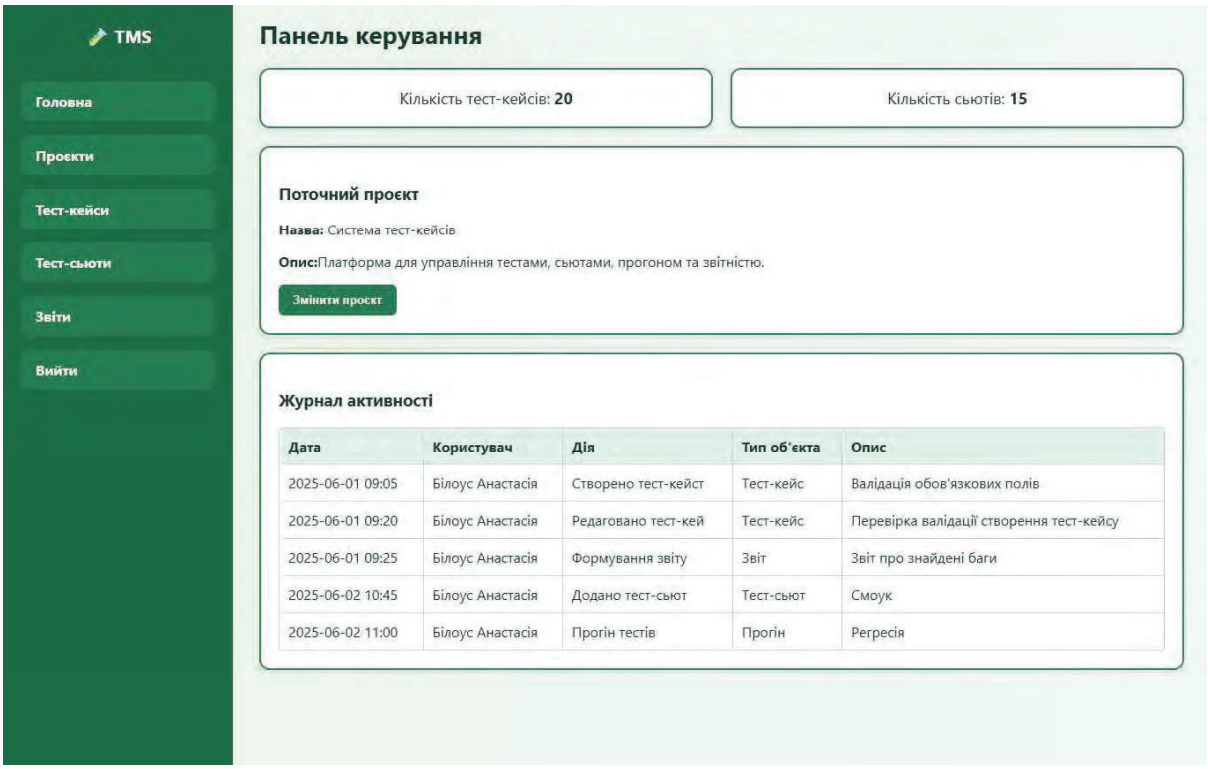

Рисунок 9.10 – Екранна форма головної сторінки

У верхній частині головної сторінки розташовані два блоки з короткою статистичною інформацією про кількість тест-кейсів та кількість тест-сьютів. Нижче представлений розділ поточного проєкту, де відображається його назва та опис, що характеризує систему. В нижній частині сторінки розташовано журнал активності користувача, представлений у вигляді таблиці, у ньому фіксуються усі ключові дії, виконані користувачем протягом місяця. Кожна подія має позначку з датою, часом, назвою дії та пов’язаним об'єктом, що дозволяє швидко оцінити історію активності та взаємодії з системою. Зліва знаходиться вертикальне меню для швидкого переходу між розділами: головна, проєкти, тест-кейси, тест-сьюти, звіти та вихід із системи. Також наявна кнопка «Змінити проєкт» яка після натискання перенаправляє на сторінку «Проєкти» зображену на рисунку 9.11.

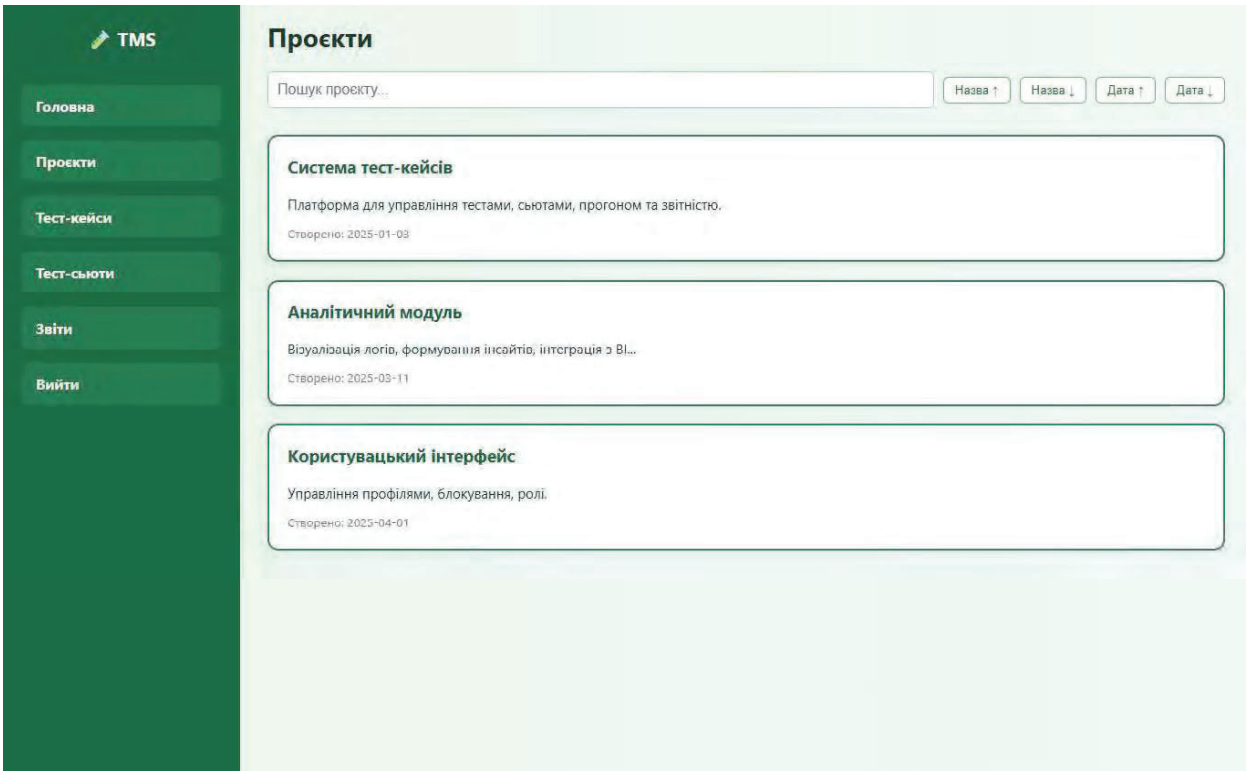

Рисунок 9.11 – Екранна форма сторінки «Проєкти»

На сторінці «Проєкти» користувачеві представлено список усіх наявних проєктів у системі у вигляді окремих карток. Кожна картка містить назву проєкту, короткий опис функціональності та дату створення. У верхній частині сторінки розміщене поле пошуку, яке дає змогу швидко знайти потрібний проєкт за ключовим словом. Поруч із полем пошуку доступні кнопки сортування за назвою або датою у зростаючому чи спадному порядку. В загальному списку проєктів користувач може натиснути на картку проєкту після чого відкриється сторінка деталей проєкту екранна форма такої сторінки зображена на рисунку 9.12.

Рисунок 9.12 – Екранна форма деталей проєкту

На сторінці деталей обраного проєкту відображається назва проєкту, його опис, автор, дата створенні та оновлення. Користувач може лише зробити обраний проєкт поточним і тоді всі відповідні тест-кейси і тест-сьюти будуть відображатись в його системі. Перейдімо до сторінки з тест-кейсами натиснувши на кнопку «Тест-кейси» в навігаційному меню, зовнішній вигляд сторінки зображено на рисунку 9.13

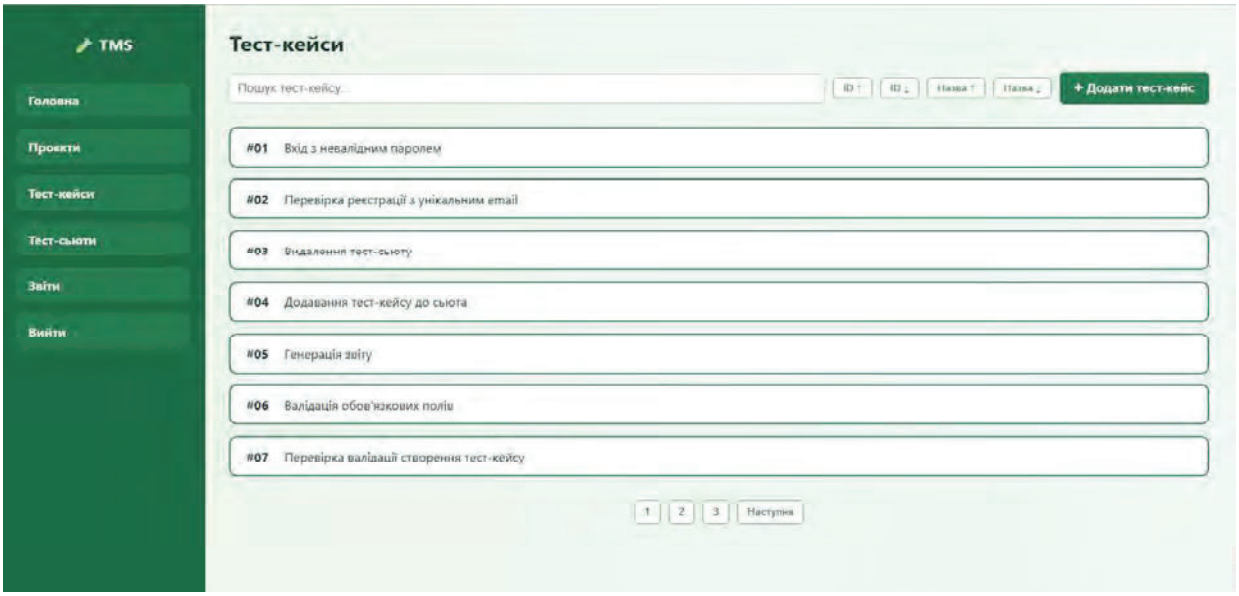

Рисунок 9.13 – Екранна форма сторінки Тест-кеси

На сторінці тест-кейсів користувач має змогу переглядати, шукати, сортувати та додавати нові тест-кейси. У верхній частині розташоване поле пошуку, яке дозволяє швидко знайти потрібний тест-кейс за ключовими словами. Поряд із полем пошуку знаходяться кнопки сортування, які дозволяють впорядкувати список за зростанням або спаданням ID або за алфавітом назви. Також реалізована пагінація яка спростить користування сервісом за великих обємів даних, бо користувач з легкістю зможе переміщатись на інші сторінки з тест-кейсами. Всі тест-кейси клікабельні і після натискання на назву тест кейсу відкривається сторінка редагування тест-кейсу, цей функціонал буде розглянуто пізніше. Справа розміщена кнопка «Додати тест-кейс», яка відкриває форму для створення нового тест-кейсу. Натиснемо на кнопку додавання нового тест-кейсу і бачимо сторінку створення нового тест-кейсу яка зображена на рисунку 9.14.

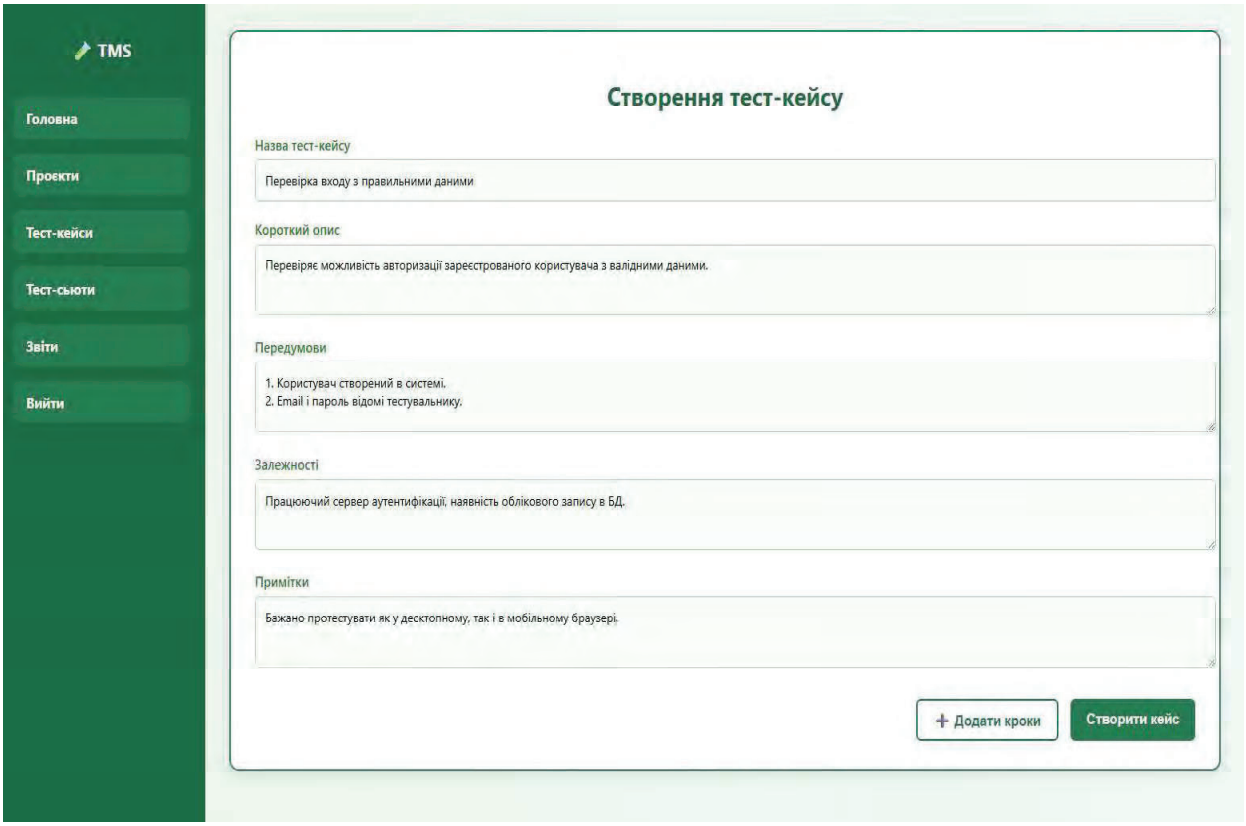

Рисунок 9.14 – Екранна форма сторінки Створення тест-кесу

На сторінці створення тест-кейсу користувач має змогу заповнити основні поля, необхідні для формування повноцінного тестового сценарію. У основній частині форми знаходиться поле для введення назви тест-кейсу, короткого опису, необхідних умови, які мають бути виконані до початку тестування, опис технічних або програмних компонентів, що мають бути доступні для коректного виконання тесту, також можна залишити додаткові вказівки щодо виконання тест-кейсу. У нижній частині форми доступні кнопки «Додати кроки» для переходу до покрокового опису тест-кейсу та «Створити кейс» для завершення створення й збереження введених даних. Після натискання на кнопку додавання кроків користувач бачить сторінку з кроками відтворення тест-кейсу зображену на рисунку 9.15.

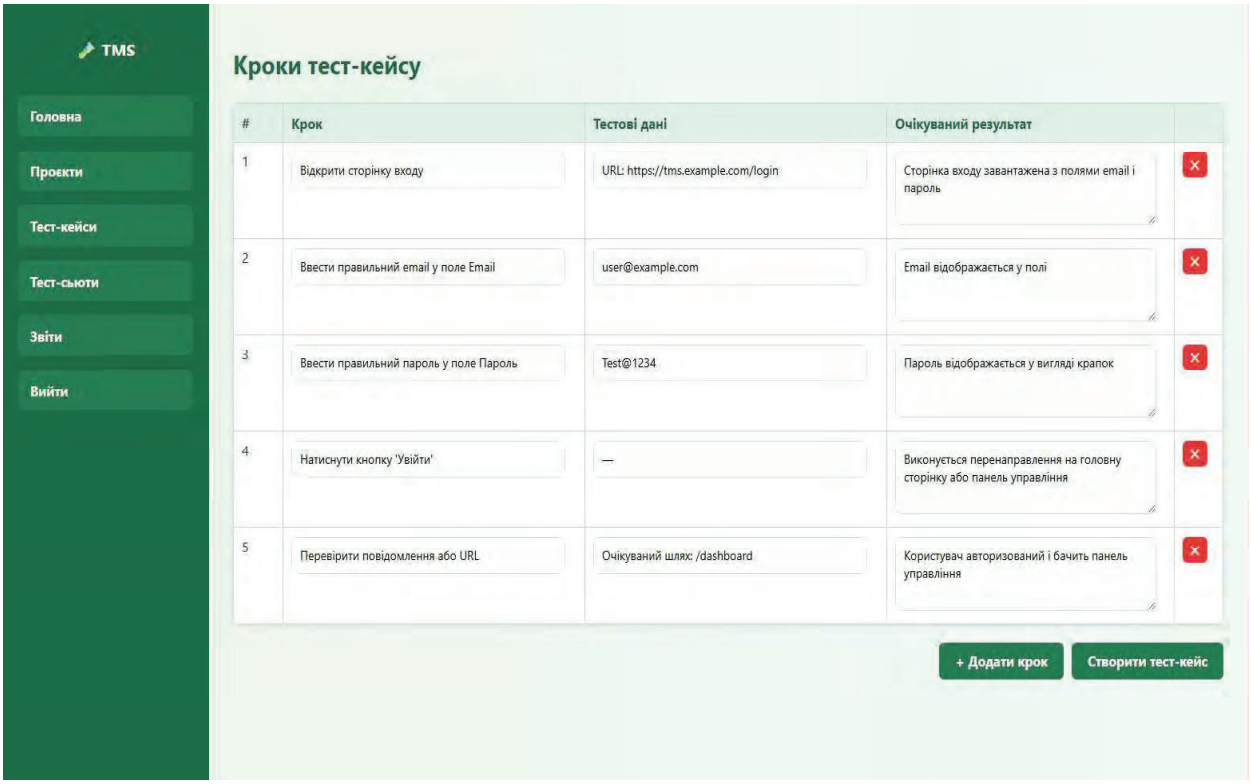

Рисунок 9.15 – Екранна форма додавання кроків до тест-кейсу

У таблиці відображено послідовні пронумеровані кроки для виконання тест-кейсу. Кожен рядок містить три ключові поля: опис дії (кроку), відповідні тестові дані та очікуваний результат. Інтерфейс дозволяє редагувати або видаляти окремі кроки за допомогою відповідних кнопок з правого боку кожного рядка. Внизу розташовано дві основні дії — «+ Додати крок» для внесення додаткових кроків у сценарій та «Створити тест-кейс» для збереження всієї послідовності як єдиного логічного тесту. Після натискання на кнопку додавання кроків користувач бачить новий пронумерований крок з пустими полями. На сторінці зі всіма тест-кейсати натиснемо на картку кейсу і побачимо сторінку редагування тест-кейсу інтерфейс якої зображено на рисунку 9.16.

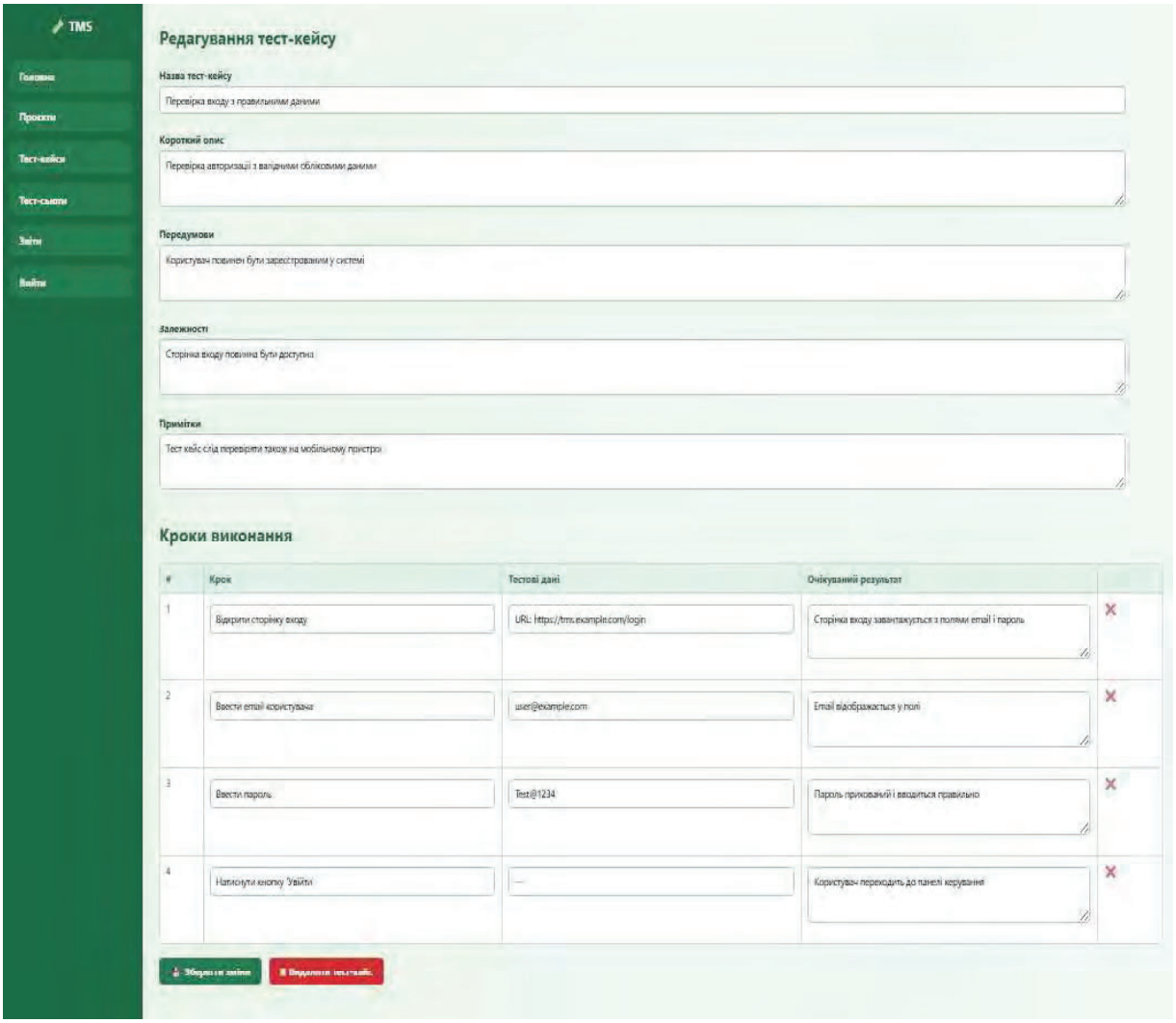

Рисунок 9.16 – Екранна форма редагування тест-кейсу

На сторінці редагування тест-кейсу користувач має змогу змінити як загальні параметри тесту, так і конкретні кроки його виконання. У верхній частині форми розміщені поля для редагування назви тест-кейсу, короткого опису, передумов, залежностей та приміток. Нижче представлена таблиця з кроками виконання, яка дає змогу редагувати кожен окремий етап: його опис, тестові дані та очікуваний результат. Для кожного кроку передбачено можливість редагування або видалення. Завершити редагування можна за допомогою кнопки «Зберегти зміни», а якщо кейс більше неактуальний —

натиснути «Видалити тест-кейс». Перейдемо на сторінку тест-сютів натиснувши кнопку Тест-сьюти в навігаційній панелі і побачимо сторінку зі всіма створеними тест-сьютами зображену на рисунку 9.17.

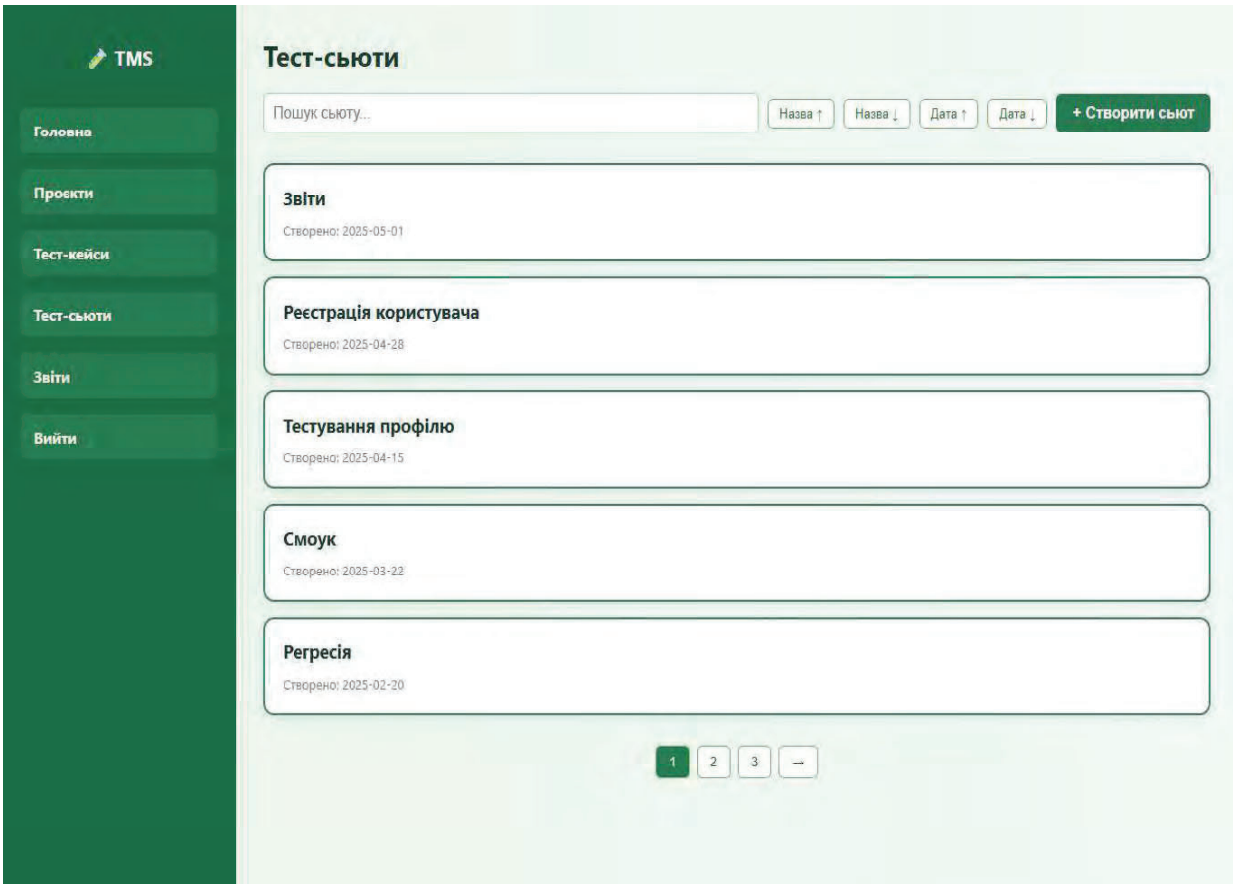

Рисунок 9.17 – Екранна форма сторінки Тест-сьюти

Користувачу представлено перелік усіх доступних сьютів, які згруповані у вигляді карток з назвою та датою створення. У верхній частині інтерфейсу розміщено поле пошуку, що дозволяє швидко знайти потрібний сьют за назвою, а також кнопки сортування за назвою та датою у зростаючому або спадному порядку. Кожна картка сьюта містить назву і вказану дату створення, що дозволяє легко відстежувати хронологію створення сьютів. В нижній частині сторінки реалізована зручна пагінація,

що дозволяє перемикатися між сторінками при великій кількості сьютів. У правому верхньому куті розміщено кнопку «+ Створити сьют», яка відкриває форму для додавання нового сьюту яку відображено на рисунку 9.18.

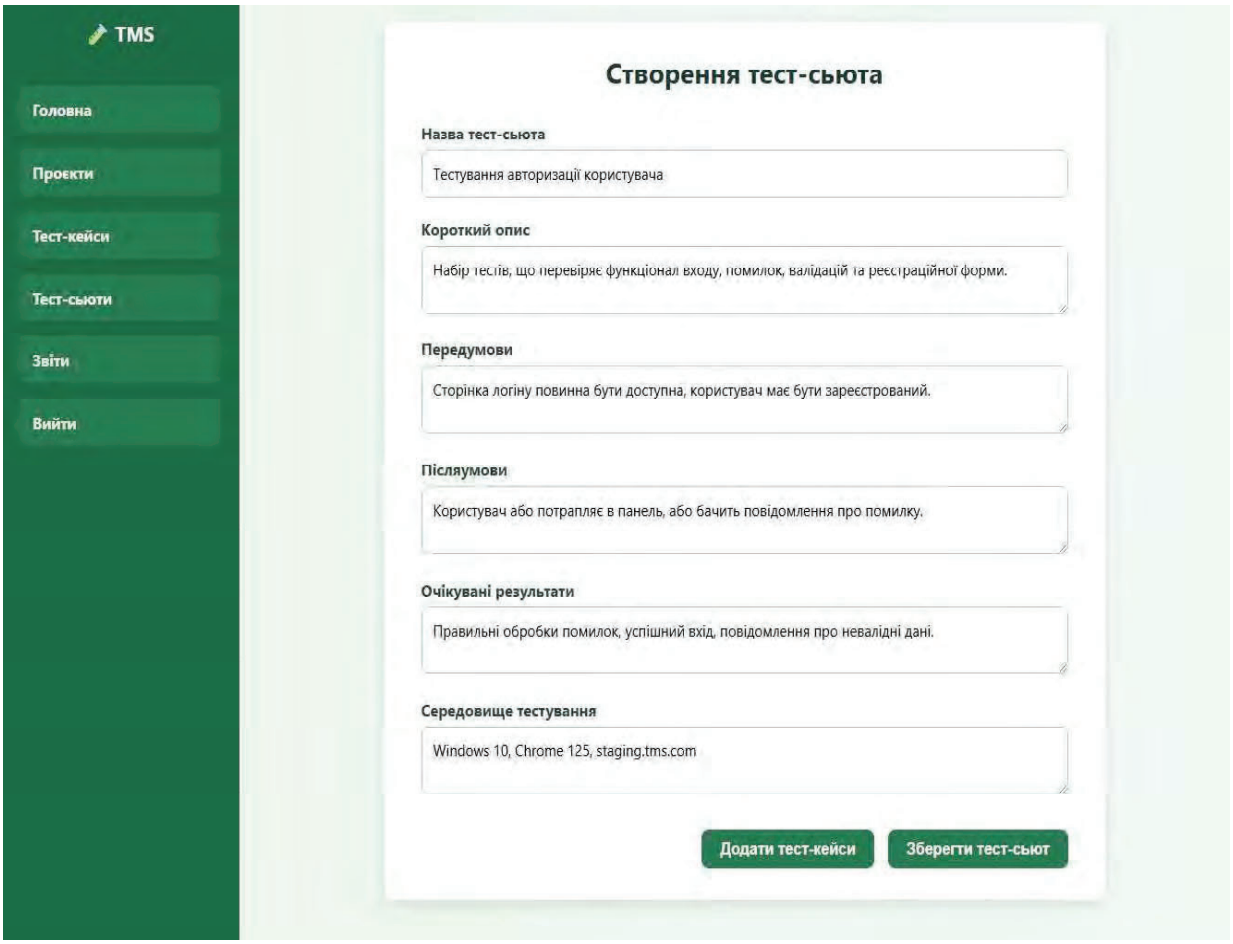

Рисунок 9.18 – Екранна форма створення тест-сьюту

На сторінці створення тест-сьюта користувач має змогу задати ключові параметри, необхідні для формування повноцінного набору тестів.Користувач вписує назву сьюта, короткий опис його призначення, передумови виконання, післяумови, очікувані результати та середовище тестування. Завершується інтерфейс двома кнопками: «Додати тест-кейси» для прикріплення до сьюта необхідних тестів та «Зберегти тест-сьют» для

фіксації всіх внесених даних. Натиснемо на кнопку додавання тест-кейсів і побачимо модальне вікно вміст якого зображено на рисунку 9.19.

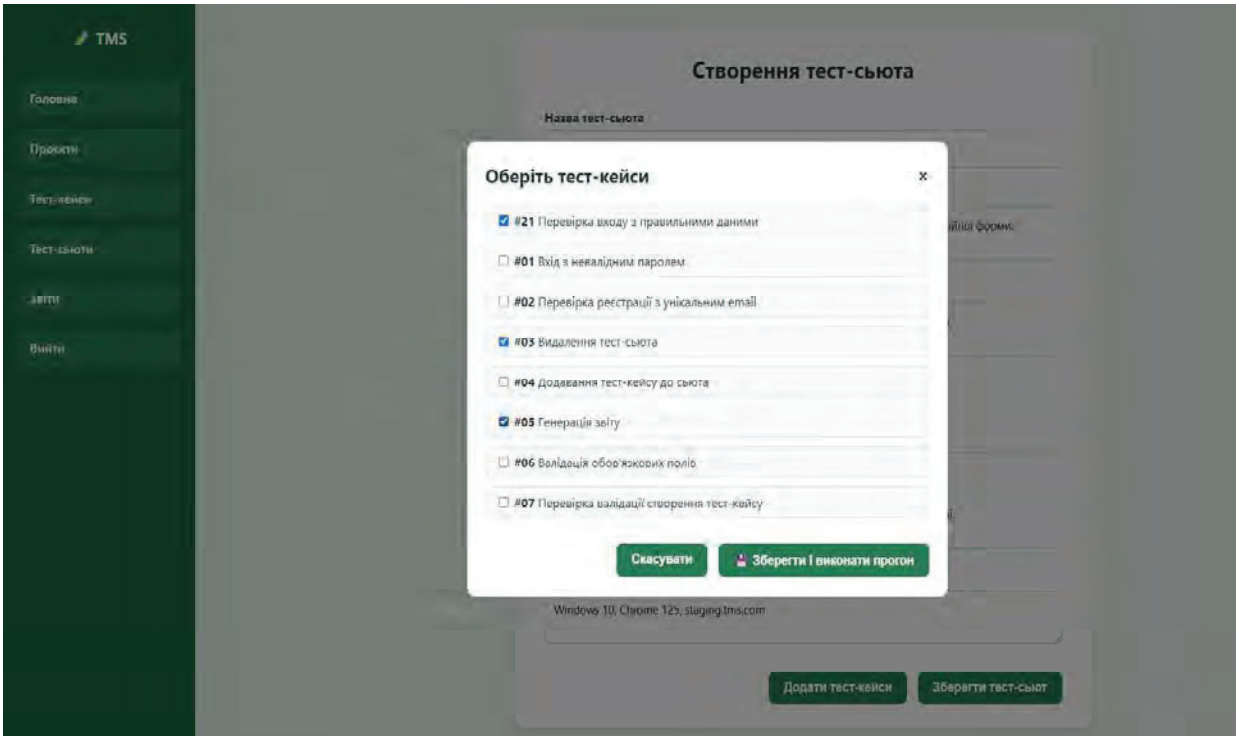

Рисунок 9.19 – Екранна форма додавання тест-кейсів до тест-сьюту

У центрі модального вікна представлено список доступних тест-кейсів, кожен із яких супроводжується чек боксом зліва, що дозволяє обирати декілька кейсів одночасно. В нижній частині вікна розташовані дві кнопки: «Скасувати», яка закриває вікно без збереження змін, та «Зберегти та виконати прогон», що фіксує вибрані кейси та запускає тестовий прогін на основі новоствореного сьюта. Збережемо зміни і почнемо прогін після чого побачимо сторінку прогону вміс якої відображено на рисунку 9.20.

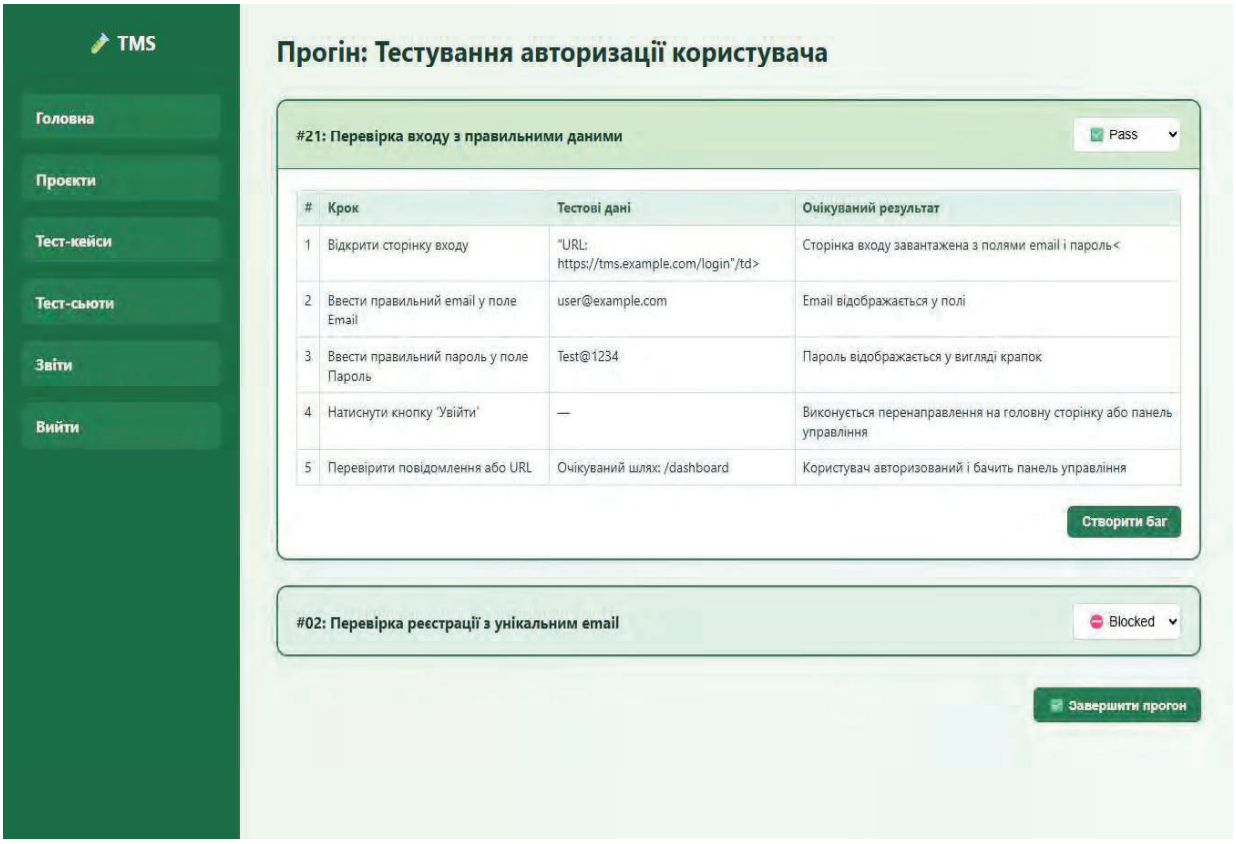

Рисунок 9.20 – Екранна форма виконання прогону

У рамках прогону виконуються окремі тест-кейси, кожен з яких має свій унікальний ідентифікатор, назву, перелік кроків виконання, тестові дані та очікувані результати. Користувач має поставити статус виконання для кожного тест кейсу і також може додати баг до відповідного тест-кейсу. Завершення тестового прогону здійснюється через кнопку «Завершити прогон» у нижній частині екрана. Після натискання на кнопку Створити баг користувач побачить модальне вікно для опису і створення багу зображене на рисунку 9.21.

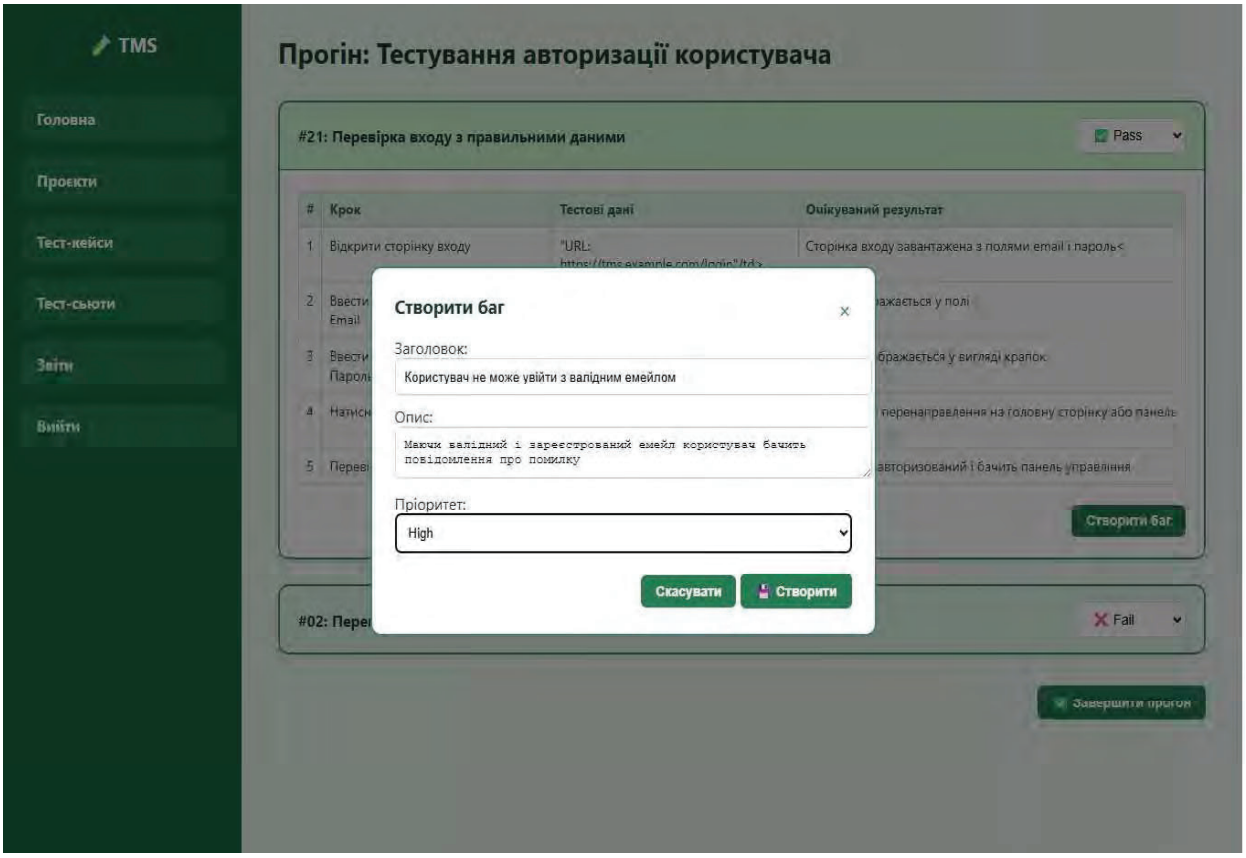

Рисунок 9.21 – Екранна форма додавання багу

На зображенні представлено процес створення бага під час прогону тестів у модулі «Тестування авторизації користувача». Тестувальник може ввести назву багу та опис для конкретизації проблеми та обрати приорітет багу. Після введення всіх даних можемо зберегти зміни натиснувши на кнопку створення або скасувати додавання багу. Створення багів саме під час прогону тест-кейсів допомагає формувати інформаційну базу про стан тестового покриття на проекті та на скільки якісно ведеться розробка. На сторінці звітів навіть є окремий вид звіту який генерує звіт якраз про ці баг створені під час прогонів. Завершимо прогон і перейдемо сторінки Звітів за допомогою панелі навігації, сторінка генерації звітів зображена на рисунку 9.22.

Рисунок 9.22 – Екранна форма сторінки створення звіту активності користувача

У верхній частині вікна користувачеві пропонується обрати тип звіту з чотирьох доступних варіантів: «Активність користувача», «Знайдені баги», «Результати тестового прогона» та «Результати тестування». Поточним активним варіантом є «Активність користувача». Нижче розташовано поля для налаштування параметрів звіту. Перше поле це дата, яка задається вручну, друге поле це тип активності він дозволяє обрати конкретний вид активності або залишити значення «Усі» для включення повної картини дій користувача. Після внесення параметрів користувач натискає зелену кнопку «Створити звіт» для генерації документа. У нижній частині інтерфейсу передбачено блок «Попередні звіти», в якому відображаються згенеровані раніше документи. На рисунку 9.23 зображено фрагмент створеного звіту про активність користувача.

ЗВІТ ПРО АКТИВНІСТЬ КОРИСТУВАЧА №___

Звіт сформовано 03.06.2025

за 28.05.2025 р.

на основі даних проєкту TMS за користувачем Анастасія

звіт сформовано автоматично на підставі журналу активності

ID	Дата	Час	Дія	Тип об'єкта	ID об'єкта	Опис
1	28.05.2025	08:45	Login	User	U001	Успішний вхід до системи
2	28.05.2025	09:10	Create	TestCase	TC101	Створено тест-кейс для реєстрації
3	28.05.2025	09:25	Update	TestCase	TC102	Оновлено тест-кейс авторизації
4	28.05.2025	09:40	Create	TestRun	TR305	Запущено прогін для регрес-тестування
5	28.05.2025	10:00	Execute	TestRun	TR305	Успішно виконано прогін
6	28.05.2025	10:20	Create	Bug	BUG009	Додано баг: некоректна валідація email
7	28.05.2025	10:45	Create	Report	REP401	Згенеровано звіт про результати
8	28.05.2025	11:00	Generate	Report	REP999	Автоматична генерація звіту активності

Анастасія _____

(підпис)

Рисунок 9.23 – Фрагмент звіту про активність користувача

Звіт відображає детальні дані що стосуються активності користувача за певний день. Вказано дату , час , дію яка виконувалась користувачем в той момент Далі оберемо звіт про найдені баги, поля налаштування звіту зображені на рисунку 9.24.

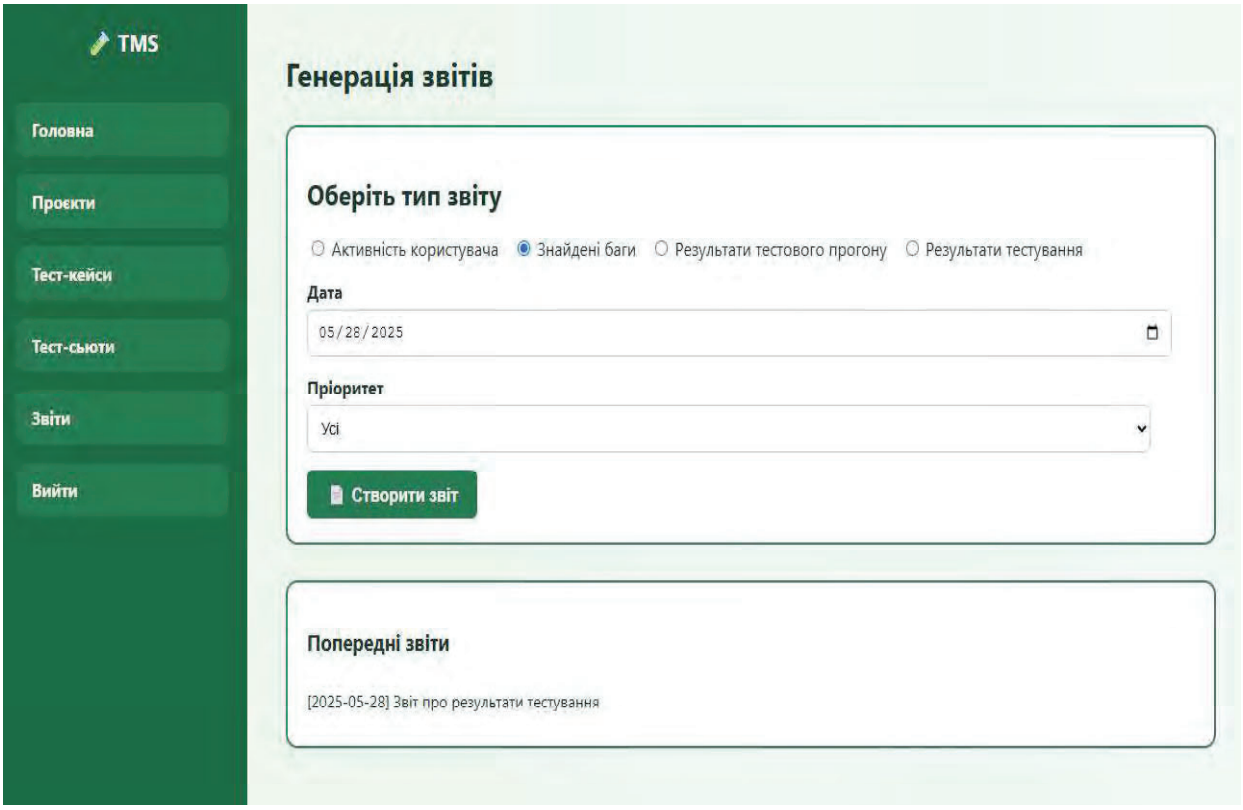

Рисунок 9.24 – Екранна форма сторінки створення звіту про знайдені баги

Обрано звіт «Знайдені баги», що дозволяє створити звіт, зосереджений на зафіксованих помилках у системі. Під цією опцією розташоване поле для вибору дати та випадаючий список для встановлення пріоритету багів. Нижче розташована кнопка «Створити звіт», яка ініціює процес формування відповідного документа. Користувач отримає детальний звіт про знайдені баги за певний період та певними приорітетами обраними під час налаштування цього звіту також звіт матиме додаткові поля такі як назва багу та автор цього багу для деталізації звіту. Фрагмент згенерованого звіту про знайдені баги з певним пріоритетом надано на рисунку 9.25.

ЗВІТ ПРО ЗНАЙДЕНІ БАГИ №___

Звіт сформовано 04.06.2025

на 28.05.2025 р.

на основі даних проєкту TMS за користувачем Анастасія

звіт сформовано автоматично на підставі зафіксованих багів у системі

№	Дата	Час	Пріоритет	Назва бага	Автор
1	2025-05-28	11:23	High	Email validation error	Білоус Анастасія
2	2025-05-28	14:05	Medium	Login redirect issue	Білоус Анастасія
3	2025-05-28	09:40	High	Save button inactive	Білоус Анастасія
4	2025-05-28	13:15	Critical	Report generation failure	Білоус Анастасія
5	2025-05-28	16:30	Low	Status display bug	Білоус Анастасія

Анастасія _____

(підпис)

Рисунок 9.25 – Фрагмент звіту про знайдені баги

Сформований звіт про знайдені баги містить в собі всю детальну інформацію про створені баги за певний період часу та певний приорітет в нашому випадку ми обрали всі можливі приорітети. Звіт деталізований датою,

часом та користувачем який створив цей баг. Тепер оберемо звіт про результати тестового прогону інтерфейс сторінки відображено на рисунку 9.26.

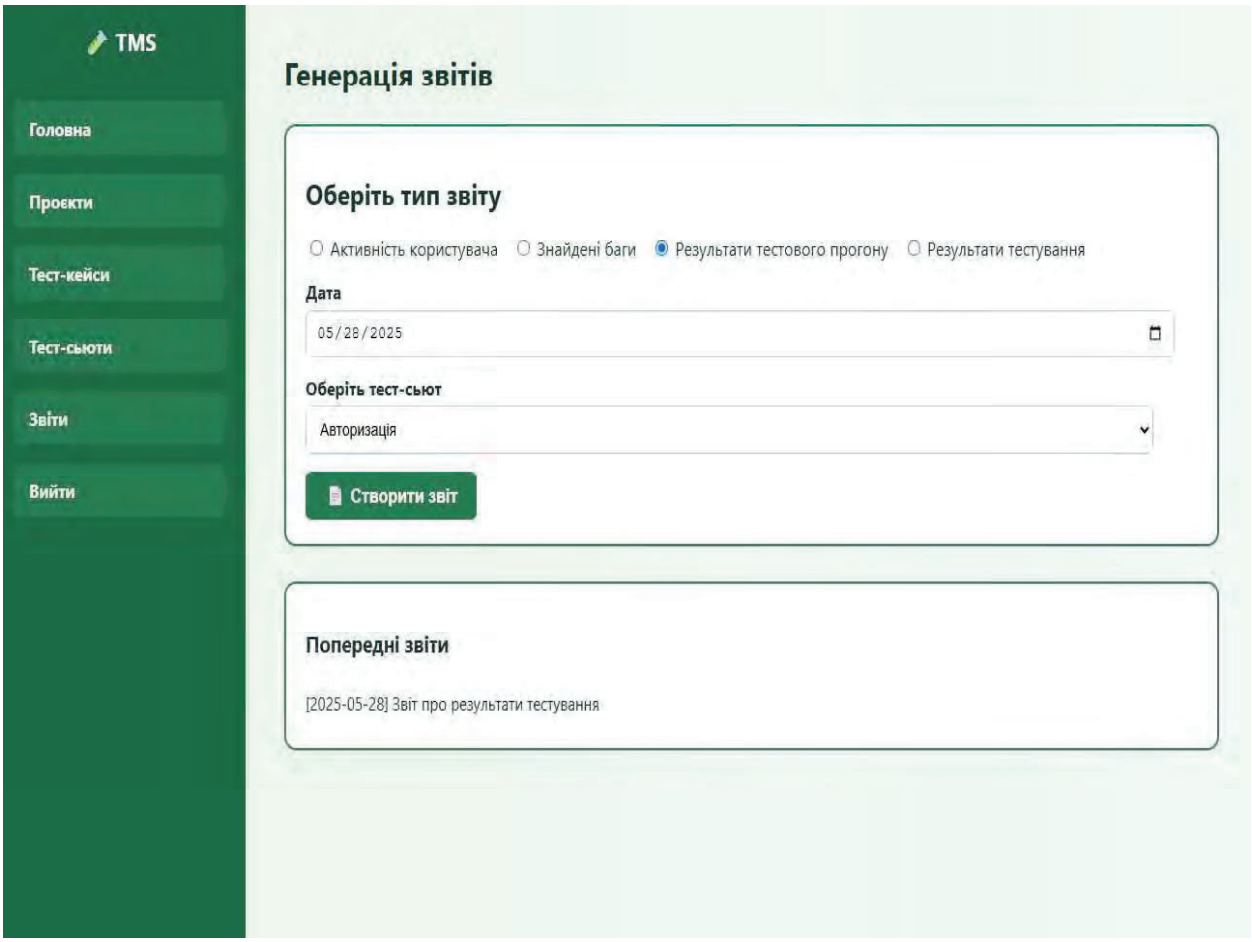

Рисунок 9.26 – Екранна форма сторінки створення звіту про результати тестового прогону

Обрано тип звіту «Результати тестового прогону», користувач може обрати дату прогону та обрати тест сьют в межах якого виконували прогон тест- кейсів. Звіт матиме в собі дані про статус проходження всіх тест-кейсів в межах обраного тест-сьюту. Кнопка «Створити звіт» запускає процес формування документа. Фрагмент сформованого звіту надано на рисунку 9.27.

ЗВІТ ПРО РЕЗУЛЬТАТИ ПРОГОНУ №___

Звіт сформовано 04.06.2025

на 28.05.2025 р.

на основі даних проєкту TMS за користувачем Анастасія

звіт сформовано автоматично на підставі виконаного тестового прогона

№	Тест-сьют	Тест-кейс	Статус	Назва кейсу
1	Авторизація	TC-01	Pass	Вхід з невалідним паролем
2	Авторизація	TC-02	Fail	Перевірка реєстрації з унікальним email
3	Авторизація	TC-03	Pass	Видалення тест-сьюту
4	Авторизація	TC-04	Pass	Додавання тест-кейсу до сьюта
5	Авторизація	TC-05	Skipped	Генерація звіту
6	Авторизація	TC-06	Fail	Валідація обов'язкових полів
7	Авторизація	TC-07	Blocked	Перевірка валідації створення тест-кейсу

Анастасія _____

(підпис)

Рисунок 9.27 – Фрагмент звіту про результати прогону

Перейдемо до звіту про результати тестування, дані для генерації обраного звіту зображено на рисунку 9.28.

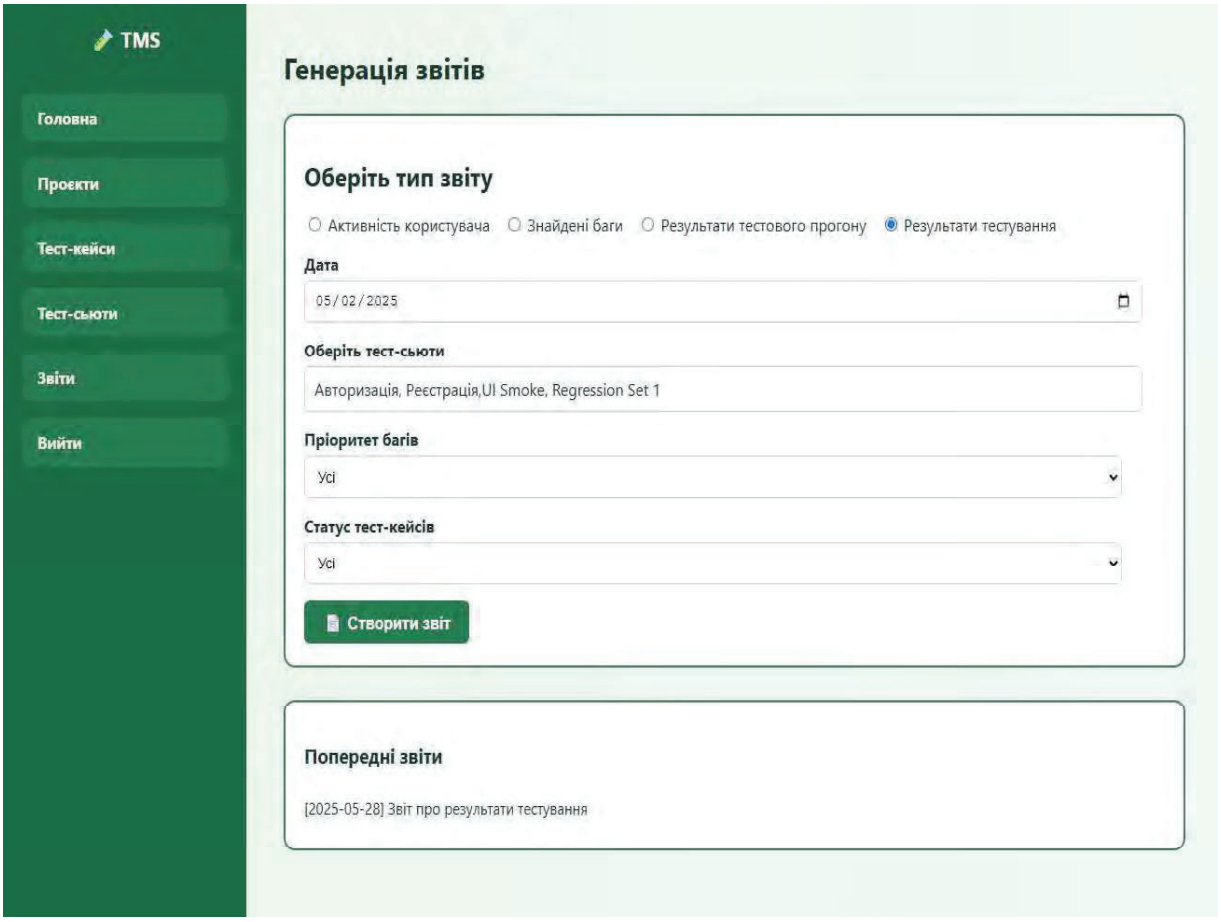

Рисунок 9.28 – Екранна форма сторінки створення звіту про результати тестування

Далі зображено налаштування даних для звіту про результати. Користувач може налаштувати всі важлтиві поля включаючи дату за яку він хочить бачити ці дані, також може обрати тест сьюти для яких ми хочемо знати цю інформацію ще є можливість обрати конкретні або всі статуси тест-кейсів та приорітет багів які були знайдені під час виконання тестового прогону. Обравши всі потрібні дані натиснемо кнопку «Створити звіт» яка дозволяє згенерувати детальний звіт за налаштованими полями, приклад звіту зображено на рисунку 9.29.

Звіт про результати тестування № ___

Дата створення звіту: 2025-05-28

Користувач: Анастасія

Тип звіту: Результати тестування

За період : 2025-05-02

Опис: Звіт містить результати тестування, що охоплюють різні тест-сьюти, їхній пріоритет та статуси виконання тест-кейсів.

Тест-сьют	Кількість багів	Пріоритет багів	Кількість кейсів	Статус
Авторизація	3	2-Low 1- Medium	5	1-Blocked 2-Pass 2-Skipped
Реєстрація	4	Medium	4	Fail
UI Smoke	1	Critical	6	3-Fail 3-Blocked
Regression Set 1	1	Critical	8	Fail

Анастасія _____

(підпис)

Рисунок 9.29 – Фрагмент звіту про результати тестування

Під час розробки інтерфейсу особливу увагу приділено навігації та швидкому доступу до основних функцій. Кожна сторінка містить панель пошуку, сортування та пагінації, що дозволяє зручно працювати навіть із великою кількістю записів. Додаткові дії, зокрема створення нового об’єкта чи зміна статусу, виконуються через модальні вікна, які відкриваються без перезавантаження сторінки, що підвищує швидкодію і забезпечує безперервність взаємодії. Інтерфейс орієнтований на принцип «мінімального

кліку», згідно з яким будь-яка основна дія повинна виконуватись не більше ніж у два-три переходи або натискання.

Дизайн системи реалізовано таким чином, щоб зберігати візуальну ієрархію, чіткість відображення структурних елементів та високу контрастність, усе це сприяє комфортному використанню інтерфейсу навіть протягом тривалого часу. Враховано й перспективу масштабування: завдяки гнучкості верстки та компонентній структурі клієнтської частини, інтерфейс може бути легко доповнений новими модулями або налаштований відповідно до потреб організації.

Інтерфейс сервісу поєднує зручність, логічну організацію та технічну цілісність, дозволяючи користувачам швидко орієнтуватися в системі, зосереджуючись на виконанні своїх основних завдань без необхідності у додаткових інструкціях чи навчанні.

У сервісі використано зелений колір як базовий у комплектуванні головного меню, заголовків розділів, кнопок і декоративних елементів. Такий вибір не є випадковим, бо зелений колір стимулює зосередженість і підвищує продуктивність під час виконання рутинних завдань. Для користувачів, які щодня працюють із тест-кейсами, прогонами, звітами й технічною документацією, це дозволяє знизити втому, підтримати увагу та скоротити час на прийняття рішень.

Зелений колір також має здатність зменшувати візуальну втомлюваність, особливо при роботі, де переважає текстова інформація, таблиці та великі обсяги даних[18]. Зелений колір не є агресивним, не привертає зайвої уваги до другорядних елементів, але водночас дозволяє легко орієнтуватися у функціональній структурі інтерфейсу. Впроваджена колірна схема дозволяє швидко відрізнити інтерактивні елементи без необхідності спеціального маркування

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено IT-сервіс «Створення та управління тест-кейсами», який включає ключові етапи процесу тестування в проєктних командах. У рамках підготовки до розробки було проведено предпроєктне обстеження предметної області, сформовано загальну організаційну модель та обрано об'єкт розробки що являє собою процес керування тестовими кейсами, тестовими прогонами, звітністю та багами.

Була побудована концептуальна модель у вигляді IDEF0-діаграми з декомпозицією першого рівня, яка дозволила чітко окреслити інформаційні потоки, рольові взаємодії та структурні компоненти системи. Враховуючи досвід існуючих інструментів керування тестуванням, проведено функціональне порівняння аналогів і сформульовано переваги розробленого рішення.

Були визначені функціональні та нефункціональні вимоги до системи, зокрема для ролей тестувальника й менеджера, та на основі цього побудовано відповідну UML-діаграму класів, що відображає основні сутності й зв’язки між ними. Архітектура рішення описана у вигляді DFD-моделі та логічної структури даних.

Розробка інтерфейсу була спрямована на зручність взаємодії з користувачем, з урахуванням реальних сценаріїв роботи з тестовою документацією. Було створено інтерфейс для кожного з функціональних модулів: керування проєктами, користувачами, створення та редагування кейсів, запуску тестових прогонів та формування звітів. У процесі роботи особливу увагу приділено питанню модульності коду, доступності, технічної гнучкості рішення та подальшої масштабованості.

Для тестувальників сервіс надає комфортне та структуроване середовище, у якому зручно організовувати всі етапи тестування починаючи з формування тестової документації до фіксації результатів. Інтерфейс дозволяє швидко орієнтуватися в поточному проєкті, бачити актуальні тест-кейси, запускати прогони і відзначати статус виконання кожного сценарію в кілька кліків без необхідності переходів між вкладками чи зовнішніми таблицями. Усі дії виконуються в єдиній системі, що позбавляє від дублювання і втрати інформації.

Окремо реалізований механізм збереження , статусів і прив’язки до багів, що дозволяє фіксувати фактичні результати в момент виконання з мінімальним відволіканням від робочого процесу. Завдяки централізованій структурі даних, кожен тестувальник має постійний доступ до актуальних тестових сценаріїв, без потреби дублювання чи перенесення вручну.

Кваліфікаційна робота була виконана згідно з стандартом ДСТУ та методичними вказівками [19].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SO/IEC/IEEE 29119-1:2013. Software and Systems Engineering — Software Testing — Part 1: Concepts and Definitions. Geneva : ISO, 2013. 48 p.

2. OWASP Foundation. OWASP Application Security Verification Standard 4.0. Baltimore : OWASP, 2019. 264 p.

3. Smith J., Taylor K. Modern Software Testing Techniques: A Practical Guide for Developers and Testers. 1st ed. London: Springer, 2021. 380 p.

4. Comparison and Evaluation of Test Management Tools for Agile Projects // IEEE Access. 2019. Vol. 7. P. 110895–110910. DOI: 10.1109/ACCESS.2019.2937823.

5. Gurock Software. TestRail Test Management Tool. URL: https://www.gurock.com/testrail/tour/ (дата звернення: 09.04.2025).

6. Zephyr Scale Test Management for Jira. Atlassian Marketplace. URL: https://marketplace.atlassian.com/apps/1014680/zephyr-scale-test-management-for-jira (дата звернення: 10.04.2025).

7. Xray Test Management for Jira. Official Documentation. URL: https://docs.getxray.app/ (дата звернення: 10.04.2025).

8. Sommerville I. Engineering Software Products: An Introduction to Modern Software Engineering. Harlow: Pearson, 2021. 320 p.

9. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva : ISO, 2011. 34 p.

10. Visual Paradigm. What is DFD (Data Flow Diagram)? URL: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/ (дата звернення: 17.04.2025).

11. Kendall K. E., Kendall J. E. Systems Analysis and Design. 10th ed.

Boston : Pearson, 2019. 576 p.

12. Hernandez M. J. Database Design for Mere Mortals: 25th Anniversary Edition. 4-th ed. Boston : Addison-Wesley Professional, 2021. 640 p.

13. Microsoft Learn. Design a relational database in Azure SQL Database. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/design-first-database-tutorial (дата звернення: 10.05.2025).

14. 15.Forgács I., Kovács A. Modern Software Testing Techniques: A Practical Guide for Developers and Testers. 1st ed. New York: Apress, 2024. 284 p.

15. Medium. What is Django ORM and How Does It Work? URL: https://medium.com/django-unleashed/what-is-django-orm-and-how-does-it-work-b91531761713 (дата звернення: 09.06.2025)

16. Fontaine D. Mastering PostgreSQL 13. 4th ed. Packt, 2020. 400 p.

17. Goodyear J. Enterprise System Architectures: Building Client Server and Web Based Systems. Boca Raton : CRC Press, 2017. 356 p.

18. InspiringApps. The Psychology of Color in Design. URL: https://www.inspiringapps.com/blog/the-importance-of-color-in-design (дата звернення: 26.05.2025).

19. Методичні вказівки до організації виконання та захисту кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти для студентів спеціальності 122 «Комп'ютерні науки» за освітньою програмою «Інформаційні технології управління». [Електронний ресурс] / Упоряд.: К.Е. Петров, А.В. Міхнова, М.С. Кудрявцева, М.В. Євланов, Т.І. Борисенко. – Електронне видання. – Харків: ХНУРЕ, 2023. – 68 с. – pdf