МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проєктування»

на тему: Web-орієнтована система автоматизації процесу аналізу якості програмного продукту

Здобувачки групи ІТ.м-22 Медведєвої Катерини Сергіївни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Катерина МЕДВЕДЄВА
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доц. Вікторія АНТИПЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проєктування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Медведєвої Катерини Сергіївни
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Web-орієнтована система автоматизації процесу аналізу якості програмного продукту

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи метрики якості тестування та розробки програмного продукту, баг репорт

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проєктування web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту, розробка web-орієнтованої системи, тестування web-орієнтованої системи

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, постановка задачі, задачі дослідження, огляд систем аналізу якості програмних продуктів, результати проведеного аналізу аналогів, функціональні вимоги, засоби реалізації, методи дослідження, структурно-функціональне моделювання, моделювання варіантів використання, розробка бази даних, практична реалізація, демонстрація роботи web-орієнтованої системи, тестування, апробація результатів, висновки.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка специфікації	до 18.09.2023	
2	Розробка web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту	до 14.11.2023	
3	Тестування функціоналу web-орієнтованої системи	до 14.11.2023	
4	Впровадження в дію web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту	до 01.12.2023	
5	Налаштування коректної роботи web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту	до 01.12.2023	

Магістрант _____ Катерина МЕДВЕДЄВА

Керівник роботи _____ к.т.н., доц. Вікторія АНТИПЕНКО

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Web-орієнтована система автоматизації процесу аналізу якості програмного продукту».

Пояснювальна записка складається зі вступу, 4 розділів, висновку, списку використаних джерел із 42 найменувань, 3 додатків. Загальний обсяг роботи – 99 сторінок, у тому числі 50 сторінок основного тексту, 4 сторінки списку використаних джерел, 45 сторінок додатків.

Сучасний світ зараз перебуває під впливом масової цифровізації послуг, де програмне забезпечення використовується майже у всіх сферах життєдіяльності. Тому забезпечення його високої якості стає критично важливим. Це пояснюється підвищеним попитом на надійність, безпеку та ефективність програмних продуктів. Саме тому цьому питанню приділяється така велика увага. Отже, дослідження причин появи дефектів, а також пошук слабких місць програмного продукту є нагальними сьогодні. Зазвичай процес аналізу відбувається вручну, шляхом ідентифікації спільних атрибутів багів. І він триває досить довго, бо треба обробити великий об’єм даних про дефекти. Тому розробка web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту є актуальною.

Метою роботи є автоматизація процесу аналізу якості програмного продукту за рахунок розробки та впровадження відповідної web-орієнтованої системи, використовуючи bug review документацію.

Ключові слова: WEB-ОРІЄНТОВАНА СИСТЕМА, АНАЛІЗ, БАГ, ДЕФЕКТ, БАГ-РЕПОРТ, МЕТРИКИ, АВТОМАТИЗАЦІЯ, PYTHON, PHP, YII2.

ЗМІСТ

АНОТАЦІЯ 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Огляд останніх досліджень і публікацій 9
1.2 Аналіз інформаційних систем 11
2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ 16
2.1 Мета та задачі дослідження 16
2.2 Методи дослідження 17
2.3 Вибір технологій 20
3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ 21
3.1 Діаграми нотації IDEF0 21
3.2 Use Case діаграма 23
3.3 Проєктування моделі бази даних 25
4 РОЗРОБКА WEB-ОРІЄНТОВАНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ 27
4.1 Архітектура web-орієнтованої системи 27
4.2 Реалізація бази даних web-орієнтованої системи 28
4.3 Програмна реалізація web-орієнтованої системи 29
4.4 Демонстрація роботи web-орієнтованої системи 35
4.5 Тестування web-орієнтованої системи 45
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50
ДОДАТОК А 54
ДОДАТОК Б 67
ДОДАТОК В 70

ВСТУП

Актуальність. Забезпечення якості програмного забезпечення (ПЗ) є невід'ємним аспектом підтримки та розвитку сучасних інформаційних технологій (ІТ). Дане питання дійсно є досить актуальним сьогодні. Якість програмного продукту впливає не лише на досвід застосування кінцевого користувача. Вона також має глобальне значення. Наприклад, вплив на комерційний успіх ПЗ на ринку. Ураховуючи це, безумовно процеси аналізу та оцінювання якості програмного забезпечення вимагають постійного вдосконалення та автоматизації сьогодні.

Швидкий розвиток web-технологій та їх інтеграція у майже всі сфери діяльності людини відкриває нові можливості для створення систем автоматизації. Вони є різноманітними. Однак у сучасному суспільстві все більшої популярності набирають саме web-орієнтовані системи. Зокрема завдяки своїй доступності та масштабованості. А, отже, їх доречне використання сприятиме детальному збору, обробці та аналізу даних про якість роботи команд розробки та тестування. У свою чергу це дозволить краще дослідити особливості та специфіку їх діяльності. У результаті, застосування web-орієнтованої системи є оптимальним рішенням для автоматизації процесів оцінки якості програмного продукту.

Для представленного дослідження важливими є дані про дефекти ПЗ. Вони є цінною інформацією для оцінки якості програмного продукту. Процес Bug review, який полягає у зборі та аналізі основних характеристик дефектів, становить невід'ємну його частину. Він виконує важливу роль у виявленні та усуненні проблемних місць у програмному продукті та роботі команди розробки. Втім, існує потреба у подальшій автоматизації цього процесу. Це необхідно з метою зниження ручної праці та підвищення об'єктивності оцінок.

Отже, на основі всього вищесказаного, можна зробити висновок, що розробка web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту є актуальною. Її застосування надасть ряд переваг. Наприклад, це дозволить ефективно інтегрувати процеси оцінювання якості в життєвий цикл розробки ПЗ, а також поліпшити продуктивність та надійність роботи розробників та тестувальників, забезпечуючи при цьому довершеність програмного продукту та підвищення його конкурентоспроможності.

Тема. Web-орієнтована система автоматизації процесу аналізу якості програмного продукту.

Мета. Автоматизація процесу аналізу якості програмного продукту за рахунок розробки та впровадження відповідної web-орієнтованої системи, використовуючи bug review документацію. Застосування запропонованого рішення буде актуальним серед сучасних IT-компаній. Його використання забезпечить підвищення продуктивності та рівня роботи розробників і тестувальників, за рахунок ефективної інтеграції процесів оцінки якості програмного продукту, на основі дослідження bug review документу, у життєвий цикл розробки ПЗ.

Для досягнення мети проєкту необхідно виконати наступні задачі:

– визначити актуальність роботи, дослідити предметну область та цільову аудиторію;

– виконати аналіз існуючих web-орієнтованих систем-аналогів, ідентифікувати їх позитивні та негативні риси;

– визначити функціональні можливості та технології для розробки даної web-орієнтованої системи;

– виконати структурно-функціональне моделювання;

– визначити технології для розробки web-орієнтованої системи;

– реалізувати структуру та функціонал web-орієнтованої системи автоматизації процесу аналізу якості програмного забезпечення;

– провести тестування власної розробки.

Об’єкт дослідження. Процес автоматизації аналізу якості програмного продукту.

Предмет дослідження. Web-орієнтована система автоматизації аналізу якості програмного продукту.

Практична новизна. Удосконалення процесу розробки та тестування. Також покращення якості програмного продукту, завдяки автоматизації аналізу дефектів. Це дозволить визначити слабкі місця в діяльності команди розробки та тестування.

Результати даної роботи були апробовані на науково-практичній конференції АПКН-2023 у Хмельницькому державному університеті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Автоматизація будь-яких процесів є важливою складовою сучасних досліджень. Її застосування є досить актуальним сьогодні. Останні десятиліття суспільство рухається до повної автоматизації тих рутинних активностей, які займають багато часу та потребують ручної роботи.

Дефекти, або баги, у програмному забезпеченні є неминучим явищем. Вони в основному виникають через людський фактор у процесі розробки або тестування. І на те є різні причини. Наприклад, невизначені вимоги, неправильне використання компонентів програми, помилки у коді тощо [1].

Процедура перегляду багів (bug review) є нестандартизованою частиною процесу розробки програмного забезпечення. Однак вона прийняла різноманітні форми в межах різних підходів до аналізу якості ПЗ.

Bug review документація – це формалізований документ, який містить в собі основні характеристики дефектів, їх причину, функціональність, ким знайдено, скільки виправляли цей баг тощо. Усі ці дані мають вплив на формування оцінки якості програмного продукту. Дізнавшись, чому дефект виникає, можна буде здійснити конкретні зміни в діяльності команд розробки чи тестування.

У сучасному світі процес оцінювання якості програмного продукту базується на різноманітних міжнародних стандартах. Однак основою для цього є підхід із використанням метрик. Він включає в себе техніки на базі експертних оцінок. Цей підхід надає безліч можливостей. Особливо для оцінювання характеристик якості програмних продуктів. А саме на ранніх стадіях їхнього проєктування та розробки. У свою чергу це дозволяє прогнозувати рівень якості програмного забезпечення. Існує ряд досліджень, які відносяться до вищезгаданої теми. Після їх вивчення та аналізу можна виокремити такі її

основні напрямки: аналіз ризиків, прогнозування дефектів та прогнозування характеристик якості програмного забезпечення [2]. Проте, якщо орієнтуватися лише на дослідження останніх двох критеріїв, то можна випустити важливий аспект життєвого циклу створення ПЗ – наявність та причини багів.

Метрики програмного продукту дозволяють охарактеризувати різні сторони програмних систем. Їх застосування в процесі оцінювання якості ПЗ сприяє зменшенню людського фактору та забезпечує об'єктивність рішень, підтриманих кількісними даними. В основному розрізняють два типи метрик – метрики процесу програмування та метрики програмного продукту. Останні фокусуються на аналізі готових результатів програмної розробки. Це такі, як технічна документація, вихідний код та дефекти. У той час метрики процесу програмування зосереджені на вимірюванні характеристик самої розробки. У тому числі методології та часу, який витрачається на створення програмного продукту [3]. Корисні метрики повинні бути простими й об'єктивними [4]. Також стійкими до незначних змін у процесі чи ПП та легко отримуватися.

Існують дослідження, які зосереджуються на використанні метрик програмного забезпечення, таких як рівень коду та дані про дефекти. Вони призначені для створення моделей прогнозування багів. Дані метрики, відображають якість кінцевого продукту. Вони, за пропозицією автора статті, створюються, використовуючи гібридний підхід вибору їх атрибутів, так як такий метод показує задовільні результати прогнозування дефектів [5].

У кожному повідомленні про помилку міститься багато інформації, яка є критичною для вирішення проблем у розробці. Два важливих аспекти – це серйозність і пріоритетність дефекту. Вони визначають рівень впливу помилки на систему та порядок її усунення [6]. Останні дослідження також описують можливість автоматизувати процес визначення пріоритетності та серйозності дефектів. Для цього пропонується застосовувати машинне навчання. Оскільки цей процес посилається лише на суб'єктивну оцінку розробників, тестувальників або менеджменту. Однак для аналізу якості ПЗ необхідно опиратись на вже формалізовані та визначені дані.

Обробка звітів про дефекти є важливою складовою процесу виправлення багів у життєвому циклі розробки програмного забезпечення. Це не робиться миттєво після їх створення. Перед початком процесу виправлення багу проходять певні етапи. Це перевірка на існування дублікату цього дефекту, визначення та сортування за пріоритетом і серйозністю. Тим не менш, через те що все це виконується вручну, даний метод виправлення помилок є досить неефективним. Саме тому дослідники цього питання підкреслюють необхідність автоматизації вищезазначених процесів. Точність існуючих методів залишається незадовільною [7].

Також звіти про баги містять цінну інформацію, яка допоможе прогнозувати первинну причину помилки за описом дефекту, для швидкого вибору розробниками відповідного інструменту для вирішення проблем з дебагінгом. Деякі причини помилок можна легко ідентифікувати, ознайомившись зі звітом, але стаття пропонує класифікувати їх за допомогою машинного навчання [8].

Результати даного дослідження було апробовано на конференції «Актуальні проблеми комп’ютерних наук» [9].

1.2 Аналіз інформаційних систем

У наш час web-орієнтовані системи використовуються IT-компаніями для аналізу якості програмного продукту. Вони застосовуються як інструменти додаткового контролю над рівнем розробки ПЗ.

Тому для порівняння було підібрано дві web-орієнтовані системи. Одна з них – це платформа «SonarQube» [10]. Друга – інструмент аналізу програмного забезпечення «SQuORE» [11].

Платформа з відкритим кодом «SonarQube» виконує безперервний аналіз та визначає якість програмного коду відповідно до метрик Seven Axes of Quality [12].

Структура даного ПЗ складається з dashboard-ів, статистики за кодом, unit-тестів та документації (рис. 1.1).

Основні функціональні можливості SonarQube наступні:

– проводить статичний аналіз коду, щоб виявити дефекти без виконання коду;

– забезпечує аналіз коду багатьох мов програмування;

– визначає «запахи коду» (ознаки потенційних проблем в коді), які можуть вказувати на необхідність проведення рефакторингу [13];

– оцінює кількість часу на усунення виявлених проблем;

– надає інтерактивну та інтуїтивну дашборду для перегляду відомостей про якість коду, включаючи оцінки, статистику та пріоритетні питання.

Web-платформа SonarQube має приємну кольорову гамму та витриманий шрифт. Також вона забезпечена простим, сучасним і зрозумілим інтерфейсом.

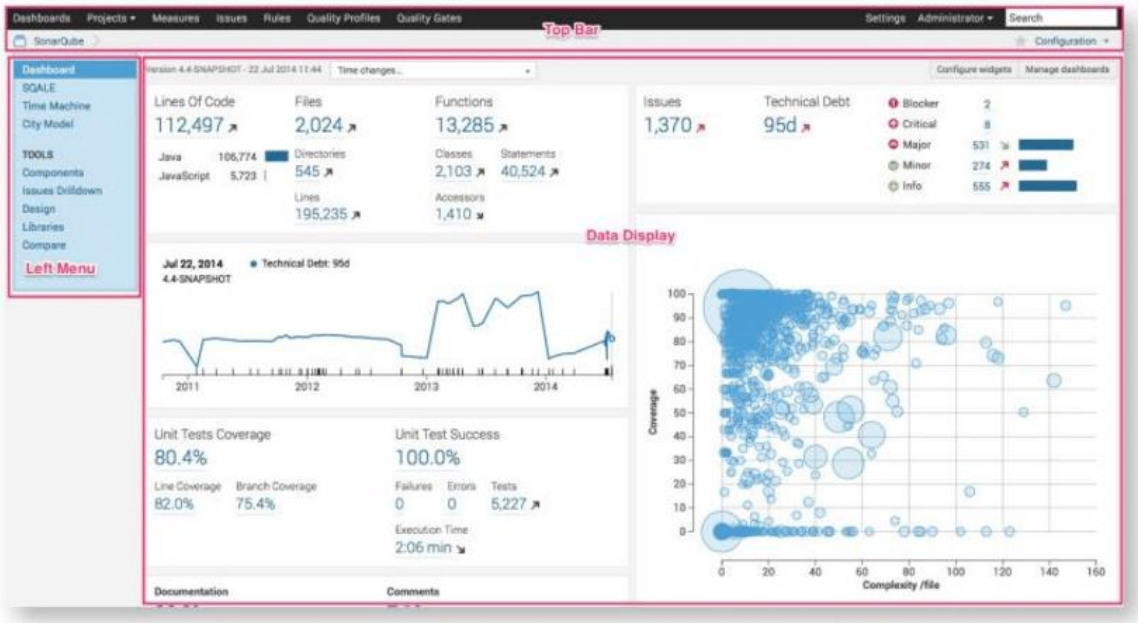


Рисунок 1.1 – Головне вікно SonarQube

Джерело: [14]

Після проведення дослідження платформи SonarQube, можна зробити висновок, що ця web-орієнтована система дійсно оцінює якість програмного забезпечення. Однак здійснює це лише з однієї сторони. А саме оцінює тільки якість коду.

Web-орієнтований інструмент для аналізу програмного забезпечення «SQuORE» (рис. 1.2-1.3) забезпечує управління якістю розробки ПЗ за допомогою наступного функціоналу:

– покращення продуктивності проєкту;

– підвищення якості програмного забезпечення;

– забезпечення дотримання процесами розробки стандартів [15].

Серед його функцій можна виділити такі:

– детальний аналіз якості коду;

– застосування метрик складності, зв’язності, дуплікації коду, тестового покриття;

– відстеження змін в якості проєкту з часом, дозволяючи досліджувати тренд змін;

– створення звітів та аудиту проєктів;

– ідентифікація та управління ризиками.

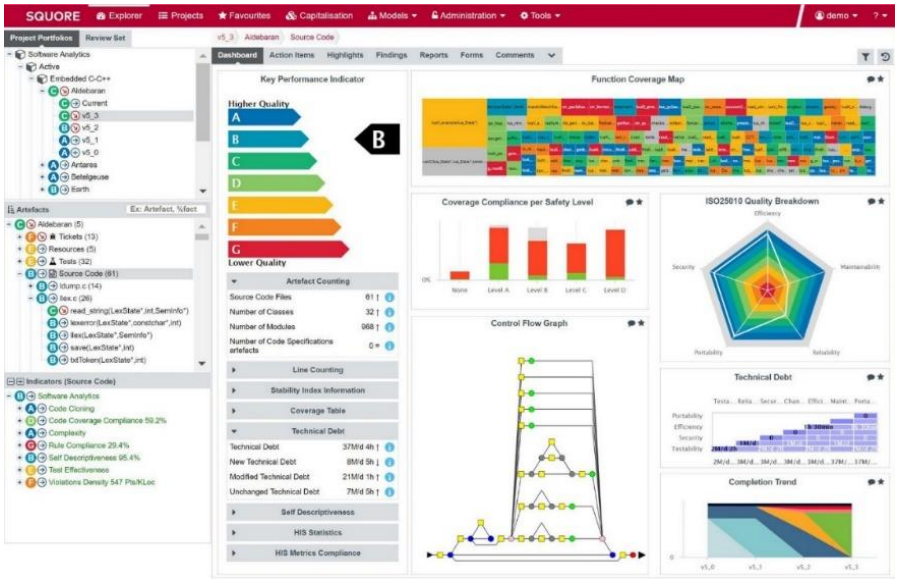


Рисунок 1.2 – Головне вікно SQuORE

Джерело: [10]

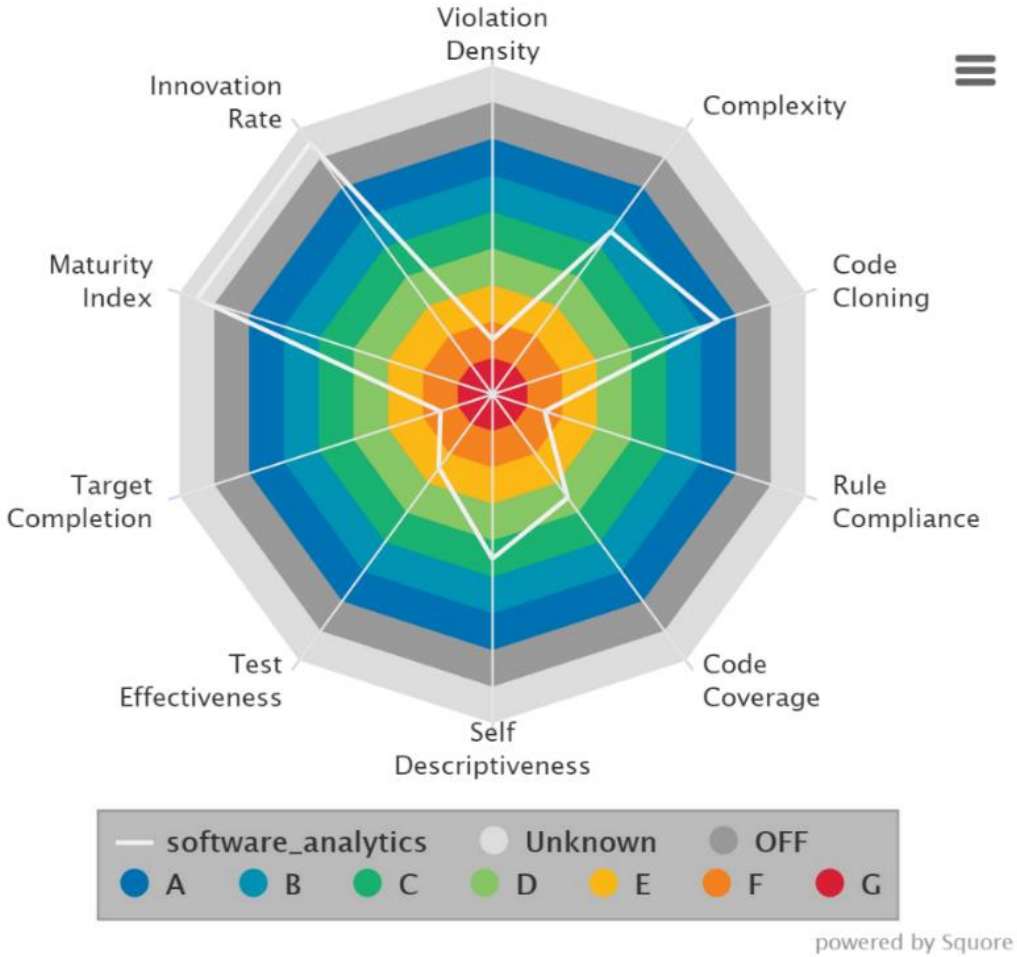


Рисунок 1.3 – Візуалізація оцінки якості ПЗ

Джерело: [10]

Серед переваг даного інструменту можна виокремити його сучасний вигляд, наявність метрик і візуалізація результатів аналізу якості програмного продукту. Також здійснення формування звітності на основі вищезазначеного. Але, наявний функціонал не забезпечує аналіз дефектів програмного продукту, що також є невід'ємною й цінною інформацією про ПЗ.

Результатом проведення порівняльного аналізу двох обраних web-орієнтованих програмних інструментів є таблиця 1.1.

Таблиця 1.1 – Порівняльна таблиця продуктів-аналогів

Характеристика	**«SonarQube»**	**«SQuORE»**
Відповідність сучасним вимогам UI/UX дизайну	+	+
Аналіз якості програмного продукту за кодом	+	+
Аналіз якості програмного продукту за метриками програмування	-	+
Аналіз якості програмного продукту за метриками дефектів	-	-
Можливість формування звіту	-	+
Відслідковування тренду змін якості програмного продукту	-	+
Оцінка кількості часу на виправлення дефекту	+	-

Джерело: побудовано автором

Проаналізувавши дані з таблиці 1.1, можна виділити переваги розглянутих ПЗ, а саме певний їхній функціонал, який варто буде реалізувати у власній розробці. А також недоліки, які варто подолати. Отже, можна зробити висновок, що запропонована web-орієнтована система повинна мати інтуїтивно-зрозумілий та сучасний інтерфейс, забезпечувати аналіз рівня розробки програмного продукту за метриками дефектів і відслідковувати тренд змін якості ПЗ.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи магістра є автоматизація процесу аналізу якості програмного продукту за рахунок розробки та впровадження web-орієнтованої системи, використовуючи bug review документацію. Створене ПЗ буде мати попит серед IT компаній. А саме його використання стане актуальним для команди розробки. Оскільки вона буде мати інформацію про причину появи дефектів і слабкі місця при створенні програмного забезпечення. Загалом, на основі вищевсказаного, дана web-орієнтована система стане корисною для діяльності розробників та тестувальників. Також, за допомогою неї можна дослідити тренд змін у кількості дефектів, базуючись на інших характеристиках помилок ПЗ. Реалізація представленого програмного продукту та подальше його використання в IT компанії дозволить автоматизувати процес Bug review. Це покращить роботу над створенням ПЗ та процеси в середині команди розробки.

Дана web-орієнтована система має відповідати наступним функціональним вимогам:

- імпорт документу bug review з .csv файлу [16] в таблицю на сторінці;
- створення, редагування та видалення полів таблиці;
- проведення аналізу за даними таблиці з дефектами за метриками;
- візуалізація результатів аналізу за допомогою діаграм;
- генерація та вивід результату аналізу якості програмного продукту.

Створювана web-орієнтована система повинна мати зручний користувацький інтерфейс із мінімальним застосуванням візуальних ефектів. Оскільки це ПЗ розглядається як внутрішній продукт IT-компанії й призначена для оптимізації її роботи. Результати аналізу мають бути легкі для їх візуального сприйняття та мати високу швидкість завантаження на сторінці.

Для досягнення мети даного проєкту потрібно реалізувати наступні задачі:

– визначити актуальність роботи, провести детальний аналіз предметної області та визначити потенційних користувачів даної системи;

– провести комплексний аналіз існуючих web-орієнтованих систем-аналогів, виявити їх сильні та слабкі сторони;

– визначити ключові функціональні можливості запропонованої web-орієнтованої системи та обрати відповідні технології та інструменти для її розробки;

– виконати структурно-функціональне моделювання процесу автоматизації аналізу якості програмного продукту, який забезпечується запропонованим програмним продуктом.

У ході визначення мети проєкту було проведено детальне планування його робіт, яке представлено у Додатку А.

2.2 Методи дослідження

Після встановлення мети проєкту, формування функціональних та нефункціональних вимог до створюваної web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту, а також виділення необхідних для реалізації задач, треба визначити методи. За їхньої допомоги буде здійснюватися дослідження.

Наукові методи поділяються на загальні та спеціальні. Загальні розділяють на такі чотири основні групи:

– емпіричні методи дослідження;

– теоретичні методи дослідження;

– комбіновані (використовуються як на теоретичному, так і на емпіричному рівнях дослідження) методи дослідження;

– метод моделювання [17].

Емпіричні методи засновані на спостереженні та експерименті. Вони дозволяють взаємодіяти з об’єктом дослідження. Їхнє використання також забезпечує отримання необхідних даних через моніторинг і вимірювання. До ключових емпіричних методів належать проведення спостережень, організація експериментів, реалізація опитувань і анкетувань, а також детальний аналіз конкретних прикладів [18]. Вони були застосовані на стадії вивчення предметної області. Також при оцінці реалізації необхідних функціональних можливостей, ґрунтуючись на консультаціях із працівниками IT-компанії.

Теоретичний метод дослідження базується на ретельному аналізі специфіки предметної області [19]. Його було використано під час роботи над проєктом для глибокого вивчення сучасних трендів в IT індустрії та визначення засобів для розробки представленого програмного продукту.

Комбінований метод дослідження використовується на різних етапах для досягнення кращого розуміння предметної області та забезпечення обґрунтованих висновків [20]. Він також був застосований для розробки діаграм варіантів використання (Use Case Diagrams) [21].

Метод моделювання, у свою чергу, відповідає за створення діаграм. Вони демонструють процес роботи над розроблюваним програмним продуктом та яким чином його буде використано надалі [22]. Його було застосовано на етапі побудови діаграм IDEF0 [23] та Use Case.

Основною задачею даної web-орієнтованої системи є аналіз якості програмного продукту. Тому було досліджено наявні методи та метрики визначення якісних показників ПП. Оскільки їх використання є важливим аспектом даної роботи.

Примітивні метрики застосовуються під час всього життєвого циклу програмного продукту. Вони можуть бути зображені за допомогою графіків, гістограм, як частина статистичного процесу контролю. Наприклад, існують наступні примітивні метрики:

– кількість баг репортів за спринтом, пріоритетом, причиною, та категорією;

– кількість створених, закритих баг репортів за певний період часу;

– час з створення, закриття баг репортів [24].

Ще однією важливою метрикою є Щільність Дефектів або Defect Density, що допомагає виділити відсоток помилок на певний продукт компанії. Для розрахунку використовується формула (2.1):

$$\text{Щільність дефектів} = \frac{\text{Кількість дефектів за продуктом}}{\text{Загальна кількість дефектів}} \cdot 100\% \quad (2.1)$$

Якщо щільність багів на певному продукті дорівнює 30% і більше – необхідно звернути увагу на процеси розробки та тестування, бо 30% є допустимим значенням для помилок. Щодо розподілу між production та development середовищами допустимих значень, для production середовища – це значення має бути біля 0% і менше 1%, а для development середовища – оптимальне значення 20% [25].

Також важливими показником є середній час життя дефекту. Ця метрика дозволяє оцінити час, необхідний для тестування, та виділити продукт, із яким виникають найбільші труднощі. Для розрахунку використовується формула (2.2):

$$\text{Середній час життя дефекту} = \frac{\text{Сумарна час виправлення дефектів}}{\text{Кількість дефектів}} \quad (2.2)$$

Середній час життя дефекту також можна обчислити для різних функціональних частин програмного продукту. Таким чином наявною є можливість виявити складні модулі для виправлення.

Наступним важливим критерієм є коефіцієнт помилок, пропущених на production середовище. Він демонструє якість тестування та ефективність виявлення помилок – яка частка багів була знайдена, а яка була пропущена. Розраховується цей показник за формулою (2.3):

$$\text{Пропущені дефекти} = \frac{\text{Кількість помилок виявлених після випуску релізу}}{\text{Загальна кількість помилок випущених до та після релізу}} \quad (2.3)$$

Допустимий відсоток помилок має бути приблизно <0.1. У іншому випадку це буде означати, що кожен десятий дефект не був знайдений під час тесту та свідчить про незадовільну роботу тестувальників.

Вагомим критерієм для обчислення також є частка непідтверджених дефектів, що показує скільки багів було заведено, але не виправлено. Він розраховується за формулою (2.4):

$$\text{Неопрацьовані дефекти} = \frac{\text{Число дефектів не прийнятих до виправлення}}{\text{Загальна кількість зареєстрованих дефектів}} \quad (2.4)$$

Якщо частка дефектів, які були не опрацьовані, перевищує 20%, то в команді є певні проблеми з розумінням що є баг, а що ні. А також із пріоритезацією їх виправлення в продукті [26].

2.3 Вибір технологій

Для програмної реалізації представленої web-орієнтованої системи було обрано наступні технології:

– HTML – для створення шаблону web-сторінок [27];
– CSS – для надання web-сторінці адаптивності [28];
– JavaScript – для створення та відображення діаграм [29];
– Yii2 – як основа для створення бекенду даного програмного продукту [30];
– бібліотеку pandas – для аналізу даних [31];
– MySQL [32] для організації збереження даних.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ

Після виконання аналізу предметної області, визначення актуальності, мети та задач розробки, проведення детального планування робіт, можна перейти до проєктування даної web-орієнтованої системи. Було створено діаграми нотації IDEF0, детально описано послідовність дій, які беруть участь у автоматизації процесу аналізу якості програмного продукту на основі Bug review документу.

3.1 Діаграми нотації IDEF0

Для моделювання функціоналу розроблюваної web-орієнтованої системи використовуються різноманітні методи структурного та об’єктно-орієнтованого підходу. А саме IDEF0. Це методологія функціонального моделювання. Вона є досить корисною. В основному застосовується для розробки функціональної моделі. Вона демонструє структуру системи. А також її функції. Результатом застосування IDEF0 до будь-якої системи є модель останньої. Вона складається з ієрархічно упорядкованого набору діаграм. У свою чергу містять такі дані, як вхідні, вихідні, управління та механізму.

Для контекстної діаграми було обрано наступні дані web-орієнтованої системи автоматизації процесу аналізу якості програмного забезпечення:

1. Вхідні дані: .csv файл із дефектами та їх основними характеристиками, або введена вручну таблиця з дефектами.
2. Управління: метрики якості програмного забезпечення, стандарти якості програмного продукту, автоматизовані скрипти обробки .csv файлів.

3. Вихідні дані: результати обчислення метрик якості програмного продукту, діаграми з візуалізованими результатами аналізу якості програмного забезпечення.

4. Механізми: користувач, web-орієнтована система, апаратне забезпечення, база даних.

На рисунку 3.1 зображена контекстна діаграма IDEF0 web-орієнтованої системи автоматизованого процесу аналізу якості програмного продукту.

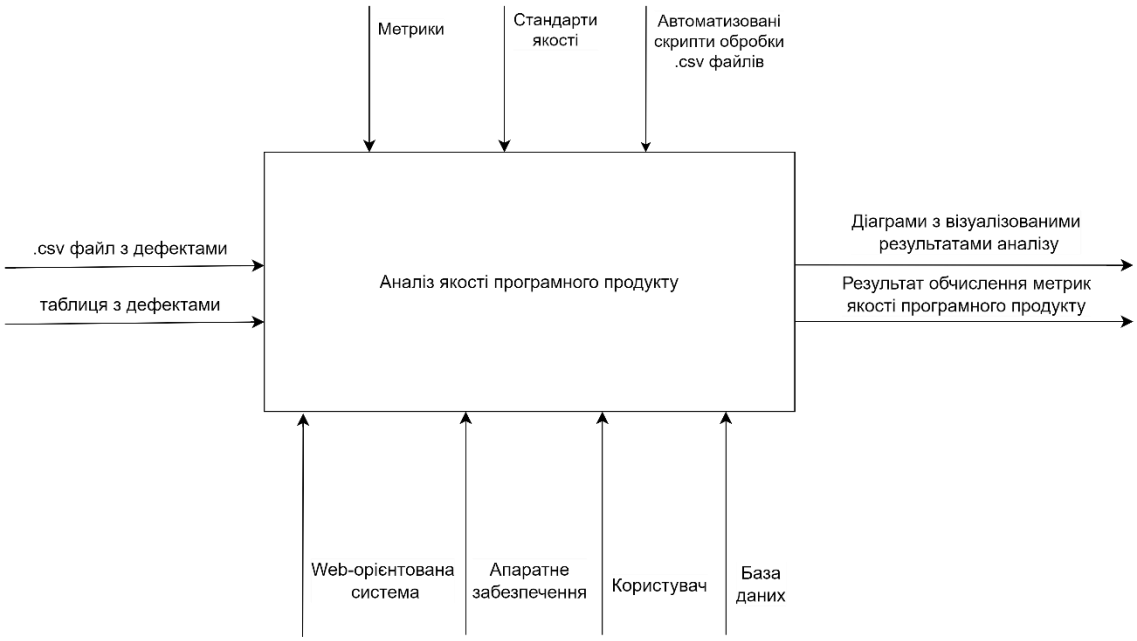


Рисунок 3.1 – Контекстна діаграма IDEF0

Джерело: побудовано автором

Для деталізації операцій web-орієнтованої системи було виконано декомпозицію першого рівня для автоматизованого процесу аналізу якості програмного продукту через імпорт завантаженого файлу .csv (рис. 3.2).

Дані для діаграми наступні:

1. Вхідні дані: .csv файл з дефектами, таблиця з дефектами.

2. Управління: автоматизовані скрипти обробки .csv файлів, метрики, стандарти якості програмних продуктів.

3. Вихідні дані: результати обчислення метрик якості програмного продукту, діаграма з результатами аналізу.

4. Механізми: апаратне забезпечення, web-орієнтована система, користувач, база даних.

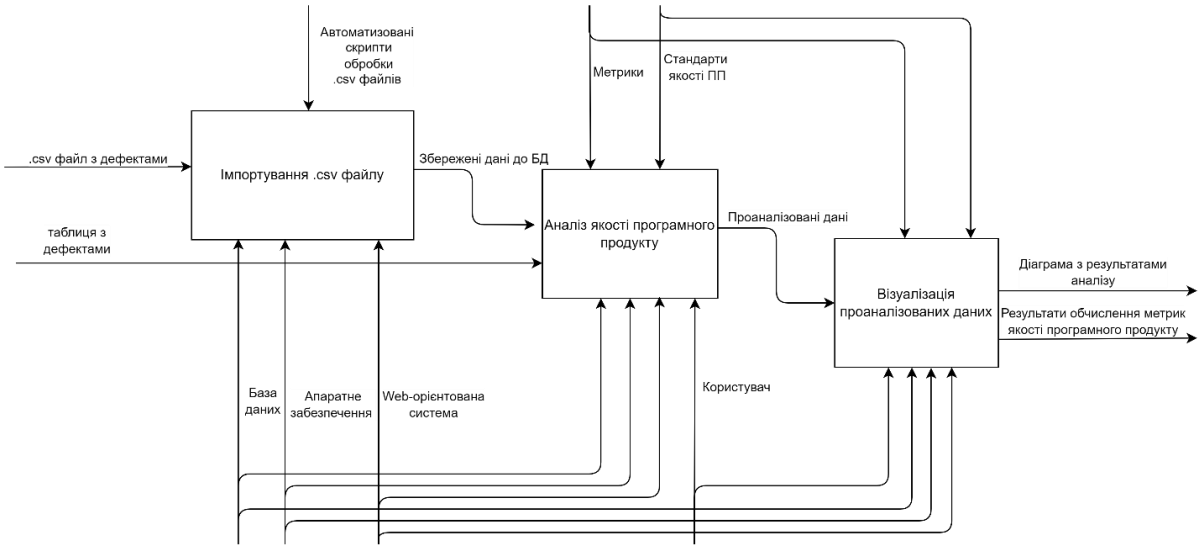


Рисунок 3.2 – Діаграма декомпозиції процесу аналізу якості ПЗ

Джерело: побудовано автором

3.2 Use Case діаграма

Після проведення процесу моделювання необхідно створити діаграму варіантів використання (Use Case diagram). Вона є графічним зображенням взаємодії між користувачами («акторами») та розробленою системою. Use Case діаграма показує різноманітні варіанти їхньої взаємодії. Вона має різні ключові складові. Це актори (actors), варіанти використання (use cases), а також відносини (relationships) між акторами та варіантами використання: включення (includes), розширення (extends), співвідношення (associations), узагальнення (generalizations).

Для web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту було визначено наступні варіанти використання:

– завантаження даних;
– імпорт даних з .csv:
– створення записів про дефект самостійно;
– редагування даних в таблицях;
– видалення відомостей про дефект;
– аналіз даних дефектів;
– побудова діаграм на основі результатів процесу аналізу;
– вивід результатів обчислення метрик якості програмного забезпечення.

Актором діаграми варіантів використання є Користувач (тестувальник) та База даних. Діаграма використання web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту зображена на рисунку 3.3.

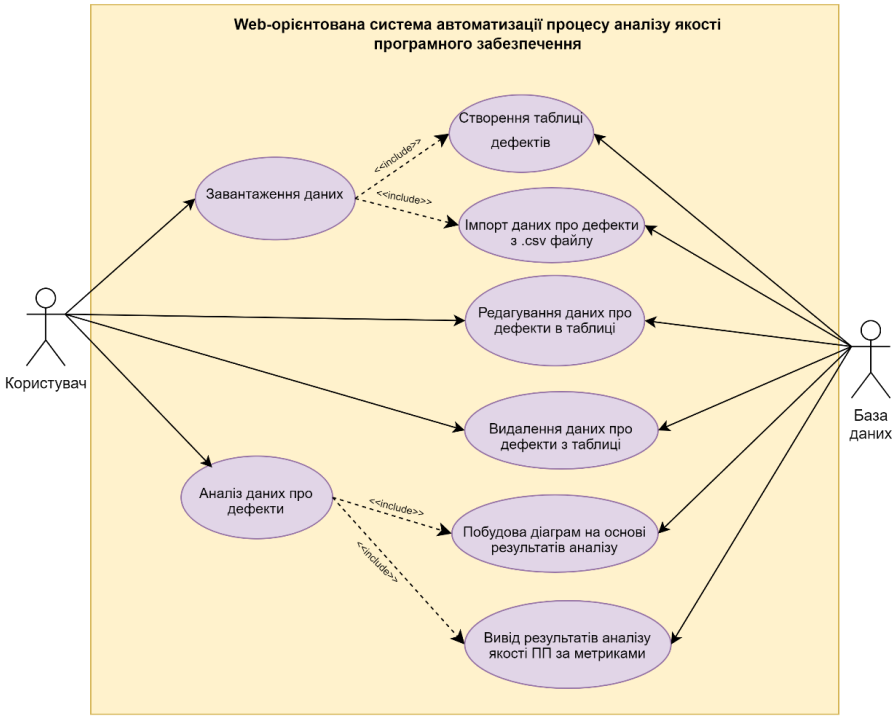


Рисунок 3.3 – Use Case діаграма

Джерело: побудовано автором

3.3 Проєктування моделі бази даних

Розробка бази даних (БД) для web-орієнтованої системи починається з визначення тісно взаємопов'язаних таблиць та зв'язків між ними [33]. Вони в сукупності формують схему бази даних. Початковим етапом створення БД є ідентифікація ключових сутностей системи. Для кожної з яких буде створена окрема таблиця.

Під час розробки структури бази даних цього проєкту було виділено наступні сутності:

– дефекти (bugs);
– статус (status);
– пріоритет (priority);
– серйозність (severity);
– платформа (platform);
– автор багу (user type);
– функціонал (functionality).

Таблиця Дефектів (bugs) має наступні атрибути:

– id (integer) – первинний ключ таблиці;
– sprint (integer) – номер спринту;
– title (text) – назва дефекту;
– createdDate (date) – дата створення дефекту;
– changedDate (date) – дата останньої зміни дефекту;
– timeForSolve (integer) – час на виправлення дефекту;
– rootCause (text) – причини виникнення дефекту;
– createdBy (integer) – foreign key автора дефекту;
– statusID (integer) – foreign key статусу дефекту;
– priorityID (integer) – foreign key пріоритету дефекту;
– severityID (integer) – foreign key серйозності дефекту;
– functionalityID (integer) – foreign key функціоналу дефекту;

– platformID (integer) – foreign key платформи дефекту.

Таблиця Статусів (status) дефектів має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar (25)) – назва статуса.

Таблиця Пріоритетів (priority) багів має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar (15)) – назва пріоритету.

Таблиця Серйозності (severity) дефекту має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar (15)) – назва серйозності.

Таблиця Автора багу (user_type) має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar (25)) – назва типу користувачів.

Таблиця Функціоналу (functionality) багу має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar (50)) – назва функціоналу.

Таблиця Платформ (platform) дефекту має наступні атрибути:

– id (integer) – первинний ключ таблиці;

– name (varchar(25)) – назва платформи.

4 РОЗРОБКА WEB-ОРІЄНТОВАНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ

4.1 Архітектура web-орієнтованої системи

Процес розробки даної web-орієнтованої системи починається з визначення архітектури програмного продукту за допомогою діаграми HLD (High Level Design). Це діаграма дизайну високого рівня. Вона призначена для відображення загальної структури та архітектури розроблюваної системи, ілюструючи її основні компоненти, їх взаємозв’язки та способи взаємодії [34].

Архітектура web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту (рис. 4.1) включає наступні компоненти:

– користувач (User) взаємодіє з даною системою відправляючи запити (Requests) та отримуючи відповіді (Responses);

– маршрутизація (Routing) приймає запити від користувачів та направляє їх до відповідного контролера;

– модель (Model), яка здійснює взаємодію з базою даних (Data Base);

– контролер (Controller), який забезпечує зв’язок між моделлю та іншими компонентами даної системи;

– сервіс для аналізу даних (Data analyzing service), який приймає дані з контролера та повертає проаналізовані дані;

– база даних (Data Base) – це сховище даних, із яким модель взаємодіє для збереження та отримання даних;

– вигляд (View) для візуалізації інтерфейсу.

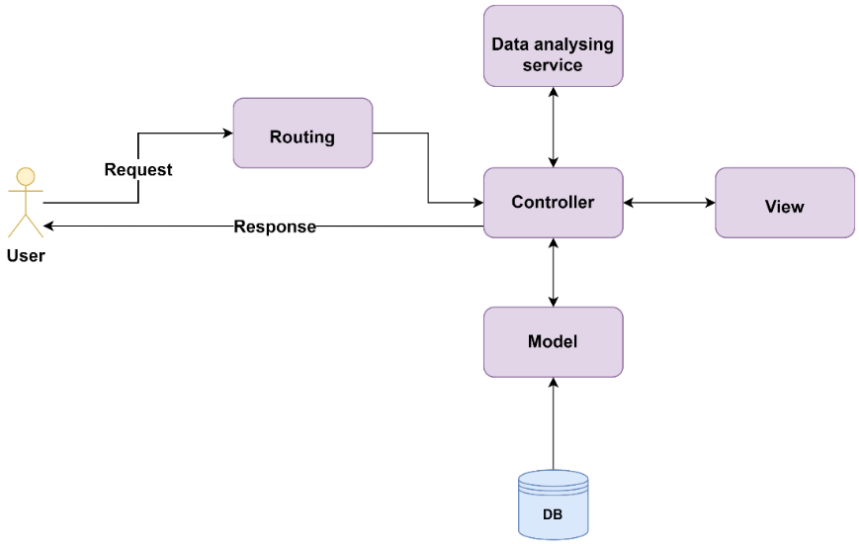


Рисунок 4.1 – Діаграма високого рівня

Джерело: побудовано автором

4.2 Реалізація бази даних web-орієнтованої системи

Базу даних було реалізовано згідно попереднього проєктування її структури у пункті 3.3. На рисунку 4.2 представлено схему реалізованої БД web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту засобами Luna Modeler [35].

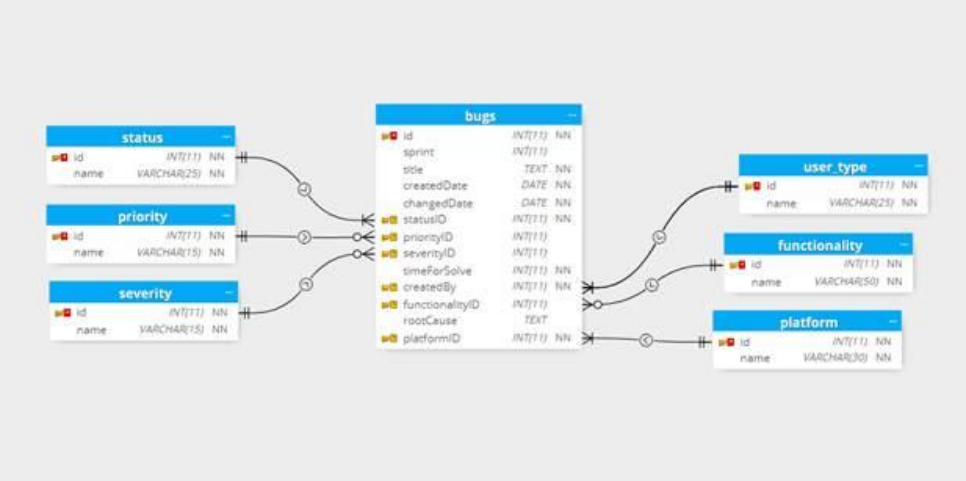


Рисунок 4.2 – Схема реалізованої бази даних засобами Luna Modeler

Джерело: побудовано автором (знімок з екрану)

4.3 Програмна реалізація web-орієнтованої системи

Для реалізації представленого програмного продукту спочатку необхідно інсталювати локальний web-сервер. У даному випадку, це Open Server [36]. Для написання програмного коду було обрано IDE PHPStorm [37]. Саме він є найзручнішим інструментом для реалізації проєкту. Після цього було розроблено базу даних.

Для управління інформацією з БД було створено модель сутностей «Дефектів» та моделі для всіх пов'язаних сутностей: статус, платформа, пріоритет і т.д. На рисунку 4.3 зображена модель Bugs, а саме визначення правил валідації «rules» та мітки для атрибутів моделі «attributeLabels». Рисунок 4.4 демонструє код моделі Bugs, який встановлює зв'язки між іншими пов'язаними таблицями.

```php
class Bugs extends \yii\db\ActiveRecord
{
    no usages
    public static function tableName()
    {
        return 'bugs';
    }
    public function rules()
    {
        return [
            [['id', 'title', 'createdDate', 'changedDate', 'statusID', 'priorityID', 'severityID', 'timeForSolve', 'createdBy', 'functionalityID', 'platformID'], 'required'],
            [['id', 'sprint', 'statusID', 'priorityID', 'severityID', 'timeForSolve', 'createdBy', 'functionalityID', 'platformID'], 'integer'],
            [['title', 'rootCause'], 'string'],
            [['createdDate', 'changedDate'], 'safe'],
            [['id'], 'unique'],
            [['functionalityID'], 'exist', 'skipOnError' => true, 'targetClass' => Functionality::class, 'targetAttribute' => ['functionalityID' => 'id']],
            [['platformID'], 'exist', 'skipOnError' => true, 'targetClass' => Platform::class, 'targetAttribute' => ['platformID' => 'id']],
            [['priorityID'], 'exist', 'skipOnError' => true, 'targetClass' => Priority::class, 'targetAttribute' => ['priorityID' => 'id']],
            [['severityID'], 'exist', 'skipOnError' => true, 'targetClass' => Severity::class, 'targetAttribute' => ['severityID' => 'id']],
            [['statusID'], 'exist', 'skipOnError' => true, 'targetClass' => Status::class, 'targetAttribute' => ['statusID' => 'id']],
        ];
    }
    no usages
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'sprint' => 'Sprint',
            'title' => 'Title',
            'createdDate' => 'Created Date',
            'changedDate' => 'Changed Date',
            'statusID' => 'Status',
            'priorityID' => 'Priority',
            'severityID' => 'Severity',
            'timeForSolve' => 'Time For Solve',
            'createdBy' => 'Created By',
            'functionalityID' => 'Functionality',
            'rootCause' => 'Root Cause',
            'platformID' => 'Platform',
        ];
```

Рисунок 4.3 – Модель Bugs

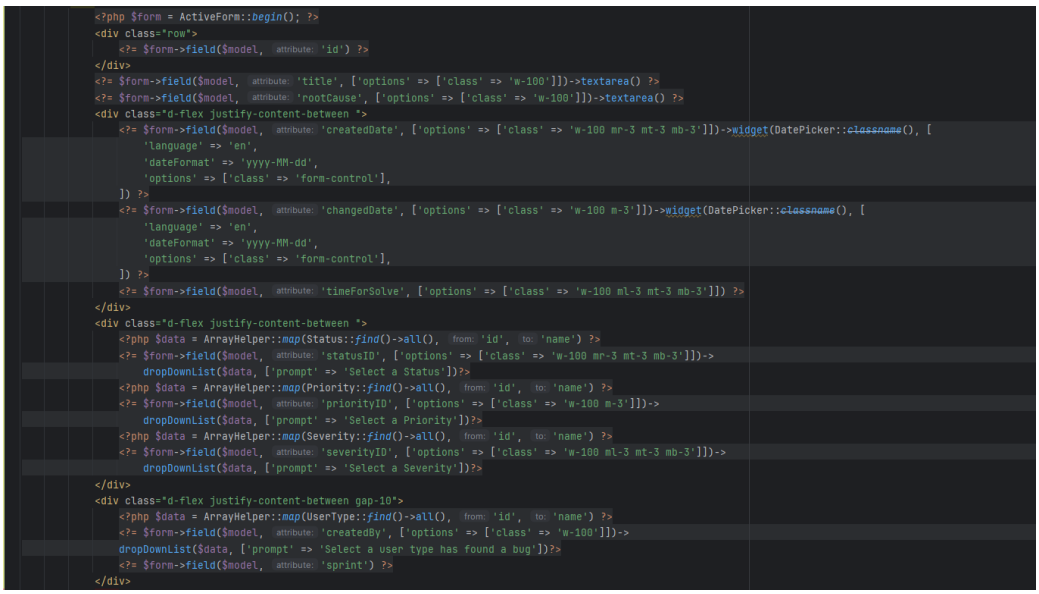
Джерело: побудовано автором (знімок з екрану)

```
    /** Gets query for [[Functionality]]. ...*/
    no usages
    public function getFunctionality()
    {
        return $this->hasOne( class: Functionality::class, ['id' => 'functionalityID']);
    }
    /** Gets query for [[Platform]]. ...*/
    no usages
    public function getPlatform()
    {
        return $this->hasOne( class: Platform::class, ['id' => 'platformID']);
    }
    /** Gets query for [[Priority]]. ...*/
    no usages
    public function getPriority()
    {
        return $this->hasOne( class: Priority::class, ['id' => 'priorityID']);
    }
    /** Gets query for [[Severity]]. ...*/
    no usages
    public function getSeverity()
    {
        return $this->hasOne( class: Severity::class, ['id' => 'severityID']);
    }
    /** Gets query for [[Status]]. ...*/
    no usages
    public function getStatus()
    {
        return $this->hasOne( class: Status::class, ['id' => 'statusID']);
    }
    /** Gets query for [[UserType]]. ...*/
    no usages
    public function getUserType()
    {
        return $this->hasOne( class: UserType::class, ['id' => 'createdBy']);
    }
}
```

Рисунок 4.4 – Продовження моделі Bugs

Джерело: побудовано автором (знімок з екрану)

Першим кроком у реалізації даного програмного продукту є створення самого дефекту. На рисунках 4.5-4.6 зображено частину коду, який представляє інтерфейс користувача для форми подання дефектів, використовуючи date picker-и, випадаючі списки.



```
<?php $form = ActiveForm::begin(); ?>
<div class="row">
    <?= $form->field($model, attribute: 'id') ?>
</div>
<?= $form->field($model, attribute: 'title', ['options' => ['class' => 'w-100']])->textarea() ?>
<?= $form->field($model, attribute: 'rootCause', ['options' => ['class' => 'w-100']])->textarea() ?>
<div class="d-flex justify-content-between ">
    <?= $form->field($model, attribute: 'createdDate', ['options' => ['class' => 'w-100 mr-3 mt-3 mb-3']])->widget(DatePicker::classname(), [
        'language' => 'en',
        'dateFormat' => 'yyyy-MM-dd',
        'options' => ['class' => 'form-control'],
    ]) ?>
    <?= $form->field($model, attribute: 'changedDate', ['options' => ['class' => 'w-100 m-3']])->widget(DatePicker::classname(), [
        'language' => 'en',
        'dateFormat' => 'yyyy-MM-dd',
        'options' => ['class' => 'form-control'],
    ]) ?>
    <?= $form->field($model, attribute: 'timeForSolve', ['options' => ['class' => 'w-100 ml-3 mt-3 mb-3']]) ?>
</div>
<div class="d-flex justify-content-between ">
    <?php $data = ArrayHelper::map(Status::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'statusID', ['options' => ['class' => 'w-100 mr-3 mt-3 mb-3']])->
        dropDownList($data, ['prompt' => 'Select a Status'])?>
    <?php $data = ArrayHelper::map(Priority::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'priorityID', ['options' => ['class' => 'w-100 m-3']])->
        dropDownList($data, ['prompt' => 'Select a Priority'])?>
    <?php $data = ArrayHelper::map(Severity::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'severityID', ['options' => ['class' => 'w-100 ml-3 mt-3 mb-3']])->
        dropDownList($data, ['prompt' => 'Select a Severity'])?>
</div>
<div class="d-flex justify-content-between gap-10">
    <?php $data = ArrayHelper::map(UserType::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'createdBy', ['options' => ['class' => 'w-100']])->
    dropDownList($data, ['prompt' => 'Select a user type has found a bug'])?>
    <?= $form->field($model, attribute: 'sprint') ?>
</div>
```

Рисунок 4.5 – Інтерфейс створення дефекту

Джерело: побудовано автором (знімок з екрану)

```
<div class="d-flex justify-content-between ">
    <?php $data = ArrayHelper::map(Functionality::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'functionalityID', ['options' => ['class' => 'w-100']])->
        dropDownList($data, ['prompt' => 'Select a Functionality area'])?>
    <?php $data = ArrayHelper::map(Platform::find()->all(), from: 'id', to: 'name') ?>
    <?= $form->field($model, attribute: 'platformID', ['options' => ['class' => 'w-100']])->
        dropDownList($data, ['prompt' => 'Select a Platform'])?>
</div>
<div class="form-group mt-5">
    <?= Html::submitButton( content: 'Submit', ['class' => 'btn btn-primary']) ?>
</div>
<?php ActiveForm::end(); ?>
```

Рисунок 4.6 – Продовження коду інтерфейсу створення дефекту

Джерело: побудовано автором (знімок з екрану)

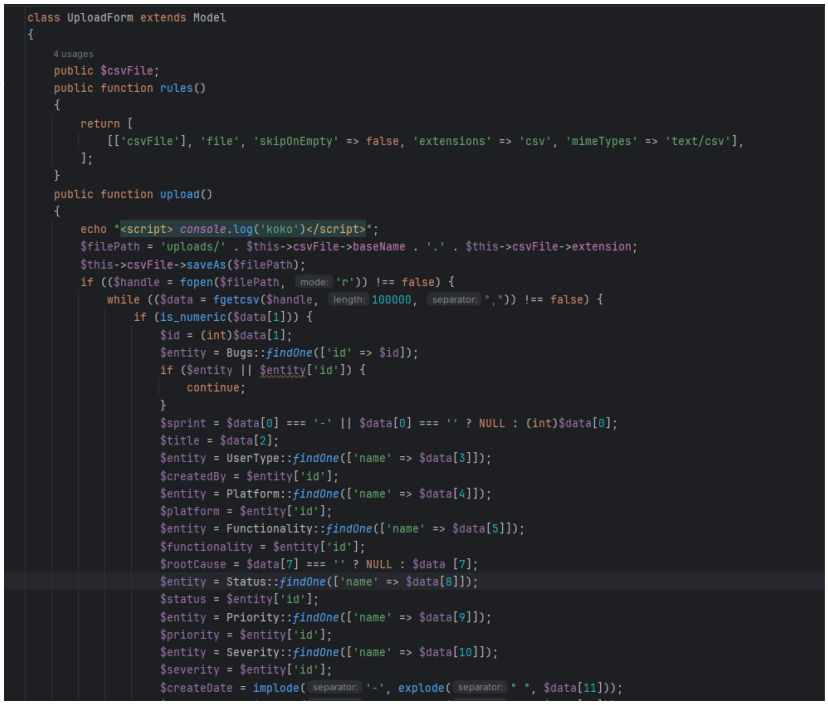
Далі було розроблено контролер ActionNew для створення нового запису дефекту (рис. 4.7).

```
public function actionNew()
{

    $model = new Bugs();

    if ($model->load(Yii::$app->request->post()) && $model->validate()) {
        if ($model->save()) {

            Yii::$app->session->setFlash( key: 'success', value: 'Data saved successfully.');
        } else {

            Yii::$app->session->setFlash( key: 'error', value: 'Error saving data.');
        }
    }

    return $this->render( view: 'new', [
        'model' => $model,
    ]);
}
```

Рисунок 4.7 – Контролер створення нового запису дефекту

Джерело: побудовано автором (знімок з екрану)

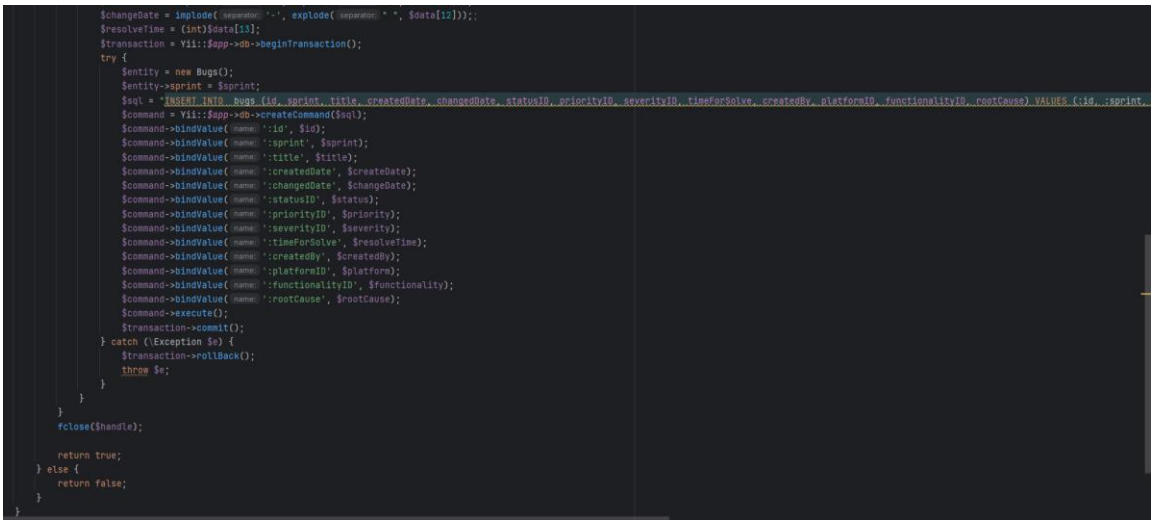
Наступна важлива функція системи – це імпорт даних про дефекти з .csv файлу, що забезпечує ефективність роботи з великою кількістю багів. Для цього було розроблено модель UploadForm. На рисунках 4.8-4.9 зображено код для обробки завантажених файлів, їх валідації, та збереження до бази даних.



```
class UploadForm extends Model
{
    4 usages
    public $csvFile;
    public function rules()
    {
        return [
            [['csvFile'], 'file', 'skipOnEmpty' => false, 'extensions' => 'csv', 'mimeTypes' => 'text/csv'],
        ];
    }
    public function upload()
    {
        echo "<script> console.log('koko')</script>";
        $filePath = 'uploads/' . $this->csvFile->baseName . '.' . $this->csvFile->extension;
        $this->csvFile->saveAs($filePath);
        if (($handle = fopen($filePath, mode: 'r')) !== false) {
            while (($data = fgetcsv($handle, length: 100000, separator: ",")) !== false) {
                if (is_numeric($data[1])) {
                    $id = (int)$data[1];
                    $entity = Bugs::findOne(['id' => $id]);
                    if ($entity || $entity['id']) {
                        continue;
                    }
                    $sprint = $data[0] === '-' || $data[0] === '' ? NULL : (int)$data[0];
                    $title = $data[2];
                    $entity = UserType::findOne(['name' => $data[3]]);
                    $createdBy = $entity['id'];
                    $entity = Platform::findOne(['name' => $data[4]]);
                    $platform = $entity['id'];
                    $entity = Functionality::findOne(['name' => $data[5]]);
                    $functionality = $entity['id'];
                    $rootCause = $data[7] === '' ? NULL : $data [7];
                    $entity = Status::findOne(['name' => $data[8]]);
                    $status = $entity['id'];
                    $entity = Priority::findOne(['name' => $data[9]]);
                    $priority = $entity['id'];
                    $entity = Severity::findOne(['name' => $data[10]]);
                    $severity = $entity['id'];
                    $createDate = implode( separator: '-', explode( separator: " ", $data[11]));
```

Рисунок 4.8 – Модель UploadForm

Джерело: побудовано автором (знімок з екрану)



```
                $changeDate = implode( separator: '-', explode( separator: " ", $data[12]));
                $resolveTime = (int)$data[13];
                $transaction = Yii::$app->db->beginTransaction();
                try {
                    $entity = new Bugs();
                    $entity->sprint = $sprint;
                    $sql = "INSERT INTO `bugs`(id, sprint, title, createdDate, changedDate, statusID, priorityID, severityID, timeForSolve, createdBy, platformID, functionalityID, rootCause) VALUES (:id, :sprint,
                    $command = Yii::$app->db->createCommand($sql);
                    $command->bindValue( name: ':id', $id);
                    $command->bindValue( name: ':sprint', $sprint);
                    $command->bindValue( name: ':title', $title);
                    $command->bindValue( name: ':createdDate', $createDate);
                    $command->bindValue( name: ':changedDate', $changeDate);
                    $command->bindValue( name: ':statusID', $status);
                    $command->bindValue( name: ':priorityID', $priority);
                    $command->bindValue( name: ':severityID', $severity);
                    $command->bindValue( name: ':timeForSolve', $resolveTime);
                    $command->bindValue( name: ':createdBy', $createdBy);
                    $command->bindValue( name: ':platformID', $platform);
                    $command->bindValue( name: ':functionalityID', $functionality);
                    $command->bindValue( name: ':rootCause', $rootCause);
                    $command->execute();
                    $transaction->commit();
                } catch (\Exception $e) {
                    $transaction->rollBack();
                    throw $e;
                }
            }
        }
        fclose($handle);

        return true;
    } else {
        return false;
    }
}
```

Рисунок 4.9 – Продовження коду моделі для імпорту .csv файлів

Джерело: побудовано автором (знімок з екрану)

Далі було розроблено метод контролер actionList, який керує завантаженням файлів .csv із записами дефектів та їх подальшою обробкою (рис. 4.10).

```php
1 usage
public function actionList()
{

    $model = new UploadForm();


    if (Yii::$app->request->isPost) {
        $model->csvFile = UploadedFile::getInstance($model, attribute: 'csvFile');
        if ($model->upload()) {
            return $this->refresh();
        }
    }

    return $this->render( view: 'list', ['model' => $model]);
}
```

Рисунок 4.10 – Метод контролеру імпорту нових дефектів з .csv файлу

Джерело: побудовано автором (знімок з екрану)

Для функцій аналізу даних про дефекти було розгорнуто Python сервер за допомогою Flask [38]. Оскільки це є досить зручним рішенням. На рисунках 4.11-4.12 представлено код для обробки та аналізу даних про дефекти. Це реалізовано за допомогою бібліотеки pandas. Після аналізу та агрегації дані використовуються для візуалізації у вигляді діаграм та результатів обчислення метрик. Повний код файлів надано у Додатку В.

diploma-modal.py

```python
1 usage
def average_time_to_solve_func(df):
    total_time_to_solve = df['timeForSolve'].sum()
    total_defects_count = df.shape[0]

    average_time_per_defect = total_time_to_solve / total_defects_count if total_defects_count > 0 else None

    average_time_per_functionality = df.groupby('functionalityID')['timeForSolve'].mean()

    result = {
        "common": {
            "avgTime": average_time_per_defect,
            "status": 1 if average_time_per_defect <= 30 else 0
        },
        "byFunctionality": []
    }

    for functionality_id, time in average_time_per_functionality.items():
        result["byFunctionality"].append({
            "id": functionality_id,
            "avgTime": time,
            "status": 1 if time <= 30 else 0
        })

    return result


@app.route( rule: '/process', methods=['POST'])
def process_data():
    data_string = request.data.decode('utf-8')
```

Рисунок 4.11 – Код обчислення середнього часу вирішення дефектів

Джерело: побудовано автором (знімок з екрану)

```python
@app.route( rule: '/process', methods=['POST'])
def process_data():
    data_string = request.data.decode('utf-8')

    intermediate_data = json.loads(data_string)

    if isinstance(intermediate_data, str):
        final_data = json.loads(intermediate_data)
    else:
        final_data = intermediate_data

    bug_data = pd.DataFrame(final_data)

    by_severity = series_to_list(bug_data['severityID'].value_counts())
    by_priority = series_to_list(bug_data['priorityID'].value_counts())
    by_sprint = series_to_list(bug_data['sprint'].value_counts().sort_index())
    most_bugs_by_functionality = series_to_list(bug_data['functionalityID'].value_counts()[:10])

    product_environment = bug_data['Platform'].str.extract(r'(\w+)\((\w+)\)')
    product_environment.columns = ['label', 'environment']
    bug_data_with_env = bug_data.join(product_environment)

    defect_density = series_to_list(bug_data_with_env.groupby(['label', 'environment']).size().unstack().fillna(0))

    missed_bugs_prod = missed_bugs_prod_func(bug_data)

    avg_time_solve = average_time_to_solve_func(bug_data)
```

Рисунок 4.12 – Аналіз та розрахунок метрик пропущених дефектів на production середовище

Джерело: побудовано автором (знімок з екрану)

Для відображення проаналізованих даних у вигляді діаграм була використана бібліотека ChartJS [39] (рис. 4.13).

```php
<div class="d-flex flex-column w-100">
    <h5> Distribution of Bugs by Priority  </h5>
    <?php
    $priorityData = array_map(function ($item) {
        return $item['count'];
    }, $data['byPriority']);
    $priorityLabels = array_map(function ($item) {
        return \app\models\Priority::findOne(['id' => $item['priorityID']])['name'];
    }, $data['byPriority']);

    $colors = ['#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0', '#9966FF', '#FF9F40', '#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0'];

    $colors = array_slice(array_merge($colors, $colors),  offset: 0, count($priorityData));
    ?>

    <?= ChartJs::widget([
        'type' => 'bar',
        'data' => [
            'labels' => $priorityLabels,
            'datasets' => [
                [
                    'data' => $priorityData,
                    'backgroundColor' => $colors
                ]
            ]
        ]
    ]) ?>

</div>
</div>
```

Рисунок 4.13 – Відображення проаналізованих даних за допомогою бібліотеки ChartJS

Джерело: побудовано автором (знімок з екрану)

4.4 Демонстрація роботи web-орієнтованої системи

Web-орієнтована система автоматизації процесу аналізу якості програмного продукту має лише одного користувача. Це – тестувальник. На головному екрані зображена сторінка з виводом результатів аналізу. Вона має фільтр і 3 вкладки: діаграми («Dashboards»), метрики у вигляді таблиць («Metrics») та висновки по кожній з метрик («Conclusions») (рис. 4.14).

Сторінка «Metrics» містить 5 основних діаграм. Перші з них – це Розподіл дефектів за Серйозністю («Distribution of Bugs by Severity») та Розподіл дефектів за Пріоритетом («Distribution of Bugs by Priority»). Вони зображені на рисунку 4.15. Далі йдуть діаграми Тренд змін дефектів за спринтами («Trends in Bug Discovery by Sprint») із лінією тренду обчисленою за лінійною регресією (рис. 4.16), Функціональність із найбільшою кількістю дефектів («Functionalities with Most bugs») (рис. 4.17) та Щільність дефектів за платформами («Defect Density») (рис. 4. 18).

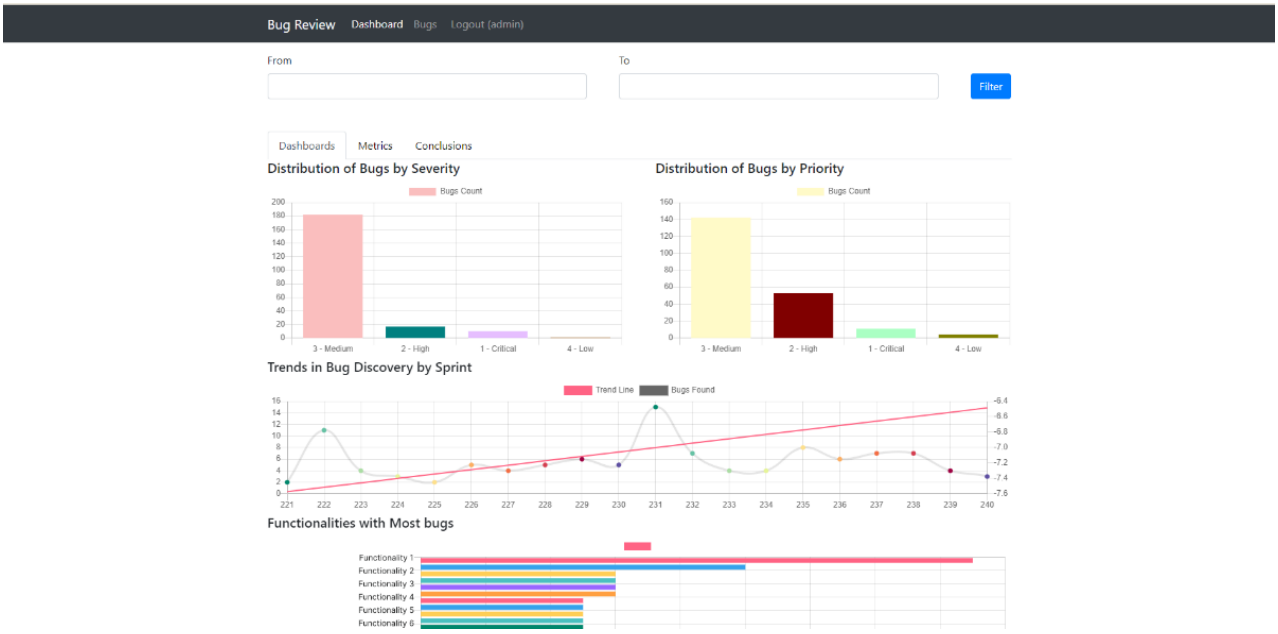


Рисунок 4.14 – Головна сторінка web-орієнтованої системи

Джерело: побудовано автором (знімок з екрану)

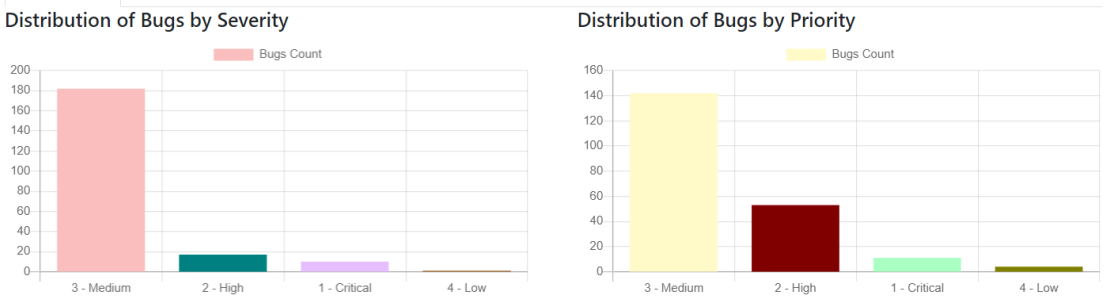


Рисунок 4.15 – Діаграма розподілу дефектів за серйозністю та пріоритетом

Джерело: побудовано автором (знімок з екрану)

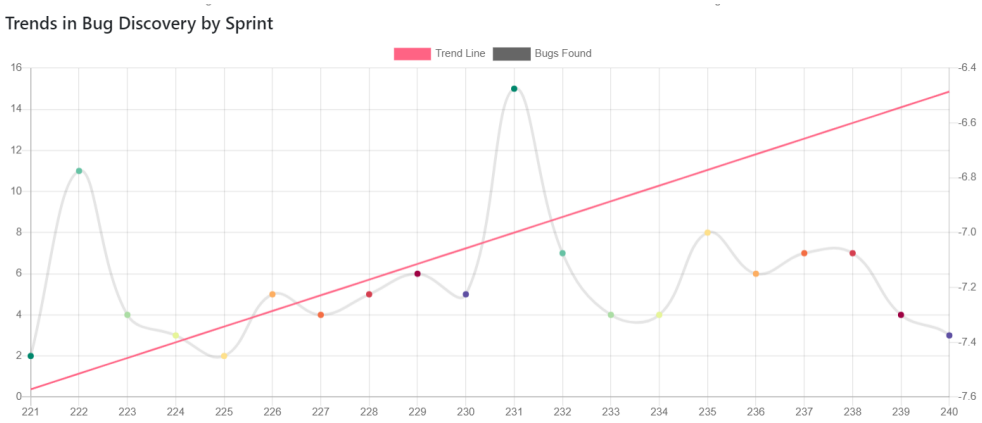


Рисунок 4.16 – Діаграма тренду змін кількості дефектів за спринтами

Джерело: побудовано автором (знімок з екрану)

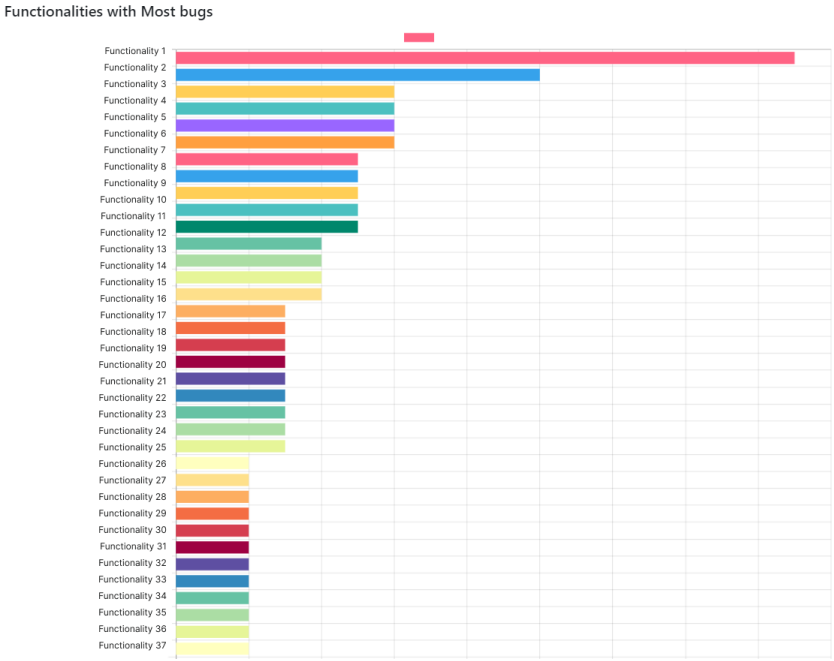


Рисунок 4.17 – Діаграма функціональностей з найбільшою кількістю дефектів

Джерело: побудовано автором (знімок з екрану)

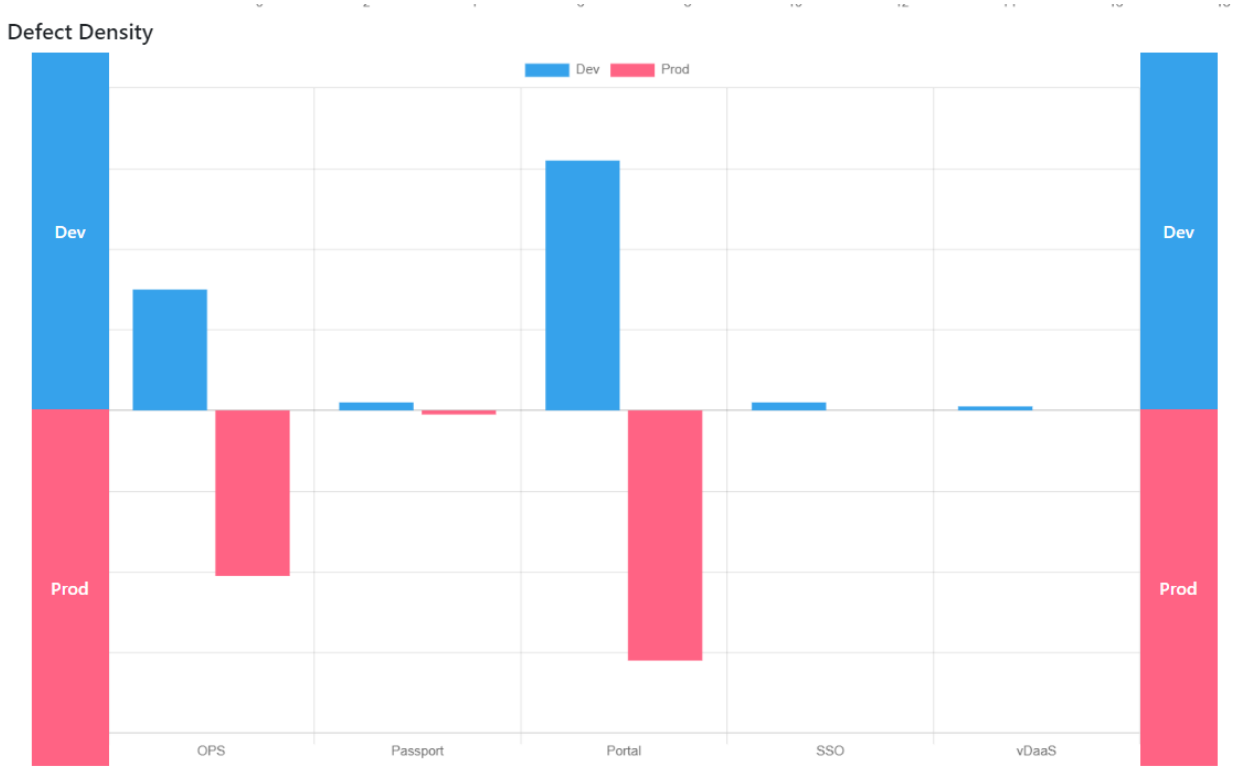


Рисунок 4.18 – Діаграма функціональностей з найбільшою кількістю дефектів

Джерело: побудовано автором (знімок з екрану)

На вкладці «Metrics» можна переглянути обчислення метрик «Defect Leakage» – відсоток пропущених дефектів на production середовище, середній час виправлення багів для функціональних частин та загальний середній час на виправлення всіх помилок (рис. 4.19). Далі працює наступне правило. Якщо значення виправлення дефекту для певного функціоналу буде більше, ніж середній час на виправлення всіх дефектів, значення колонки «Trend» буде червоним, і це є індикатором, на який треба звернути увагу. У іншому ж випадку значення буде зеленим.

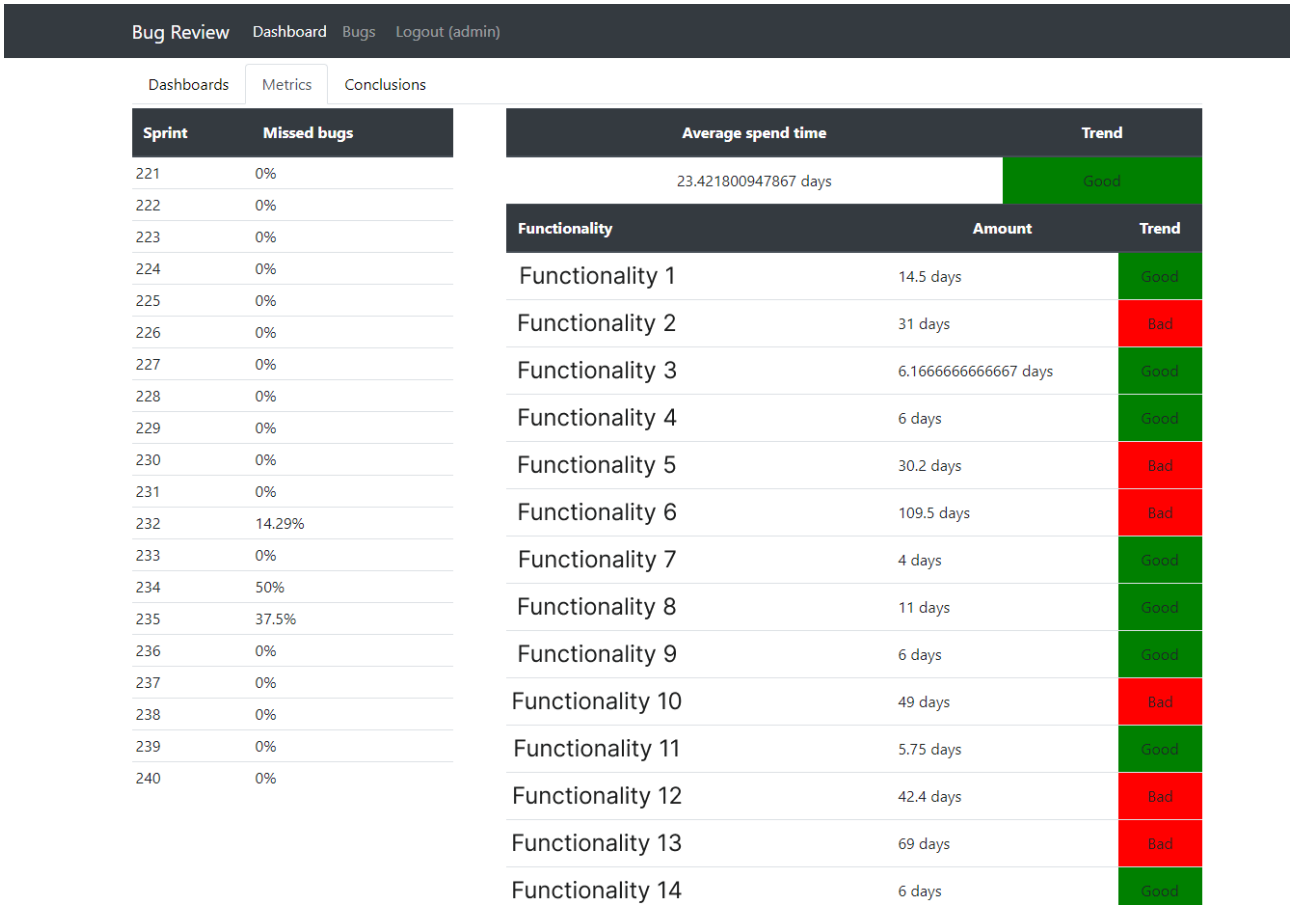


Рисунок 4.19 – Вкладка з метриками обрахунку середнього часу виправлення дефекту та пропущених дефектів на production продукту

Джерело: побудовано автором (знімок з екрану)

Вкладка «Conclusions» демонструє висновки та рекомендації по кожній з діаграм та метрик. Вона також включає в себе особливу функцію. Це обрахунок деяких метрик, які не потребують візуалізації. Сама вкладка та результати аналізу помилок за серйозністю та пріоритетом представлені на рисунку 4.20. Підсумки дослідження тенденцій та аналіз топ-5 функціоналу за кількістю помилок зображені на рисунку 4.21. Результати аналізу щільності дефектів за платформами та середовищами показано на рисунку 4.22. Підсумки аналізу показника неопрацьованих дефектів, багів пропущених на production і середнього часу на виправлення дефектів продемонстровано на рисунку 4.23.

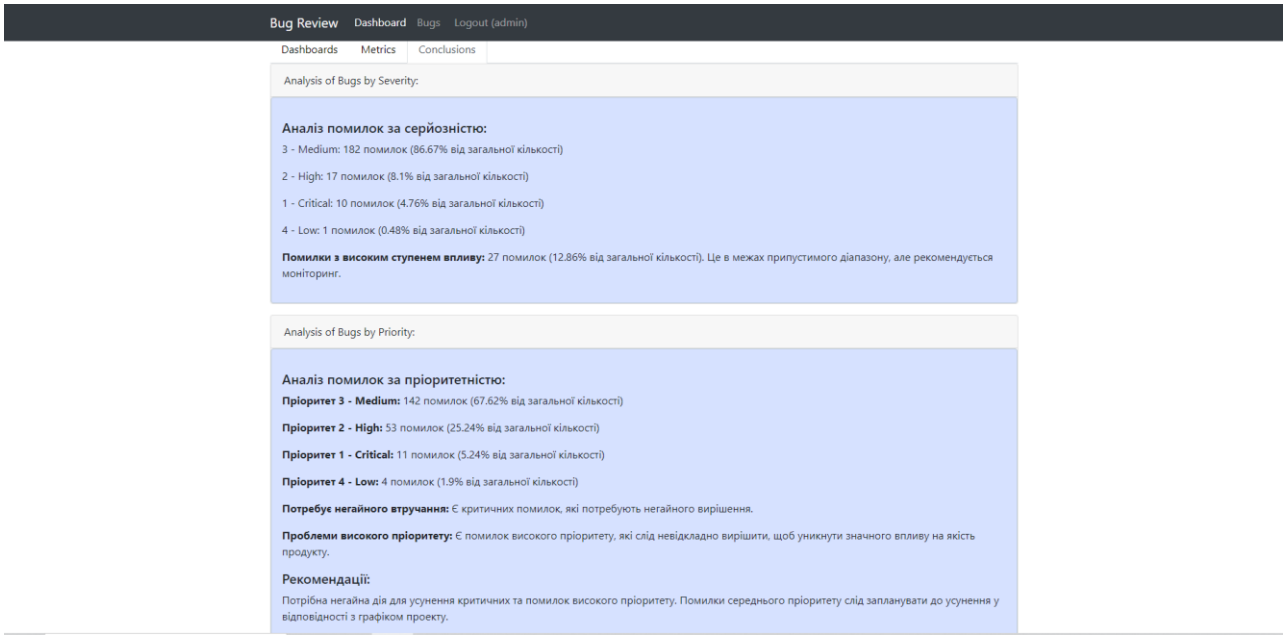


Рисунок 4.20 – Вкладка «Conclusions», та висновки з аналізу дефектів за серйозністю та пріоритетом

Джерело: побудовано автором (знімок з екрану)

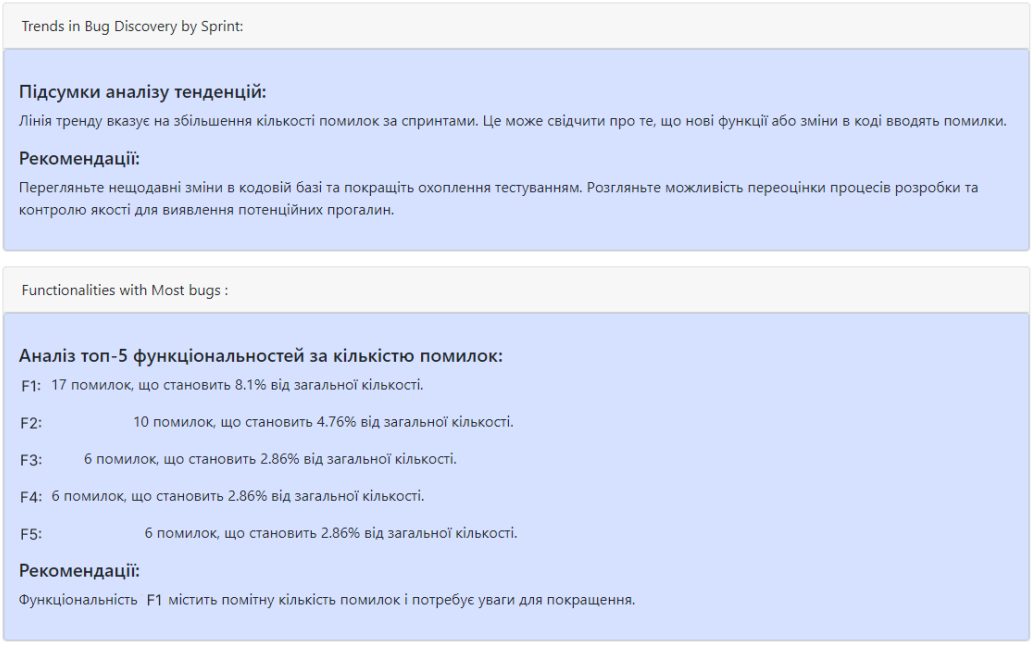


Рисунок 4.21 – Результати аналізу тенденцій та функціоналу за кількістю помилок

Джерело: побудовано автором (знімок з екрану)

Рисунок 4.22 – Результат аналізу щільності дефектів за платформами та середовищем розробки

Джерело: побудовано автором (знімок з екрану)

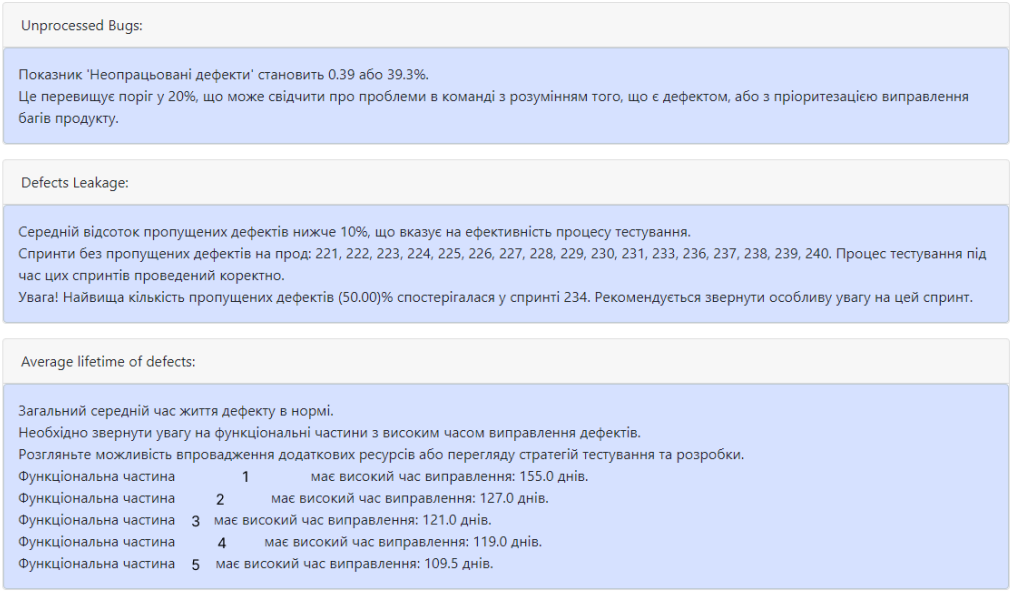


Рисунок 4.23 – Результати аналізу показнику неопрацьованих дефектів, пропущених дефектів на production, та середнього часу виправлення дефектів

Джерело: побудовано автором (знімок з екрану)

Застосувавши фільтр по даті можна оцінити, які показники дефектів програмного продукту були більш популярні у той чи інший проміжок часу. Результат його використання показано на рисунку 4.24. Фільтр також застосовується й для розрахунку метрик та результатів аналізу.

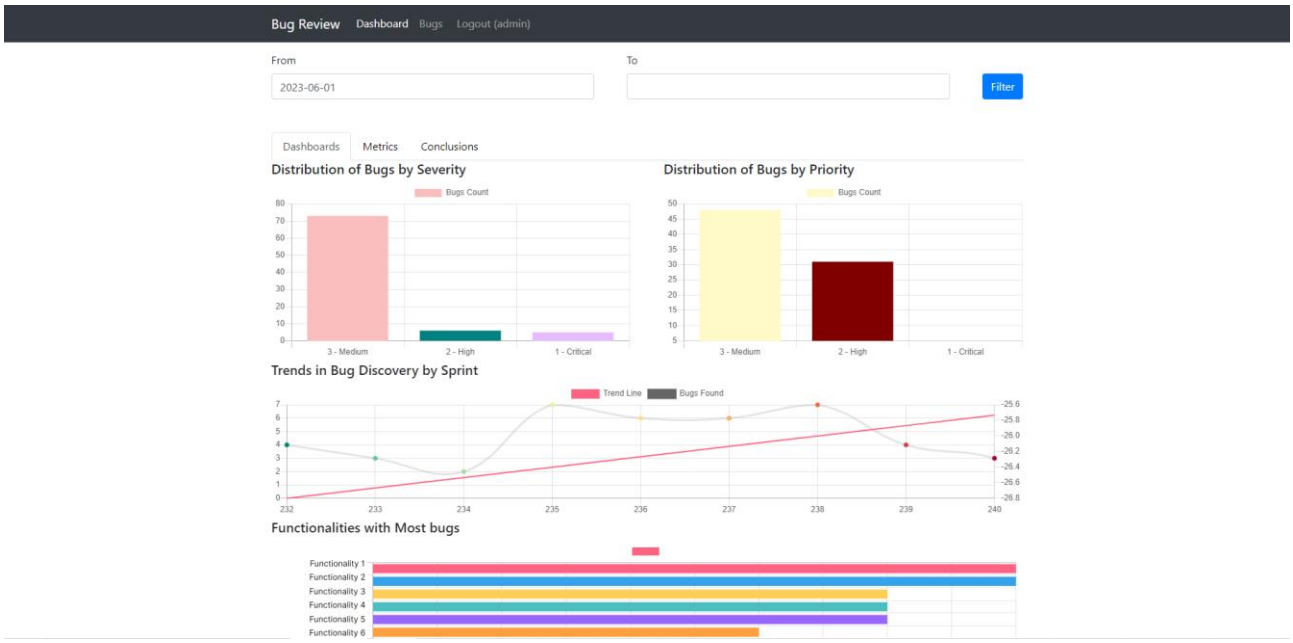


Рисунок 4.24 – Застосування часового фільтру до метрик

Джерело: побудовано автором (знімок з екрану)

Наступною, не менше важливою складовою даної web-орієнтованої системи, є сторінка Bugs. Вона дозволяє переглядати, додавати, редагувати, видаляти та імпортувати дані про дефекти. Потім, використовуючи саме цю інформацію, і формуються всі аналітичні розрахунки метрик, візуалізація підсумків аналізу за допомогою діаграм, а також формування результатів аналізу значень метрик. Сторінка Bugs представлена на рисунку 4.25. Вона має 4 основні кнопки, які відповідають за додавання нових записів до БД дефектів, обрання та імпорт файлу .csv з баг трекінгової системи, який містить інформацію про помилки, та видалення всіх записів про баги. Нижче зображена певна таблиця. Вона містить ID – унікальний ідентифікатор дефекту, його назву, ім’я, ким був знайдений баг, платформу, дату створення, серйозність та пріоритет помилки.

Таблиця також має «Actions» колонку. За її допомогою можна переглядати, редагувати та видаляти запис про дефект.

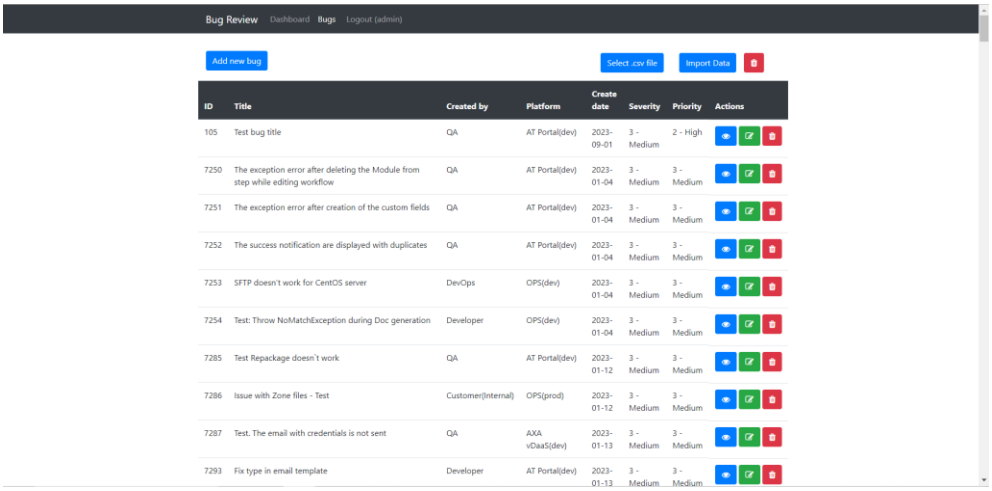


Рисунок 4.25 – Сторінка «Bugs» з основною інформацією за засобами управління дефектами

Джерело: побудовано автором (знімок з екрану)

Для створення нового запису у базі даних про дефект необхідно натиснути на кнопку «Add new bug» і на сторінці, яка відкриється, увести в форму основні відомості про баг (рис. 4.26).

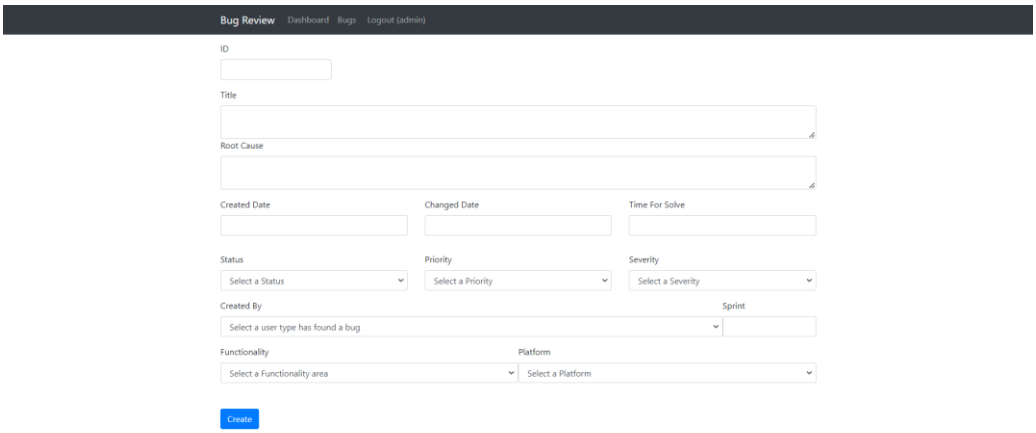


Рисунок 4.26 – Сторінка створення запису про новий дефект

Джерело: побудовано автором (знімок з екрану)

Після заповнення всіх необхідних полів і натиснення кнопки «Create» повідомлення про успішно збережені дані відобразиться на екрані (рис. 4.27) та новий запис буде показано вгорі таблиці дефектів на сторінці «Bugs» (рис. 4.28).

Рисунок 4.27 – Повідомлення про успішне збереження даних про новий дефект

Джерело: побудовано автором (знімок з екрану)

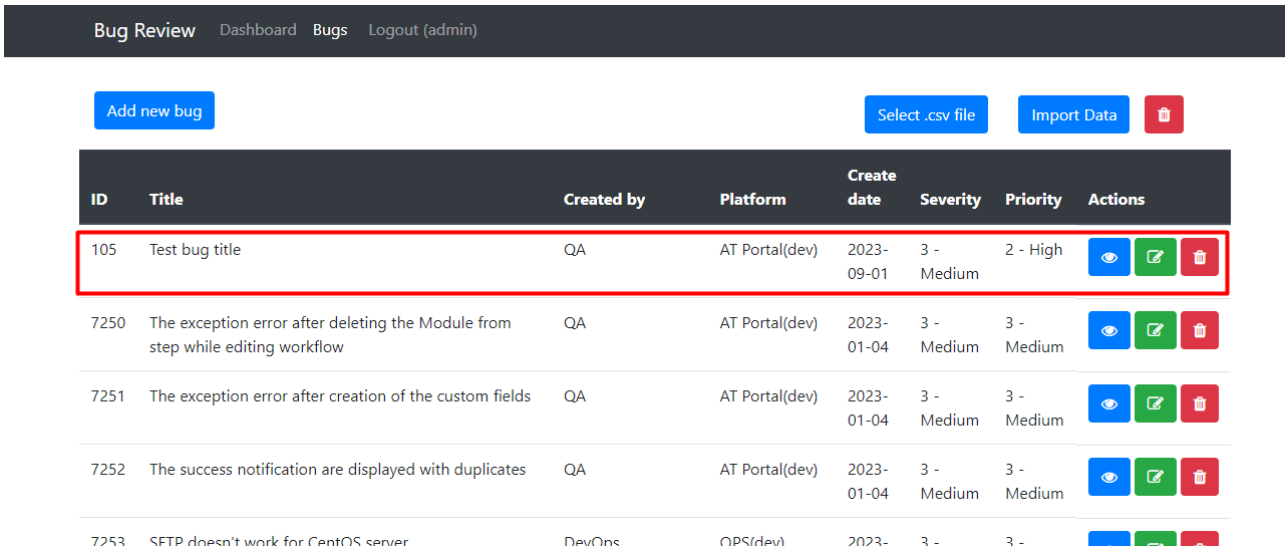


Рисунок 4.28 – Відображення створеного нового дефекту в таблиці «Bugs»

Джерело: побудовано автором (знімок з екрану)

Для перегляду інформації про дефект необхідно натиснути на кнопку «Переглянути». Вона продемонстрована за допомогою іконки у вигляді ока. У результаті буде відкрита нова сторінка зі всією інформацією про баг (рис. 4.29).

Для редагування інформації про дефект необхідно натиснути на кнопку «Редагування». Вона представлена за допомогою іконки пера. У результаті буде відкрита нова сторінка із input полями для редагування інформації про помилку (рис. 4.30). Після натиску на кнопку «Update» буде показане відповідне повідомлення про успішне оновлення даних (рис. 4.31). Оновлена інформація відобразиться на сторінці з таблицею багів.

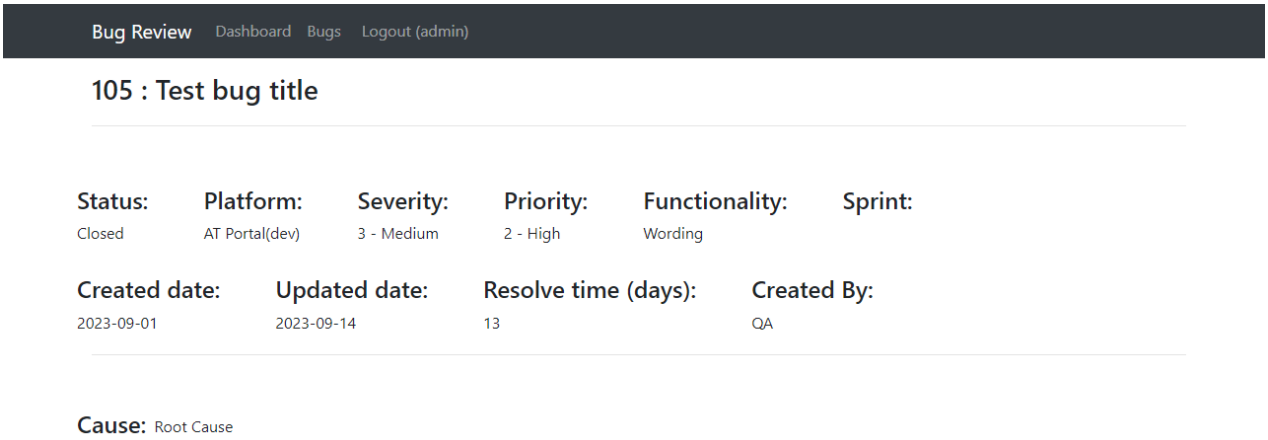


Рисунок 4.29 – Сторінка перегляду інформації про дефект

Джерело: побудовано автором (знімок з екрану)

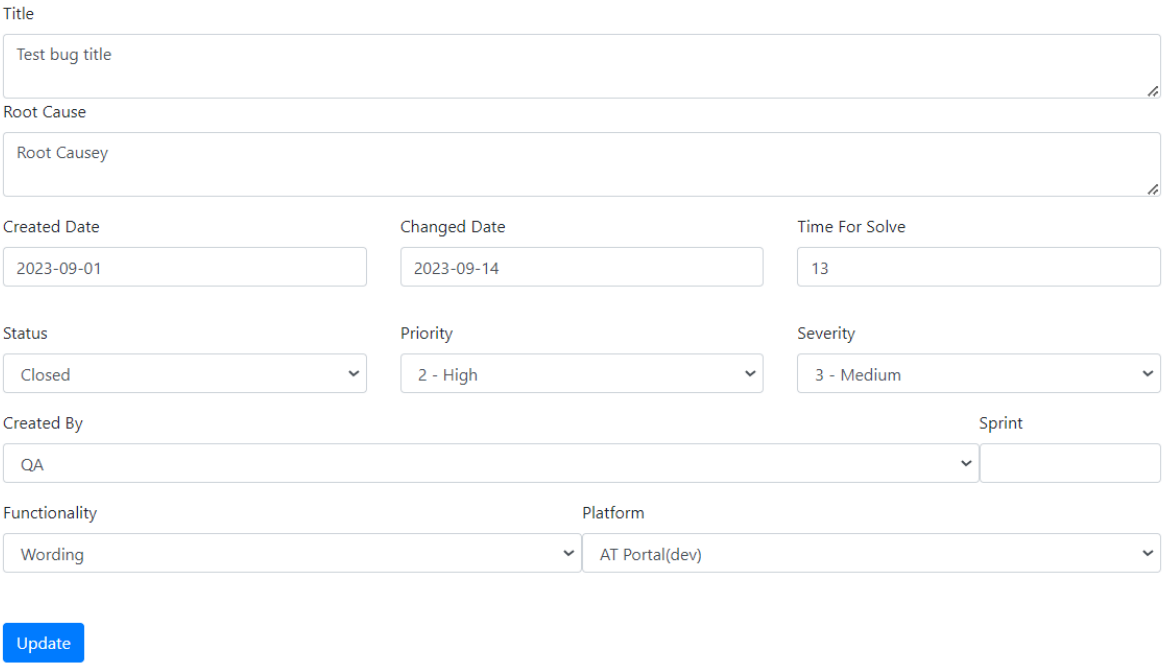


Рисунок 4.30 – Форма редагування інформації про дефект

Джерело: побудовано автором (знімок з екрану)

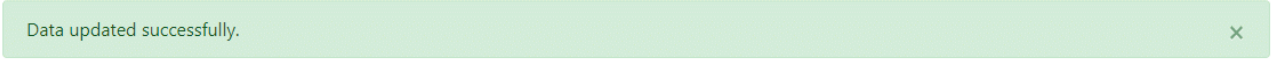


Рисунок 4.31 – Повідомлення про успішне оновлення даних про баг

Джерело: побудовано автором (знімок з екрану)

Для видалення інформації про дефект необхідно натиснути на кнопку «Видалення». Вона зображена за допомогою іконки сміттєвого бака. У результаті запис буде миттєво видалено з таблиці.

Для імпорту .csv файлу необхідно натиснути на кнопку «Select .csv file». Після цього відкриється вікно з вибором певного об’єкту з файлового носія (рис. 4.32). Далі необхідно натиснути на кнопку «Import data» і дані успішно будуть відображені у таблиці «Bugs».

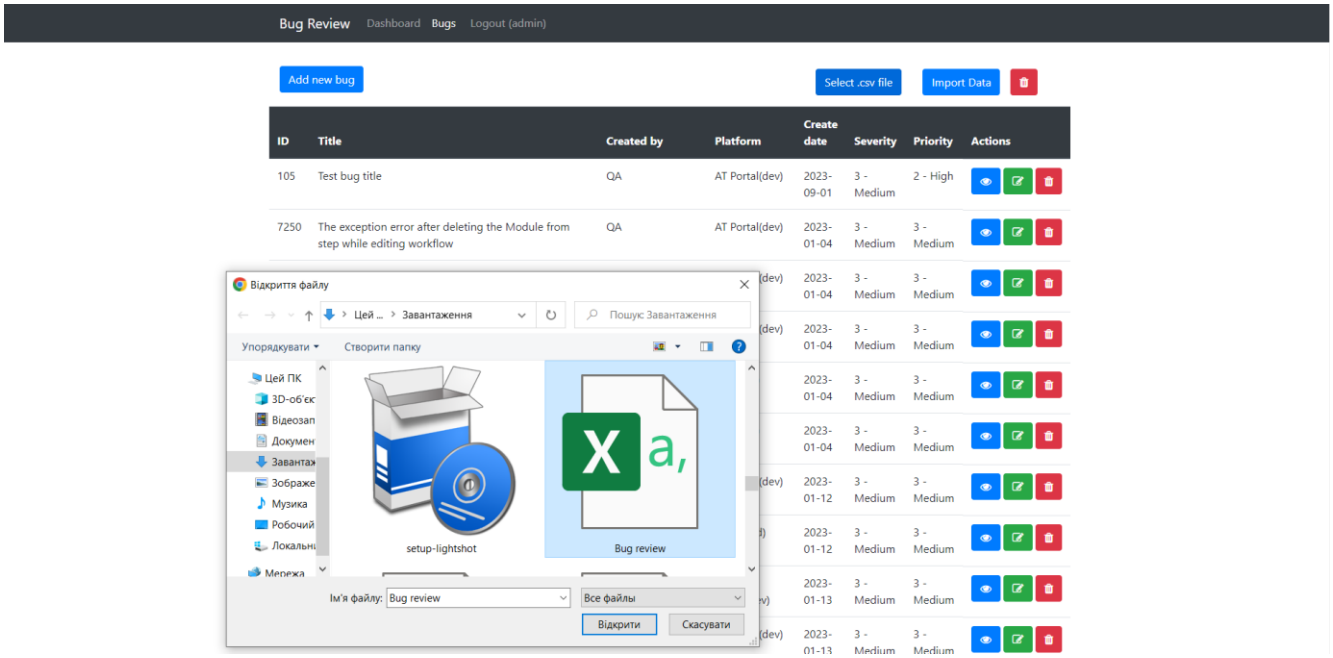


Рисунок 4.32 – Вибір .csv файлу з дефектами для імпорту

Джерело: побудовано автором (знімок з екрану)

4.5 Тестування web-орієнтованої системи

Для забезпечення надійності та ефективності функцій розробленої web-орієнтованої системи було виконано функціональне тестування. Цей процес включає перевірку відображення інформації та коректності обробки вхідних даних у формах для створення багів, а також аналіз правильності формату

завантажених файлів. Ключовий функціонал даної web-орієнтованої системи також був протестований.

Першою задачею було перевірити працездатність гіперпосилань у головному меню («Bugs» та «Dashboard»), щоб упевнитися, що вони коректно перенаправляють на відповідні сторінки та відображають актуальну інформацію.

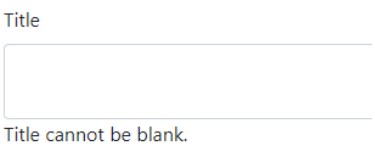
Далі були протестовані процеси створення, редагування та видалення записів про дефекти. Також здійснено перевірку їх відображення на сторінці «Bugs». Помилок виявлено не було У підсумку аналіз даних у вигляді діаграм та метрик на сторінці «Dashboards» також показав задовільні результати.

Під час тестування була перевірена функція завантаження файлів, їх обробка та зберігання в базі даних, яка відбулася належним чином.

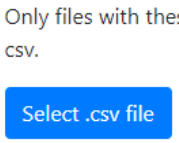
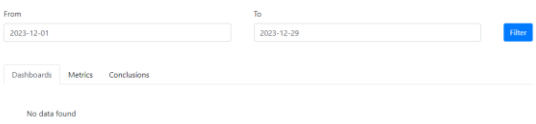
Використаний метод тестування – «чорний ящик» – дозволив оцінити працездатність даної web-орієнтованої системи без втручання у її внутрішню структуру та код, зосередившись виключно на аналізі вихідних результатів. А також на поведінці запропонованого програмного продукту [40].

Результати проведеного тестування надано в таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування валідними та невалідними даними

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
1	Введення коректних даних до форми створення баг репорту	Баг репорт успішно створений	Створений баг репорт відображається у таблиці на сторінці «Bugs»	1
2	Перевірка форми створення баг репорту введенням некоректних даних	Відображається текст повідомлення щодо обов'язковості введення значення		1

Продовження табл. 4.1

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
3	Імпорт .csv файлу	Файл успішно завантажений.	Файл успішно завантажений, та дані з нього коректно імпортовані до бази даних.	1
4	Імпорт будь якого, але не .csv файлу	Повідомлення про неприйнятний тип файлу, яке відображається біля кнопки імпорту		1
5	Фільтрація результатів аналізу валідними даними	Діаграми, метрики, та висновки формуються на основі обмеження по даті	Усі дані на сторінці «Dashboards» відображаються у відповідності з датами введеними до фільтру	1
6	Фільтрація результатів аналізу невалідними даними	Повідомлення про те, що немає даних для відображення виводиться на сторінку.		1

Джерело: побудовано автором

Після проведення функціонального тестування критичних дефектів не виявлено. Web-орієнтована система працює коректно.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено проєктування та реалізацію web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту, орієнтуючись на наступні визначені функціональні вимоги:

– імпортування таблиці дефектів з .csv файлу;
– можливість створення та заповнення таблиці з дефектами на сайті;
– можливість редагування та видалення даних про дефекти;
– можливість аналізу якості програмного продукту за метриками та стандартами якості ПЗ;
– візуалізація за допомогою діаграм результатів аналізу;
– відображення результатів аналізу за метриками програмного забезпечення.

Представлений програмний продукт матиме високий попит серед IT компаній, які зацікавлені у процесі Bug review та поліпшенні роботи команди розробки та тестування шляхом знаходження слабких місць у життєвому циклі створення ПЗ.

Було виконано аналіз предметної області. У ході цього визначено потребу в розробці web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту. На даний момент немає повних реалізованих аналогів запропонованого рішення. Також було здійснено аналіз програмних продуктів зі схожим призначенням. Виділено їх переваги, на які треба звернути увагу під час створення даного проєкту, і недоліки, які варто доопрацювати та подолати.

Далі було визначено мету, сформовано функціональні та нефункціональні вимоги до створюваної системи, а також виконано постановку задач на реалізацію. Крім того, здійснено детальне планування робіт проєкту. У свою чергу, це дозволило оцінити часові рамки його виконання.

Наступним етапом було проєктування роботи та сценаріїв використання даної web-орієнтованої системи за допомогою діаграми нотації IDEF0 та її деталізації через декомпозицію. Також із застосуванням Use Case діаграми. Вона демонструє різні варіанти використання представленого продукту. Дана методика дозволила чітко сформувати функції представленого ПЗ і розподілити їх на менші задачі.

Також було спроєктовано структуру бази даних, і виділено основні її сутності.

Представлена web-орієнтована система була створена з використанням фреймворку Yii2 та Flask для розробки бекенду. MySQL використано в якості системи управління базами даних. Bootstrap задіяно для розробки фронтенду. Javascript використано для надання програмному продукту динамічності, а Python та бібліотека pandas – для швидкого аналізу даних.

Після розробки web-орієнтованої системи було проведено функціональне тестування. Після нього не було виявлено критичних дефектів.

Отже, розроблена web-орієнтована система автоматизації процесу аналізу якості програмного забезпечення дозволить виконати ряд речей, а саме: вдосконалити процедуру Bug Review процесу, інтегрувати аналіз якості програмного продукту в життєвий цикл розробки ПЗ, а також покращити рівень та ефективність роботи розробників та тестувальників.

Результати даного дослідження апробовано на конференції АПКН-2023. Опубліковані тези надано в Додатку Б.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fraser G., Rojas J.M. Software Testing. P. 1–72.
2. Кузь М.В. et al. ПРОГНОЗУВАННЯ ЯКОСТІ ПРОГРАМНИХ ЗАСОБІВ НА ОСНОВІ АНАЛІЗУ ЯКОСТІ ВИМОГ // METHODS AND DEVICES OF QUALITY CONTROL. 2023. № 1(50).
3. Смелик Яна. Аналіз якості програмного забезпечення за допомогою програмних метрик. Київ: КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО, 2022. 1–114 p.
4. Singh M., Mittal A., Kumar S. Survey on Impact of Software Metrics on Software Quality // International Journal of Advanced Computer Science and Applications. 2012. Vol. 3, № 1.
5. Gao K. et al. Choosing software metrics for defect prediction: An investigation on feature selection techniques // Softw Pract Exp. 2011. Vol. 41, № 5.
6. Tran H.M. et al. An Analysis of Software Bug Reports Using Machine Learning Techniques // SN Comput Sci. 2020. Vol. 1, № 1.
7. Lee D.G., Seo Y.S. Systematic review of Bug report processing techniques to improve software management performance // Journal of Information Processing Systems. 2019. Vol. 15, № 4.
8. Hirsch T., Hofer B. Root cause prediction based on bug reports // Proceedings - 2020 IEEE 31st International Symposium on Software Reliability Engineering Workshops, ISSREW 2020. 2020.
9. Медведєва К.С., Антипенко В.П. WEB-ОРІЄНТОВАНА СИСТЕМА АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ // Актуальні проблеми комп’ютерних наук АПКН-2023. 2023. P. 187–189.
10. Software Analytics - Dashboard for Project Quality and Performance | Vector [Electronic resource]. URL: https://www.vector.com/int/en/products/products-a-

z/software/squore/squore-software-analytics-for-project-monitoring/#c121789 (accessed: 23.11.2023).

11. Code Quality, Security & Static Analysis Tool with SonarQube | Sonar [Electronic resource]. URL: https://www.sonarsource.com/products/sonarqube/ (accessed: 23.11.2023).

12. What are the 7 axes of source code quality? [Electronic resource]. URL: https://www.panel.es/en/software-quality-the-7-axes-of-source-code-quality/ (accessed: 07.11.2023).

13. Lacerda G. et al. Code smells and refactoring: A tertiary systematic review of challenges and observations // Journal of Systems and Software. 2020. Vol. 167.

14. SonarQube Web Interface. | Download Scientific Diagram [Electronic resource]. URL: https://www.researchgate.net/figure/SonarQube-Web-Interface_fig24_280565995 (accessed: 08.12.2023).

15. Squore - Augmented Software Analytics | Vector [Electronic resource]. URL: https://www.vector.com/int/en/products/products-a-z/software/squore/# (accessed: 08.11.2023).

16. What is a .CSV file and what does it mean for my ecommerce business? | BigCommerce [Electronic resource]. URL: https://www.bigcommerce.com/ecommerce-answers/what-csv-file-and-what-does-it-mean-my-ecommerce-business/ (accessed: 29.11.2023).

17. What Are the Different Types of Scientific Research? [Electronic resource]. URL: https://akjournals.com/page/types-of-scientific-research (accessed: 30.11.2023).

18. Шатова В.М. ПРАКТИЧНЕ ЗАСТОСУВАННЯ ЕМПІРИЧНИХ МЕТОДІВ ДОСЛІДЖЕННЯ // Економічні проблеми сталого розвитку. 2014.

19. Методи теоретичного дослідження — Українська педагогіка [Electronic resource]. URL: https://ukped.com/statti/skarbnichka/420-metody-teoretychnoho-doslidzhennia.html (accessed: 16.11.2023).

20. Комплексно-комбіновані методи дослідження систем управління [Electronic resource]. URL:

https://stud.com.ua/16701/investuvannya/kompleksno_kombinovani_metodi_doslidzhennya_sistem_upravlinnya (accessed: 29.11.2023).

21. What is Use Case Diagram? [Electronic resource]. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (accessed: 13.11.2023).

22. Д.В. Ломотько et al. Методи наукових досліджень. УкрДАЗТ, 2014. P. 1–79.

23. Нотація IDEF0 [Electronic resource]. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%202%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B2%D0%B0/page9.html (accessed: 10.11.2023).

24. Lazić L. Software Errors Analysis and Prevention. 1–38 p.

25. 34 Software Testing Metrics And KPIs: Complete Guide 2023 | ThinkSys Inc. [Electronic resource]. URL: https://thinksys.com/qa-testing/software-testing-metrics-kpis/ (accessed: 30.11.2023).

26. Василенко І. С. ВИВЧЕННЯ ЗАСОБІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ДЛЯ ОЦІНЮВАННЯ ПОКАЗНИКІВ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ: Кваліфікаційна робота магістра. Харків: Харківський національний університет радіоелектроніки, 2021. 1–88 p.

27. Що таке html? [Electronic resource]. URL: https://css.in.ua/article/shcho-take-css_3 (accessed: 23.11.2023).

28. Що таке CSS [Electronic resource]. URL: https://css.in.ua/article/shcho-take-html_10 (accessed: 23.11.2023).

29. JavaScript: The World’s Most Misunderstood Programming Language. Crockford.com.

30. Фреймворк Yii2 що це? [Electronic resource]. URL: https://avada-media.ua/ua/services/yii-yii2/ (accessed: 09.11.2023).

31. pandas - Python Data Analysis Library [Electronic resource]. URL: https://pandas.pydata.org/ (accessed: 09.11.2023).

32. MySQL [Electronic resource]. URL: https://www.mysql.com/ (accessed: 09.11.2023).

33. What Is a Database [Electronic resource]. URL: https://www.oracle.com/database/what-is-database/ (accessed: 05.12.2023).

34. Ved Mulkalwar. How to create a good High Level Design [Electronic resource] // Medium. 2023. URL: https://medium.com/@vedmkw/how-to-create-a-good-high-level-design-hld-fddba7f6ae18 (accessed: 03.12.2023).

35. Luna Modeler | Database Design Tool [Electronic resource]. URL: https://www.datensen.com/ (accessed: 14.12.2023).

36. Open Server: Reviews, Features [Electronic resource]. URL: https://alternativeto.net/software/open-server/about/ (accessed: 03.12.2023).

37. PhpStorm: PHP IDE and Code Editor from JetBrains [Electronic resource]. URL: https://www.jetbrains.com/phpstorm/ (accessed: 03.12.2023).

38. Welcome to Flask [Electronic resource]. URL: https://flask.palletsprojects.com/en/3.0.x/ (accessed: 14.12.2023).

39. Chart.js [Electronic resource]. URL: https://www.chartjs.org/ (accessed: 05.12.2023).

40. White/black/grey box-тестування [Electronic resource]. URL: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/ (accessed: 08.12.2023).

41. What is a Work Breakdown Structure (WBS) | Project Management [Electronic resource]. URL: https://www.workbreakdownstructure.com/ (accessed: 01.12.2023).

42. Organization Breakdown Structure (OBS) - PSA - EN [Electronic resource]. URL: https://uplandsoftware.com/psa/resources/glossary/organization-breakdown-structure-obs/ (accessed: 01.12.2023).

ДОДАТОК А

Планування робіт

За останні роки кожна IT компанія прагне автоматизувати всі можливі бізнес процеси. Особлива увага приділяється тим, які займають досить багато часу та мануальної роботи працівника. Це здійснюється заради оптимізації внутрішніх процесів. Також для вивільнення додаткового часу на розробку чи тестування. Автоматизація процесу аналізу якості програмного продукту принесе IT-компанії ряд переваг. Наприклад, дозволить підвищити ефективність виявлення та усунення дефектів. Також забезпечить стандартизацію операцій оцінки якості програмного продукту, підкріплюючи їх точними метриками та аналітикою. У свою чергу, це дозволяє приймати обґрунтовані рішення щодо поліпшення роботи команд розробки та тестування.

Деталізація мети методом SMART. Визначення чіткої мети на етапі концептуального проєктування – це показник швидкої та якісної реалізації всього проєкту. Визначена за допомогою методу SMART мета має наступне формулювання: «Автоматизація процесу аналізу якості програмного продукту за рахунок розробленої відповідної web-орієнтованої системи з використанням bug review документації для підвищення як продуктивності роботи команд розробки та тестування, так і якості процесу створення програмних продуктів до 15 грудня 2023 року». Результати деталізації мети проєкту методом SMART відображено у таблиці А.1.

Таблиця А.1 – Деталізація мети проєкту методом SMART

Specific (Конкретна)	Автоматизація процесу аналізу якості програмного продукту за рахунок розробки відповідної web-орієнтованої системи з використанням bug review документації.
Measurable (Вимірювана)	Результатом роботи дипломного проєкту є створена web-орієнтована система автоматизації процесу аналізу якості програмного продукту з реалізованими наступними функціями: створення баг репортів за допомогою web-орієнтованої системи, імпорт даних про баги з .csv файлу, аналіз інформації та подальша візуалізація проаналізованих даних у вигляді графіків, та вивід результатів обчислення метрик якості програмного продукту.
Achievable (Досяжна)	Для розробки проєкту необхідними є знання HTML, CSS, мов програмування PHP, JavaScript, Python, бібліотеки pandas, фреймворку Yii-2, бази даних MySQL, та навичок заведення документації.
Relevant (Реалістична)	Застосування даної web-орієнтованої системи дозволить підвищити як продуктивність роботи команд розробки та тестувальників, так і якість процесу розробки програмних продуктів.
Time-framed (Обмежена у часі)	Проєкт має часове обмеження. Термін досягнення його мети визначено з замовником – до 15 грудня 2023 року.

Джерело: побудовано автором

Планування змісту робіт. WBS або Work Breakdown Structure представляє собою декомпозицію загального обсягу робіт проєкту на більш дрібні задачі [41]. Ними легше управляти. Ця ієрархічна структура є ключовим елементом у плануванні проєкту, який допомагає систематизувати всі його задачі в окремі частини. Метою її розробки є оцінка термінів виконання завдань та організація командної роботи. WBS визначає задачі як ієрархічну структуру. На найвищому рівні знаходиться програмний продукт. Кожен нижчий представляє більше деталізоване визначення робочих задач. Декомпозиція завдань виконується до того моменту, поки вони не будуть елементарними, тобто доступними для ефективного управління та відстеження, але достатньо великими, щоб бути значущими. На рисунку А.1 зображено WBS на розробку web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту.

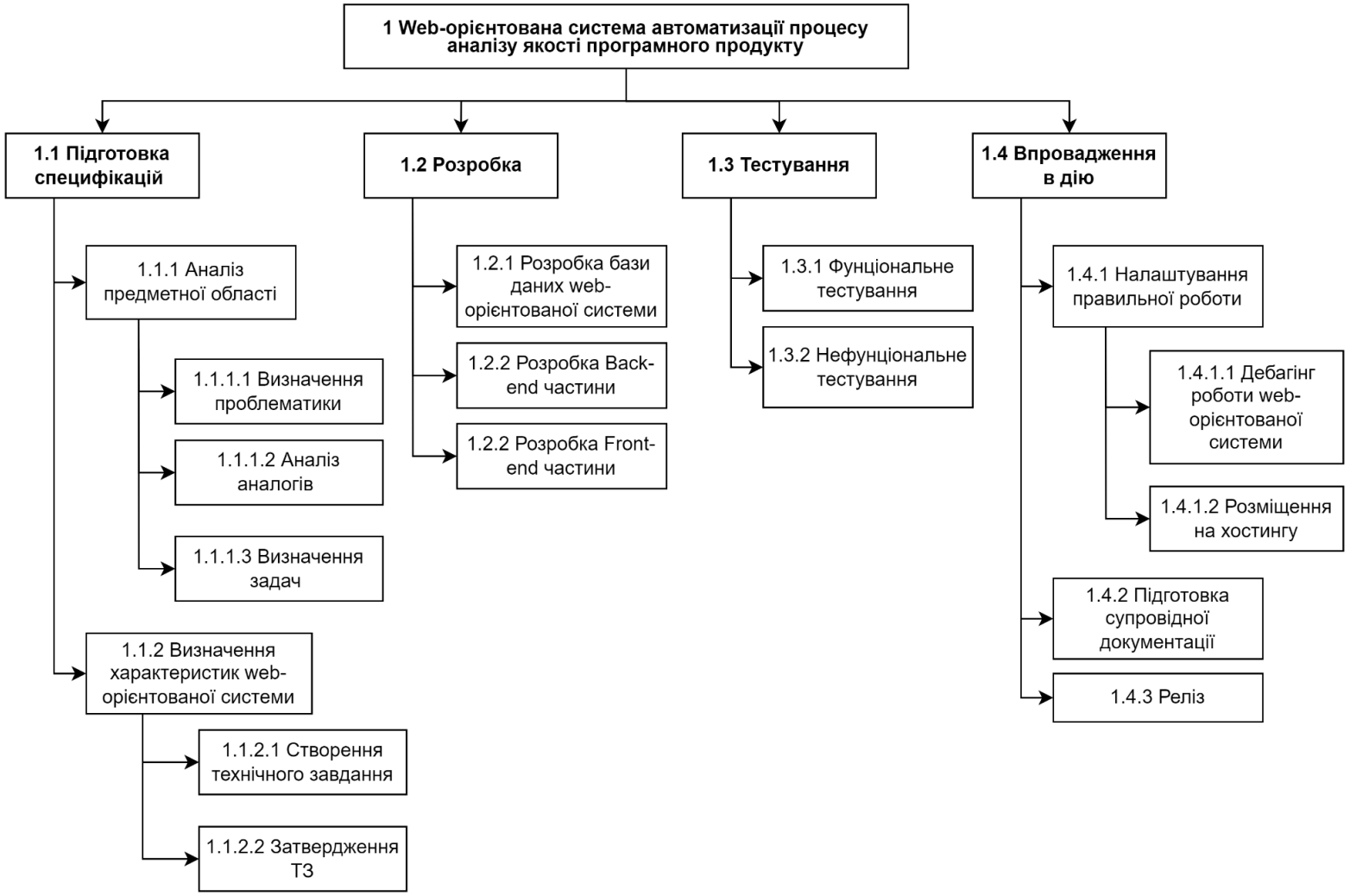


Рисунок А.1 – WBS-структура робіт проєкту

Джерело: побудовано автором

Планування структури виконавців. Розробка організаційної структури виконавців або OBS (Organization Breakdown Structure) є наступним етапом після декомпозиції робіт [42]. Вона використовується для представлення ієрархії команди в контексті проєкту. OBS також визначає її організацію та розподіл ресурсів у відповідності для досягнення мети. На рисунку А.2 представлено організаційну структуру даного проєкту. Інформацію про учасників проєкту було описано у таблиці А.2.

Таблиця А.2 – Виконавці проєкту

Роль	Ім’я	Проєктна роль
Розробник	Медведєва К.С.	Забезпечення frontend та backend розробок.
Тестувальник	Медведєва К.С.	Забезпечення тестування web-орієнтованої системи
Проєктувальник	Медведєва К.С.	Проведення проєктування структури програмного продукту та БД
Керівник проєкту	Антипенко В.П.	Формує завдання на розробку проєкту, рецензує готовий програмний продукт.
Менеджер	Медведєва К.С.	Контролює дотримання термінів, розподіл ресурсів та задач між виконавцями проєкту. Проводить аналіз та збір даних.

Джерело: побудовано автором

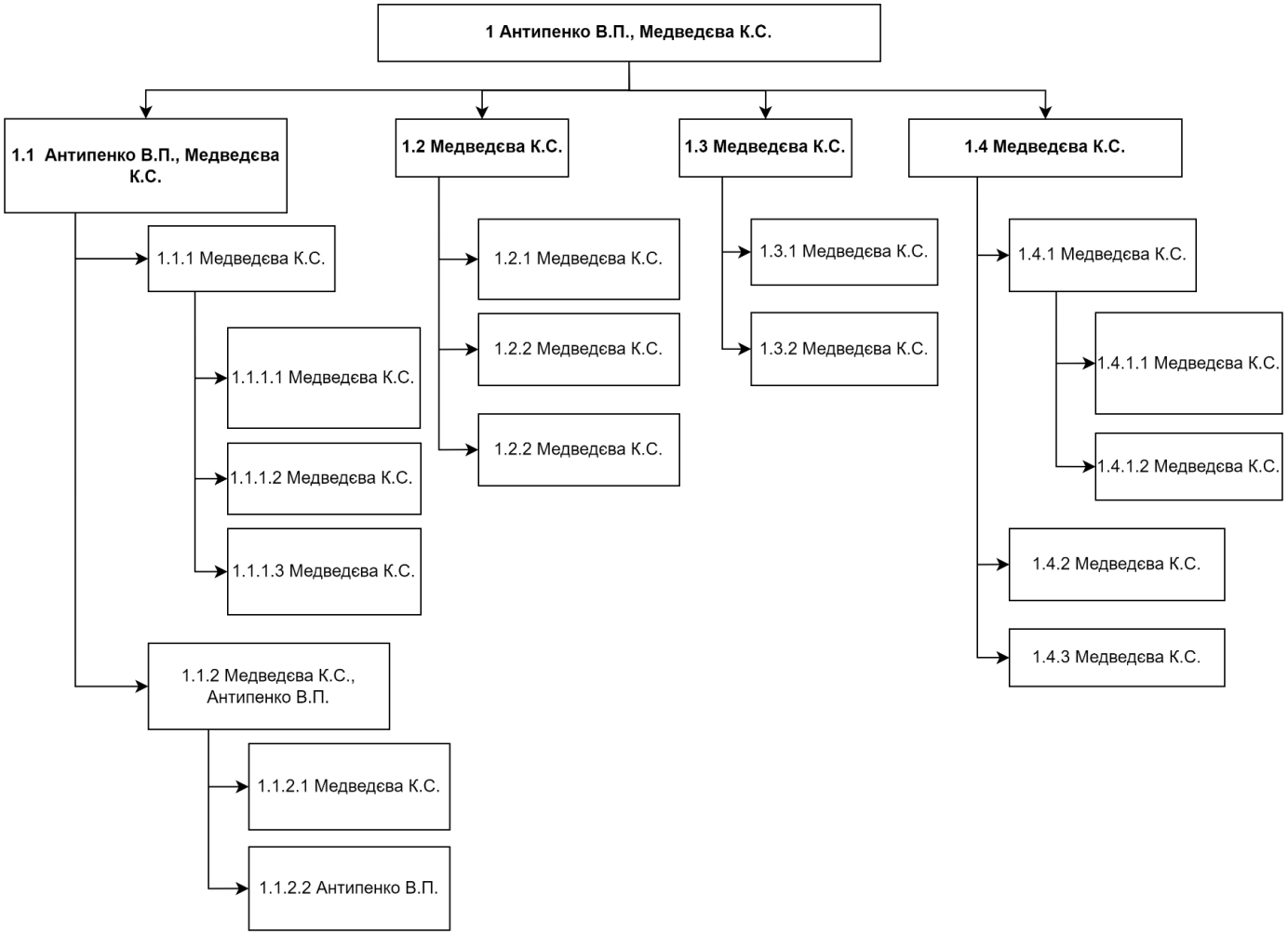


Рисунок А.2 – OBS-структура робіт проєкту

Джерело: побудовано автором

Діаграма Ганта. Створення календарного плану проєкту є невід'ємною частиною роботи над ним. Він має вигляд графіку з розподіленням дат між задачами. Календарний план допомагає оцінити тривалість виконання завдань проєкту. При цьому враховується наявність ресурсів.

Календарний графік даного проєкту зображено на рисунку А.3.

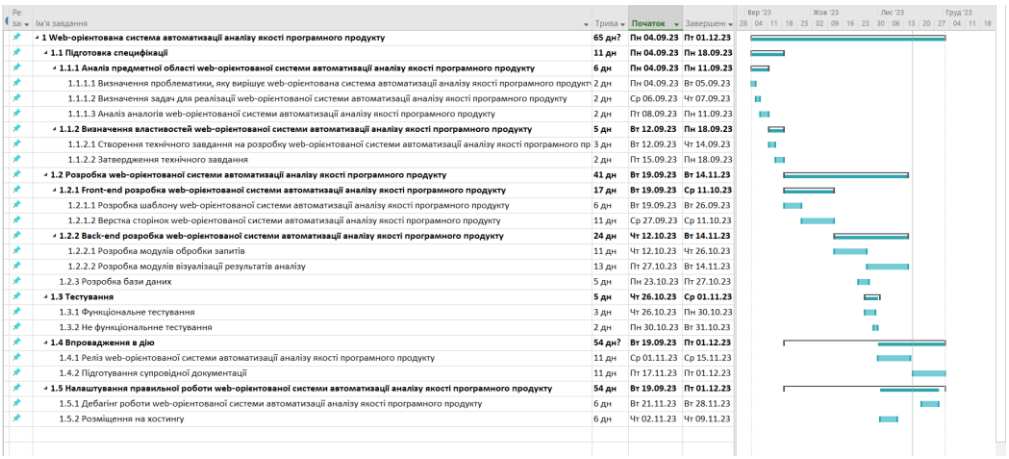


Рисунок А.3 – Діаграма Ганта проєкту

Джерело: побудовано автором (знімок з екрану)

Управління ризиками проєкту. Наступним етапом у плануванні робіт проєкту є оцінювання потенційних ризиків та розробка стратегій реагування на них.. Ризики даного проєкту було перелічено у таблиці А.3. У таблиці А.4 надано їхню оцінки. Таблиця А.5 показує шкалу ризиків за типом, величиною впливу та ймовірністю.

Таблиця А.3 – Ризики проєкту

№ ризику	**Назва (опис) ризику**
1	Недостатні знання у розробника
2	Зміни вимог замовником
3	Проблеми з доступом до мережі Інтернет

Продовження табл. А.3

№ ризику	**Назва (опис) ризику**
4	Недостатній обсяг фінансування
5	Недостатня підтримка проєкту з боку замовника
6	Хвороба виконавця
7	Відсутність механізму резервного копіювання
8	Поява альтернативного продукту
9	Виявлення складних для виправлення дефектів розробки проєкту
10	Відключення світла

Джерело: побудовано автором

Таблиця А.4 – Результати оцінки ймовірності, впливу та рангу ризику проєкту

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
1	Недостатні знання у розробника	0,1	0,3	0,03
2	Зміни вимог замовником	0,2	0,3	0,06
3	Проблеми з інтернетом	0,4	0,1	0,04
4	Недостатній обсяг фінансування	0,35	0,35	0,12
5	Недостатня підтримка проєкту з боку замовника	0,15	0,35	0,05
6	Хвороба розробника	0,15	0,7	0,10
7	Відсутність механізму резервного копіювання	0,15	0,4	0,06

Продовження табл. А.4.

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
8	Поява альтернативного продукту	0,5	0,25	0,12
9	Виявлення складних для виправлення дефектів розробки проєкту	0,4	0,3	0,12
10	Відключення світла	0,12	0,3	0,03

Джерело: побудовано автором

Таблиця А.5 – Шкала ризиків за типом, ймовірністю та величиною впливу.

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: побудовано автором

Для мінімізації негативного впливу потенційних ризиків на проєкт, необхідно розробити план реагування на них. Це включає оцінку ефективності стратегій впорядкування ризиків та аналіз можливих наслідків їхнього впливу на проєкт. Оцінка визначається за показниками, які описані в таблиці А.5. У результаті планування реагування було отримано матрицю ймовірностей впливу ризиків та виникнення ризиків, яка втілена у таблиці А.6. Зелений колір відповідає за прийнятні ризики, жовтий – за виправдані, червоний – недопустимі ризики.

Таблиця А.6 – Матриця ймовірності виникнення ризиків та їх впливу

Ймовірність виникнення ризику	**Вплив ризику**				
	0,05	**0,1**	**0,2**	**0,4**	**0,8**
0,9	0,045	0,09	0,18	0,3	0,72
0,7	0,035	0,07	0,14	0,28	0,56
0,5	0,025	0,05 **RS-5**	0,1 **RS-6**	0,2	0,4
0,3	0,015	0,03 **RS-1, RS-10**	0,06 **RS-2, RS-7**	0,12 **RS-4, RS-8, RS-9**	0,24
0,1	0,005	0,01	0,02	0,04 **RS-3**	0,08

Джерело: побудовано автором

Ризики класифіковано за рівнем відповідно до їх індексних значень, які представлені в Таблиці А.7. У Таблиці А.8 наведено детальний опис кожного ризику та розроблені стратегії реагування на них.

Таблиця А.7 – Шкала оцінювання ризику за рівнем

№	**Назва**	**Межі**	**Ризики, які входять(номера)**
1	Прийнятні	$0,005 \leq R \leq 0,05$	RS-1, RS-3, RS-5, RS-10
2	Виправдані	$0,05 \leq R \leq 0,14$	RS-2, RS-4, RS-6, RS-7, RS-8, RS-9
3	Недопустимі	$0,14 \leq R \leq 0,72$	

Джерело: побудовано автором

Таблиця А.8 – Ризики та стратегії реагування

ID	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**Тип стратегії реагування**	**План А**	**План Б**
RS-1	Відкритий	Недостатні знання у розробника	Низька	Середній	0,03	Зменшення	1. Підвищити кваліфікацію розробника. 2. Надати необхідні ресурси для підвищення знань. 3. Надати тестове завдання для розвитку необхідних навичок.	Видати необхідні ресурси(курси, література), виділити час на саморозвиток.
RS-2	Відкритий	Зміна вимог замовником	Низька	Середній	0,06	Зменшення	1. Домовитись про всі вимоги на початкових етапах проєктування. 2.Чітко окреслити необхідні характеристики проєкту.	Переоцінка вимог до проєкту, коли вимоги від замовника змінились.
RS-3	Відкритий	Проблеми з інтернетом	Середня	Низький	0,04	Зменшення	1.Підбір надійного провайдера.	Альтернативне джерело інтернету, або скористатись коворкінгом.
RS-4	Відкритий	Недостатній обсяг фінансування	Середня	Середній	0,12	Зменшення	1.Підписати контракт з замовником, який забезпечує виконання фінансових вимог.	Зв’язатись з замовником, переглянути контракт, обґрунтувати необхідність додаткового фінансування на проєкт.

Продовження табл. А.8

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS-5	Відкритий	Недостатня підтримка проєкту з боку замовника	Низька	Середній	0,05	Зменшення	1. Налагодження процес моніторингу прогресу виконання завдань, створення комфортних умов для роботи та співпраці, гарні відносини між розробником і замовником.	Влаштувати перемови з замовником, щодо владнання проблеми недостатнього фідбеку між розробником та замовником.
RS-6	Відкритий	Хвороба розробника	Низька	Високий	0,10	Зменшення	1. Забезпечення комфортних умов для роботи, медичне страхування.	Виділити необхідні засоби для допомоги розробнику.
RS-7	Відкритий	Відсутність механізму резервного копіювання	Низька	Середній	0,06	Зменшення	1. Забезпечення увімкнення автоматичного збереження даних, створення локальних копій.	Створення локальних та хмарних копій.
RS-8	Відкритий	Поява альтернативного продукту	Середня	Середній	0,12	Зменшення	Забезпечення імплементації унікального функціоналу.	Проведення маркетингової стратегії.
RS-9	Відкритий	Виявлення складних для виправлення дефектів розробки проєкту	Середня	Середній	0,12	Зменшення	Постійна комунікація з замовником, забезпечити процес review для проміжних етапів розробки.	Проміжний контроль розробки під час виконання проєкту.

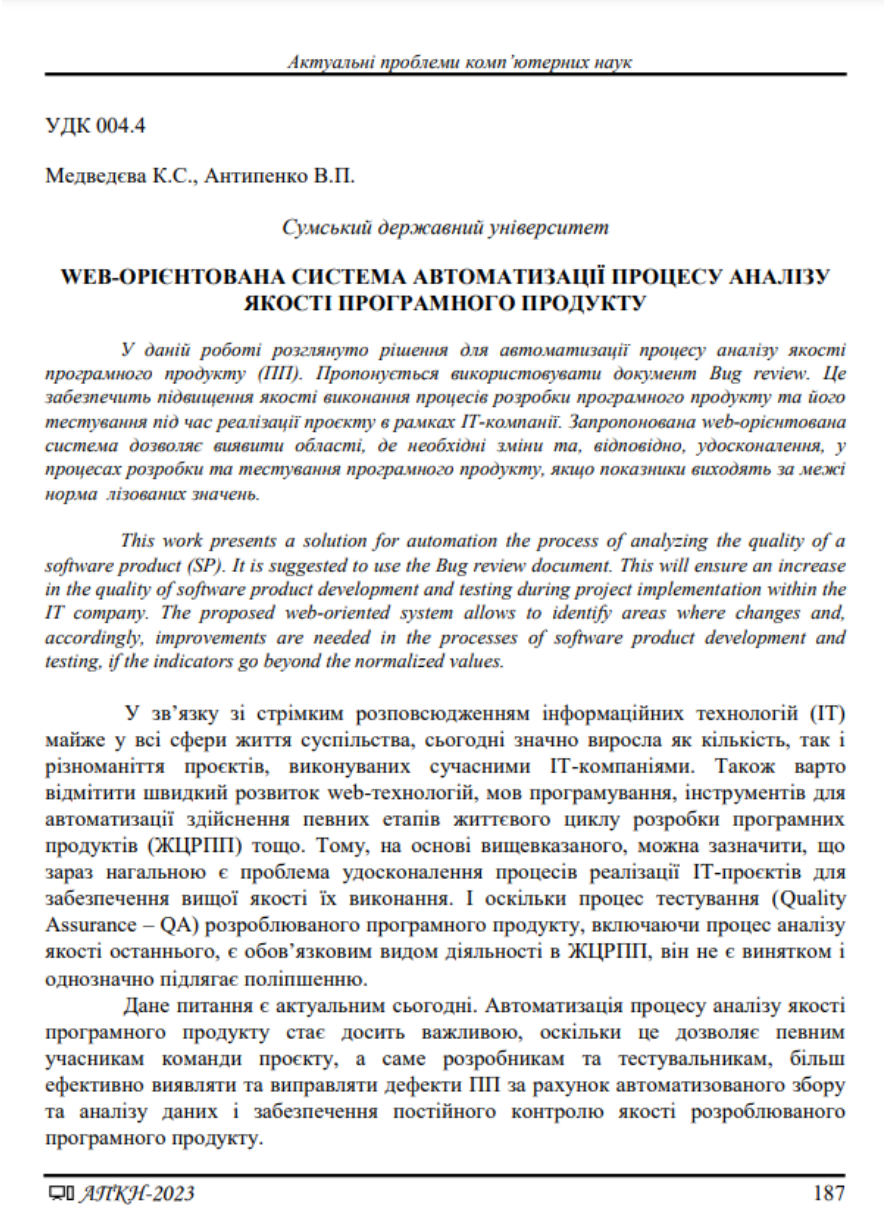
Продовження табл. А.8

ID	**Статус ризику**	**Опис ризику**	**Ймовірність виникнення**	**Вплив ризику**	**Ранг ризику**	**Тип стратегії реагування**	**План А**	**План Б**
RS-10	Відкритий	Відключення світла	Низька	Середній	0,03	Ухилення	Забезпечити альтернативне джерело енергії.	Дізнатись про розташування забезпечених альтернативними джерелами енергії місць.

Джерело: побудовано автором

ДОДАТОК Б

Апробація результатів дослідження



Актуальні проблеми комп'ютерних наук

УДК 004.4

Медведєва К.С., Антипенко В.П.

Сумський державний університет

WEB-ОРІЄНТОВАНА СИСТЕМА АВТОМАТИЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ЯКОСТІ ПРОГРАМНОГО ПРОДУКТУ

У даній роботі розглянуто рішення для автоматизації процесу аналізу якості програмного продукту (ПП). Пропонується використовувати документ Bug review. Це забезпечить підвищення якості виконання процесів розробки програмного продукту та його тестування під час реалізації проєкту в рамках IT-компанії. Запропонована web-орієнтована система дозволяє виявити області, де необхідні зміни та, відповідно, удосконалення, у процесах розробки та тестування програмного продукту, якщо показники виходять за межі норма лізованих значень.

This work presents a solution for automation the process of analyzing the quality of a software product (SP). It is suggested to use the Bug review document. This will ensure an increase in the quality of software product development and testing during project implementation within the IT company. The proposed web-oriented system allows to identify areas where changes and, accordingly, improvements are needed in the processes of software product development and testing, if the indicators go beyond the normalized values.

У зв'язку зі стрімким розповсюдженням інформаційних технологій (ІТ) майже у всі сфери життя суспільства, сьогодні значно виросла як кількість, так і різноманіття проєктів, виконуваних сучасними ІТ-компаніями. Також варто відмітити швидкий розвиток web-технологій, мов програмування, інструментів для автоматизації здійснення певних етапів життєвого циклу розробки програмних продуктів (ЖЦРПП) тощо. Тому, на основі вищевказаного, можна зазначити, що зараз нагальною є проблема удосконалення процесів реалізації ІТ-проєктів для забезпечення вищої якості їх виконання. І оскільки процес тестування (Quality Assurance – QA) розроблюваного програмного продукту, включаючи процес аналізу якості останнього, є обов'язковим видом діяльності в ЖЦРПП, він не є винятком і однозначно підлягає поліпшенню.

Дане питання є актуальним сьогодні. Автоматизація процесу аналізу якості програмного продукту стає досить важливою, оскільки це дозволяє певним учасникам команди проєкту, а саме розробникам та тестувальникам, більш ефективно виявляти та виправляти дефекти ПП за рахунок автоматизованого збору та аналізу даних і забезпечення постійного контролю якості розроблюваного програмного продукту.

АПКН-2023 187

Рисунок Б.1 – Перша сторінка тез

Джерело: побудовано автором (знімок з екрану)

Сучасні методи оцінки якості програмного забезпечення ґрунтуються на встановлених міжнародних і національних стандартах [1]. У сфері автоматизації аналізу якості ПП існують дослідження, які фокусуються на застосуванні технологій машинного навчання та штучного інтелекту [2]. Їх використання сприяє вдосконаленню процесів тестування та оцінки коду. Незважаючи на те, що статичний аналіз коду [3] застосовується для виявлення деяких видів вразливостей і помилок, він має обмеження в контексті симуляції реальних умов експлуатації ПП. Розвиток web-технологій сприяє створенню більш гнучких і доступних інструментів для забезпечення безперервного моніторингу якості програмного продукту в реальному часі.

Метою роботи є розробка web-орієнтованої системи автоматизації процесу аналізу якості програмного продукту для підвищення ефективності роботи розробників і тестувальників під час виконання проєктів у рамках IT-компанії. Вона генерує результати на основі дослідження документа Bug review. Це необхідно для ідентифікації слабких місць у процесі розробки і подальшого їх удосконалення наряду з відповідними процесами тестування. У свою чергу це також дозволить ефективно управляти дефектами ПП.

Таблиця 1 – Метрики для аналізу документа Bug review

Назва метрики	**Пояснення значень результатів метрики**
Defect density	Щільність дефектів за конкретним програмним продуктом. Якщо показник >0,3 необхідно внести зміни в процеси команди розробки.
Time bugs solving by Priority	Час на виправлення дефекту відносно його пріоритету. Відображається за допомогою тренду змін.
Bugs Environment	Місце знаходження дефектів відносно середовища розробки. Якщо більшість дефектів було виявлено на dev/stage середовищі – команда QA працює правильно, якщо більшість дефектів знайдено на production середовищі – необхідні зміни в QA процесах.
Bug Discovery Source	Аналіз ким були знайдені дефекти. Якщо більшість дефектів було знайдено QA та розробниками це свідчить про налагодженість та надійність процесів в команді розробки, якщо дефект був знайдений користувачем – необхідно поліпшити методи тестування та розробки.

Далі розглянемо суть документа Bug review. Це інструмент для структурованого аналізу дефектів. Останні збираються під час тестування та розробки програмного продукту. Документ Bug review містить основні

Рисунок Б.2 – Друга сторінка тез

Джерело: побудовано автором (знімок з екрану)

характеристики кожного дефекту: його ID та назву, номер спринту, хто знайшов дефект, оточення, функціонал, скільки разів відтворювався дефект, його причина, статус, пріоритет, серйозність, дата створення, дата зміни, час на виправлення дефекту. Це дозволяє ретельно дослідити якість розробки та тестування програмного продукту.

Для ефективного аналізу документа Bug review використовуються різні метрики тестування. Їх опис надано в таблиці 1.

Використання web-орієнтованої системи для аналізу якості програмного продукту має численні переваги. Перш за все, її застосування дозволяє здійснювати дослідження дефектів ПП в режимі реального часу, забезпечуючи доступ до даних з будь-якого місця і в будь-який час. Крім того, дана web-орієнтована система дає можливість візуалізації результатів аналізу документа Bug review у зручному форматі, що сприяє кращому розумінню проблеми та прийняттю відповідних рішень.

Автоматизація процесу аналізу якості програмного продукту є важливим завданням у ході підтримки етапів ЖЦРПП. Використання запропонованої web-орієнтованої системи, яка базується на аналізі документа Bug review, дозволяє підвищити якість розробки та тестування програмного продукту. У свою чергу це веде до удосконалення всього процесу реалізації сучасних IT-проєктів у цілому. Впровадження метрик тестування дозволяє ефективно виявляти та управляти дефектами. Це впливає на підвищення якості ПП та процесів розробки. Такі підходи допомагають забезпечити надійність і конкурентоспроможність програмного продукту в умовах постійних змін і вимог ринку та користувачів.

Перелік посилань

1. Незамай Б. Прогнозування якості програмного забезпечення. In 2022 International Conference on Innovative Solutions in Software Engineering (ICISSE), 2022. С. 86.
2. Tran H. M., Le S. T., Nguyen S. V., & Ho P. T. An analysis of software bug reports using machine learning techniques. SN Computer Science, 1, 2020. P. 1-11.
3. Смелик Я. М. Аналіз якості програмного забезпечення за допомогою програмних метрик. Магістерська дис.:121 Інженерія програмного забезпечення, Київ, 2022. 114 С.

Рисунок Б.3 – Третя сторінка тез

Джерело: побудовано автором (знімок з екрану)

ДОДАТОК В

Лістинг програмного коду

BugController.php

```
class BugController extends \yii\web\Controller
{
  public function actionIndex($id)
  {
    $model = Bugs::findOne($id);

    return $this->render('index', [
      'model' => $model
    ]);
  }

  public function actionList()
  {

    $model = new UploadForm();


    if (Yii::$app->request->isPost) {
      $model->csvFile = UploadedFile::getInstance($model,
'csvFile');
      if ($model->upload()) {
        return $this->refresh();
      }
    }

    return $this->render('list', ['model' => $model]);
  }

  /**
   * @throws StaleObjectException
   * @throws \Throwable
   */
  public function actionDelete($id = null)
  {
    if ($id) {
      $bug = Bugs::findOne(['id' => $id]);

      if ($bug) {
        $bug->delete();
      }
    } else {
```

```
      Bugs::deleteAll();
    }

    return $this->actionList();
  }

  /**
   * Displays bugs page.
   *
   * @return string
   */
  public function actionNew()
  {

    $model = new Bugs();

    if ($model->load(Yii::$app->request->post()) && $model-
>validate()) {
      if ($model->save()) {

        Yii::$app->session->setFlash('success', 'Data saved
successfully.');
      } else {

        Yii::$app->session->setFlash('error', 'Error saving
data.');
      }
    }

    return $this->render('new', [
      'model' => $model,
    ]);
  }


  /**
   * Displays bugs page.
   *
   * @return string
   */
  public function actionEdit($id)
  {

    $model = Bugs::findOne($id);

    if ($model->load(Yii::$app->request->post()) && $model-
>validate()) {
      if ($model->save()) {
        // data is saved successfully
        Yii::$app->session->setFlash('success', 'Data updated
successfully.');
      } else {
```

```
        // error in saving data
        Yii::$app->session->setFlash('error', 'Error saving
data.');
      }
    }

    return $this->render('edit', [
      'model' => $model,
    ]);
  }

}
```

services/AnalysingService.py

```
from flask import Flask, request, jsonify
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
import json

app = Flask(__name__)


def series_to_list(series):
  return list(series.reset_index().to_dict(orient='records'))


def unprocessed_defects(df):
  num_defects_to_do = df[df['statusID'] == 3].shape[0]
  total_defects = df.shape[0]

  unprocessed_defects_metric_corrected = num_defects_to_do /
total_defects if total_defects > 0 else 0

  def analyze_unprocessed_defects(num_defects_to_do,
total_defects):
    if total_defects > 0:
      metric = num_defects_to_do / total_defects
      analysis_text = f"Показник 'Неопрацьовані дефекти' становить
{metric:.2f} або {metric * 100:.1f}%. <br />"
      if metric > 0.20:
        analysis_text += "Це перевищує поріг у 20%, що може
свідчити про проблеми в команді з розумінням того, що є дефектом,
або з пріоритезацією виправлення багів продукту.<br />"
      else:
        analysis_text += "Це менше за поріг у 20%, що свідчить про
адекватне управління дефектами в команді.<br />"
    else:
```

```
      analysis_text = "Загальна кількість зареєстрованих дефектів
дорівнює 0, тому розрахунок показника неможливий.<br />"

    return analysis_text

  analysis_result = analyze_unprocessed_defects(num_defects_to_do,
total_defects)
  return analysis_result


def missed_bugs_prod_func(df):
  sprints = df['sprint'].dropna().unique()

  df['createdDate'] = pd.to_datetime(df['createdDate'])
  missed_defects_metrics_updated = {}
  start_date = pd.Timestamp('2023-01-11')

  def calculate_sprint_end_date(start_date):
    return start_date + pd.Timedelta(weeks=2)

  def calculate_missed_defects_for_sprint(sprint_data,
sprint_end_date):
    bugs_after_sprint_prod = sprint_data[
      (sprint_data['createdDate'] > sprint_end_date) &
(sprint_data['Platform'].str.contains('prod'))]

    total_bugs_sprint = sprint_data.shape[0]

    if total_bugs_sprint > 0:
      missed_defects = bugs_after_sprint_prod.shape[0] /
total_bugs_sprint
    else:
      missed_defects = None

    return missed_defects

  for sprint in sprints:
    sprint_data = df[df['sprint'] == sprint]

    end_date = calculate_sprint_end_date(start_date)
    missed_defects_metrics_updated[sprint] =
calculate_missed_defects_for_sprint(sprint_data, end_date)

    start_date = end_date

  missed_defects_df =
pd.DataFrame(list(missed_defects_metrics_updated.items()),
                    columns=['sprint', 'metric'])
  missed_defects_df.sort_values(by='sprint', inplace=True)
  missed_defects_df.reset_index(drop=True, inplace=True)

  return missed_defects_df
```

```
def generate_missed_bugs_conclusion(missed_defects_df):
  missed_defects_df['sprint'] =
missed_defects_df['sprint'].astype(int)
  no_missed_bugs_sprints =
missed_defects_df[missed_defects_df['metric'] ==
0]['sprint'].tolist()

  average_metric = missed_defects_df['metric'].mean()
  max_missed_bugs = missed_defects_df['metric'].max()
  sprint_with_max_missed_bugs =
missed_defects_df[missed_defects_df['metric'] ==
max_missed_bugs]['sprint'].iloc[0]

  if average_metric < 0.1:
    conclusion = "Середній відсоток пропущених дефектів нижче 10%,
що вказує на ефективність процесу тестування.<br />"
  else:
    conclusion = "Середній відсоток пропущених дефектів є більшим
ніж 10%, що свідчить про проблеми в процесі тестування.<br />"

  if no_missed_bugs_sprints:
    sprints_str = ', '.join(map(str, no_missed_bugs_sprints))
    conclusion += f" Спринти без пропущених дефектів на прод:
{sprints_str}. Процес тестування під час цих спринтів проведений
коректно.<br />"

  if max_missed_bugs > 0.1:
    conclusion += f" Увага! Найвища кількість пропущених дефектів
({max_missed_bugs*100:.2f})% спостерігалася у спринті
{sprint_with_max_missed_bugs}. Рекомендується звернути особливу
увагу на цей спринт.<br />"
  else:
    conclusion += f" Найвища кількість пропущених дефектів у
спринті {sprint_with_max_missed_bugs} не перевищує 0.1, що є
прийнятним.<br />"

  return conclusion


def average_time_to_solve_func(df):
  total_time_to_solve = df['timeForSolve'].sum()
  total_defects_count = df.shape[0]

  average_time_per_defect = total_time_to_solve /
total_defects_count if total_defects_count > 0 else None

  average_time_per_functionality =
df.groupby('functionalityID')['timeForSolve'].mean()

  result = {
```

```
    "common": {
      "avgTime": average_time_per_defect,
      "status": 1 if average_time_per_defect <= 30 else 0
    },
    "byFunctionality": []
  }

  for functionality_id, time in
average_time_per_functionality.items():
    if time > 0:
      result["byFunctionality"].append({
        "id": functionality_id,
        "avgTime": time,
        "status": 1 if time <= 30 else 0
    })

  return result


def analyze_average_time_to_solve(result):
  conclusions = []
  specific_issues = []

  avg_time = result["common"]["avgTime"]
  if avg_time and avg_time <= 30:
    conclusions.append("Загальний середній час життя дефекту в
нормі.<br />")
  else:
    conclusions.append("Загальний середній час життя дефекту
високий.<br />")

  sorted_functionalities = sorted(result["byFunctionality"],
key=lambda x: x["avgTime"], reverse=True)

  for functionality in sorted_functionalities[:5]:
    specific_issues.append(f"Функціональна частина
${functionality['id']}$ має високий час виправлення:
{functionality['avgTime']} днів.<br />")

  if specific_issues:
    conclusions.append("Необхідно звернути увагу на функціональні
частини з високим часом виправлення дефектів.<br />")
    conclusions.append("Розгляньте можливість впровадження
додаткових ресурсів або перегляду стратегій тестування та
розробки.<br />")

  return "\n".join(conclusions + specific_issues)


@app.route('/process', methods=['POST'])
def process_data():
  data_string = request.data.decode('utf-8')
```

```
  intermediate_data = json.loads(data_string)

  if isinstance(intermediate_data, str):
    final_data = json.loads(intermediate_data)
  else:
    final_data = intermediate_data

  bug_data = pd.DataFrame(final_data)

  by_severity =
series_to_list(bug_data['severityID'].value_counts())
  by_priority =
series_to_list(bug_data['priorityID'].value_counts())
  by_sprint =
series_to_list(bug_data['sprint'].value_counts().sort_index())
  most_bugs_by_functionality =
series_to_list(bug_data['functionalityID'].value_counts())

  product_environment =
bug_data['Platform'].str.extract(r'(\w+)\((\w+)\)')
  product_environment.columns = ['label', 'environment']
  bug_data_with_env = bug_data.join(product_environment)

  defect_density =
series_to_list(bug_data_with_env.groupby(['label',
'environment']).size().unstack().fillna(0))

  missed_bugs_prod = missed_bugs_prod_func(bug_data)

  avg_time_solve = average_time_to_solve_func(bug_data)

  unprocessed_bugs = unprocessed_defects(bug_data)

  missed_bugs_conclusion =
generate_missed_bugs_conclusion(missed_bugs_prod)

  analysis_conclusion =
analyze_average_time_to_solve(avg_time_solve)

  data_json = {
    "bySeverity": by_severity,
    "byPriority": by_priority,
    "bySprint": by_sprint,
    "mostBugsByFunctionality": most_bugs_by_functionality,
    "defectDensity": defect_density,
    "missedBugs": series_to_list(missed_bugs_prod),
    "missedBugsConclusion": missed_bugs_conclusion,
    "avgResolveTime": avg_time_solve,
    "analysisConclusion": analysis_conclusion,
    "unprocessedBugs": unprocessed_bugs,
  }
```

```
  return jsonify(data_json)


if __name__ == "__main__":
  app.run(debug=True)
```

site/index.php

```
$this->title = 'Dashboard';
?>

<div class="site-index">

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>

  <div class="body-content">
    <?php
    $form = ActiveForm::begin([
      'action' => Url::to(['site/index']),
      'method' => 'get',
    ]);
    ?>


    <div class="d-flex justify-content-between align-items-center
mb-5">
      <?= $form->field($model, 'startDate', ['options' => ['class'
=> 'w-100 mr-5']])->widget(DatePicker::classname(), [
        'language' => 'en',
        'dateFormat' => 'yyyy-MM-dd',
        'options' => ['class' => 'form-control'],
      ]) ?>
      <?= $form->field($model, 'endDate', ['options' => ['class'
=> 'w-100 mr-5']])->widget(DatePicker::classname(), [
        'language' => 'en',
        'dateFormat' => 'yyyy-MM-dd',
        'options' => ['class' => 'form-control'],
      ]) ?>
      <?= Html::submitButton('Filter', ['class' => 'btn btn-
primary', 'style' => 'margin-top: 32px'])?>
    </div>

    <?php ActiveForm::end() ?>
    <ul class="nav nav-tabs" id="myTab" role="tablist">
      <li class="nav-item" role="presentation">
        <button class="nav-link active" id="dashboards-tab" data-
bs-toggle="tab" data-bs-target="#dashboards" type="button"
```

```
role="tab" aria-controls="dashboards" aria-
selected="true">Dashboards</button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="metrics-tab" data-bs-
toggle="tab" data-bs-target="#metrics" type="button" role="tab"
aria-controls="metrics" aria-selected="false">Metrics</button>
      </li>
      <li class="nav-item" role="presentation">
        <button class="nav-link" id="conclusion-tab" data-bs-
toggle="tab" data-bs-target="#conclusion" type="button" role="tab"
aria-controls="conclusion" aria-
selected="false">Conclusions</button>
      </li>
    </ul>
    <div class="tab-content">
      <div class="tab-pane active" id="dashboards" role="tabpanel"
aria-labelledby="dashboards-tab">
        <?php if($data): ?>
          <div class="d-flex w-100">
            <div class="d-flex flex-column w-100 mr-5">
              <h5> Distribution of Bugs by Severity </h5>
              <?php
              $priorityData = array_map(function ($item) {
                return $item['count'];
              }, $data['bySeverity']);
              $priorityLabels = array_map(function ($item) {
                return \app\models\Severity::findOne(['id' =>
$item['severityID']])['name'];
              }, $data['bySeverity']);
              $colors = ['#fabebe', '#008080', '#e6beff',
'#9a6324', '#fffac8', '#800000', '#aaffc3', '#808000', '#ffd8b1'];

              $colors = array_slice(array_merge($colors, $colors),
0, count($priorityData));
              $totalBugs = array_sum($priorityData);
              $highImpactBugsCount = 0;
              $highImpactLabels = ['1 - Critical', '2 - High'];

              foreach ($priorityLabels as $index => $label) {
                if (in_array($label, $highImpactLabels)) {
                  $highImpactBugsCount += $priorityData[$index];
                }
              }

              $highImpactPercentage = ($highImpactBugsCount /
$totalBugs) * 100;
                $analysisTextSeverity = "<h5>Аналіз помилок за
серйозністю:</h5>";

              foreach ($priorityData as $index => $count) {
                $label = $priorityLabels[$index];
```

```
                $percentage = ($count / $totalBugs) * 100;
                $analysisTextSeverity .= "<p>$label: $count
помилок (" . round($percentage, 2) . "% від загальної
кількості)</p>";
              }

              if ($highImpactPercentage > 20) {
                $analysisTextSeverity .= "<p><strong>Помилки з
високим ступенем впливу:</strong> $highImpactBugsCount помилок ("
. round($highImpactPercentage, 2) . "% від загальної кількості).
Це критично та потребує негайного реагування.</p>";
              } else {
                $analysisTextSeverity .= "<p><strong>Помилки з
високим ступенем впливу:</strong> $highImpactBugsCount помилок ("
. round($highImpactPercentage, 2) . "% від загальної кількості).
Це в межах припустимого діапазону, але рекомендується
моніторинг.</p>";
              }

              ?>

              <?= ChartJs::widget([
                'type' => 'bar',
                'data' => [
                  'labels' => $priorityLabels,
                  'datasets' => [
                    [
                      'label' => "Bugs Count",
                      'data' => $priorityData,
                      'backgroundColor' => $colors
                    ]
                  ]
                ],
                'options' => [
                  'plugins' => [
                    'legend' => [
                      'display' => true,
                    ],
                  ],
                ]
              ]) ?>

            </div>

            <div class="d-flex flex-column w-100">
              <h5> Distribution of Bugs by Priority </h5>
              <?php
              $priorityData = array_map(function ($item) {
                return $item['count'];
              }, $data['byPriority']);

              $priorityLabels = array_map(function ($item) {
```

```
                return \app\models\Priority::findOne(['id' =>
$item['priorityID']])['name'];
            }, $data['byPriority']);

            $colors = ['#fffac8', '#800000', '#aaffc3',
'#808000', '#ffd8b1'];

            $colors = array_slice(array_merge($colors, $colors),
0, count($priorityData));

            $totalBugs = array_sum($priorityData);
            $analysisTextPriority = "<h5>Аналіз помилок за
пріоритетністю:</h5>";

            $priorityIndexes = array_flip($priorityLabels);

            foreach ($priorityIndexes as $priorityName =>
$index) {
              if (array_key_exists($index, $priorityData)) {
                $count = $priorityData[$index];
                $percentage = ($count / $totalBugs) * 100;
                $analysisTextPriority .= "<p><b>Пріоритет
$priorityName:</b> $count помилок (" . round($percentage, 2) . "%
від загальної кількості)</p>";
              }
            }

            if ($priorityData[$priorityIndexes['1 - Critical']]
> 0) {
              $analysisTextPriority .= "<p><b>Потребує негайного
втручання:</b> Є " . $priorityData[$priorityIndexes['Critical']] .
" критичних помилок, які потребують негайного вирішення.</p>";
            } else {
              $analysisTextPriority .= "<p><b>Всі критичні
помилки під контролем:</b> Немає критичних помилок, що свідчить
про стабільне становище продукту.</p>";
            }

            if ($priorityData[$priorityIndexes['2 - High']] > 0)
{
              $analysisTextPriority .= "<p><b>Проблеми високого
пріоритету:</b> Є " . $priorityData[$priorityIndexes['High']] . "
помилок високого пріоритету, які слід невідкладно вирішити, щоб
уникнути значного впливу на якість продукту.</p>";
            }

            $recommendationsPriority = "";
            if ($totalBugs > 0) {
              if ($priorityData[$priorityIndexes['1 -
Critical']] > 0 || $priorityData[$priorityIndexes['High']] > 0) {
                $recommendationsPriority .= "Потрібна негайна
дія для усунення критичних та помилок високого пріоритету. ";
```

```
                }
                if ($priorityData[$priorityIndexes['3 - Medium']]
> 0) {
                  $recommendationsPriority .= "Помилки середнього
пріоритету слід запланувати до усунення у відповідності з графіком
проєкту. ";
                }
              } else {
                $recommendationsPriority .= "Помилок немає -
чудова робота з підтримки якості!";
              }

              ?>


              <?= ChartJs::widget([
                'type' => 'bar',
                'data' => [
                  'labels' => $priorityLabels,
                  'datasets' => [
                    [
                      'label' => 'Bugs Count',
                      'data' => $priorityData,
                      'backgroundColor' => $colors,
                    ]
                  ]
                ],
                'options' => [
                  'plugins' => [
                    'datalabels' => [
                      'anchor' => 'end',
                      'align' => 'top',
                      'formatter' => new
\yii\web\JsExpression('function(value, context) {
                              return value;
                              }'),
                      'color' => '#444',
                    ]
                  ]
                ],
              ])
              ?>

              </div>
            </div>
          <div class="d-flex flex-column w-100 mr-5 h-200px">
            <h5> Trends in Bug Discovery by Sprint </h5>

            <?php
            $bySprintData = array_map(function ($item) {
              return $item['count'];
            }, $data['bySprint']);
```

```
            if (!in_array(0, $bySprintData)) {
              $bySprintData[] = 0;
            }
            $bySprintLabels = array_map(function ($item) {
              return $item['sprint'];
            }, $data['bySprint']);
            $colors = ['#00876c', '#66c2a4', '#abdda4', '#e6f598',
'#fee08b', '#fdae61', '#f46d43', '#d53e4f', '#9e0142', '#5e4fa2'];

            $colors = array_slice(array_merge($colors, $colors),
0, count($bySprintData));

            function linearRegression($x, $y) {
              $n = count($x);
              $x_sum = array_sum($x);
              $y_sum = array_sum($y);
              $xx_sum = 0;
              $xy_sum = 0;

              for($i = 0; $i < $n; $i++){
                $xy_sum += ($x[$i]*$y[$i]);
                $xx_sum += ($x[$i]*$x[$i]);
              }

              $slope = (($n * $xy_sum) - ($x_sum * $y_sum)) / (($n
* $xx_sum) - ($x_sum * $x_sum));
              $intercept = ($y_sum - ($slope * $x_sum)) / $n;

              return [$slope, $intercept];
            }

            list($slope, $intercept) =
linearRegression($bySprintLabels, $bySprintData);

            $trendData = [];
            foreach ($bySprintLabels as $i => $label) {
              $trendData[] = $slope * $i + $intercept;
            }

            $increase = false;
            if ($slope > 0) {
              $increase = true;
            }

            $analysisSummary = "<h5>Підсумки аналізу
тенденцій:</h5>";
            $recommendationsTrend = "<h5>Рекомендації:</h5>";

            if ($increase) {
              $analysisSummary .= "<p>Лінія тренду вказує на
збільшення кількості помилок за спринтами. Це може свідчити про
те, що нові функції або зміни в коді вводять помилки.</p>";
```

```
            $recommendationsTrend .= "<p>Перегляньте нещодавні
зміни в кодовій базі та покращіть охоплення тестуванням.
Розгляньте можливість переоцінки процесів розробки та контролю
якості для виявлення потенційних прогалин.</p>";
          } else {
            $analysisSummary .= "<p>Лінія тренду вказує на
стабільну або зменшуючу кількість помилок за спринтами. Це
свідчить про те, що команда ефективно керує помилками та усуває їх
по мірі виникнення.</p>";
            $recommendationsTrend .= "<p>Продовжуйте
використовувати поточні стратегії управління помилками та
зосередьтеся на тих областях, де помилки все ще звітуються, для
подальшого покращення якості.</p>";
          }

          ?>

          <?= ChartJs::widget([
            'type' => 'line',
            'options' => [
              'height' => 200,
              'responsive' => true,
            ],
            'data' => [
              'labels' => $bySprintLabels,
              'datasets' => [
                [
                  'label' => "Trend Line",
                  'data' => $trendData,
                  'borderColor' => 'rgba(255, 99, 132, 1)',
                  'backgroundColor' => 'rgba(0, 0, 0, 0)',
                  'type' => 'line',
                  'fill' => false,
                  'borderWidth' => 2,
                  'pointRadius' => 0,
                  'yAxisID' => 'y-axis-2',
                ],
                [
                  'label' => "Bugs Found",
                  'data' => $bySprintData,
                  'backgroundColor' => $colors,
                  'borderColor' => $colors,
                  'fill' => false,
                  'type' => 'line',
                  'yAxisID' => 'y-axis-1',
                ]
              ]
            ],
            'clientOptions' => [
              'scales' => [
                'yAxes' => [
                  [
```

```
                    'id' => 'y-axis-1',
                    'type' => 'linear',
                    'position' => 'left',
                  ],
                  [
                    'id' => 'y-axis-2',
                    'type' => 'linear',
                    'position' => 'right',
                    'gridLines' => [
                      'drawOnChartArea' => false,
                    ],
                  ],
                ],
              ],
            ],
          ]) ?>


        </div>

        <div class="d-flex flex-column w-100 mr-5">
          <h5> Functionalities with Most bugs </h5>
          <?php
          $bySprintData = array_map(function ($item) {
            return $item['count'];
          }, $data['mostBugsByFunctionality']);
          if (!in_array(0, $bySprintData)) {
            $bySprintData[] = 0;
          }
          $bySprintLabels = array_map(function ($item) {
            return \app\models\Functionality::findOne(['id' =>
$item['functionalityID']])['name'];
          }, $data['mostBugsByFunctionality']);
          $colors = [
            '#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0',
'#9966FF', '#FF9F40',
            '#9e0142', '#5e4fa2', '#3288bd',
];

          $colors = array_slice(array_merge($colors, $colors),
0, count($bySprintData));
          $analysisTextFunctionality = "<h5>Аналіз топ-5
функціональностей за кількістю помилок:</h5>";
          $recommendationsFunctionalities =
"<h5>Рекомендації:</h5>";


          $urgentThreshold = 20;
          $attentionThreshold = 15;
          $topFunctionalities =
array_slice($data['mostBugsByFunctionality'], 0, 5);
```

```
            $urgentFixes = [];
            $needsAttention = [];

            foreach ($topFunctionalities as $functionality) {
              $funcName = \app\models\Functionality::findOne(['id'
=> $functionality['functionalityID']])['name'];
              $bugsCount = $functionality['count'];

              if ($bugsCount >= $urgentThreshold) {
                $urgentFixes[] = $funcName;
              } elseif ($bugsCount >= $attentionThreshold) {
                $needsAttention[] = $funcName;
              }
              $percentageOfTotal = ($bugsCount / $totalBugs) *
100;
              $analysisTextFunctionality .=
"<p><b>{$funcName}:</b> {$bugsCount} помилок, що становить " .
round($percentageOfTotal, 2) . "% від загальної кількості.</p>";
            }

            if (!empty($urgentFixes)) {
              $funcNames = implode(', ', $urgentFixes);
              $recommendationsFunctionalities .=
"<p>Функціональність {$funcNames} має значну кількість помилок, що
вимагає негайного аналізу та вирішення.</p>";
            }

            if (!empty($needsAttention)) {
              $funcNames = implode(', ', $needsAttention);
              $recommendationsFunctionalities .=
"<p>Функціональність {$funcNames} містить помітну кількість
помилок і потребує уваги для покращення.</p>";
            }

            if (empty($urgentFixes) && empty($needsAttention)) {
              $recommendationsFunctionalities .= "<p>Всі
функціональності знаходяться в задовільному стані. Продовжуйте
моніторинг та уважно ставтеся до нових помилок.</p>";
            }

            ?>

            <?= ChartJs::widget([
              'type' => 'horizontalBar',
              'options' => [
                'height' => 1000,
              ],
              'data' => [
                'labels' => $bySprintLabels,
                'datasets' => [
                  [
                    'label' => '',
```

```
                'data' => $bySprintData,
                'backgroundColor' => $colors
              ]
            ]
          ]
        ]) ?>


      </div>

      <div class="d-flex flex-column w-100 mr-5">
        <h5> Defect Density </h5
          <?php

          $dev = [];
          $prod = [];
          $defectDensityData = [];
          foreach ($data['defectDensity'] as $i => $item) {
            $dev[] = $item['dev'];
            $prod[] = -$item['prod'];
            $defectDensityData[] = [
              'label' => $item['label'],
              'data' => [$item['dev'], - $item['prod']],
              'borderColor' => $colors[$i % count($colors)],
            ];
          }

          $defectDensityLabels = array_map(function ($item) {
            return $item['label'];
          }, $data['defectDensity']);


          $colors = ['#FF6384', '#36A2EB', '#FFCE56',
'#4BC0C0', '#9966FF', '#FF9F40',
            '#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0',
'#00876c', '#66c2a4'];

          $colors = array_slice(array_merge($colors, $colors),
0, count($defectDensityData));
          foreach ($defectDensityData as $key => $dataset) {
            $defectDensityData[$key]['backgroundColor'] =
$colors;
          }

          $analysisTextDensity = "<h5>Аналіз щільності
дефектів за платформами та середовищами:</h5>";

          $devDefectsByPlatform = [];
          $prodDefectsByPlatform = [];


          foreach ($defectDensityData as $d) {
```

```
                $platform = htmlspecialchars($d['label'],
ENT_QUOTES, 'UTF-8');
                $prodDefects = $d['data'][1] * -1;
                $devDefects = $d['data'][0];
                $totalDefects = $devDefects + $prodDefects;

                $analysisTextDensity .= "<p>Платформа
<b>{$platform}</b>:</p>";
                $analysisTextDensity .= "<ul>";
                $analysisTextDensity .= "<li>Dev середовище:
{$devDefects} дефектів.</li>";
                $analysisTextDensity .= "<li>Prod середовище:
{$prodDefects} дефектів.</li>";
                $analysisTextDensity .= "<li>Загальна кількість
дефектів: " . ($devDefects + $prodDefects) . ".</li>";
                $analysisTextDensity .= "</ul>";
              }

              $recommendationsDensity = "<h5>Рекомендації:</h5>";

              $percentDev = 0.80;
              $percentProd = 0.20;

                $expectedDevDefects = $totalDefects * $percentDev;
                $expectedProdDefects = $totalDefects *
$percentProd;

                if ($devDefects > $expectedDevDefects) {
                  $recommendationsDensity .= "<p>Платформа
<b>{$platform}</b> у dev середовищі має більшу, ніж очікувалося,
кількість дефектів. Рекомендується провести ретельний аналіз
причин дефектів та оптимізувати процеси розробки та
тестування.</p>";
                }

                if ($prodDefects > $expectedProdDefects) {
                  $recommendationsDensity .= "<p>Платформа
<b>{$platform}</b> у prod середовищі має більшу, ніж очікувалося,
кількість дефектів. Необхідно негайно вжити заходів для
виправлення та запобігання виникненню критичних помилок.</p>";
                }
              ?>

                <!--           --><?php //=
print_r($defectDensityData) ?>

              <div class="d-flex flex-row">
                <div
                  class="d-flex flex-column justify-content-
around"
                  style="left: 22px; position: relative; z-index:
11"
```

```
                >
                  <div
                    style="background: #36A2EB; width: 70px;
color: white; font-weight: 500;"
                    class="h-50 p-2 d-flex justify-content-center
align-items-center"
                  >
                    Dev
                  </div>
                  <div
                    style="background: #FF6384; width: 70px;
color: white; font-weight: 500;"
                    class="h-50 p-2 d-flex justify-content-center
align-items-center"
                  >
                    Prod
                  </div>
                </div>
                <div style="width: calc(100% - 106px); position:
relative; left: -12px; z-index: 10">
                <?= ChartJs::widget([
                  'type' => 'bar',
                  'options' => [
                    'style' => "",
                    'height' => 200,
                    'responsive' => true,
                    'interaction' => [
                     'intersect' => false
                    ],
                    'scales' => [
                      'xAxes' => [[
                        'stacked' => true
                      ]],
                      'yAxes' => [[
                        'stacked' => true
                      ]]
                    ],
                    'plugins' => [
                      'legend' => ['display' => false,]
                    ]
                  ],
                  'data' => [
                    'labels' => $defectDensityLabels,
                    'datasets' => [
                      [
                        'label' => 'Dev',
                        'data' => $dev,
                        'backgroundColor' => '#36A2EB',
                      ],
                      [
                        'label' => 'Prod',
                        'data' => $prod,
```

```
                        'backgroundColor' => '#FF6384',
                      ]
                    ]
                  ]
                ]) ?>
                </div>
                <div
                  class="d-flex flex-column justify-content-
around"
                  style="right: 14px; position: relative;"
                >
                  <div
                      style="background: #36A2EB; width: 70px;
color: white; font-weight: 500;"
                      class="h-50 p-2 d-flex justify-content-
center align-items-center"
                  >
                    Dev
                  </div>
                  <div
                      style="background: #FF6384; width: 70px;
color: white; font-weight: 500;"
                      class="h-50 p-2 d-flex justify-content-
center align-items-center"
                  >
                    Prod
                  </div>
                </div>
              </div>

          </div>
        <?php else: ?>
          <div class="text m-5"> No data found </div>
        <?php endif; ?>
      </div>
      <div class="tab-pane" id="metrics" role="tabpanel" aria-
labelledby="metrics-tab">
        <?php if($data): ?>
          <div class="d-flex justify-content-between flex-row
align-items-baseline mt-1" style="gap: 20px">
            <table class="table" style="width: 30%">
              <thead class="thead-dark">
              <tr>
                <th> Sprint </th>
                <th> Missed bugs </th>
              </tr>
              </thead>

              <tbody>
              <?php
                foreach ($data['missedBugs'] as $metric) {
                echo "<tr>";
```

```
                echo "<td class='p-1'>" . $metric['sprint'] .
"</td>";
                echo "<td class='p-1'>" .
round($metric['metric']*100,2) . "%</td>";
                echo "</tr>";

              }
              ?>
              </tbody>
            </table>

            <table class="table" style="width: 65%">
              <thead class="thead-dark">
              <tr>
                <th colspan="3" class="text-center"> Average spend
time </th>
                <th colspan="3" class="text-center"> Trend </th>
              </tr>
              </thead>

              <tbody>
              <tr>
                <?php $status =
$data['avgResolveTime']['common']['status'] ?>
                <td class='text-center' colspan="3"><?=
$data['avgResolveTime']['common']['avgTime'] . ' days' ?></td>
                <td class='text-center' style="background: <?=
$status == 1 ? 'green' : 'red' ?>" colspan="3"><?= $status == 1 ?
'Good' : 'Bad' ?></td>
              </tr>
              </tbody>

              <thead class="thead-dark">
              <tr>
                <th colspan="2"> Functionality </th>
                <th colspan="2" class="text-center"> Amount </th>
                <th colspan="2" class="text-center"> Trend </th>
              </tr>
              </thead>

              <tbody>
              <?php
              foreach ($data['avgResolveTime']['byFunctionality']
as $d) {
                $status = $d['status'];
                $status_color = $status == 1 ? "green" : 'red';
                $functionalityEntity =
\app\models\Functionality::findOne(['id' => $d['id']]);

                echo "<tr>";
                echo "<td colspan='2'>" .
$functionalityEntity['name'] . "</td>";
```

```
                echo "<td colspan='2'>" . $d['avgTime'] . '
days'."</td>";
                echo "<td class='text-center' colspan='2'
style='background: ".$status_color."' colspan='2'>";
                echo $status == 1 ? 'Good' : 'Bad';
                echo "</td>";
                echo "</tr>";

              }
              ?>
              </tbody>
            </table>

          </div>

        <?php else: ?>
          <div class="text m-5"> No data found </div>
        <?php endif; ?>
      </div>
      <div class="tab-pane" id="conclusion" role="tabpanel" aria-
labelledby="conclusion-tab">
        <?php if($data): ?>


          <div class="card text-bg-info mb-3">
            <div class="card-header">Analysis of Bugs by
Severity:</div>
            <div class="card-body">
              <p class="card-text">
                <?= $analysisTextSeverity;?>
            </div>
          </div>
          <div class="card text-bg-info mb-3">
            <div class="card-header">Analysis of Bugs by
Priority:</div>
            <div class="card-body">
              <p class="card-text">
                <?= $analysisTextPriority;
                echo "<h5>Рекомендації:</h5>";
                echo "<p>$recommendationsPriority</p>";?>
            </div>
          </div>

          <div class="card text-bg-info mb-3">
            <div class="card-header">Trends in Bug Discovery by
Sprint:</div>
            <div class="card-body">
              <p class="card-text">
                <?= $analysisSummary;
                echo $recommendationsTrend;
                ?>
            </div>
```

```
          </div>
          <div class="card text-bg-info mb-3">
            <div class="card-header">Functionalities with Most
bugs :</div>
            <div class="card-body">
              <p class="card-text">
                <?= $analysisTextFunctionality;
                echo $recommendationsFunctionalities;
                ?>
            </div>
          </div>
          <div class="card text-bg-info mb-3">
            <div class="card-header">Defect Density :</div>
            <div class="card-body">
              <p class="card-text">
                <?= $analysisTextDensity;
                echo $recommendationsDensity;
                ?>
            </div>
          </div>
          <div class="card text-bg-info mb-3">
            <div class="card-header"> Unprocessed Bugs:</div>
            <div class="card-body">
              <?= $data['unprocessedBugs'] ?>
            </div>
          </div>

          <div class="card text-bg-info mb-3">
            <div class="card-header">Defects Leakage:</div>
            <div class="card-body">
              <?= $data['missedBugsConclusion'] ?>
            </div>
          </div>

          <div class="card text-bg-info mb-3">
            <div class="card-header">Average lifetime of
defects:</div>
            <div class="card-body">
              <?php
              $result =
preg_replace_callback('/\$(\d+(\.\d+)?)\$/', function($matches) {
                $id = $matches[1];
                $name = Functionality::findOne(['id' =>
$id])['name'];
                return "<b>".$name."</b>";
              }, $data['analysisConclusion']);
              echo $result ?>
            </div>
          </div>


        <?php else: ?>
```

```
          <div class="text m-5"> No data found </div>
        <?php endif; ?>
      </div>
    </div>
  </div>
```

bug/new.php

```
$this->title = 'Create new defect';
?>
<div class="site-index">

  <div class="body-content">

    <div class="column">
      <?php $form = ActiveForm::begin(); ?>
      <div class="d-flex justify-content-between ">
        <?= $form->field($model, 'id') ?>
      </div>
      <?= $form->field($model, 'title', ['options' => ['class' =>
'w-100']])->textarea() ?>
      <?= $form->field($model, 'rootCause', ['options' => ['class'
=> 'w-100']])->textarea() ?>
      <div class="d-flex justify-content-between ">
        <?= $form->field($model, 'createdDate', ['options' =>
['class' => 'w-100 mr-3 mt-3 mb-3']])-
>widget(DatePicker::classname(), [
          'language' => 'en',
          'dateFormat' => 'yyyy-MM-dd',
          'options' => ['class' => 'form-control'],
        ]) ?>
        <?= $form->field($model, 'changedDate', ['options' =>
['class' => 'w-100 m-3']])->widget(DatePicker::classname(), [
          'language' => 'en',
          'dateFormat' => 'yyyy-MM-dd',
          'options' => ['class' => 'form-control'],
        ]) ?>
        <?= $form->field($model, 'timeForSolve', ['options' =>
['class' => 'w-100 ml-3 mt-3 mb-3']]) ?>
      </div>
      <div class="d-flex justify-content-between ">
        <?php $data = ArrayHelper::map(Status::find()->all(),
'id', 'name') ?>
        <?= $form->field($model, 'statusID', ['options' =>
['class' => 'w-100 mr-3 mt-3 mb-3']])->
          dropDownList($data, ['prompt' => 'Select a Status'])?>
        <?php $data = ArrayHelper::map(Priority::find()->all(),
'id', 'name') ?>
        <?= $form->field($model, 'priorityID', ['options' =>
['class' => 'w-100 m-3']])->
          dropDownList($data, ['prompt' => 'Select a Priority'])?>
```

```
        <?php $data = ArrayHelper::map(Severity::find()->all(),
'id', 'name') ?>
        <?= $form->field($model, 'severityID', ['options' =>
['class' => 'w-100 ml-3 mt-3 mb-3']])->
          dropDownList($data, ['prompt' => 'Select a Severity'])?>
      </div>
      <div class="d-flex justify-content-between gap-10">
        <?php $data = ArrayHelper::map(UserType::find()->all(),
'id', 'name') ?>
        <?= $form->field($model, 'createdBy', ['options' =>
['class' => 'w-100']])->
        dropDownList($data, ['prompt' => 'Select a user type has
found a bug'])?>
        <?= $form->field($model, 'sprint') ?>
      </div>
      <div class="d-flex justify-content-between ">
        <?php $data = ArrayHelper::map(Functionality::find()-
>all(), 'id', 'name') ?>
        <?= $form->field($model, 'functionalityID', ['options' =>
['class' => 'w-100']])->
          dropDownList($data, ['prompt' => 'Select a Functionality
area'])?>
        <?php $data = ArrayHelper::map(Platform::find()->all(),
'id', 'name') ?>
        <?= $form->field($model, 'platformID', ['options' =>
['class' => 'w-100']])->
          dropDownList($data, ['prompt' => 'Select a Platform'])?>
      </div>
      <div class="form-group mt-5">
        <?= Html::submitButton('Create', ['class' => 'btn btn-
primary']) ?>
      </div>
      <?php ActiveForm::end(); ?>
    </div>

  </div>
</div>

<?php

$script = <<< JS
$(document).ready(function() {
  $('#bugs-createddate, #bugs-changeddate').change(function() {
    var createdVal = $('#bugs-createddate').val();
    var changedVal = $('#bugs-changeddate').val();

    if (!createdVal || !changedVal) {
      return
    }

    const createdDate = new Date(createdVal)
    const changedDate = new Date(changedVal)
```

```
    var amountOfDays = calculateDays(createdDate, changedDate);

    $('#bugs-timeforsolve').val(amountOfDays);
  });
});

function calculateDays(d1, d2) {
  const oneDay = 24 * 60 * 60 * 1000;

  const differenceInTime = Math.abs(d1 - d2);
  return Math.ceil(differenceInTime / oneDay);
}

JS;
$this->registerJs($script, View::POS_END);
?>
```

models/Bugs.php

```
class Bugs extends \yii\db\ActiveRecord
{
  public static function tableName()
  {
    return 'bugs';
  }
  public function rules()
  {
    return [
      [['id', 'title', 'createdDate', 'changedDate', 'statusID',
'priorityID', 'severityID', 'timeForSolve', 'createdBy',
'functionalityID', 'platformID'], 'required'],
      [['id', 'sprint', 'statusID', 'priorityID', 'severityID',
'timeForSolve', 'createdBy', 'functionalityID', 'platformID'],
'integer'],
      [['title', 'rootCause'], 'string'],
      [['createdDate', 'changedDate'], 'safe'],
      [['id'], 'unique'],
      [['functionalityID'], 'exist', 'skipOnError' => true,
'targetClass' => Functionality::class, 'targetAttribute' =>
['functionalityID' => 'id']],
      [['platformID'], 'exist', 'skipOnError' => true,
'targetClass' => Platform::class, 'targetAttribute' =>
['platformID' => 'id']],
      [['priorityID'], 'exist', 'skipOnError' => true,
'targetClass' => Priority::class, 'targetAttribute' =>
['priorityID' => 'id']],
      [['severityID'], 'exist', 'skipOnError' => true,
'targetClass' => Severity::class, 'targetAttribute' =>
['severityID' => 'id']],
```

```
      [['statusID'], 'exist', 'skipOnError' => true, 'targetClass'
=> Status::class, 'targetAttribute' => ['statusID' => 'id']],
    ];
  }
  public function attributeLabels()
  {
    return [
      'id' => 'ID',
      'sprint' => 'Sprint',
      'title' => 'Title',
      'createdDate' => 'Created Date',
      'changedDate' => 'Changed Date',
      'statusID' => 'Status',
      'priorityID' => 'Priority',
      'severityID' => 'Severity',
      'timeForSolve' => 'Time For Solve',
      'createdBy' => 'Created By',
      'functionalityID' => 'Functionality',
      'rootCause' => 'Root Cause',
      'platformID' => 'Platform',
    ];
  }

  /**
   * Gets query for [[Functionality]].
   *
   * @return \yii\db\ActiveQuery
   */
  public function getFunctionality()
  {
    return $this->hasOne(Functionality::class, ['id' =>
'functionalityID']);
  }
  /**
   * Gets query for [[Platform]].
   *
   * @return \yii\db\ActiveQuery
   */
  public function getPlatform()
  {
    return $this->hasOne(Platform::class, ['id' => 'platformID']);
  }
  /**
   * Gets query for [[Priority]].
   *
   * @return \yii\db\ActiveQuery
   */
  public function getPriority()
  {
    return $this->hasOne(Priority::class, ['id' => 'priorityID']);
  }
  /**
```

```
    * Gets query for [[Severity]].
    *
    * @return \yii\db\ActiveQuery
    */
   public function getSeverity()
   {
     return $this->hasOne(Severity::class, ['id' => 'severityID']);
   }
   /**
    * Gets query for [[Status]].
    *
    * @return \yii\db\ActiveQuery
    */
   public function getStatus()
   {
     return $this->hasOne(Status::class, ['id' => 'statusID']);
   }
   /**
    * Gets query for [[UserType]].
    *
    * @return \yii\db\ActiveQuery
    */
   public function getUserType()
   {
     return $this->hasOne(UserType::class, ['id' => 'createdBy']);
   }
}
```

models/UploadForm.php

```
class UploadForm extends Model
{
  public $csvFile;
  public function rules()
  {
    return [
      [['csvFile'], 'file', 'skipOnEmpty' => false, 'extensions'
=> 'csv', 'mimeTypes' => 'text/csv'],
    ];
  }
  public function upload()
  {
    $filePath = 'uploads/' . $this->csvFile->baseName . '.' .
$this->csvFile->extension;
    $this->csvFile->saveAs($filePath);
    if (($handle = fopen($filePath, 'r')) !== false) {
      while (($data = fgetcsv($handle, 100000, ",")) !== false) {
        if (is_numeric($data[1])) {
          $id = (int)$data[1];
          $entity = Bugs::findOne(['id' => $id]);
```

```
            if ($entity || $entity['id']) {
              continue;
            }
            $sprint = $data[0] === '-' || $data[0] === '' ? NULL :
(int)$data[0];
            $title = $data[2];
            $entity = UserType::findOne(['name' => $data[3]]);
            $createdBy = $entity['id'];
            $entity = Platform::findOne(['name' => $data[4]]);
            $platform = $entity['id'];
            $entity = Functionality::findOne(['name' => $data[5]]);
            $functionality = $entity['id'];
            $rootCause = $data[7] === '' ? NULL : $data [7];
            $entity = Status::findOne(['name' => $data[8]]);
            $status = $entity['id'];
            $entity = Priority::findOne(['name' => $data[9]]);
            $priority = $entity['id'];
            $entity = Severity::findOne(['name' => $data[10]]);
            $severity = $entity['id'];
            $crDate = DateTime::createFromFormat('d-m-Y', implode('-
', explode(".", $data[11])));
            $createDate = $crDate->format('Y-m-d');
            echo($createDate);
            $upDate = DateTime::createFromFormat('d-m-Y', implode('-
', explode(".", $data[12])));
            $changeDate = $upDate->format('Y-m-d');
            $resolveTime = (int)$data[13];
            $transaction = Yii::$app->db->beginTransaction();
            try {
              $entity = new Bugs();
              $entity->sprint = $sprint;
              $sql = "INSERT INTO bugs (id, sprint, title,
createdDate, changedDate, statusID, priorityID, severityID,
timeForSolve, createdBy, platformID, functionalityID, rootCause)
VALUES (:id, :sprint, :title, :createdDate, :changedDate,
:statusID, :priorityID, :severityID, :timeForSolve, :createdBy,
:platformID, :functionalityID, :rootCause)";
              $command = Yii::$app->db->createCommand($sql);
              $command->bindValue(':id', $id);
              $command->bindValue(':sprint', $sprint);
              $command->bindValue(':title', $title);
              $command->bindValue(':createdDate', $createDate);
              $command->bindValue(':changedDate', $changeDate);
              $command->bindValue(':statusID', $status);
              $command->bindValue(':priorityID', $priority);
              $command->bindValue(':severityID', $severity);
              $command->bindValue(':timeForSolve', $resolveTime);
              $command->bindValue(':createdBy', $createdBy);
              $command->bindValue(':platformID', $platform);
              $command->bindValue(':functionalityID',
$functionality);
              $command->bindValue(':rootCause', $rootCause);
```

```
            $command->execute();
            $transaction->commit();
          } catch (\Exception $e) {
            echo($e);
            $transaction->rollBack();
            throw $e;
          }
        }
      }
      fclose($handle);

      return true;
    } else {
      return false;
    }
  }
}
```

Повний код знаходиться за посиланням:

https://github.com/KateMedvedieva/BugReview