МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____ проф. Приходько С.Б.
«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Математична модель для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, та розробка програми для її реалізації._____

Виконав: студент групи 6151м
Скакунов Г.О._____
(підпис, ПІБ)

Керівник роботи:
Завідувач кафедри ПЗАС, проф.____
(посада, науковий ступень вчене звання)
Приходько С.Б._____
_____
(підпис, ПІБ)

Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«14» жовтня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Скакунову Георгію Александровичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Математична модель для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, та розробка програми для її реалізації

Керівник роботи: завідувач кафедри ПЗАС, проф. Приходько Сергій Борисович _____

Затверджені наказом ректора № 1070-УЧ від « 11 » _____10_____ 2024 р.

2. Термін подання роботи: 09.12.2024 р._____

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів) _____

Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Розділ з розробки програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів: Титульний слайд, Мета та завдання дослідження, Об'єкт та Предмет дослідження, Наукова новизна одержаних результатів, Практичне значення одержаних результатів, Особистий внесок здобувача, Апробація результатів досліджень та Публікації, Аналіз існуючих методів та моделей для оцінювання якості програмного забезпечення, Математична модель оцінювання якості Java-застосунків для онлайн-покупок, Аналіз вимог до програмного забезпечення, Ескізний проект ПЗ, Технічний проект ПЗ,

Динамічна модель ПЗ, Вибір інструментів розробки, Реалізація та Тестування програмного забезпечення, Результати розробки, Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 14.10.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	16.10.2024	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	18.10.2024	НС*
3.	Підготовка розділу (ів) за результатами власних досліджень	21.10.2024	НС*
4.	Підготовка розділу з розробки програмного забезпечення	27.11.2024	
5.	Підготовка організаційно-економічного розділу	29.11.2024	
6.	Підготовка розділу з охорони праці	02.12.2024	
7.	Підготовка розділу з охорони навколишнього середовища	04.12.2024	
8.	Підготовка розділу ВИСНОВКИ	05.12.2024	
9.	Оформлення списку використаних джерел та додатків	06.12.2024	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	09.12.2024	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	13.12.2024	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	16.12.2024	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	19.12.2024	

* - за результатами наукового стажування (НС), яке було з 02.09.2024 по 13.10.2024 р.

Студент _____ Скакунов Г.О.
(підпис) (ПІБ)

Керівник роботи _____ Приходько С.Б.
(підпис) (ПІБ)

РЕФЕРАТ

Скакунов Георгій Олександрович

«Математична модель для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, та розробка програми для її реалізації»

Кваліфікаційна (магістерська) робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2024 р.

Обсяг роботи: 109 стор., 10 табл., 17 рис., 34 використаних джерел, 5 додатків.

Актуальність теми: Розвиток програмного забезпечення для онлайн-покупок, особливо відкритих Java-застосунків, вимагає достовірного оцінювання якості таких рішень. Удосконалення існуючих моделей оцінювання сприятиме підвищенню конкурентоспроможності та надійності програмного продукту.

Мета та завдання дослідження: підвищити достовріність оцінювання якості Java-застосунків для онлайн-покупок шляхом удосконалення математичної моделі та розробки програмного забезпечення для її реалізації. Завдання включають критичний огляд підходів, обґрунтування вибору метрик (RFC, CBO, WMC), збір даних, створення регресійної моделі, її тестування і розробку відповідного ПЗ.

Об'єкт дослідження: Процес оцінювання якості Java-застосунків для онлайн-покупок.

Предмет дослідження: Математична модель для оцінювання якості Java-застосунків для онлайн-покупок за допомогою метрики RFC залежно від метрик CBO і WMC.

Методи дослідження: Включають статистичну обробку даних, видалення викидів, метод найменших квадратів, логарифмічне перетворення та побудову регресійних моделей.

Наукова новизна: Удосконалено математичну модель для оцінювання якості Java-застосунків із використанням нових параметрів і логарифмічного перетворення для підвищення достовірності прогнозування.

Практична значущість: Розроблена модель дозволяє оцінювати якість Java-застосунків для онлайн-покупок та використовувати її для вдосконалення ПЗ у цій сфері.

Апробація результатів дослідження: Результати представлені на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи», 2024 рік.

Публікації: За темою роботи опубліковано одну наукову працю.

Ключові слова: Java, онлайн-покупки, оцінювання якості, RFC, CBO, WMC, математична модель, регресія.

REVIEW

Skakunov Heorhii

«A mathematical model for assessing the quality of open source online shopping applications in Java and development of a programme for its implementation»

Qualification (master's) thesis for a master's degree in the speciality 121 ‘Software Engineering’. Admiral Makarov National University of Shipbuilding. Mykolaiv, 2024.

Volume of work: 109 pages, 10 tables, 17 figures, 34 references, 5 appendices.

Relevance of the topic: The development of online shopping software, especially open source Java applications, requires a reliable assessment of the quality of such solutions. Improving existing evaluation models will help to increase the competitiveness and reliability of the software product.

The purpose and objectives of the study: to increase the reliability of quality assessment of Java applications for online shopping by improving the mathematical model and developing software for its implementation. The tasks include a critical review of approaches, justification of the choice of metrics (RFC, CBO, WMC), data collection, creation of a regression model, its testing and development of the corresponding software.

Object of study: The process of evaluating the quality of Java applications for online shopping.

Subject of research: A mathematical model for evaluating the quality of Java applications for online shopping using the RFC metric depending on the CBO and WMC metrics.

Scientific novelty: A mathematical model for assessing the quality of Java applications has been improved using new parameters and logarithmic transformation to improve the reliability of forecasting.

Practical significance: The developed model allows to assess the quality of Java applications for online shopping and use it to improve software in this area.

Testing of the research results: The results were presented at the V All-Ukrainian Scientific and Practical Internet Conference ‘Information Technologies: Models, Algorithms, Systems’, 2024.

Publications: One scientific paper was published on the topic of the work.

Keywords: Java, online shopping, quality assessment, RFC, CBO, WMC, mathematical model, regression.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CBO – Coupling Between Object Classes;

RFC – Response for Class;

WMC – Weighted Methods per Class.

ВСТУП

Актуальність теми. Розвиток програмного забезпечення для онлайн-покупок набуває все більшої популярності, зокрема у сфері відкритих Java-застосунків, що використовуються як в Україні, так і за її межами. Оцінка якості таких застосунків є важливим завданням, оскільки від їх надійності та ефективності залежить користувацький досвід і конкурентоспроможність продукту. Виникає задача, яка пов’язана з оцінюванням якості програмного забезпечення. Якість програмного забезпечення залежить від різних характеристик коду, які можна оцінювати за допомогою метрик, що відображають його складність, зв'язність та відповідність архітектурним вимогам. Для вирішення цієї задачі було розроблено різноманітні підходи, методи та моделі [1 - 7]. Одним із підходів до оцінювання якості є метод [2], який використовує довірчі та прогнозні інтервали нелінійної регресії для прогнозування метрики RFC (Response for Class) на основі метрик CBO (Coupling Between Object Classes) та WMC (Weighted Methods per Class) [8–10]. Для обґрунтування вибору нелінійної регресійної моделі було проведено аналіз існуючих моделей оцінювання RFC в залежності від CBO та WMC для Kotlin [11]. Порівняльний аналіз, проведений у дослідженні [12], показав значні відмінності у значеннях інтервалів метрик RFC, CBO та WMC для Java- і Kotlin-застосунків. Ці відмінності свідчать про те, що існуючі моделі оцінювання, побудовані для Kotlin, є непридатними для використання в Java-проектах через їхню невідповідність. Це підкреслює необхідність розробки нелінійної регресійної моделі, яка враховує особливості архітектури Java-застосунків, підвищуючи достовірність оцінювання та враховуючи унікальні характеристики Java-застосунків, саме для онлайн-покупок.

Мета дослідження. Метою даного дослідження є підвищення достовірності оцінювання якості Java-застосунків для онлайн-покупок шляхом удосконалення математичної моделі, яка враховує інші діапазони значень метрик RFC, CBO та WMC. Для досягнення мети дослідження необхідно виконати такі завдання:

1. Провести критичний огляд існуючих підходів та моделей оцінювання якості програмного забезпечення, зокрема для Java-застосунків, а також вивчити підходи до використання метрик RFC, CBO та WMC.

2. Обґрунтувати вибір метрик RFC, CBO та WMC для оцінювання якості Java-застосунків, з’ясувавши їхній вплив на основні параметри якості програмного забезпечення.

3. Зібрати дані про метрики RFC, CBO та WMC з відкритих Java-застосунків для онлайн-покупок. Провести обробку даних, зокрема видалення викидів.

4. Розробити математичну модель для оцінювання метрики RFC у залежності від метрик CBO та WMC, використовуючи перетворення у вигляді десяткового логарифму.

5. Використати метод найменших квадратів для оцінки параметрів побудованої моделі та визначення її якості на основі таких показників, як *MMRE* та *PRED(0.25)*.

6. Провести тестування розробленої моделі на окремій вибірці Java-застосунків, щоб підтвердити її точність та ефективність у прогнозуванні якості застосунків для онлайн-покупок з відкритим кодом на Java.

Об'єкт дослідження. Об'єктом дослідження є процес оцінювання якості програмного забезпечення для онлайн-покупок, розробленого на Java, на основі аналізу метрик коду, таких як RFC, CBO та WMC.

Предмет дослідження. Предметом дослідження є математична модель для оцінювання якості Java-застосунків у залежності від метрик RFC, CBO та

WMC, що дозволяє врахувати специфічні архітектурні особливості Java-застосунків у сфері онлайн-покупок та розробка програми для її реалізації.

Методи дослідження. У роботі застосовано наступні методи: статистичну обробку даних для аналізу і відбору вхідних даних про метрики RFC, CBO та WMC з відкритих Java-застосунків; метод видалення викидів за допомогою квадратів відстаней Махаланобіса для очищення вибірки від аномальних значень; метод найменших квадратів для побудови і оцінки параметрів нелінійної регресійної моделі; нормалізуючі перетворення, зокрема десятковий логарифм, для відображення нелінійних залежностей між метриками.

Наукова новизна. Наукова новизна полягає у удосконаленні математичної моделі для оцінювання якості Java-застосунків для онлайн-покупок, яка враховує особливості метрик RFC, CBO та WMC. Модель враховує специфічні характеристики для метрик RFC, CBO та WMC цих застосунків та використовує нові значення параметрів. Удосконалена модель, побудована із застосуванням логарифмічного перетворення, дозволяє достовірно визначити залежності між цими метриками, підвищуючи достовірність оцінювання якості програмного забезпечення.

Особистий внесок здобувача. Всі результати, викладені в роботі, отримані автором самостійно. У спільній публікації [13], підготовленій разом із науковим керівником, автору належить: проведення аналізу існуючих моделей оцінювання якості, збір та попередня обробка даних, розробка нелінійної регресійної моделі, отримання параметрів моделі та аналіз її якості.

Апробація результатів досліджень. Результати досліджень були опубліковані на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» яка проводилась 30-31 жовтня 2024 року, організованій Національним університетом кораблебудування імені адмірала Макарова.

Публікації. За результатами досліджень опубліковано одну наукову працю, а саме матеріали інтернет-конференції.

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТРИК ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ JAVA-ЗАСТОСУНКІВ

1.1 Аналіз існуючих підходів та моделей оцінювання якості програмного забезпечення

Оцінка якості програмного забезпечення є важливим етапом у розробці та підтримці програм, оскільки вона дозволяє визначити ступінь відповідності програмного продукту вимогам користувачів, а також виявити потенційні проблеми, що можуть вплинути на ефективність та безпеку програм. Існують різні підходи та моделі для оцінювання якості програмного забезпечення, кожен з яких має свої переваги і недоліки, зокрема методика метрик програмного коду, що дозволяє об'єктивно вимірювати складність, підтримуваність та ефективність коду. Основні метрики включають:

- Цикломатична складність (Cyclomatic Complexity), яка визначає кількість незалежних шляхів у програмі, оцінюючи складність логіки коду та його тестування.
- CBO (Coupling Between Object Classes) вимірює рівень зв'язності між класами, де високий рівень вказує на складнощі у підтримці коду.
- LOC (Lines of Code) оцінює обсяг коду, але має обмеження у врахуванні структури.
- RFC (Response For Class) відображає кількість методів, що можуть бути викликані через клас, а високий рівень може свідчити про надмірну відповідальність класу.
- WMC (Weighted Methods per Class) вимірює складність методів у класі, допомагаючи оцінити його підтримуваність.

– Додаткові метрики, як-от NOC (Number of Children) і DIT (Depth of Inheritance Tree), допомагають оцінити масштабованість та глибину спадкування класу.

Ці метрики дозволяють комплексно оцінити якість програмного забезпечення, виявляючи потенційні проблеми для подальшої оптимізації.

Модель FURPS [3]: Ця модель була розроблена компанією IBM і описує якість програмного забезпечення через п'ять основних аспектів:

– F (Functionality) – функціональність, яка включає в себе відповідність програмного продукту вимогам, можливість виконувати задані функції.

– U (Usability) – зручність користування, яка оцінює, наскільки програмне забезпечення просте і ефективне для кінцевих користувачів.

– R (Reliability) – надійність, що включає стабільність роботи програми, здатність виконувати функції без збоїв протягом певного часу.

– P (Performance) – продуктивність, що визначає швидкість виконання програмних операцій і ресурсну ефективність.

– S (Supportability) – підтримуваність, тобто здатність програмного забезпечення адаптуватися до змін, бути обслуговуваним та тестованим.

Модель CMMI (Capability Maturity Model Integration) [4]: Це модель, яка оцінює рівень зрілості процесів розробки програмного забезпечення. Вона включає п’ять рівнів зрілості, від початкового (немає стандартизованих процесів) до оптимізованого (процеси постійно вдосконалюються). Модель допомагає організаціям визначити, на якому етапі розвитку знаходяться їхні процеси, та які напрямки потребують покращення.

Модель 6-Sigma [5]: Ця модель фокусується на зменшенні дефектів у процесі розробки програмного забезпечення та досягненні максимальної якості продукту. Вона базується на статистичних методах, спрямованих на варіативність в процесах та їх оптимізацію для досягнення найменшого числа дефектів.

Модель Dromey [6] стверджує, що оцінка є різною для кожного продукту, отже, необхідна динамічна ідея для моделювання процесу. Тому основна ідея запропонованої моделі полягала в тому, щоб отримати модель, достатньо широку для роботи з різними системами. Модель спрямована на покращення розуміння взаємозв'язку між атрибутами (характеристиками) та субатрибутами (підхарактеристиками) якості. У цій моделі визначено два рівні: атрибути високого рівня та підпорядковані атрибути. Тому ця модель страждає від відсутності критеріїв для вимірювання якості програмного забезпечення.

Модель МакКолла [7] була розроблена Управлінням електронних систем ВПС США (ESD), Римським центром розвитку авіації (RADC) та компанією General Electric з метою підвищення якості програмних продуктів. Ця модель була розроблена для оцінки взаємозв'язків між зовнішніми факторами та критеріями якості продукту. Тому характеристики якості були класифіковані за трьома основними типами: 11 факторів, які описують зовнішній вигляд програмного забезпечення (погляд користувача), 23 критерії якості, які описують внутрішній вигляд програмного забезпечення (погляд розробника), і метрики, які визначають і використовуються для забезпечення шкали і методу вимірювання. Для спрощення кількість факторів було скорочено до одинадцяти. Цими факторами є коректність, надійність, ефективність, цілісність, зручність використання, ремонтопридатність, тестованість, гнучкість, портативність, можливість повторного використання та інтероперабельність. Основним внеском цієї моделі є взаємозв'язок між характеристиками якості та метриками. Однак, модель не розглядає безпосередньо функціональність програмних продуктів.

Також для оцінки якості програмного забезпечення використовуються лінійні моделі регресії, які базуються на простих математичних зв'язках між різними метриками коду. Однак ці моделі часто не здатні адекватно враховувати складні взаємозв'язки між характеристиками програмних продуктів, особливо коли йдеться про такі специфічні метрики, як RFC, CBO та WMC для Java-

застосунків. Лінійні моделі можуть бути недостатньо достовірними, оскільки часто не враховують нелінійні залежності між показниками, що знижує їхню достовірність для реальних проектів.

Нелінійні моделі оцінювання якості [2] програмного забезпечення є більш ефективними, коли необхідно врахувати складні взаємозв'язки між метриками, такими як RFC, CBO та WMC. Ці моделі здатні відображати нелінійні залежності між характеристиками програми та її якістю, що дозволяє досягати вищої точності у прогнозуванні. Одним з поширених підходів є використання методів, таких як перетворення в логарифмічну шкалу або експоненційні моделі, які дозволяють лінійно представляти нелінійні залежності. Однак для Java-застосунків, зокрема для онлайн-покупок, важливо розглянути, як ці моделі адаптуються до специфічних особливостей цього середовища.

Для Java-застосунків були розроблені кілька підходів, що враховують специфіку цієї мови програмування, зокрема її об'єктно-орієнтовану природу. Однією з основних метрик для оцінки якості Java-коду є RFC (Response for Class), яка визначає, скільки методів може бути викликано класом. Важливо, що ця метрика безпосередньо впливає на тестованість, підтримуваність і розширюваність програмного забезпечення. Метрика CBO (Coupling Between Object classes) вимірює зв'язність між класами, що також є важливим аспектом для оцінки складності програми, адже висока зв'язність може ускладнювати внесення змін у систему без небажаних побічних ефектів. WMC (Weighted Methods per Class) — оцінює складність класу на основі кількості і складності його методів.

Попри великий досвід у застосуванні цих метрик для різних типів програм, важливо зазначити, що для Java-застосунків, особливо в контексті онлайн-покупок, існуючі моделі можуть бути недостатньо точними. Це підтверджується результатами порівняльного аналізу моделей оцінки для Kotlin, де з'ясувалося, що інтервали значень метрик RFC, CBO та WMC для Kotlin та Java значно

відрізняються. Відповідно, існуючі моделі для Kotlin не можуть бути безпосередньо використані для Java-застосунків, що підкреслює необхідність адаптації методів до специфіки Java.

Модель оцінювання якості [2] базується на застосуванні довірчих та прогнозних інтервалів для метрики RFC, яка прогнозується на основі значень CBO та WMC. Довірчий інтервал дозволяє окреслити діапазон, у межах якого з високою ймовірністю перебувають середні значення RFC, що дає змогу оцінити загальну якість програмного забезпечення. Прогнозний інтервал, у свою чергу, визначає можливий діапазон коливань RFC для нових даних, забезпечуючи більш гнучкий і точний підхід до аналізу якості конкретних застосунків. Такий підхід сприяє класифікації якості програмного забезпечення на три рівні: високий, середній та низький, залежно від розташування значення RFC відносно меж довірчого та прогнозного інтервалів. Зокрема:

– Якщо значення RFC знаходиться всередині довірчого інтервалу, це вказує на середню якість застосунку.

– Якщо значення RFC розташоване між межами довірчого та прогнозного інтервалів, це свідчить про високу якість.

– Вихід значення RFC за верхню межу прогнозного інтервалу може свідчити про низьку якість програмного продукту.

Аналіз існуючих підходів показує, що для Java-застосунків необхідно розробити математичну модель, яка б враховувала специфічні характеристики метрик RFC, CBO та WMC для онлайн-покупок. Для цього потрібно використовувати методи нелінійного регресійного аналізу, зокрема перетворення у вигляді десяткового логарифму для відображення залежностей між метриками, що дозволяє більш точно оцінювати якість програмних продуктів.

1.2 Особливості розробки застосунків для онлайн покупок на Java

Розробка застосунків для онлайн покупок на Java включає в себе кілька важливих аспектів, що залежать від специфіки електронної комерції та вимог до функціональності платформи.

Основним етапом є правильна архітектура. Для цього важливо застосовувати модульність, розділяючи функції на окремі частини, такі як обробка замовлень, оплата, каталог товарів тощо. Це дозволяє легше управляти кодом і здійснювати підтримку. Для великих платформ з високим трафіком часто використовують мікросервісну архітектуру [13], де кожен сервіс відповідає за певний функціонал, наприклад, обробку платежів або управління товарами. Також застосовуються розподілені системи, зокрема технології, як RabbitMQ або Kafka для асинхронної обробки великих обсягів запитів.

Одним із найпоширеніших інструментів для серверної частини є Spring Framework, зокрема Spring Boot, завдяки своїй простоті та потужності для створення масштабованих веб-застосунків. Використовуються також додаткові компоненти, як Spring Security для забезпечення безпеки (автентифікація та авторизація), Spring Data для роботи з базами даних і Spring Cloud для мікросервісних рішень. Для роботи з базами даних часто використовують Hibernate або JPA для об'єктно-орієнтованого підходу, а для створення фронтенду можна застосовувати JavaFX або Thymeleaf для шаблонів веб-додатків. Для інтеграцій з іншими системами зручно використовувати Apache Camel або Spring Integration.

Вибір між SQL і NoSQL базами даних залежить від потреб проекту. Реляційні бази даних, як MySQL або PostgreSQL, зазвичай використовуються для збереження структурованих даних, в той час як нереляційні бази даних, як MongoDB, можуть бути кращими для зберігання великих обсягів

неструктурованої інформації [14]. Усі транзакції повинні бути правильно оброблені, щоб забезпечити цілісність даних, особливо під час операцій, таких як оформлення замовлення або здійснення оплати. Окрім того, для покращення швидкості роботи часто використовують кешування (наприклад, Redis).

Забезпечення безпеки є важливим аспектом у розробці застосунків для онлайн покупок. Для цього використовуються протоколи HTTPS для захисту передачі даних. Потрібно також здійснювати валідацію введених даних для захисту від таких загроз, як SQL-ін'єкції та XSS-атаки. Для автентифікації користувачів часто застосовують OAuth2 та JWT (JSON Web Tokens).

Застосунки для онлайн покупок повинні підтримувати інтеграцію з популярними платіжними системами, такими як PayPal, Stripe, LiqPay чи ПриватБанк. Це дозволяє користувачам безпечно здійснювати платежі. Усі платіжні транзакції мають бути захищені за допомогою платіжних шлюзів і відповідати стандартам безпеки, таким як PCI DSS.

Незважаючи на те, що Java здебільшого використовується для розробки серверної частини, важливо також враховувати мобільну сумісність застосунку. Більшість покупок здійснюється через смартфони, тому важливо, щоб серверна частина могла ефективно взаємодіяти з мобільними додатками. Для цього можна використовувати Java для Android або кросплатформенні фреймворки, як React Native.

Для забезпечення високої якості застосунку необхідно проводити юніт-тести за допомогою JUnit та мокінг за допомогою Mockito. Важливо також налагодити процеси постійної інтеграції та розгортання (CI/CD) з використанням таких інструментів, як Jenkins або GitLab CI. Для моніторингу продуктивності застосовуються інструменти на зразок Prometheus, Grafana та ELK stack (Elasticsearch, Logstash, Kibana).

Для забезпечення швидкої роботи застосунку, особливо в умовах високого навантаження, використовуються технології асинхронної обробки запитів, черги

повідомлень, а також налаштовується балансування навантаження. Це дозволяє ефективно обробляти велику кількість запитів і забезпечувати стабільну роботу навіть за високої активності користувачів.

Хоча розробка інтерфейсів користувача не є прямим завданням для Java, важливо забезпечити швидку та ефективну взаємодію між сервером і фронтендом. Для цього застосовуються REST API або GraphQL, що дозволяє забезпечити інтерактивність і швидкість роботи інтерфейсу.

Для покращення користувацького досвіду важливо вести історію покупок, зберігати профілі користувачів і використовувати ці дані для персоналізації контенту та рекомендацій. Для цього можуть застосовуватися різні алгоритми для рекомендацій товарів на основі попередніх покупок або переглядів.
Розробка застосунків для онлайн покупок на Java вимагає комплексного підходу, що включає вибір правильних технологій, забезпечення безпеки, оптимізацію продуктивності і підтримку масштабованості системи.

1.3 Обґрунтування вибору нелінійної регресії для оцінювання RFC застосунків для онлайн покупок на Java

Для оцінювання метрики RFC (Response for Class) Java-застосунків, призначених для онлайн покупок, було обрано підхід із використанням нелінійної регресії на основі нормалізуючого перетворення у вигляді десяткового логарифму. Таке рішення базується на кількох ключових аспектах.

Лінійні моделі регресії широко застосовуються в програмній інженерії через їхню простоту та інтерпретованість. Однак для специфічних метрик, таких як RFC, CBO (Coupling Between Object Classes) і WMC (Weighted Methods per Class), вони часто виявляються недостовірними. Основні причини цього полягають у наступному:

Високий рівень мультиколінеарності [15] між предикторами вказує на значну залежність між змінними, що ускладнює інтерпретацію моделі та визначення впливу кожного предиктора окремо. Метрики RFC (8,2), CBO (6,19) і WMC (4,22) демонструють високі значення VIF, що ускладнюють застосування лінійної регресії та підкреслює необхідність модифікації моделі та використання методів, здатних зменшити вплив мультиколінеарності.

Відсутність нормальності даних. Тест Мардія на нормальність показав значну асиметрію (1132,1919) і ексцес (85,5667) у розподілі метрик, що свідчить про непридатність припущення нормальності залишків, необхідного для лінійної регресії.

Нелінійні залежності між метриками. Залежності між RFC, CBO та WMC є нелінійними. Наприклад, збільшення CBO може експоненціально впливати на значення RFC, що не враховується у лінійних моделях.

Широкий діапазон значень метрик. Метрики мають широкий діапазон значень: CBO [0,4; 6,39], WMC [1,6; 20,85], RFC [1,6; 13,95]. Лінійні моделі не здатні адекватно врахувати такі варіації, що призводить до високої похибки.

Нелінійна регресія дозволяє моделювати складні залежності між метриками, підвищуючи точність прогнозування. Для вирівнювання масштабу та зменшення асиметрії розподілу було застосовано десяткове логарифмічне перетворення. Його ефективність підтверджено такими результатами:

Покращення розподілу даних. Після логарифмічного перетворення значення асиметрії знизилося до 99,0492, а ексцесу – до 23,474. Це свідчить про значне наближення даних до нормального розподілу.

Стабілізація залежностей. Логарифмічне перетворення дозволяє згладити варіації значень, забезпечуючи побудову більш стабільної та інтерпретованої моделі. Наприклад, метрика RFC у логарифмічній шкалі краще корелює з CBO та WMC, що спрощує аналіз.

Урахування архітектурних особливостей Java-застосунків. Java-застосунки, особливо для онлайн покупок, мають високі вимоги до продуктивності, масштабованості та підтримуваності. Нелінійна модель краще відображає взаємозв'язки між складністю (CBO), кількістю методів (WMC) та здатністю класу до взаємодії (RFC).

Застосування нелінійних регресійних моделей для оцінки якості програмного забезпечення вже успішно використовується в інших мовах програмування, зокрема в Kotlin. Такі дослідження підтверджують ефективність нелінійних методів для моделювання залежностей між метриками якості програмного коду. У роботі [12] представлено модель, яка використовує десятковий логарифм для врахування нелінійних залежностей між метриками CBO, WMC та RFC. Модель має вигляд:

$$Y = 10^{\varepsilon + b_0} X_1^{b_1} X_2^{b_2} \quad (1.1),$$

де Y – це значення метрики RFC;

X_1 та X_2 – це значення метрик CBO і WMC відповідно;

ε – гаусівська випадкова величина з нульовим математичним сподіванням та середньоквадратичним відхиленням, оцінка якого дорівнює $0{,}1128$;

b_0, b_1 та b_2 є оцінками параметрів, які мають відповідно такі значення: $-0{,}306892$, $0{,}228584$ та $1{,}36206$.

Модель для Kotlin не враховує всіх особливостей Java-застосунків, зокрема тих, що стосуються онлайн-платформ. Дані, отримані з Java-проектів, мають наступні діапазони значень для метрик:

- CBO: [0,4; 6,39];
- WMC: [1,6; 20,85];
- RFC: [1,6; 13,95].

Для Kotlin ці ж метрики мають значення в діапазонах:

- CBO: [0,995; 4,724];
- WMC: [2,167; 13,020];
- RFC: [2,118; 19,487].

Це підкреслює важливість розробки спеціалізованої моделі для оцінки RFC саме для Java-проектів у контексті онлайн покупок. Існуючі моделі, застосовані до Kotlin, не забезпечують достатньої достовірності для середовища Java, оскільки Java-застосунки мають інші вимоги щодо масштабування, складності коду та взаємодії компонентів.

2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ ДЛЯ ОНЛАЙН-ПОКУПОК З ВІДКРИТИМ КОДОМ НА JAVA

2.1 Збір та попередня обробка даних метрик Java-застосунків для онлайн покупок

Для розробки нелінійної регресійної моделі, яка оцінюватиме метрику RFC для Java-застосунків з відкритим кодом, що використовуються в онлайн-покупках, було зібрано дані з 100 таких проектів, доступних на GitHub. Метрики RFC, CBO та WMC на рівні застосунків були отримані за допомогою інструменту SourceMeter [16]. Для кожного проекту було визначено відносні значення метрик, що залежать від кількості класів у проекті: RFC виступає як залежна змінна Y, яку необхідно передбачити за допомогою моделі, а CBO та WMC є незалежними змінними X_1 та X_2, на основі яких здійснюється прогнозування RFC. Для перевірки багатовимірних даних на нормальність було використано тест Мардія [17], який враховує характеристики багатовимірної асиметрії та ексцесу. У рамках цього тесту обчислюються два ключові параметри: асиметрія та ексцес. Оцінювання асиметрії базується на відстанях Махаланобіса [18]:

$$d_i^2 = (X_i - \bar{X})^2 S_N^{-1} (X_i - \bar{X}), \quad (2.1),$$

де $\bar{X}$ – вектор середніх значень вибірки для змінних X_1, X_2, Y;

X_i – вектор спостережень (значення змінних X_1, X_2, Y, для кожного застосунку);

S_N – коваріаційна матриця:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})(X_i - \bar{X})^T. \quad (2.2),$$

Набір даних, де CBO є змінною X_1, WMC – змінною X_2, а RFC – змінною Y, D^2 – розрахований квадрат відстаней Махаланобіса представлений у таблиці 2.1.

Таблиця 2.1 – Набір даних

#	URL	RFC	CBO	WMC	D^2
1	2	3	4	5	6
1	https://github.com/bmvermeer/JavaCoffeeShop	7,9750	1,9250	7,2750	0,3437
2	https://github.com/npolyakova/shop	4,2500	0,6250	3,5000	1,4077
3	https://github.com/visalsrimanga/Shopping-Cart-Web-App	1,8148	0,7037	1,5926	1,4452
4	https://github.com/marcoman2/CafeShopManagementSystem	11,6250	1,3750	22,5000	2,0224
5	https://github.com/songsimo/pion-shop	3,8000	1,2667	3,2000	0,8912
6	https://github.com/Roshan-Patro/Online-Shopping-Application	4,9057	3,0566	5,6792	1,3593
7	https://github.com/stellariumImpl/Yoyoshop	9,5526	3,4474	7,5789	1,3222
8	https://github.com/BogomolovISerg/ShopCarts	5,5789	2,1579	4,6316	0,4347
9	https://github.com/Shubh2-0/Shopzilla-Online-Shopping-Platform	3,3536	2,1381	3,7514	0,8951
10	https://github.com/alokrai0607/EazyShop	4,1837	2,4898	4,2041	0,9352
11	https://github.com/tusimegod123/cs490-online-shopping-cart	2,3297	1,3516	3,4231	1,0467
12	https://github.com/edison-lhk/instamall	4,2143	2,1429	4,0857	0,6944
13	https://github.com/arpanghosh2414/E-Commerce-Website	2,3000	2,1250	2,7500	1,1192
14	https://github.com/MaarkNassef/E-Shop	3,6218	1,2017	2,8824	0,958
15	https://github.com/ycshang123/shop_online	2,0659	2,1538	2,1978	1,1708
16	https://github.com/itwill-project/Shopping-Mall	4,7040	1,8880	3,0800	0,5883
17	https://github.com/NHN-YesAladin/yesaladin_shop	6,2473	4,4236	4,0382	2,4701
18	https://github.com/Yashmerino/online-shop	4,2639	2,4583	3,6389	0,8776
19	https://github.com/David-Garrancho/CoffeeShopSystem	7,4035	1,4035	8,2105	0,6541
20	https://github.com/livelysb/coffee-shop	2,4103	2,4103	1,5897	1,2147
21	https://github.com/devendrasanghavi404/shoppers-retail	4,8333	0,5667	3,7333	1,4893
22	https://github.com/WalaaASA/ItesaqShop	3,4224	1,2241	4,2328	0,9299
23	https://github.com/jwanderson314/Pizza_Shop	1,1000	0,2000	1,2000	1,8644

Продовження таблиці 2.1

1	2	3	4	5	6
24	https://github.com/Advanced-Programming-1401/Internet-Shop	13,3333	2,0000	11,6667	1,4662
25	https://github.com/OOP-Project-ShopingApp/PeaShop	11,1176	1,1765	15,2353	1,4039
26	https://github.com/jsyGitTest/jingShop	8,7079	2,3258	10,3034	0,4108
27	https://github.com/Vitaly2022/shop	10,4915	3,3220	8,2373	1,2519
28	https://github.com/coderzhn/shop	3,0769	2,0962	3,2692	0,9271
29	https://github.com/codaholichq/shop	3,8000	0,8000	2,8000	1,2749
30	https://github.com/BrandanBurgess/ShopTrack	4,7717	1,5870	4,4239	0,5815
31	https://github.com/UnsafeDodo/gui-shop	5,9259	2,7778	7,1481	0,9723
32	https://github.com/JohnWCCI/pizza-shop-forClass	7,9333	1,5333	5,4000	0,9101
33	https://github.com/longleDevops/Car-Shop-Application	12,8571	2,5714	9,8286	1,3418
34	https://github.com/ym1085/springboot-shopping-mall	4,7071	2,6061	4,5859	0,9278
35	https://github.com/dasswayam/e-shopping-cart	7,4444	1,9444	5,5556	0,4674
36	https://github.com/josdem/shopping-cart-service	1,4400	1,0400	1,2400	1,2941
37	https://github.com/jairoArh/WorkShopBTS	15,0000	1,2000	11,8000	2,4362
38	https://github.com/Yujin051/shopping-mall-team-project	3,9286	2,2143	3,0893	0,7862
39	https://github.com/kousen/shopping_v3	6,0625	1,1250	5,1875	0,959
40	https://github.com/Skirtek/ShoppingAppApi	4,6154	1,3462	4,1538	0,7575
41	https://github.com/Omarmasalmah/COMP333-Coffe-shop	10,5200	1,5600	16,3200	1,1997
42	https://github.com/ahmedemad3/shopping-cart-event-sourcing	3,4444	1,4444	3,4444	0,8231
43	https://github.com/newbee-ltd/newbee-mall	14,3846	3,2154	14,6923	1,5553
44	https://github.com/roadbook2/shop	3,6296	2,7407	3,9815	1,2604
45	https://github.com/shopizer-ecommerce/shopizer	16,1460	6,3879	11,5182	4,0786
46	https://github.com/lrwinx/shop	1,5833	0,7917	2,1667	1,3989
47	https://github.com/Manphil/shop	20,5100	2,4700	18,6300	2,8441
48	https://github.com/Joryun/Shop	10,4286	2,6327	8,2245	0,8563
49	https://github.com/RojerAlone/shop	12,6491	3,1754	11,6140	1,312
50	https://github.com/albfan/jEdit	13,9507	4,8335	20,8522	3,1251
51	https://github.com/chau-nm/ShoppingApp	1,6000	0,4000	1,6000	1,6775
52	https://github.com/BizSpringSource/bizspring-vue3-opensource	3,4922	2,4974	7,4844	1,4284
53	https://github.com/KBTG-Kampus-ClassNest-SE-Java/workshopb2-group-1	2,6250	1,6000	2,9250	0,9313

Продовження таблиці 2.1

1	2	3	4	5	6
54	https://github.com/khanhduzz/matcha	2,5349	0,7907	2,0465	1,3297
55	https://github.com/digantv/shoping-management-mono-repo	6,5333	1,1333	4,6333	1,0727
56	https://github.com/NaeDdoCo/nutritional-shopping-mall-web	13,2830	3,3585	10,0566	1,5875
57	https://github.com/f-lab-edu/shopping-mall-fashion	4,0361	3,1031	3,0103	1,4609
58	https://github.com/chamithKavinda/Coffee-Shop-POS-JavaEE-Backend	3,8400	2,5200	4,7200	1,0718
59	https://github.com/JavaGraduationProject/BeautyShoppingMallManagementSystem	16,7500	1,5000	16,5500	2,3652
60	https://github.com/JavaGraduationProject/ChinaShoppingManagementSystem	18,7500	2,8438	16,0938	2,4463
61	https://github.com/idiotman-2212/EJB3_Stateful_ShoppingCart	7,0000	1,0000	13,5000	1,3243
62	https://github.com/JavaGraduationProject/HuoyingAnimationShoppingManagementSystem	10,9524	2,7857	9,5238	0,9205
63	https://github.com/OlegRyazancev/online-shop	4,0750	2,6393	3,2107	1,0512
64	https://github.com/StarJ0405/ShoppingMall	3,4578	2,5964	6,2470	1,3693
65	https://github.com/deeptesh-rout/dream-shops	3,8000	2,7250	3,4250	1,1845
66	https://github.com/Jonny0301/Shopping	3,4468	1,2340	3,5532	0,9258
67	https://github.com/JavaGraduationProject/CakeShoppingMallManagementSystem	21,2000	2,7500	16,9000	3,086
68	https://github.com/JavaGraduationProject/OnlineShoppingMallManagementSystem	19,4528	1,4906	18,8113	2,9694
69	https://github.com/The-inhabito/inhabito-ar-mobileApp	3,3684	0,8158	4,0000	1,217
70	https://github.com/shidubei/SA59-ShoppingCart	6,8333	2,3333	6,2083	0,3456
71	https://github.com/Ravinduchathuranga/Textile-shop	2,9259	0,9259	2,7222	1,189
72	https://github.com/yan-jordan/Oline-Shop	18,5000	0,5000	71,0000	9,2174
73	https://github.com/akiltipu/secure-shop-devops	5,7778	1,2778	5,3889	0,7686
74	https://github.com/saravanan81java/shopping-card-validation	5,5833	0,8333	6,9167	1,1319
75	https://github.com/MariiaLobanova/E-Commerce-Shop-RESTful-API	2,0816	2,3469	2,8367	1,3026
76	https://github.com/JavaGraduationProject/TmallShoppingMallManagementSystem	10,1429	4,5102	8,7959	2,3315
77	https://github.com/le0n67/XinFur_Mall	11,9286	2,5000	9,0714	1,1595
78	https://github.com/Beksultan2020/Online_Shop_Project_Java	5,7619	2,6667	4,8571	0,7942

Продовження таблиці 2.1

1	2	3	4	5	6
79	https://github.com/every821/mobile-shop-android	6,4711	2,0667	5,6533	0,2518
80	https://github.com/phani-sriram/shop	4,1111	1,1667	4,5000	0,899
81	https://github.com/LeeHSeoK/Shop	2,3636	0,5455	2,0909	1,5211
82	https://github.com/balllm/shop	3,0000	0,2500	4,2500	1,7012
83	https://github.com/leechistest/shop	2,9744	2,3333	3,4872	1,0931
84	https://github.com/michael-thoughtworker/Shop	4,4000	1,0000	4,0000	1,0461
85	https://github.com/JavierSantiagoSalazar/shop	2,2000	1,4000	1,8000	1,0447
86	https://github.com/fengyongge/shopcar	5,7407	1,5185	5,4074	0,5478
87	https://github.com/xvmingyuan/shop	18,2957	2,6348	15,9478	2,3651
88	https://github.com/GGsnake/ShopMall	8,5217	2,2957	9,6870	0,3356
89	https://github.com/louisgeek/LouisShopCart	4,6897	1,6552	4,7586	0,5493
90	https://github.com/lilishop/lilishop	6,0912	5,1509	6,8961	3,3479
91	https://github.com/nashtech-garage/yas	4,9055	2,3160	4,0212	0,6517
92	https://github.com/myxh/CoolShopping	5,9460	1,8993	7,8345	0,4002
93	https://github.com/funtl/spring-cloud-alibaba-my-shop	9,0980	1,0588	9,3333	1,2343
94	https://github.com/atguigu01/Shopping	6,4784	2,0784	8,5333	0,3994
95	https://github.com/sczyh30/vertx-blueprint-microservice	13,0000	2,2000	13,5375	1,1875
96	https://github.com/chengzhx76/weixin-shop-spring-cloud	10,6238	1,8218	9,4455	0,9591
97	https://github.com/sjr7/seckill	10,2000	2,6667	8,5333	0,7885
98	https://github.com/codingxin/OnlineSchoolShop	19,9239	2,5635	18,2335	2,6835
99	https://github.com/MaJesTySA/miaosha_Shop	11,3485	2,5455	9,4242	0,9621
100	https://github.com/shashirajraja/onlinebookstore	8,0769	3,5385	7,3077	1,4515

Формули для обчислення асиметрії та ексцесу за методом Мардія мають такий вигляд [19]:

$$\beta_{1,k} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[\left(\mathrm{X}_i - \overline{\mathrm{X}}\right)^T \mathrm{S}_N^{-1}\left(\mathrm{X}_j - \overline{\mathrm{X}}\right)\right]^3, \quad (2.3),$$

$$\beta_{2,k} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\mathrm{X}_i - \overline{\mathrm{X}}\right)^T \mathrm{S}_N^{-1}\left(\mathrm{X}_i - \overline{\mathrm{X}}\right)\right]^2, \quad (2.4),$$

де X — вектор випадкових величин розмірності $k \times 1$, $X = (X_1, X_2, \dots, X_k)^T$, а S_N — зміщена вибіркова коваріаційна матриця.

Для перевірки асиметрії $\beta_{1,k}$ (2.3) обчислюють тестову статистику [19]:

$$\frac{N}{6}\beta_{1,k}, \tag{2.5}$$

яка наближається до розподілу χ^2 зі ступенями свободи $k(k+1)(k+2)/6$ і рівнем значущості $\alpha = 0{,}005$.

Для ексцесу $\beta_{2,k}$ (2.4) використовують тестову статистику $z_{2,k}$, що відповідає $1-\alpha$-квантилю нормального розподілу із середнім значенням $k(k+2)$ і дисперсією $8k(k+2)/N$.

Перевірка нормальності виконується шляхом порівняння:

- значення (2.5) із квантилем $\chi^2_{(k+1)(k+2)/6,\alpha}$,
- значення $\beta_{2,k}$ із $z_{2,k}$.

Якщо хоча б одна з тестових статистик перевищує відповідний критичний рівень, розподіл не є нормальним.

У цьому дослідженні тестової статистики асиметрії Мардія становило 1132,1919, що перевищує критичне значення 25,1882, а ексцес дорівнював 85,5667, також перевищуючи критичне значення 17,8217. Це свідчить про значну асиметрію даних, що не відповідають нормальному розподілу [20]. Через невідповідність умовам лінійної регресії (зокрема, лінійності та нормальності залишків) метод асиметрії Мардія виявився непридатним.

Для корекції було використано нелінійну регресійну модель із нормалізуючим перетворенням у вигляді десяткового логарифму, що дозволило краще описати залежності між метриками.

Після застосування перетворення значення асиметрії та ексцесу значно знизилося до 99,0492 та 23,474 відповідно. Це свідчить про ефективність застосування логарифмічного перетворення для вирівнювання розподілу даних.

Наступним кроком є ідентифікація та видалення викидів із вихідних даних. Для цього виконаємо наступні кроки [21]:

1. Нормалізація даних за допомогою логарифмування значень метрик у десятковій системі.

2. Розраховуємо коваріаційну матрицю S_N (2.2) для трьох змінних на нормалізованих даних.

3. Знаходимо зворотну коваріаційну матрицю ${S_N}^{-1}$.

4. Для кожного спостереження обчислюємо квадрат відстані Махаланобіса на нормалізованих даних за формулою (2.1).

5. Обчислюємо тестову статистику (TS) для кожного спостереження, яка дозволяє оцінити, наскільки воно відрізняється від середнього значення з урахуванням коваріаційної структури даних [22]:

$$T_i = (N-k)Nd_i^2/\big((N^2-1)k\big), \tag{2.6}$$

7. Обчислюємо $F_{\text{крит}}$ як [23]:

$$F_{\text{крит}} = F(\alpha, 3, n-3), \tag{2.7}$$

де α – рівень значущості (0,005),

3 – кількість ступенів вільності,

n – розмір вибірки.

8. Якщо T_i (2.6) $>$ $F_{\text{крит}}$ (2.7), то і–те спостереження є викидом і його вилучаємо з вибірки.

9. Повторяємо крок 2 – 8 поки не буде вилучено усі викиди.

10. Далі потрібно побудувати рівняння нелінійної регресії (1.1) та розрахувати інтервал прогнозування який обчислюється як:

$$10^{\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{1+\frac{1}{N}+\left(z_X^+\right)^T S_Z^{-1}\left(z_X^+\right)\right\}^{\frac{1}{2}}\right)} \tag{2.8}$$

де $\hat{Z}_Y$ – прогнозне значення, отримане з рівняння лінійної регресії (1.1) для нормалізованих даних;

$Z_1 = log_{10}(X_1)$ та $Z_2 = log_{10}(X_2)$ – нормалізовані значення метрик CBO та WMC відповідно;

$t_{\alpha/2,v}$ – квантиль розподілу Стьюдента з рівнем значущості $\alpha/2$ та $v = N - 3$ ступенями вільності;

z_X^+ – вектор з компонентами $(Z_{1_i} - \bar{Z}_1, Z_{2_i} - \bar{Z}_2)$ для і-го спостереження;

$\bar{Z}_j = \frac{1}{N}\sum_{i=1}^{N} Z_{j_i}, j = 1,2$ – середні значення нормалізованих змінних;

$S_{Z_Y}^2 = \frac{1}{v}\sum_{i=1}^{N}(Z_{Y_i} - \hat{Z}_{Y_i})^2$ – залишкова дисперсія;

$$S_Z = \left(S_{Z_1Z_1}\ S_{Z_1Z_2}\ S_{Z_1Z_2}\ S_{Z_2Z_2}\ \right),$$

де $S_{Z_qZ_r} = \sum_{i=1}^{N}\left[\left(Z_{q_i} - \bar{Z}_q\right)\left(Z_{r_i} - \bar{Z}_r\right)\right], q, r = 1,2.$

Якщо є точки даних які виходять за інтервал прогнозування то потрібно вилучити їх із вибірки. Цей крок повторюється доти, доки всі викиди не будуть усунуті.

Нижче представлено видалення викидів для нормалізованих даних:

Ітерація 1:

- 72 застосунок: 10,4919 > 4,5508 (TS > $F_{\text{крит}}$).

Ітерація 2:

- 4 застосунок: 11,625 < 11,8622 (Y < Інтервалу прогнозування);
- 52 застосунок: 3,4922 < 4,8316 (Y < Інтервалу прогнозування);
- 61 застосунок: 7 < 7,2958 (Y < Інтервалу прогнозування);
- 64 застосунок: 3,4578 < 4,1315 (Y < Інтервалу прогнозування);

Ітерація 3:

- 11 застосунок: 2,3297 < 2,4484 (Y < Інтервалу прогнозування);
- 41 застосунок: 10,52 < 10,5982 (Y < Інтервалу прогнозування);

- 50 застосунок: 13,9507 < 14,7445 (Y < Інтервалу прогнозування);
- 75 застосунок: 2,0816 < 2,1526 (Y < Інтервалу прогнозування);

Після усунення викидів вибірка налічує 91 застосунок.

2.2 Побудова моделі для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java

Після попередньої обробки даних і видалення аномальних значень було відібрано 91 застосунків для аналізу. Потім ці дані розподілили на навчальну та тестову вибірки: 60% загальної вибірки (54 застосунок) призначили для навчання моделі, а решту 40% (37 застосунків) — для перевірки її роботи. Під час розподілу враховували наявність мінімальних і максимальних значень метрик у навчальній вибірці, а в тестовій залишали значення, близькі до граничних значень, щоб забезпечити репрезентативність і надійність моделі. У таблиці 2.2 представлено вибірку для побудови моделі. У таблиці 2.3 представлено вибірку для тестування моделі

Таблиця 2.2 – Вибірка для побудови моделі

#	URL	RFC	CBO	WMC
1	2	3	4	5
1	https://github.com/jwanderson314/Pizza_Shop	1,1000	0,2000	1,2000
2	https://github.com/JavaGraduationProject/CakeShoppingMallManagementSystem	21,2000	2,7500	16,9000
3	https://github.com/shopizer-ecommerce/shopizer	16,1460	6,3879	11,5182
4	https://github.com/JavaGraduationProject/OnlineShoppingMallManagementSystem	19,4528	1,4906	18,8113
5	https://github.com/chengzhx76/weixin-shop-spring-cloud	10,6238	1,8218	9,4455
6	https://github.com/Advanced-Programming-1401/Internet-Shop	13,3333	2,0000	11,6667
7	https://github.com/sczyh30/vertx-blueprint-microservice	13,0000	2,2000	13,5375
8	https://github.com/UnsafeDodo/gui-shop	5,9259	2,7778	7,1481

Продовження таблиці 2.2

1	2	3	4	5
9	https://github.com/Shubh2-0/Shopzilla-Online-Shopping-Platform	3,3536	2,1381	3,7514
10	https://github.com/JavaGraduationProject/TmallShoppingMallManagementSystem	10,1429	4,5102	8,7959
11	https://github.com/roadbook2/shop	3,6296	2,7407	3,9815
12	https://github.com/shashirajraja/onlinebookstore	8,0769	3,5385	7,3077
13	https://github.com/lilishop/lilishop	6,0912	5,1509	6,8961
14	https://github.com/OOP-Project-ShopingApp/PeaShop	11,1176	1,1765	15,2353
15	https://github.com/jsyGitTest/jingShop	8,7079	2,3258	10,3034
16	https://github.com/WalaaASA/ItesaqShop	3,4224	1,2241	4,2328
17	https://github.com/le0n67/XinFur_Mall	11,9286	2,5000	9,0714
18	https://github.com/saravanan81java/shopping-card-validation	5,5833	0,8333	6,9167
19	https://github.com/deeptesh-rout/dream-shops	3,8000	2,7250	3,4250
20	https://github.com/xvmingyuan/shop	18,2957	2,6348	15,9478
21	https://github.com/BrandanBurgess/ShopTrack	4,7717	1,5870	4,4239
22	https://github.com/codingxin/OnlineSchoolShop	19,9239	2,5635	18,2335
23	https://github.com/josdem/shopping-cart-service	1,4400	1,0400	1,2400
24	https://github.com/JavaGraduationProject/BeautyShoppingMallManagementSystem	16,7500	1,5000	16,5500
25	https://github.com/louisgeek/LouisShopCart	4,6897	1,6552	4,7586
26	https://github.com/fengyongge/shopcar	5,7407	1,5185	5,4074
27	https://github.com/KBTG-Kampus-ClassNest-SE-Java/workshopb2-group-1	2,6250	1,6000	2,9250
28	https://github.com/Yujin051/shopping-mall-team-project	3,9286	2,2143	3,0893
29	https://github.com/chau-nm/ShoppingApp	1,6000	0,4000	1,6000
30	https://github.com/bmvermeer/JavaCoffeeShop	7,9750	1,9250	7,2750
31	https://github.com/alokrai0607/EazyShop	4,1837	2,4898	4,2041
32	https://github.com/MaarkNassef/E-Shop	3,6218	1,2017	2,8824
33	https://github.com/ycshang123/shop_online	2,0659	2,1538	2,1978
34	https://github.com/JohnWCCI/pizza-shop-forClass	7,9333	1,5333	5,4000
35	https://github.com/dasswayam/e-shopping-cart	7,4444	1,9444	5,5556
36	https://github.com/NaeDdoCo/nutritional-shopping-mall-web	13,2830	3,3585	10,0566
37	https://github.com/balllm/shop	3,0000	0,2500	4,2500
38	https://github.com/JavierSantiagoSalazar/shop	2,2000	1,4000	1,8000
39	https://github.com/digantv/shoping-management-mono-repo	6,5333	1,1333	4,6333
40	https://github.com/LeeHSeoK/Shop	2,3636	0,5455	2,0909
41	https://github.com/Manphil/shop	20,5100	2,4700	18,6300
42	https://github.com/akiltipu/secure-shop-devops	5,7778	1,2778	5,3889
43	https://github.com/Joryun/Shop	10,4286	2,6327	8,2245
44	https://github.com/Ravinduchathuranga/Textile-shop	2,9259	0,9259	2,7222

Продовження таблиці 2.2

1	2	3	4	5
45	https://github.com/atguigu01/Shopping	6,4784	2,0784	8,5333
46	https://github.com/myxh/CoolShopping	5,9460	1,8993	7,8345
47	https://github.com/coderzhn/shop	3,0769	2,0962	3,2692
48	https://github.com/michael-thoughtworker/Shop	4,4000	1,0000	4,0000
49	https://github.com/Beksultan2020/Online_Shop_Project_Java	5,7619	2,6667	4,8571
50	https://github.com/longleDevops/Car-Shop-Application	12,8571	2,5714	9,8286
51	https://github.com/leechistest/shop	2,9744	2,3333	3,4872
52	https://github.com/chamithKavinda/Coffee-Shop-POS-JavaEE-Backend	3,8400	2,5200	4,7200
53	https://github.com/kousen/shopping_v3	6,0625	1,1250	5,1875
54	https://github.com/David-Garrancho/CoffeeShopSystem	7,4035	1,4035	8,2105

Таблиця 2.3 – Вибірка для тестування моделі

#	URL	RFC	CBO	WMC
1	2	3	4	5
1	https://github.com/itwill-project/Shopping-Mall	4,7040	1,8880	3,0800
2	https://github.com/Yashmerino/online-shop	4,2639	2,4583	3,6389
3	https://github.com/f-lab-edu/shopping-mall-fashion	4,0361	3,1031	3,0103
4	https://github.com/funtl/spring-cloud-alibaba-my-shop	9,0980	1,0588	9,3333
5	https://github.com/devendrasanghavi404/shoppers-retail	4,8333	0,5667	3,7333
6	https://github.com/stellariumImpl/Yoyoshop	9,5526	3,4474	7,5789
7	https://github.com/JavaGraduationProject/HuoyingAnimationShoppingManagementSystem	10,9524	2,7857	9,5238
8	https://github.com/RojerAlone/shop	12,6491	3,1754	11,6140
9	https://github.com/jairoArh/WorkShopBTS	15,0000	1,2000	11,8000
10	https://github.com/ahmedemad3/shopping-cart-event-sourcing	3,4444	1,4444	3,4444
11	https://github.com/codaholichq/shop	3,8000	0,8000	2,8000
12	https://github.com/sjr7/seckill	10,2000	2,6667	8,5333
13	https://github.com/npolyakova/shop	4,2500	0,6250	3,5000
14	https://github.com/nashtech-garage/yas	4,9055	2,3160	4,0212
15	https://github.com/edison-lhk/instamall	4,2143	2,1429	4,0857
16	https://github.com/Jonny0301/Shopping	3,4468	1,2340	3,5532
17	https://github.com/Vitaly2022/shop	10,4915	3,3220	8,2373
18	https://github.com/phani-sriram/shop	4,1111	1,1667	4,5000
19	https://github.com/lrwinx/shop	1,5833	0,7917	2,1667
20	https://github.com/visalsrimanga/Shopping-Cart-Web-App	1,8148	0,7037	1,5926
21	https://github.com/newbee-ltd/newbee-mall	14,3846	3,2154	14,6923
22	https://github.com/MaJesTySA/miaosha_Shop	11,3485	2,5455	9,4242
23	https://github.com/shidubei/SA59-ShoppingCart	6,8333	2,3333	6,2083

Продовження таблиці 2.3

1	2	3	4	5
24	https://github.com/arpanghosh2414/E-Commerce-Website	2,3000	2,1250	2,7500
25	https://github.com/GGsnake/ShopMall	8,5217	2,2957	9,6870
26	https://github.com/Skirtek/ShoppingAppApi	4,6154	1,3462	4,1538
27	https://github.com/livelysb/coffee-shop	2,4103	2,4103	1,5897
28	https://github.com/The-inhabito/inhabito-ar-mobileApp	3,3684	0,8158	4,0000
29	https://github.com/OlegRyazancev/online-shop	4,0750	2,6393	3,2107
30	https://github.com/JavaGraduationProject/ChinaShoppingManagementSystem	18,7500	2,8438	16,0938
31	https://github.com/khanhduzz/matcha	2,5349	0,7907	2,0465
32	https://github.com/Roshan-Patro/Online-Shopping-Application	4,9057	3,0566	5,6792
33	https://github.com/every821/mobile-shop-android	6,4711	2,0667	5,6533
34	https://github.com/ym1085/springboot-shopping-mall	4,7071	2,6061	4,5859
35	https://github.com/songsimo/pion-shop	3,8000	1,2667	3,2000
36	https://github.com/BogomolovISerg/ShopCarts	5,5789	2,1579	4,6316
37	https://github.com/NHN-YesAladin/yesaladin_shop	6,2473	4,4236	4,0382

Через те, що емпіричні дані не мають нормальний розподіл, для вирішення задачі прогнозування якості на основі кількості класів було вирішено побудувати рівняння нелінійної регресії з використанням нормалізуючих перетворень, зокрема десяткового логарифма. Рівняння лінійної регресії має наступний вигляд:

$$\hat{z}_y = \beta_0 + \beta_1 Z_{x_1} + \beta_2 Z_{x_2}, \quad (2.9),$$

Z_{x_1} – нормалізована змінна (CBO);

Z_{x_2} – нормалізована змінна (WMC);

$\beta_0, \beta_1, \beta_2$ – коефіцієнти регресії.

Оцінка параметрів моделі β_0, β_1, β_2 буде здійснена методом найменших квадратів [24]:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y, \quad (2.10),$$

де $\hat{\beta}$ – оцінка МНК;

X – змінна матричного регресора X;

y – вектор значення змінної відгуку.

За допомогою зворотного перетворення отримаємо модель нелінійної регресії (1.1). Після обчислень були отримані наступні значення коефіцієнтів: $\beta_0 = 0{,}0177$, β_1 = 0,0856, β_2 = 0,9729, ε – гаусівська випадкова величина з нульовим математичним сподіванням та середньоквадратичним відхиленням, оцінка якого дорівнює 0,078.

Таким чином, остаточне рівняння має вигляд:

$$\hat{Y} = 10^{0{,}0177}\, {X_1}^{0{,}0856} {X_2}^{0{,}9729}. \qquad (2.11),$$

Модель функціонує в межах таких значень метрик: RFC [1,1 – 21,2], CBO [0,2 – 6,3879] та WMC [1,2 – 18,8113]. Для нормалізованих даних додатково обчислюється квадрат відстані Махаланобіса, який має бути меншим за критичне значення хі-квадрат розподілу $\chi^2_{0.005,3} = 12{,}8382$ [25].

Якість побудованої нелінійної регресійної моделі оцінюються за допомогою R^2, *MMRE* та *PRED(0,25)*. Вони дозволяють оцінити точність та адекватність моделі для прогнозування. Результати для навчальної та тестової вибірок такі:

- Навчальна вибірка: R^2 = 0,9326, MMRE = 0,1502, PRED(0,25) = 0,7963.
- Тестова вибірка: R^2 = 0,9335, MMRE = 0,1458, PRED(0,25) = 0,7838.

Значення R^2, близьке до 0,9, свідчить про те, що модель ефективно описує змінність метрики RFC, вказуючи на високий рівень відповідності між фактичними та передбаченими значеннями як для навчальної, так і для тестової вибірок. Низьке значення MMRE для обох вибірок демонструє, що середня відносна похибка моделі є низькою, що підтверджує її надійність у прогнозуванні. Значення PRED(0,25) для навчальної та для тестової вибірок вказують на те, що модель забезпечує високий відсоток прогнозів із відносною помилкою не більше 25%, що є прийнятним для практичних застосувань. Ці

результати свідчать про адекватність та стабільність моделі в умовах прогнозування якості Java-застосунків.

Для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, на основі побудованої регресійної моделі пропонується використовувати довірчі та прогнозні інтервали [2]. Такий підхід дозволяє не тільки отримати точкову оцінку метрики RFC, але й визначити діапазон можливих значень з певною ймовірністю, що сприяє підвищенню надійності оцінювання. Прогнозний інтервал для нелінійної регресії, з урахуванням нормалізації даних за допомогою десяткового логарифму, визначається за формулою (2.8).

Довірчий інтервал для нелінійної регресії розраховується аналогічно, але без доданка «1» у фігурних дужках:

$$10^{\left(\hat{Z_Y} \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y} \left\{\frac{1}{N} + \left(z_X^+\right)^T S_Z^{-1} \left(z_X^+\right)\right\}^{\frac{1}{2}}\right)}$$

Для розрахунку інтервалів використовуючи нормалізовані дані для побудови моделі було отримано квантиль розподілу Стьюдента: 2,0076, залишкове стандартне відхилення: 0,078, та обернена коваріаційна матриця: $\begin{pmatrix} 0{,}3269 & -0{,}1535 \\ -0{,}1535 & 0{,}2772 \end{pmatrix}$. У таблиці 2.4 представлено значення Y (RFC), передбачені значення $\hat{Y}$ та розраховані прогнозні та довірчі інтервали для кожного застосунку.

Таблиця 2.4 – Передбачені значення та інтервали

#	Y	$\hat{Y}$	Довірчий нижня	Довірчий верхня	Прогнозний нижня	Прогнозний верхня
1	2	3	4	5	6	7
1	1,1000	1,0837	0,9088	1,2923	0,7255	1,6190
2	21,2000	17,7792	16,2363	19,4688	12,2567	25,7900
3	16,1460	13,1605	11,7656	14,7208	9,0206	19,2004
4	19,4528	18,7244	16,6764	21,0239	12,8196	27,3490
5	10,6238	9,7454	9,1560	10,3728	6,7580	14,0534

Продовження таблиці 2.4

1	2	3	4	5	6	7
6	13,3333	12,0642	11,2262	12,9648	8,3513	17,4279
7	13,0000	14,0567	12,9858	15,2159	9,7161	20,3363
8	5,9259	7,7044	7,2462	8,1916	5,3436	11,1081
9	3,3536	4,0235	3,7564	4,3096	2,7870	5,8086
10	10,1429	9,8266	8,9886	10,7428	6,7770	14,2486
11	3,6296	4,3550	4,0219	4,7158	3,0100	6,3010
12	8,0769	8,0365	7,4557	8,6625	5,5598	11,6164
13	6,0912	7,8440	7,0726	8,6995	5,3896	11,4160
14	11,1176	14,9460	13,3451	16,7390	10,2406	21,8135
15	8,7079	10,8296	10,1572	11,5465	7,5076	15,6214
16	3,4224	4,3140	4,0763	4,5655	2,9943	6,2152
17	11,9286	9,6271	9,0636	10,2255	6,6783	13,8778
18	5,5833	6,7307	6,1580	7,3566	4,6421	9,7589
19	3,8000	3,7598	3,4439	4,1047	2,5938	5,4499
20	18,2957	16,7425	15,3410	18,2720	11,5513	24,2666
21	4,7717	4,6046	4,3683	4,8537	3,1980	6,6299
22	19,9239	19,0278	17,2829	20,9489	13,0997	27,6386
23	1,4400	1,2885	1,1415	1,4544	0,8807	1,8851
24	16,7500	16,5398	14,8746	18,3915	11,3564	24,0892
25	4,6897	4,9610	4,7138	5,2212	3,4463	7,1415
26	5,7407	5,5766	5,3041	5,8631	3,8744	8,0265
27	2,6250	3,0810	2,8667	3,3113	2,1328	4,4509
28	3,9286	3,3409	3,0780	3,6262	2,3079	4,8363
29	1,6000	1,5214	1,3373	1,7309	1,0372	2,2316
30	7,9750	7,5951	7,2119	7,9987	5,2756	10,9344
31	4,1837	4,5541	4,2425	4,8887	3,1532	6,5774
32	3,6218	2,9637	2,7645	3,1773	2,0526	4,2794
33	2,0659	2,3932	2,1577	2,6544	1,6444	3,4831
34	7,9333	5,5738	5,3022	5,8593	3,8726	8,0223
35	7,4444	5,8476	5,5590	6,1513	4,0625	8,4173
36	13,2830	10,9152	10,1511	11,7368	7,5551	15,7697
37	3,0000	3,7803	3,1922	4,4767	2,5381	5,6303
38	2,2000	1,8993	1,7174	2,1005	1,3060	2,7621
39	6,5333	4,6797	4,4120	4,9636	3,2470	6,7444
40	2,3636	2,0269	1,8228	2,2540	1,3917	2,9522
41	20,5100	19,3685	17,5579	21,3659	13,3275	28,1479
42	5,7778	5,4765	5,1836	5,7858	3,8022	7,8879
43	10,4286	8,7903	8,2812	9,3308	6,0985	12,6703
44	2,9259	2,7416	2,5399	2,9593	1,8961	3,9640
45	6,4784	8,9287	8,4390	9,4469	6,1977	12,8632
46	5,9460	8,1534	7,7258	8,6047	5,6617	11,7417
47	3,0769	3,5135	3,2575	3,7897	2,4304	5,0793

Продовження таблиці 2.4

1	2	3	4	5	6	7
48	4,4000	4,0129	3,7606	4,2821	2,7816	5,7894
49	5,7619	5,2719	4,9234	5,6450	3,6519	7,6104
50	12,8571	10,4330	9,7927	11,1153	7,2337	15,0474
51	2,9744	3,7757	3,4945	4,0795	2,6108	5,4603
52	3,8400	5,1022	4,7751	5,4517	3,5358	7,3626
53	6,0625	5,2200	4,9150	5,5439	3,6212	7,5248

Таким чином, успішно було побудовано таблицю довірчих і прогнозних інтервалів яка буде використовуватися у наступному підрозділі для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java.

2.3 Оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java

Для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, на основі довірчих і прогнозних інтервалів у Таблиці 2.4 було встановлено чіткі критерії.

Застосунки зі середнім рівнем якості визначаються такими, у яких фактичне значення метрики RFC знаходиться в межах довірчого інтервалу, тобто не менше його нижньої межі та не перевищує верхню. Це вказує на типову складність коду для відповідної комбінації метрик CBO та WMC.

До високоякісних застосунків належать ті, у яких фактичне значення RFC не менше нижньої межі прогнозного інтервалу, але менше нижньої межі довірчого інтервалу. Такі результати свідчать про те, що складність коду є меншою, ніж очікувалося.

Застосунки з низькою якістю характеризуються значенням RFC, що перевищує верхню межу довірчого інтервалу. Це вказує на підвищену складність

коду, яка може негативно впливати на підтримку та розширюваність програмного забезпечення.

Цей підхід забезпечує врахування статистичної невизначеності, дозволяє виявляти застосунки з нетиповою складністю коду та приймати обґрунтовані рішення щодо необхідності рефакторингу.

Аналіз вибірки з 91 застосунків без викидів показав, що 61,5% з них мають високий рівень якості, 35,2% — середній, і лише 1,1% потрапляють до категорії низької якості.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ МАТЕМАТИЧНОЇ МОДЕЛІ

3.1 Постановка задачі та вимоги до програмного забезпечення

Програмне забезпечення для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java отримало назву «JavaMetricsShop». Його основною метою є забезпечення інструментарію для аналізу якості програмних проектів за метриками RFC, CBO та WMC, а також для побудови математичних моделей, що дозволяють прогнозувати якість застосунків. Технічне завдання на розробку програмного забезпечення «JavaMetricsShop» наведено в Додатку А. Програмне забезпечення повинно включати такі функціональні можливості:

- Збір метрик: автоматизоване завантаження метрик RFC, CBO, WMC із вихідного коду Java-застосунків або з файлів у форматі CSV.
- Попередня обробка даних**:** нормалізація метрик за допомогою логарифмічного перетворення та підготовка даних для моделювання.
- Моделювання: побудова як лінійної, так і нелінійної регресійної моделі на основі зібраних даних.
- Прогнозування якості: визначення рівня якості застосунків на основі моделі. Програма повинна класифікувати якість за трьома рівнями (висока, середня, низька) відповідно до розташування значення RFC щодо довірчого та прогнозного інтервалів.

Вхідні дані:

- Таблиця метрик у форматі CSV, які містять значення RFC, CBO та WMC для окремих проектів.
- Метрики RFC, CBO, WMC для прогнозування якості.

Вихідні дані:

- Лінійна регресійна модель (2.9): базова математична модель залежності метрик.
- Нелінійна регресійна модель (2.11): удосконалена модель для прогнозування значень RFC з урахуванням нелінійних залежностей.
- Якість нелінійної моделі: оцінка точності моделі за показниками MMRE та PRED(0.25).
- Прогнози якості: результати аналізу, що класифікують якість проекту за рівнями (високий, середній, низький).

Прогнозування якості здійснюється шляхом порівняння значень **RFC** із межами довірчого та прогнозного інтервалів, визначених за допомогою нелінійної моделі. Програмне забезпечення **«JavaMetricsShop»** спрямоване на інтеграцію з іншими інструментами, надаючи можливість для автоматизації аналізу та моніторингу якості програмних проектів.

3.2 Ескізний проект програмного забезпечення

На основі аналізу вимог до програмного забезпечення було розроблено ескізний проект програмного забезпечення. Прототипи вікон програми приведено на рисунках 3.1 – 3.3.

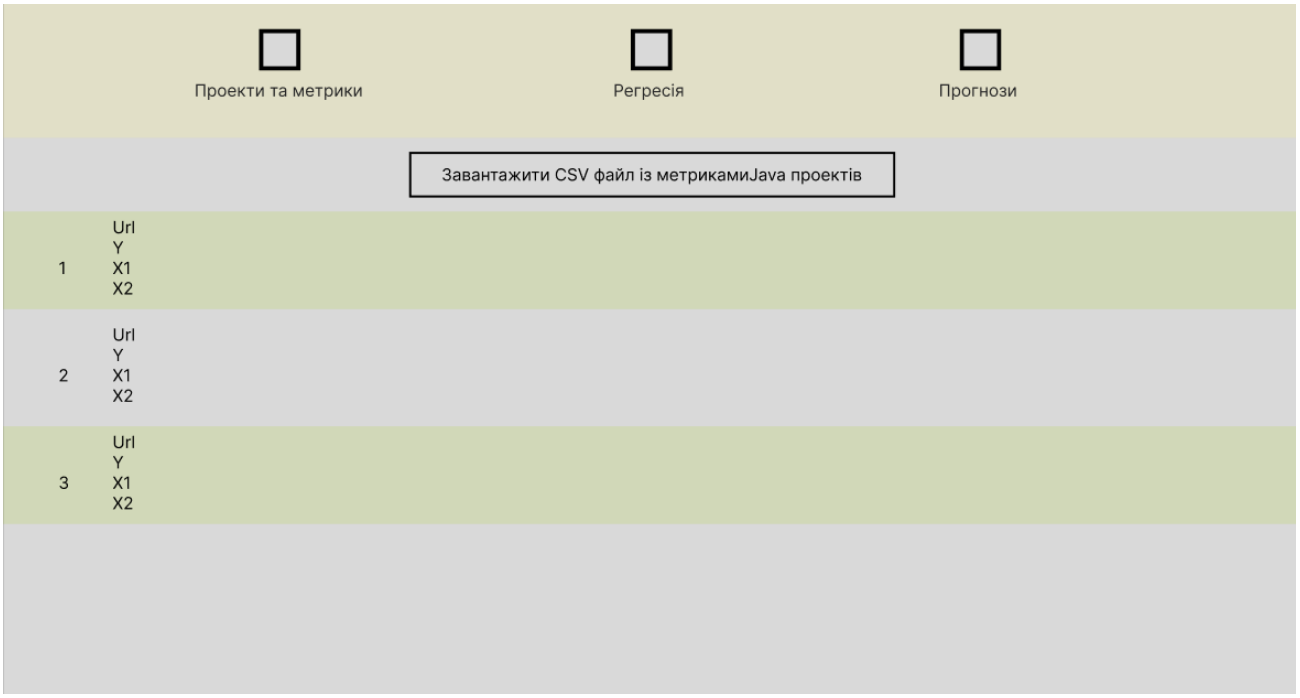

Рисунок 3.1 – Прототип головного экрана програмного забезпечення для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java «JavaMetricsShop»

Головне вікно має меню зверху із опціями «Завантажити файл із метриками» та пунктами «Проекти», «Регресія» та «Прогнози». Основна область містить завантажені програмні проекти. Кожен проекти містить нумерацію, URL та метрики:

- Y (RFC: Response for a Class).
- X1 (CBO: Coupling between Object Classes).
- X2 (WMC: Weighted Methods per Class).

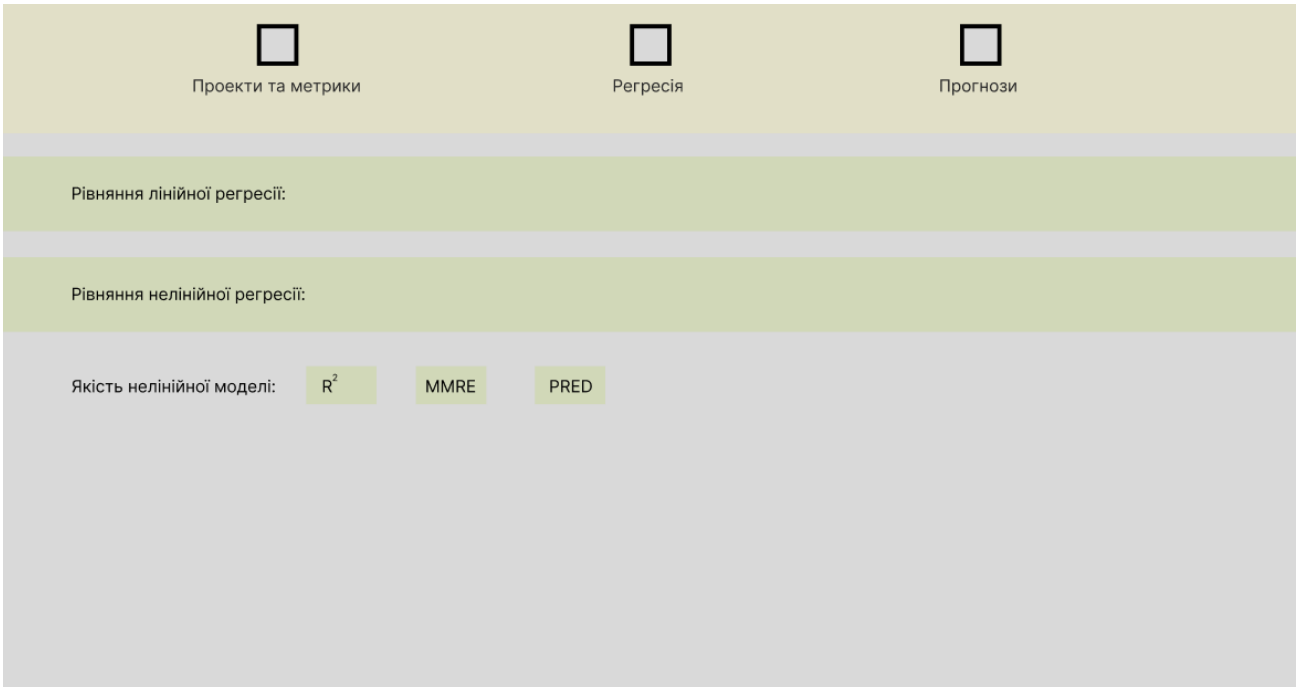

Рисунок 3.2 – Прототип екрана «Регресія» програмного забезпечення для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java «JavaMetricsShop»

Вкладка «Регресійний аналіз» відображає рівняння лінійної та нелінійної регресійної моделі. Нижче показано метрики якості нелінійної моделі: коефіцієнт детермінації R^2, середня відносна похибка *MMRE* та відсоток передбачень у допустимому діапазоні *PRED(0.25)*.

Рисунок 3.3 – Прототип екрана «Прогнози» програмного забезпечення для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java «JavaMetricsShop»

Є поля для введення RFC, CBO, WMC як незалежних змінних для передбачення якості на основі побудованої моделі. Ця вкладка дозволяє виконувати прогнози, для оцінки якості застосунків для онлайн-покупок з відкритим кодом на Java.

3.3 Технічний проект програмного забезпечення

Після ескізного проекту було розроблено технічний проект програмного забезпечення для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java. Цей проект включав деталізацію компонентів системи, їх взаємозв'язків та ключових функціональних можливостей.

На рисунку 3.4 зображено технічний проект програмного забезпечення для аналізу розміру програмних проектів інтернет магазинів, реалізованих за допомогою мови програмування Java представлений у вигляді діаграми класів.

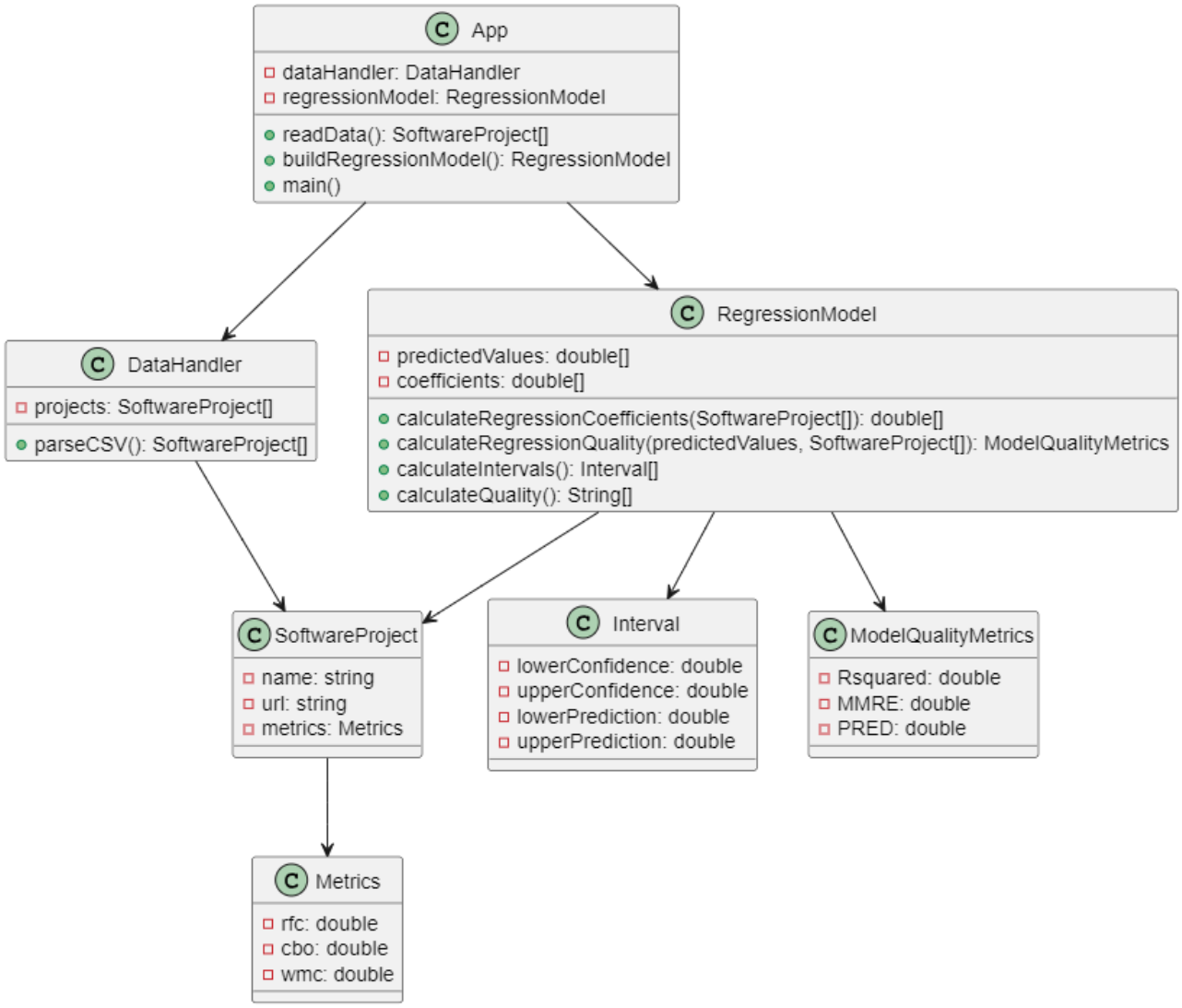

Рисунок 3.4 – Діаграма класів програмного забезпечення для аналізу розміру програмних проектів інтернет магазинів, реалізованих за допомогою мови програмування Java

На діаграмі класів, зображено основні класи та їхні відносини в системі. Центральним класом є «App», який координує роботу інших компонентів, таких

як «DataHandler» та «RegressionModel». Клас «DataHandler» відповідає за обробку вхідних даних. Клас «RegressionModel» реалізує функціональність для побудови регресійної моделі, обчислення її коефіцієнтів (2.10), прогнозування значень та оцінки якості моделі за допомогою різних метрик, таких як R^2, MMRE та PRED. Він також розраховує інтервали та прогнозує якість. Класс Intervals містить інтервали. Клас «SoftwareProject» представляє окремий програмний проект, що містить його назву, URL та відповідні метрики коду, які зберігаються в класі «Metrics». Клас «ModelQualityMetrics» використовується для зберігання метрик якості регресійної моделі, таких як R^2, MMRE та PRED. Ця діаграма класів слугує основою для подальшої реалізації системи та допомагає забезпечити коректну взаємодію між різними компонентами під час виконання ключових функцій, таких як імпорт даних, побудова моделі, оцінка якості та оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java.

3.4 Динамічна модель

Наступний етап розробки технічного проекту програмного забезпечення, полягає в побудові динамічної моделі для розроблюваного програмного забезпечення. Використовуючи раніше розроблені моделі, зокрема модель варіантів використання та діаграму класів, будується динамічна модель. На рисунках 3.5 – 3.9 представлені діаграми послідовності для основних варіантів використання, які втілюють цю модель.

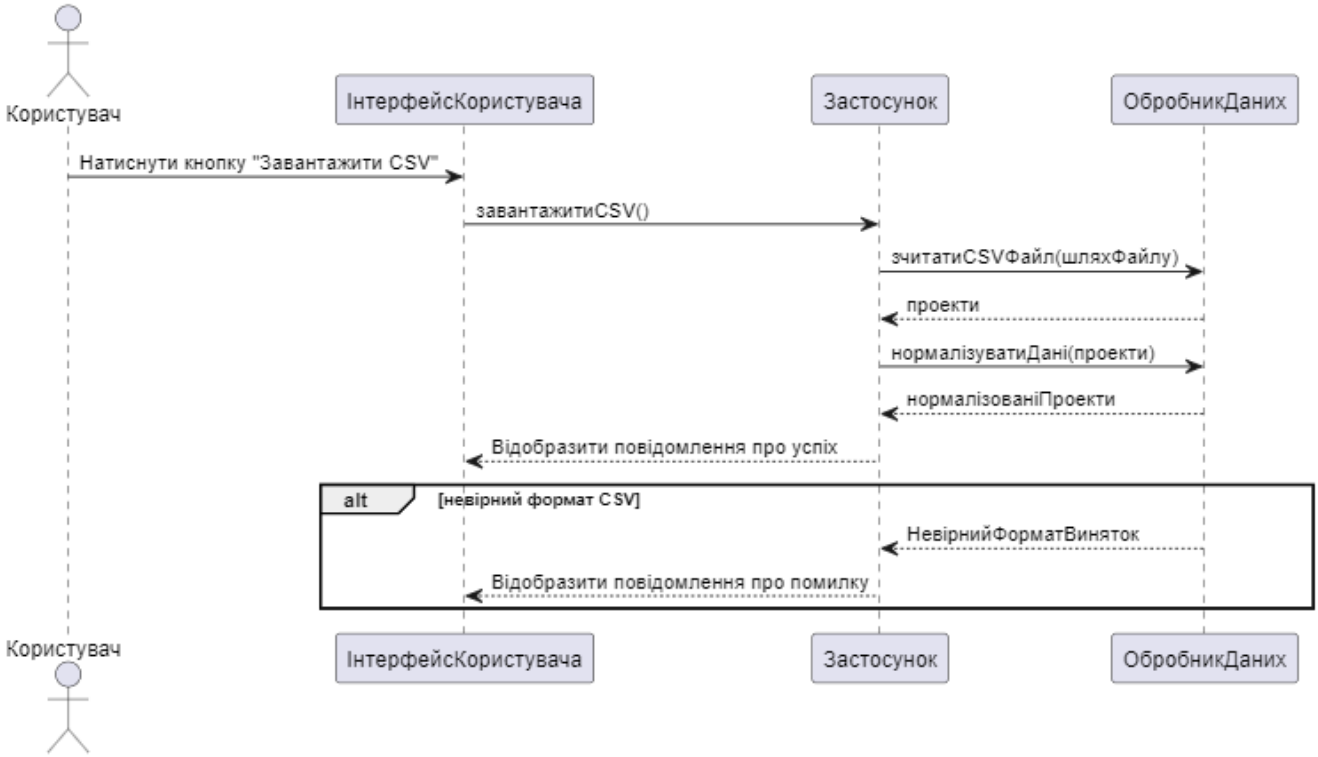

Рисунок 3.5 – Діаграма послідовності для прецеденту «Завантажити CSV»

Дана діаграма послідовності візуалізує процес завантаження та обробки CSV файлу в застосунку. Користувач ініціює завантаження CSV файлу через інтерфейс користувача, після чого застосунок зчитує файл, нормалізує дані та відображає результат операції в інтерфейсі. У випадку невірного формату CSV файлу, система відображає відповідне повідомлення про помилку.

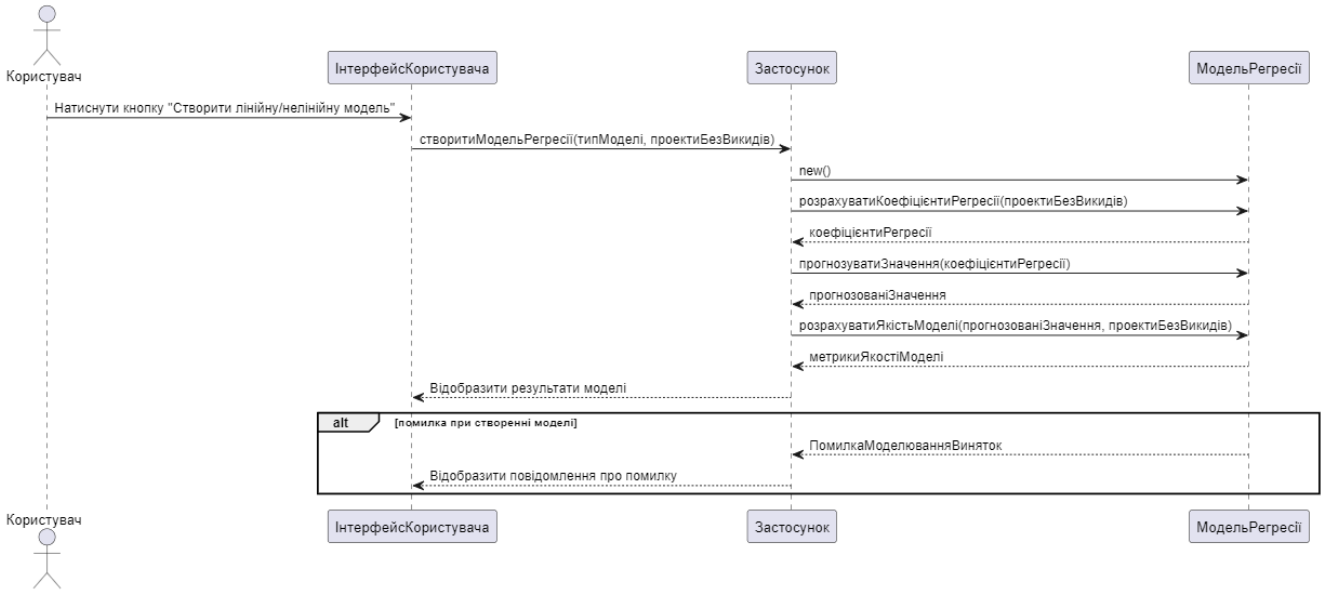

Рисунок 3.6 – Діаграма послідовності для прецеденту «Побудувати Лінійну та Нелінійну Модель»

Ця діаграма послідовності описує процес створення лінійної або нелінійної моделі регресії на основі даних проектів у застосунку. Після ініціації користувачем, застосунок створює об'єкт Модель Регресії відповідного типу, розраховує коефіцієнти регресії, прогнозує значення, оцінює якість моделі та відображає результати в інтерфейсі користувача. У випадку виникнення помилки під час створення моделі, відображається відповідне повідомлення про помилку.

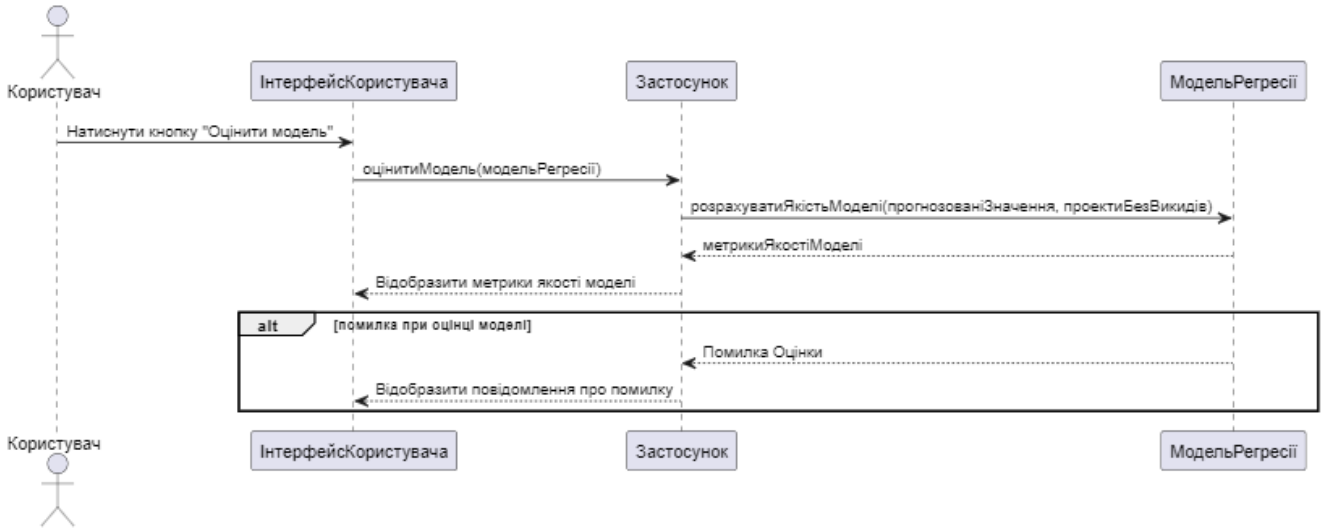

Рисунок 3.7 – Діаграма послідовності для прецеденту «Оцінка якості моделі»

Ця діаграма послідовності ілюструє процес оцінки якості вже створеної моделі регресії на основі даних проектів. Користувач ініціює оцінку моделі через інтерфейс користувача, після чого застосунок викликає метод розрахувати якість моделі в об'єкті Модель Регресії, передаючи прогнозовані значення та дані проектів. Модель обчислює відповідні метрики якості та повертає їх до застосунку, який відображає ці метрики в інтерфейсі користувача. У разі виникнення помилки під час оцінки моделі, система відображає відповідне повідомлення про помилку.

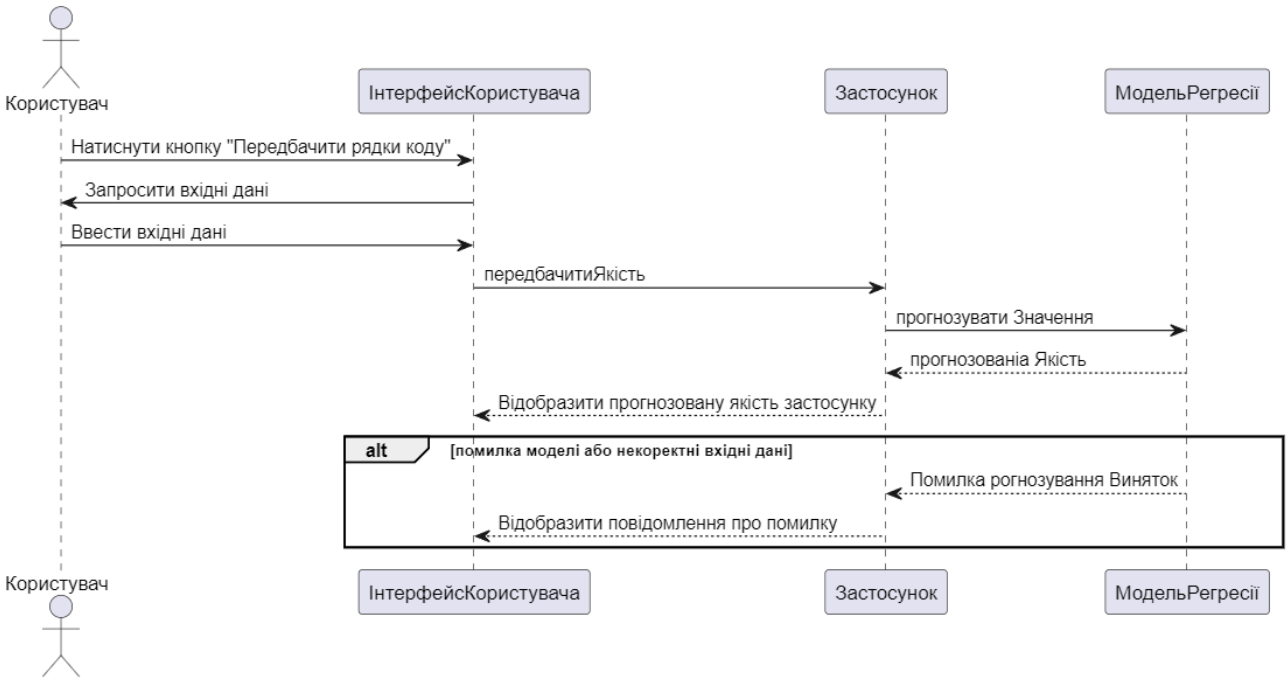

Рисунок 3.8 – Діаграма послідовності для прецеденту «Прогнозувати якість»

Ця діаграма послідовності описує процес прогнозування якості на основі введених користувачем вхідних даних метрик RFC, CBO та WMC. Після ініціації користувачем, інтерфейс користувача запитує у користувача необхідні вхідні дані. Далі застосунок передає ці вхідні дані та модель регресії до методу передбачити якість, який, у свою чергу, викликає метод прогнозувати Значення в об'єкті Модель Регресії, щоб отримати прогнозовану якість. Результат відображається в інтерфейсі користувача. У випадку некоректних вхідних даних або помилки моделі система відображає відповідне повідомлення про помилку.

3.5 Реалізація програмного забезпечення

Для розробки програмного забезпечення «JavaShopMetrics» було обрано фреймворк Flutter з використанням мови програмування Dart та середовища розробки Visual Studio Code [26].

Flutter є сучасним фреймворком для розробки кросплатформених застосунків, що дозволяє створювати високопродуктивні та візуально привабливі додатки для різних платформ, таких як iOS, Android, Web та настільні платформи, використовуючи єдину кодову базу.

Мова програмування Dart, розроблена компанією Google, забезпечує високу продуктивність виконання коду та легкість у навчанні завдяки зрозумілому синтаксису, схожому на інші сучасні мови програмування, такі як JavaScript та C#.

Visual Studio Code було обрано як основне середовище розробки завдяки його популярності, великій кількості розширень та інтеграції з Flutter. Ця IDE (Integrated Development Environment) надає розробникам зручний інтерфейс для написання коду, відлагодження та тестування застосунків, а також підтримує численні плагіни, які полегшують процес розробки.

Таким чином, комбінація Flutter, Dart та Visual Studio Code забезпечує оптимальні умови для швидкої та ефективної розробки програмного забезпечення «JavaMetricsShop», що дозволяє розробникам зосередитися на створенні функціоналу та інтерфейсу користувача, не витрачаючи зайвий час на технічні нюанси, пов'язані з підтримкою різних платформ.

Для ініціалізації проекту Flutter та налаштування необхідних залежностей будуть виконані наступні кроки:

1. Flutter SDK буде завантажений та встановлений, щоб мати можливість використовувати Flutter у проекті.

2. Новий проект Flutter буде створений у Visual Studio Code. Це можна буде зробити за допомогою команди `flutter create <назва_проекту>` у вбудованому терміналі або через командний рядок операційної системи.

3. Файл `pubspec.yaml` у проекті буде відкритий, щоб додати необхідні пакети зі списку залежностей, такі як `csv`, `flutter_math_fork`, `go_router`, `json_annotation`, `logger`, `ml_linalg`, `provider`.

4. Код програмного забезпечення буде написаний за допомогою мови програмування Dart. Повинно створити різні класи, функції та компоненти, щоб відповідати потребам програмного забезпечення.

5. Після завершення розробки додатку, його буде можна скомпілювати для виконання у веб-браузері Chrome. Для цього використовуватиметься команда `flutter build web` у терміналі, після чого додаток буде готовий до запуску у веб-браузері.

Ці кроки дозволять розпочати розробку програмного забезпечення «JavaShopMetrics» для аналізу розміру програмних проектів інтернет-магазинів, реалізованих за допомогою мови програмування Java.

У результаті розробки було створено функціональне програмне забезпечення «JavaShopMetrics» для аналізу розміру програмних проектів інтернет-магазинів, реалізованих за допомогою мови програмування Java. Код програмного забезпечення «JavaShopMetrics» наведено в Додатку Б. На рисунку 3.9 представлено головну сторінку застосунку «JavaShopMetrics».

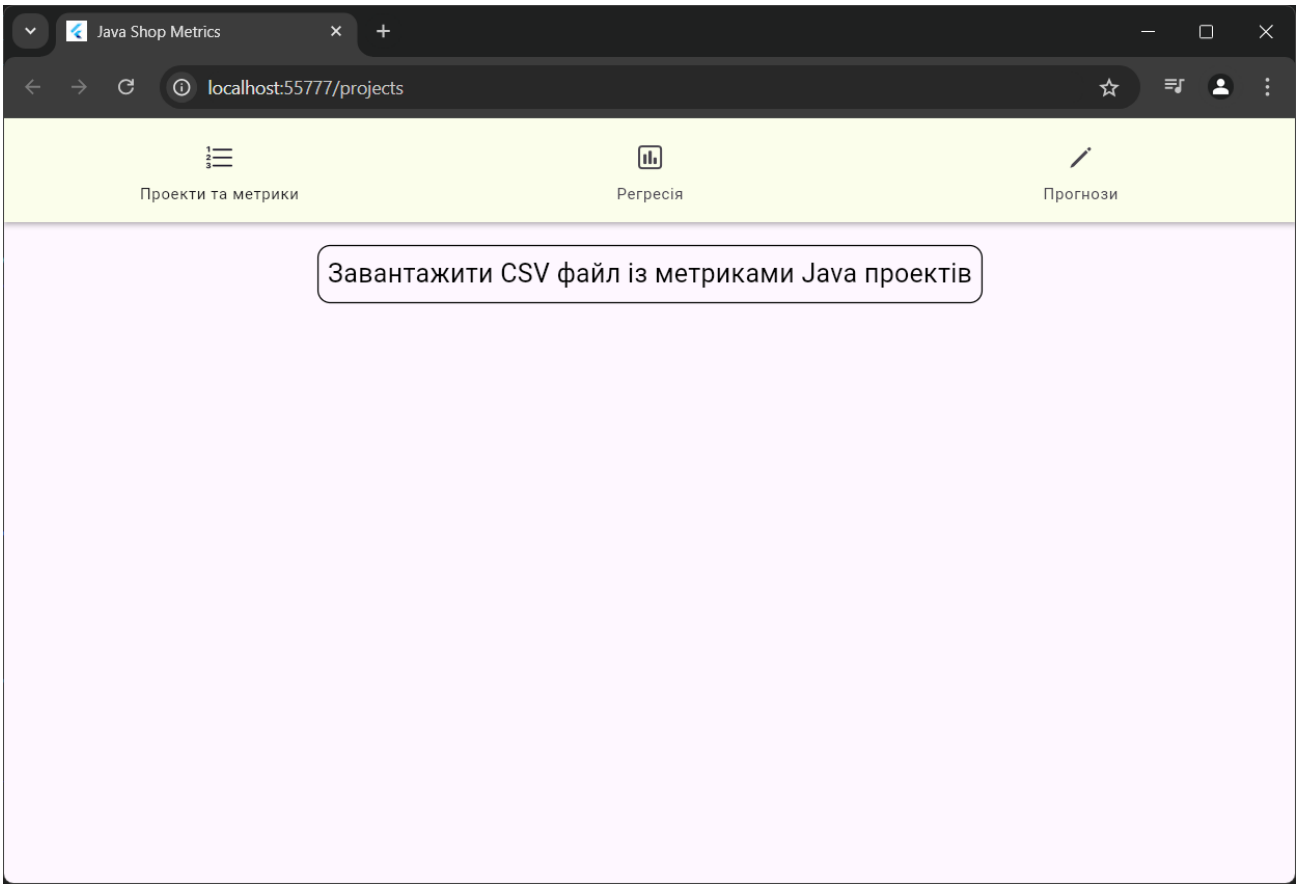

Рисунок 3.9 – Головна сторінка застосунку «JavaShopMetrics»

На головній сторінці можна побачити кнопку із завантаженням даних метрик Java із .csv файлу та компоненти меню. Після того як данні були завантаженні, їх обробляє застосунок та відображає у вкладці «Проекти та метрики». На рисунку 3.10 представлено вкладку «Проекти та метрики» із завантаженими даними.

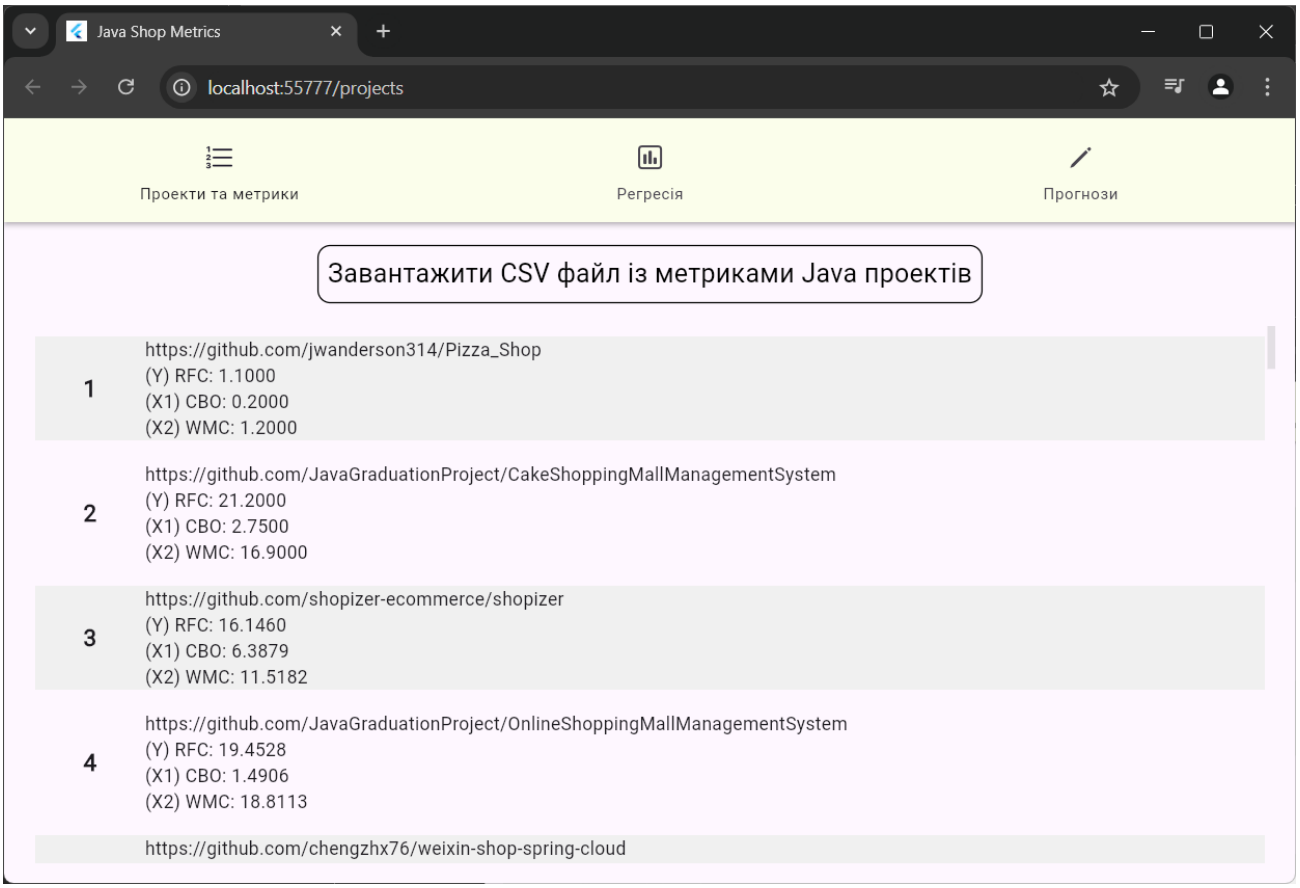

Рисунок 3.10 – Вкладка «Проекти та метрики» застосунку «JavaShopMetrics»

У вкладці «Проекти та метрики» можна здійснити перегляд завантажених проектів. На рисунку 3.11 представлена вкладка «Регресійний аналіз».

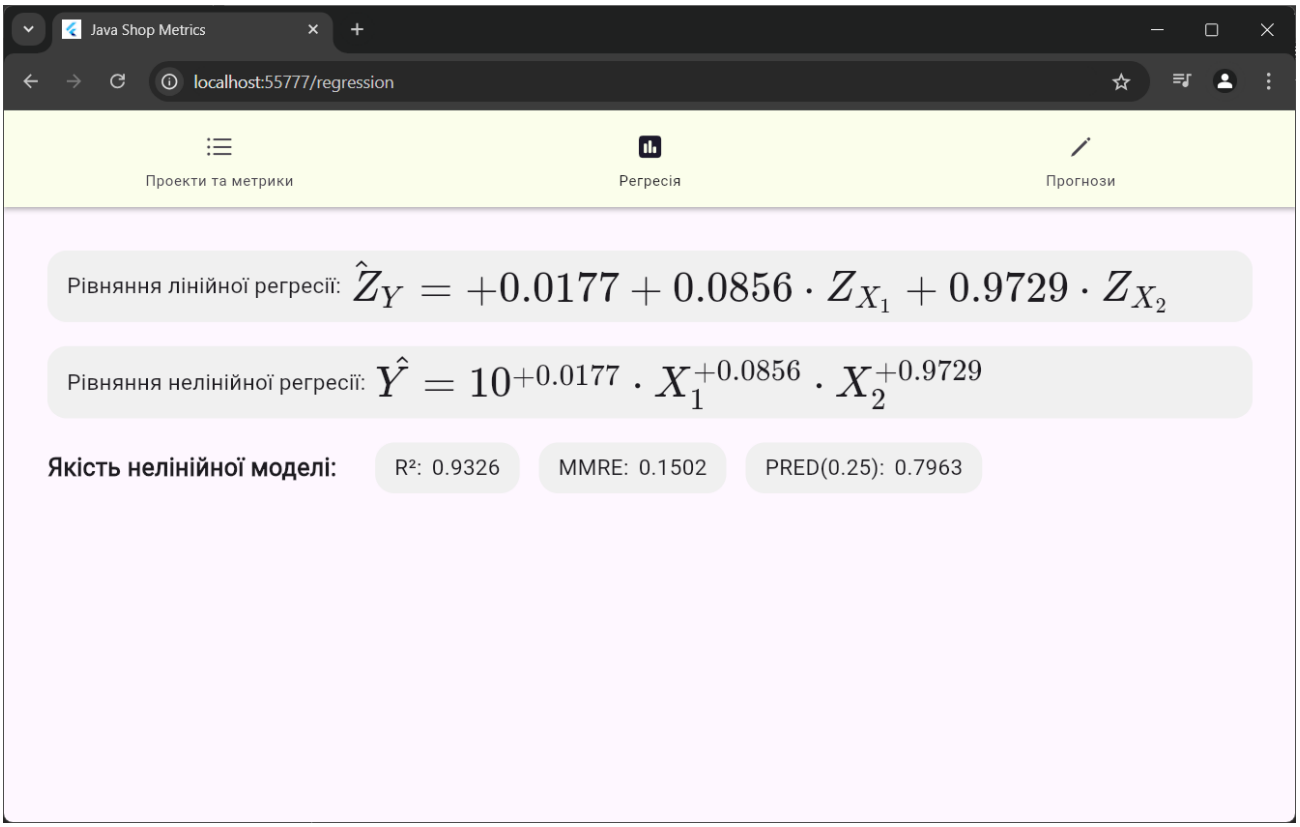

Рисунок 3.11 – Вкладка «Регресія» застосунку «JavaShopMetrics»

У вкладці «Регресійний аналіз» здійснюється перегляд побудованих лінійної та нелінійної регресійних моделей, її коефіцієнтів та статистик якості нелінійної моделі. На рисунку 3.12 представлено вкладку «Прогнози» застосунку «JavaShopMetrics». Також представлені допустимі значення для метрик RFC [1,1 – 21,2], CBO [0,2 – 6,3879] та WMC [1,2 – 18,8113]. Можна побачити що при введені недопустимих значень застосунок видає помилку про це, а саме «Введено недопустимі значення».

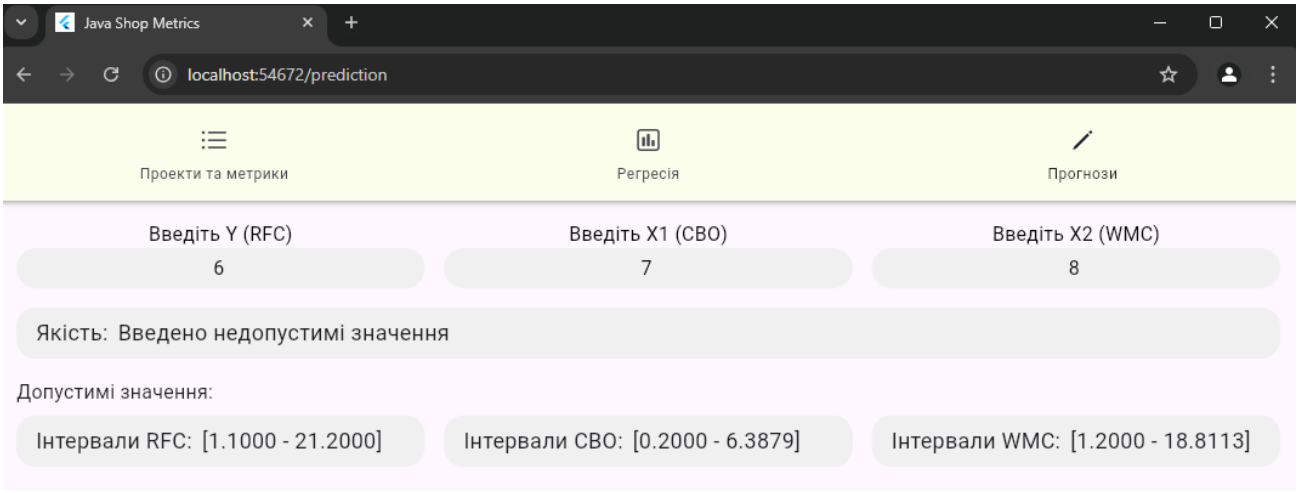

Рисунок 3.12 – Вкладка «Прогнози» застосунку «JavaShopMetrics»

На рисунку 3.13 представлено вкладку «Прогнози» застосунку «JavaShopMetrics».

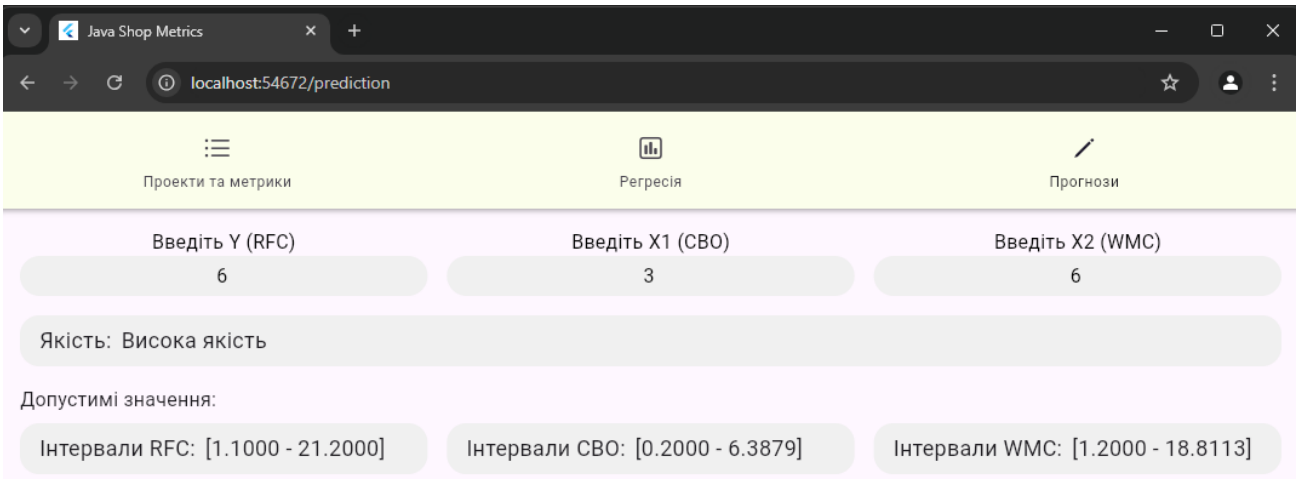

Рисунок 3.13 – Вкладка «Прогнози» застосунку «JavaShopMetrics»

Можна побачити поля із метриками RFC = 6, CBO = 3, WMC = 9, передбачена якість: висока. Опис та інструкція користувача для програмного забезпечення «JavaShopMetrics» зазначені у Додатку В та у Додатку Г.

3.6 Випробування програмного забезпечення

Під час випробування програмного забезпечення «JavaShopMetrics» для оцінювання розміру застосунків проектів інтернет-магазинів, реалізованих за допомогою мови програмування Java, були проведені наступні тести для перевірки вимог до функціональних характеристик:

- Завантаження даних метрик проектів з CSV файлу;
- Побудова лінійної та нелінійної регресійної моделі;
- Здійснення прогнозів якості за метриками RFC, CBO, WMC.

У таблиці 3.1 представлено випробування програмного забезпечення.

Таблиця 3.1 – Випробування програмного забезпечення

№	Дія	Очікуваний результат	Результат
1	2	3	4
1	Завантаження даних з CSV файлу	Система повинна завантажити та зберегти дані для подальшого використання.	Успішно
2	Побудова лінійної та нелінійної регресійної моделі	Система повинна побудувати регресійні моделі та обчислити їхні коефіцієнти.	Успішно
3	Здійснення прогнозів якості за метриками RFC, CBO, WMC	Система повинна здійснити прогнози якості та відобразити результати.	Успішно

Під час випробування програмного забезпечення «JavaShopMetrics» для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java, були проведені тести для перевірки вимог до функціональних характеристик. Зокрема, перевірялися завантаження даних з CSV файлу, побудова лінійної та нелінійної регресійної моделі, а також здійснення прогнозів за метриками RFC, CBO та WMC.

Під час випробування система успішно виконувала завдання з обробки та аналізу даних, що підтверджувалося очікуваними та фактичними результатами, зазначеними в таблиці. Завантаження даних з CSV файлу відбувалося без помилок, регресійні моделі будувалися без помилок, а прогнози якості за метриками RFC, CBO, WMC виконивалися. Усі тестові випадки продемонстрували, що «JavaShopMetrics» був надійним інструментом для аналізу якості застосунків, що підтверджувало його відповідність функціональним вимогам та здатність ефективно підтримувати процес розробки програмного забезпечення для інтернет-магазинів. Програма і методика випробувань програмного забезпечення у повному обсязі зазначені у Додатку Д.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЗАСТОСУНКІВ ДЛЯ ОНЛАЙН-ПОКУПОК З ВІДКРИТИМ КОДОМ НА JAVA

Розробка програмного забезпечення для оцінювання якості застосунків із відкритим кодом є важливою для забезпечення ефективної роботи програмних продуктів, особливо в контексті онлайн-покупок. Основним завданням є скорочення витрат на ручну перевірку якості, підвищення надійності та конкурентоспроможності продукту, а також оптимізація трудових ресурсів завдяки автоматизації процесу аналізу метрик якості.

Економічна ефективність розраховується шляхом аналізу витрат на розробку програмного забезпечення, допоміжні матеріали та вигоди, отримані внаслідок його впровадження. Для цього враховуються прямі витрати, такі як заробітна плата розробника, вартість матеріалів, а також непрямі вигоди, як-от скорочення часу на аналіз і підвищення якості програмного забезпечення.

У процесі економічного аналізу особливу увагу приділяють порівнянню витрат на традиційні методи тестування та впровадження розробленого програмного забезпечення. Автоматизація процесів тестування дозволяє значно зменшити трудомісткість перевірки якості, скоротити терміни розробки та мінімізувати ризики виникнення людських помилок. Крім того, використання інструментів з відкритим вихідним кодом надає додаткові переваги у вигляді можливості налаштування та адаптації програмного забезпечення під специфічні вимоги конкретного проекту без додаткових витрат на придбання ліцензій.

Розрахунок витрат на допоміжні матеріали для розробки програмного забезпечення наведено в таблиці 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

№	Найменування матеріалу	Кількість	Вартість одиниці, грн	Загальна вартість, грн
1	Папір для друку (500 аркушів)	2 пачки	150	300
2	Картридж для принтера	1 шт.	1200	1200
3	Канцелярські товари (ручки, блокноти)	3 комплекти	200	600
4	Електроенергія (розрахунок за місяць)	1 місяць	500	500

Разом: 2600 грн. Розрахунок витрат на розробку програмного забезпечення для одного розробника наведено в таблиці 4.2.

Таблиця 4.2 – Витрати на розробку програмного забезпечення

№	Стаття витрат	Кількість годин	Ставка, грн/год	Загальна вартість, грн
1	Аналіз вимог та проектування	40	300	12000
2	Програмування та відлагодження	100	300	30000
3	Тестування	30	300	9000
4	Документування	20	300	6000

Разом: 57000 грн. Передбачається продаж програмного забезпечення на основі ліцензійної моделі. Очікувано, що за перший рік буде продано 50 ліцензій за ціною 2500 грн кожна. У наступні роки кількість ліцензій зросте на 20% щорічно, а ціна залишиться незмінною.

Розрахунок ROI (Return on investment) [27]. ROI це грошова вартість інвестицій у порівнянні з їхньою вартістю. ROI визначається за формулою:

$$ROI = \frac{\text{Чистий прибуток}}{\text{Загальні інвестиції}} \times 100\%. \quad (4.1)$$

Дохід за 1 рік: $50 \times 2500 = 125000$ грн.

Витрати: 57000 грн (розробка) $+ 2600$ грн (матеріали) $= 59600$ грн.

Чистий прибуток: $125000 - 59600 = 65400$ грн.

$$ROI = \frac{65400}{59600} \times 100\% = 109{,}6\%.$$

Розрахунок NPV (Net Present Value) [28]. NPV враховує поточну вартість майбутніх грошових потоків, дисконтованих на певну ставку, і розраховується за формулою:

$$NPV = \sum_{t=1}^{n} \frac{CF_t}{(1+r)^t} - C_0, \quad (4.2)$$

де CF_t – грошовий потік у році t, r – дисконтна ставка, C_0 – початкові інвестиції.

Дисконтна ставка (r) = 10%.

Грошові потоки (CF_t):

– Рік 1: 125000,

– Рік 2: $125000 \times 1.2 = 150000$,

– Рік 3: $150000 \times 1.2 = 180000$.

Початкові інвестиції (C_0) = 59600.

$$NPV = \frac{125000}{(1+0.1)^1} + \frac{150000}{(1+0.1)^2} + \frac{180000}{(1+0.1)^3} - 59600.$$

$$NPV = \frac{125000}{1.1} + \frac{150000}{1.21} + \frac{180000}{1.331} - 59600.$$

$NPV \approx 113636.36 + 123966.94 + 135235.42 - 59600 = 313238.72$ грн.

ROI = 109,6% показує, що інвестиція повністю окупиться та забезпечить прибуток. NPV = 313238,72 грн свідчить про високу економічну доцільність розробки програмного забезпечення. Впровадження даного програмного забезпечення не тільки знижує витрати на оцінку якості застосунків, але й створює значні можливості для отримання прибутку завдяки ліцензійній моделі продажів.

5 ОХОРОНА ПРАЦІ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

За допомогою охорони праці вирішується наступні завдання:

– інженерно-технічне – направлено на зменшення ризиків при роботі, за допомогою впровадження нових технологій та заміни матеріалів на менш небезпечні;

– соціальне – направлено на відшкодування шкоди, яка була завдана під час трудового процесу, внаслідок нещасних випадків, та захисту прав працівника.

Усі права працівника на безпечні умови праці захищаються законом України “Про охорону праці” [29].

Охорона праці є важливим аспектом будь-якого робочого середовища і забезпечує захист працівників від шкідливих і небезпечних факторів, пов'язаних з їхньою трудовою діяльністю. Вона базується на системі правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів.

Інженерно-технічне завдання охорони праці спрямоване на запобігання та мінімізацію ризиків для працівників. Для цього впроваджуються нові технології, застосовуються безпечні матеріали, розробляються ергономічні робочі місця, встановлюються системи вентиляції та освітлення, проводяться навчання та тренінги з питань безпеки та здоров'я.

Соціальне завдання охорони праці полягає у відшкодуванні шкоди, яка була завдана працівникам у результаті нещасних випадків або професійних

захворювань. Це включає медичне обслуговування, реабілітацію, компенсаційні виплати, страхування, соціальну підтримку та захист прав працівників.

Охорона праці регулюється законом України "Про охорону праці", який визначає права та обов'язки працівників і роботодавців у сфері охорони праці. Закон встановлює вимоги до організації робочого процесу, проведення оцінки ризиків, забезпечення безпеки та здоров'я працівників, надання медичної допомоги, розслідування нещасних випадків тощо.

Важливим елементом охорони праці є система нагляду та контролю, що забезпечує виконання вимог охорони праці і запобігає порушенням. Державні органи, які відповідають за охорону праці, здійснюють перевірки підприємств, надають консультації та рекомендації з питань безпеки та здоров'я працівників.

Отже, охорона праці є системою заходів, спрямованих на забезпечення безпеки, здоров'я та працездатності працівників під час трудової діяльності. Вона включає інженерно-технічні та соціальні заходи, регулюється законодавством та передбачає нагляд і контроль з боку відповідних органів.

5.1 Аналіз шкідливих та небезпечних факторів під час роботи з мобільними пристроями

Існують небезпечні та шкідливі фактори роботи з мобільним телефоном які потрібно знати. До головних можна віднести роботу з самим мобільним телефоном та робоче місце працівника [30].

Нижче наведено деякі з них:

1) Вплив на зір: Використання мобільних пристроїв може призводити до напруження очей, сухості, подразнення та втоми. Довготривале спрямоване уваги на екран може спричинити втому очей, головний біль та навіть проблеми зі зіром.

Рекомендується регулярно робити перерви, виконувати вправи для очей та налаштовувати яскравість та контрастність екрану на комфортний рівень.

2) Постійний стрес: Спостереження за мобільними пристроями може призводити до постійного стресу, особливо в ситуаціях, коли потрібно бути постійно на зв'язку або доступним. Це може впливати на психічне здоров'я та загальний стан. Важливо встановити режими відпочинку, вимкнути сповіщення, коли вони необхідні, і виробити звичку розслаблятися і відпочивати від мобільного пристрою.

3) Негативний вплив на позу: Використання мобільних пристроїв може приводити до поганої пози тіла, особливо коли людина схиляється вперед або незручно тримає пристрій. Це може призводити до болю в шиї, спині та плечах. Важливо дотримуватися правильної пози, тримати пристрій на відстані від очей та регулярно робити перерви для розтягування та вправ для м'язів.

4) Вплив електромагнітного випромінювання: Мобільні пристрої випромінюють електромагнітне випромінювання, яке може бути шкідливим для здоров'я в довготривалій перспективі. Хоча наукові дослідження не дають однозначних доказів про шкідливість електромагнітного випромінювання на низьких рівнях, деякі люди можуть відчувати негативний вплив. Можна зменшити експозицію, використовуючи гарнітуру або вбудовані функції гучномовця, а також обмежуючи тривалість розмов та тримаючи пристрій подалі від тіла.

5) Залежність від мобільних пристроїв: Часте використання мобільних пристроїв може призводити до залежності, яка може негативно позначитися на психічному стані та міжособистових відносинах. Важливо обмежувати час, проведений з мобільним пристроєм, встановлювати правила для себе і приймати свідомі рішення щодо використання пристроїв.

Норми мікроклімату зазначені у санітарних нормах мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [31].

Температура повітря може відрізнятися від норм залежно від пори року та інших умов. Щоб уникнути небажаних відхилень потрібно встановлювати допоміжні засоби регулювання мікроклімату: кондиціонери, обігрівачі тощо.

Найбільш прийнятними методами захисту від шуму є використання акустичних екранів і звукопоглинальних облицювань.

Акустичний екран є перешкодою для звукових хвиль, що знижує рівень звуку шляхом утворення акустичної тіні за екраном в зоні розташування робочого місця.

Ергономічна безпека праці досягається за допомогою нескладних правил. Спина людини повинна бути нахилена назад під кутом в декілька градусів для збільшення кута між тулубом і стегнами, посилення кровообігу і зменшення тиску на хребет. Руки розслаблені й вільно опущені уздовж боків, передпліччя і кисті розташовані паралельно підлозі. Стегна знаходяться під прямим кутом до тулуба. Коліна - під прямим кутом до стегон.

Спинка стільця повторює лінію вигину нижньої частини спини. Сидіння злегка нахилене вперед для перенесення тиску з хребта на стегна і ноги. Край подушки сидіння заломлений вниз для ослаблення тиску на стегна.

Максимальний час безперервної роботи з телефоном не повинен перевищувати 2-х годин в день. Через кожні 7 хвилин потрібно робити коротку перерву на 10-15 секунд. Можна подивитися у вікно (у далечінь) або на картину з приємним для очей пейзажом, яку необхідно повісити за комп'ютером. У картині повинні переважати неяскраві, заспокійливі зір кольори, такі, наприклад, як зелені, блакитні. Також потрібно робити декілька простих вправ для очей.

2) Вимоги безпеки після закінчення роботи:

2.1) Після закінчення роботи фахівець зобов’язаний: відключити від електромережі мобільний телефон , засоби оргтехніки та інше устаткування, крім обладнання, яке використовується цілодобово (апарати факсимільного зв’язку,

мережеві сервери тощо); упорядкувати робоче місце, провітрити приміщення; привести себе у порядок, вимити руки й обличчя;

2.2) Зберегти інформацію в телефоні;

2.3) Прибрати робоче місце.

5.2 Методи нейтралізації або зменшення впливу негативних факторів під час роботи зі мобільними пристроями

Нейтралізація шкідливих та небезпечних факторів при роботі з мобільними пристроями може бути досягнута за допомогою наступних заходів:

1) Зменшення ефекту на зір:

– Регулярно робіть перерви під час роботи з мобільним пристроєм. Загалом рекомендується робити перерву кожні 20-30 хвилин і протягом 5-10 хвилин відводити погляд від екрану;

– Виконуйте вправи для очей, такі як погляд у далечину, обертання очей у різні напрямки і покладання теплих компресів на очі;

– Правильно налаштуйте яскравість, контрастність та розмір шрифту на своєму пристрої для забезпечення комфортного читання;

2) Менеджмент стресу:

– Встановіть часові обмеження для використання мобільних пристроїв і дотримуйтесь їх;

– Вимкніть сповіщення, коли вони необхідні, або встановіть режим "Не турбувати" під час важливих зустрічей або відпочинку;

– Займіться релаксаційними практиками, такими як медитація, глибоке дихання або йога, для зняття стресу і напруження;

3) Збереження правильної пози:

– Підтримуйте правильну позу тіла під час використання мобільного пристрою. Завжди стежте за положенням голови, шиї та спини;

– Використовуйте підставку або тримайте пристрій на такій висоті, щоб ваша голова була в природному положенні і ви могли дивитися прямо на екран без похилення голови;

4) Зменшення впливу електромагнітного випромінювання:

– Використовуйте гарнітуру або вбудовані функції гучномовця, щоб зменшити близький контакт пристрою з головою під час розмов;

– Обмежте тривалість розмов по мобільному телефону і використовуйте функцію високоякісного звуку (HD Voice) для зниження рівня експозиції;

5) Керування залежністю від мобільних пристроїв:

– Встановіть правила для себе щодо використання мобільних пристроїв, наприклад, обмеження часу, проведеного з пристроєм, або встановлення безпосередніх "безмобільних" періодів;

– Підтримуйте здорові звички, такі як виконання фізичних вправ, соціальні контакти та заняття хобі, які не пов'язані з мобільними пристроями;

Загалом, баланс і свідоме використання мобільних пристроїв, разом з регулярними перервами і здоровими звичками, допоможуть нейтралізувати більшість шкідливих та небезпечних факторів, пов'язаних з роботою з мобільними пристроями.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Сучасна інженерія програмного забезпечення має враховувати не лише технічні вимоги, але й екологічні аспекти. Використання ресурсомістких алгоритмів, енергоємних серверів і нерозумне поводження з обладнанням значно збільшують викиди вуглекислого газу та створюють загрозу для довкілля. Тому завдання зниження впливу інформаційних технологій на природу стає актуальним у всьому світі.

6.1 Енергоспоживання програмного забезпечення

Однією з найважливіших екологічних проблем є енергоспоживання серверів, які використовуються для роботи програмних продуктів. Під час їх функціонування значна кількість електроенергії витрачається не лише на обробку даних, але й на охолодження обладнання. Наприклад, дата-центри генерують тепло, що потребує використання додаткових ресурсів для підтримки оптимальної температури.

З метою зменшення впливу на навколишнє середовище розробники програмного забезпечення впроваджують енергоефективні підходи, такі як:

– оптимізація програмного коду для зменшення обчислювальних витрат;

– використання хмарних технологій, які дозволяють централізувати обробку даних;

– розробка алгоритмів із низьким рівнем споживання пам'яті та процесорних ресурсів.

6.2 Утилізація обладнання та е-сміття

Швидкий розвиток ІТ-індустрії також призводить до збільшення кількості електронного сміття. Пристрої, які використовуються для роботи програмного забезпечення, швидко морально застарівають, і їх утилізація стає проблемою. Невідповідне поводження з технікою спричиняє потрапляння токсичних речовин у навколишнє середовище.

Вирішенням цієї проблеми є розробка програмних продуктів, які підтримують довготривалу роботу на старих апаратних засобах. Крім того, важливим є впровадження політик відповідального поводження з е-сміттям, наприклад:

– використання програмного забезпечення, яке автоматично аналізує стан обладнання;

– підтримка вторинного використання техніки шляхом її оновлення та ремонту;

– створення мереж збору та переробки старої техніки.

6.3 Зелені технології у програмуванні

Розробка програмного забезпечення повинна враховувати принципи зелених технологій, які передбачають мінімізацію негативного впливу на природу. Одним із таких підходів є використання хмарних платформ, що дають змогу ефективно розподіляти ресурси між користувачами.

Зелені технології також включають в себе впровадження енергоощадних практик у процес розробки, таких як:

– зменшення кількості зайвих обчислень за рахунок оптимізації алгоритмів;

– застосування автоматизованих інструментів для моніторингу продуктивності програм;

– використання віртуалізації для зменшення навантаження на апаратне забезпечення.

6.4 Міжнародні стандарти для захисту навколишнього середовища

З метою впровадження системного підходу до екологічного менеджменту широко використовуються міжнародні стандарти ISO. Вони встановлюють загальні правила та вимоги, спрямовані на зменшення екологічного впливу.

ISO 14001 [32] – стандарт, який регулює систему екологічного управління підприємствами, включаючи IT-компанії. Він допомагає оптимізувати використання ресурсів і знижувати обсяги відходів.

ISO 50001 [33] – стандарт, спрямований на ефективне управління енергоспоживанням. У програмному забезпеченні його застосовують для моніторингу споживання енергії дата-центрами та пошуку способів її зниження.

ISO 26000 [34] – керівництво щодо соціальної відповідальності, яке охоплює також аспекти екологічної відповідальності, зокрема раціональне використання природних ресурсів та впровадження стійких практик у всіх етапах життєвого циклу програмного продукту.

Впровадження цих стандартів дозволяє компаніям, що займаються розробкою програмного забезпечення, не лише відповідати сучасним екологічним вимогам, але й демонструвати відповідальність перед суспільством та клієнтами.

Інженерія програмного забезпечення відіграє важливу роль у захисті навколишнього середовища. Розробка енергоефективних програм, оптимізація ресурсів та впровадження принципів стійкого розвитку, підтриманих

міжнародними стандартами ISO, дозволяють мінімізувати екологічний вплив інформаційних технологій. Таким чином, інтеграція екологічно орієнтованих підходів у процес створення програмного забезпечення сприятиме збереженню природи для майбутніх поколінь.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно створено програмне забезпечення «JavaShopMetrics» для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java. Результати випробування показали, що розроблене програмне забезпечення успішно виконує всі заплановані функції.

У даній кваліфікаційній роботі було удосконалено нелінійну регресійну модель, яка враховує метрики RFC, CBO та WMC, що відображають основні характеристики якості програмного коду. Аналіз існуючих підходів показав обмеженість лінійних моделей для таких задач, що зумовило необхідність розробки нелінійної регресії з логарифмічними перетвореннями.

Зібрані дані з відкритих Java-проєктів дозволили побудувати достовірну математичну модель. Для її налаштування було проведено обробку даних, включаючи видалення викидів, що підвищило надійність результатів. Застосування нелінійної моделі дозволяє об'єктивніше оцінювати якість Java-застосунків для онлайн-покупок та підвищує достовірність прогнозування їх характеристик, важливих для користувацького досвіду і конкурентоспроможності продукту. Розроблена модель підтвердила свою якість на тестовій вибірці, що вказує на її придатність для використання у практичних умовах.

У процесі розробки було використано сучасні технології та інструменти, такі як фреймворк Flutter та мова програмування Dart, що дозволило забезпечити високу ефективність та зручність використання програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laing, V., Coleman, C. Principal Components of Orthogonal Object-Oriented Metrics. White Paper Analyzing Results of NASA Object-Oriented Data (Task 323-08-14). Greenbelt, Maryland (US): NASA Goddard Space Flight Center, the Software Assurance Technology Center (SATC), 2001.

2. Prykhodko S., Prykhodko N. A Technique for Detecting Software Quality Based on the Confidence and Prediction Intervals of Nonlinear Regression for RFC Metric // 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT). Lviv, Ukraine, 2022. P. 499–502. DOI: https://doi.org/10.1109/CSIT56902.2022.10000532.

3. FURPS software quality model [Електронний ресурс]. URL: 3 (дата звернення: 15.10.2024).

4. Capability Maturity Model Integration (CMMI) [Електронний ресурс]. URL: https://www.geeksforgeeks.org/capability-maturity-model-integration-cmmi/ (дата звернення: 16.10.2024).

5. Six Sigma in Software Engineering [Електронний ресурс]. URL: https://www.geeksforgeeks.org/six-sigma-in-software-engineering/ (дата звернення: 17.10.2024).

6. Dromey, G.: A Model for Software Product Quality // IEEE Transactions on Software Engineering 146, 21 (1995).

7. McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality // Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command (1977).

8. Bagnulo, M., Claise, B., Eardley, P., Morton, A., Akhter, A. RFC 8911 [Електронний ресурс]. URL: https://www.rfc-editor.org/rfc/rfc8911.html (дата звернення: 15.10.2024).

9. Shyam, R., Chris, F. Metrics suite for object oriented design [Електронний ресурс]. URL: https://www.eso.org/~tcsmgr/oowg-forum/TechMeetings/Articles/OOMetrics.pdf (дата звернення: 28.10.2024).

10. Aanchal, S., Kumar**.** Metrics for Software Components in Object Oriented Environments: A Survey [Електронний ресурс] / Aanchal, S., Kumar // International Journal of Scientific Research in Computer Science and Engineering. 2013. ISSN 2320-7639. URL: https://isroset.org/pub_paper/IJSRCSE/ISROSET-IJSRCSE-00043.pdf. (дата звернення: 27.10.2024).

11. Приходько С.Б., Кольцов А.В., Грабовський Є.В. Нелінійна регресійна модель для оцінювання метрики RFC застосунків з відкритим кодом на Kotlin // Інформаційні технології: моделі, алгоритми, системи : IV-а Всеукраїнська науково-практична інтернет-конференція (ITMAS–2023). Миколаїв, Україна, 2023. С. 20–21. URL: https://itconf.nuos.edu.ua/2023/wp-content/uploads/sites/6/NUOS-ITMAS-2023-Proceedings.pdf

12. Приходько С.Б., Скакунов Г.О. Нелінійна регресійна модель для оцінювання метрики RFC застосунків для онлайн-покупок з відкритим кодом на Java // Інформаційні технології: моделі, алгоритми, системи : V-а Всеукраїнська науково-практична інтернет-конференція (ITMAS–2024). Миколаїв, Україна, 2024. С. 59–60. URL: https://itconf.nuos.edu.ua/2024/wp-content/uploads/sites/7/NUOS-ITMAS-2024-Proceedings.pdf

13. Кисилевич В., Усата О., Сікора Я., Вербівський Д., Іванов Д. Мікросервісна архітектура: переваги та недоліки її практичного застосування [Електронний ресурс] / Кисилевич В., Усата О., Сікора Я., Вербівський Д., Іванов Д. // Information technology. Україна. 2024. С. 50–51. DOI: https://doi.org/10.32782/IT/2024-2-7.

14. Сем В. SQL чи NoSQL – ось в чому питання [Електронний ресурс]. URL: https://alternativescience.net/programming/242-sql-chy-nosql-os-v-chomu-pytannya/ (дата звернення: 15.10.2024).

15. Detecting Multicollinearity Using Variance Inflation Factors [Електронний ресурс]. URL: https://online.stat.psu.edu/stat462/node/180/ (дата звернення: 18.10.2024).

16. Sourcemeter [Електронний ресурс]. URL: https://sourcemeter.com/ (дата звернення: 18.10.2024).

17. Mardia K.V. Measures of multivariate skewness and kurtosis with applications // Biometrika, 1970. Vol. 57. С. 519–530. DOI: https://doi.org/10.1093/biomet/57.3.519.

18. Mahalanobis Distance [Електронний ресурс]. URL: https://www.machinelearningplus.com/statistics/mahalanobis-distance/ (дата звернення: 22.10.2024).

19. D., Wulandari, S., Sutrisno. Mardia's Skewness and Kurtosis [Електронний ресурс] / D., Wulandari, S., Sutrisno. // Enthusiastic: International Journal of Applied Statistics and Data Science. 2021. С. 2-4. URL: https://www.researchgate.net/publication/353213860_Mardia's_Skewness_and_Kurtosis_for_Assessing_Normality_Assumption_in_Multivariate_Regression (дата звернення: 22.10.2024).

20. Normal distribution [Електронний ресурс]. URL: https://www.investopedia.com/terms/n/normaldistribution.asp (дата звернення: 15.10.2024).

21. S. Prykhodko, N. Prykhodko, L. Makarova, and K. Pugachenko, “Detecting Outliers in Multivariate Non-Gaussian Data on the basis of Normalizing Transformations”, in Proceedings of the 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON) «Celebrating 25 Years of IEEE Ukraine Section», May 29 – June 2, 2017, Kyiv, Ukraine, 2017, pp. 846-849. https://doi.org/10.1109/UKRCON.2017.8100366

22. Afifi A.A., Azen S.P. Statistical analysis: a computer-oriented approach. New York; London: Academic Press, 1972.

23. F Distribution [Електронний ресурс]. URL: https://www.sciencedirect.com/topics/mathematics/f-distribution (дата звернення: 15.10.2024).

24. Least squares method [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Least_squares (дата звернення: 15.10.2024).

25. Chi-squares [Електронний ресурс]. URL: https://www.investopedia.com/terms/c/chi-square-statistic.asp (дата звернення: 15.10.2024).

26. Visual Studio Code [Електронний ресурс]. URL: https://code.visualstudio.com/ (дата звернення: 19.10.2024).

27. Що таке ROI: визначення терміну, приклади та поради [Електронний ресурс]. URL: https://esputnik.com/uk/slovnyk-email-marketologa/roi (дата звернення: 22.10.2024).

28. How to Calculate Net Present Value (NPV) [Електронний ресурс]. URL: https://www.theforage.com/blog/skills/npv (дата звернення: 25.10.2024).

29. Про охорону праці: Закон України від 21.11.2002 р. № 229-IV [Електронний ресурс]. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 23.11.2024).

30. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників: Закон України від 14.02.2018 № 207 [Електронний ресурс]. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14 (дата звернення: 23.11.2024).

31. Санітарні норми мікроклімату виробничих приміщень (ДСН 3.3.6.042-99) [Електронний ресурс]. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 23.11.2024).

32. ISO 14001:2015 - Environmental management systems [Електронний ресурс]. URL: https://www.iso.org/standard/60857.html (дата звернення: 27.10.2024).

33. ISO 50001 — Energy management [Електронний ресурс]. URL: https://www.iso.org/iso-50001-energy-management.html (дата звернення: 30.10.2024).

34. ISO 26000 — Social responsibility [Електронний ресурс]. URL: https://www.iso.org/iso-26000-social-responsibility.html (дата звернення: 31.10.2024).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JavaMetricsShop»

Вступ

Повна назва програмного продукту: Програмне забезпечення «JavaMetricsShop» для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java.

Коротка назва: «JavaMetricsShop».

Сфера застосування: організації, котрі використовують мову Java для розробки програмних продуктів, а саме інтернет магазини.

1. Підстави для розробки

Підставою для розробки «JavaMetricsShop» є наказ на затвердження тем кваліфікаційних робіт магістрів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням використання в навчальних цілях та аналізу яксоті програмних проектів, розроблених з використанням Java.

2.2 Функціональне призначення

Програмне забезпечення дозволяє здійснювати аналіз якості програмних проектів, реалізованих з використанням Java, за допомогою визначених метрик, забезпечувати можливість прогнозування якість майбутніх проектів на основі аналізу вхідних даних.

3. Вимоги до програми або програмного виробу

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

− Завантаження проектів інтернет-магазинів, розроблених на Java та їх метрик, таких як RFC, CBO, WMC.

− Побудова лінійної регресійної моделі.

− Побудова нелінійної регресійної моделі.

− Визначення показників якості нелінійної регресійної моделі.

− Прогнозування якості за метриками RFC, CBO, WMC.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані:

− CSV-файл із зібраними метриками проектів.

− Метрики RFC, CBO, WMC.

Вихідні дані:

− Лінійна регресійна модель.

− Нелінійна регресійна модель.

− Якість нелінійної моделі.

− Прогнози якості.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (сталого) функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

− захист від некоректних дій користувача (під час аналізу);

− Flutter не нижче версії 3.20.

3.2.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

3.2.3 Відмови через некоректні дії оператора

Відмова можлива при наступних умовах:

– внаслідок закриття програми під час аналізу, в цьому випадку користувачеві показується додаткове вікно, чи впевнений він що хоче закрити програму;

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до чисельності та кваліфікації користувачів

Програму використовує один Користувач, котрий повинен мати базові навички роботи з комп’ютером.

3.4 Вимоги до складу і параметрів технічних засобів

Для роботи із програмним забезпеченням необхідний наступний склад технічних засобів з мінімальними параметрами:

– Процесор Intel Pentium 1Ghz;

– Оперативна пам'ять 512Мб;

– Дискового простору 1 Гб;

– Роздільна здатність екрану 800x600 пікселів;

– Інтернет-з’єднання. Програмне забезпечення вимагає доступу до Інтернету.

3.5 Вимоги до інформаційної та програмної сумісності

3.5.1 Вимоги до вихідних кодів і мов програмування

Системні програмні засоби, які використовуються програмою, повинні бути представлені ліцензійною версією операційної системи Windows (версії не нижче Windows 7). В якості інтегрованого середовища розробки програми буде використовуватися середовище Microsoft Visual Studio Code. Вихідні коди програми повинні бути реалізовані за допомогою фреймворку Flutter на мові Dart.

3.6 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою графічного інтерфейсу користувача, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програму та методику випробувань;
- керівництво користувача;
- опис програми;
- код програми.

5. Техніко-економічні показники

У даній курсовій роботі техніко-економічні показники не розраховуються.

6. Етапи роботи

Етапи роботи представлені у таблиці А.1

Таблиця А.1 – Етапи роботи

№	Назва етапів роботи створення програмного забезпечення	Кінцевий термін виконання
1.	Аналіз вимог до ПЗ	01.09.2024
2.	Розробка архітектури ПЗ	21.09.2024
3.	Розробка проекту ПЗ	16.10.2024
4.	Реалізація та випробування ПЗ	26.10.2024
5.	Випробування ПЗ	1.11.2024

7. Порядок контролю та приймання

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у

технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JavaShopMetrics»

main.dart

```
import 'dart:async';

import 'package:flutter/foundation.dart';
import 'package:flutter/widgets.dart';
import 'package:flutter_web_plugins/url_strategy.dart';
import 'package:java_shop_metrics/app/app.dart';
import 'package:java_shop_metrics/services/logger.dart';

final _log = AppLogger().logger;

Future<void> main() async {
  await runZonedGuarded(() async {
    WidgetsFlutterBinding.ensureInitialized();

    usePathUrlStrategy();

    runApp(const App());
  }, (error, stackTrace) {
    _log.e(
      'Error',
      error: error,
      stackTrace: stackTrace,
    );
  });
}
```

router.dart

```
import 'package:flutter/widgets.dart';
import 'package:go_router/go_router.dart';
import 'package:java_shop_metrics/providers/regression_model_provider.dart';
import 'package:java_shop_metrics/router/scaffold_nav.dart';
import 'package:java_shop_metrics/ui/views/metrics_view.dart';
import 'package:java_shop_metrics/ui/views/regression_view.dart';
import 'package:java_shop_metrics/ui/widgets/prediction_widget.dart';
import 'package:provider/provider.dart';

GlobalKey<NavigatorState> _rootNavigatorKey = GlobalKey<NavigatorState>();

GoRouter router = GoRouter(
  initialLocation: '/projects',
```

```
  navigatorKey: _rootNavigatorKey,
  routes: [
    StatefulShellRoute.indexedStack(
      builder: (context, state, navigationShell) {
        return ScaffoldWithNestedNavigation(navigationShell: navigationShell);
      },
      branches: [
        buildBranch('/projects', const MetricsView()),
        buildBranch('/regression', const RegressionView()),
        buildBranch('/prediction', const PredictionWidget()),
      ],
    ),
  ],
  redirect: (context, state) {
    if (state.path != '/projects') {
      final projects  = context.read<RegressionModelProvider>().projects;
      if (projects.isEmpty) {
        return '/projects';
      }
    }
    return null;
  },
);

StatefulShellBranch buildBranch(String path, Widget page) {
  return StatefulShellBranch(
    routes: [
      GoRoute(
        path: path,
        pageBuilder: (context, state) => CustomTransitionPage(
          transitionsBuilder: (context, animation, secondaryAnimation, child) =>
              FadeTransition(
            opacity: animation,
            child: child,
          ),
          child: page,
        ),
      ),
    ],
  );
}
```

project_list.dart

```
import 'package:flutter/widgets.dart';
import 'package:java_shop_metrics/models/project/project.dart';
import 'package:java_shop_metrics/services/utils.dart';
```

```
import 'package:java_shop_metrics/ui/components/app_list_tile.dart';

class ProjectsList extends StatelessWidget {
  const ProjectsList({
    required this.projects,
    this.outliersIndexes,
    super.key,
  });

  final List<Project> projects;
  final List<int>? outliersIndexes;

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: projects.length,
      itemBuilder: (context, index) {
        if (outliersIndexes != null && !outliersIndexes!.contains(index)) {
          return const SizedBox.shrink();
        }
        final project = projects[index];
        final metric = project.metrics!;
        var subtitle = '(Y) RFC: ${metric.rfc!.toStringAsFixed(4)}';
        if (metric.cbo != null) {
          var value = metric.cbo!.toStringAsFixed(4);
          if (metric.cbo! % 1 == 0) {
            value = metric.cbo!.toStringAsFixed(0);
          }
          subtitle += '\n(X1) CBO: $value';
        }
        if (metric.wmc != null) {
          var value = metric.wmc!.toStringAsFixed(4);
          if (metric.wmc! % 1 == 0) {
            value = metric.wmc!.toStringAsFixed(0);
          }
          subtitle += '\n(X2) WMC: $value';
        }
        return AppListTile(
          index: index,
          title: project.url ?? subtitle,
          subtitle: project.url != null ? subtitle : null,
          onTap: () async {
            if (project.url != null) {
              await Utils.openUrl(project.url!);
            } else {
              Utils.copyToClipboard(subtitle);
            }
          },
```

```
      );
    },
  );
 }
}
```

prediction_widget.dart

```
import 'dart:math';

import 'package:flutter/cupertino.dart' show CupertinoTextField;
import 'package:flutter/services.dart';
import 'package:flutter/widgets.dart';
import 'package:java_shop_metrics/providers/regression_model_provider.dart';
import 'package:java_shop_metrics/ui/widgets/metrics_card.dart';
import 'package:provider/provider.dart';

class PredictionWidget extends StatefulWidget {
 const PredictionWidget({super.key});

 @override
 State<PredictionWidget> createState() => _PredictionWidgetState();
}

class _PredictionWidgetState extends State<PredictionWidget> {
 late RegressionModelProvider provider;
 final yController = TextEditingController();
 final x1Controller = TextEditingController();
 final x2Controller = TextEditingController();
 num? _prediction;

 Widget _buildTextField(TextEditingController controller, String label) {
  return Column(
   children: [
    Text(
     label,
     style: const TextStyle(
      fontSize: 16,
      color: Color.fromARGB(255, 0, 0, 0),
     ),
    ),
    Expanded(
     child: CupertinoTextField(
      controller: controller,
      keyboardType: TextInputType.number,
      textAlign: TextAlign.center,
      inputFormatters: [
```

```
            FilteringTextInputFormatter.digitsOnly,
          ],
          style: const TextStyle(
            fontSize: 16,
            color: Color.fromARGB(255, 0, 0, 0),
          ),
          decoration: BoxDecoration(
            color: const Color.fromARGB(255, 240, 240, 240),
            borderRadius: BorderRadius.circular(32),
          ),
          onChanged: (_) => makePrediction(),
        ),
      ),
    ],
  );
}

Widget _buildPredictionOutput() {
  if (_prediction == null ||
      _prediction!.isNaN ||
      _prediction!.isInfinite ||
      _prediction! < 0) {
    return const SizedBox();
  }

  return FutureBuilder(
    future: provider.calculateQuality(
      yHat: _prediction?.toDouble(),
      y: double.tryParse(yController.text),
      x1: double.tryParse(x1Controller.text),
      x2: double.tryParse(x2Controller.text),
    ),
    builder: (context, snapshot) {
      final quality = snapshot.data;
      if (quality == null) return const SizedBox();

      return MetricsCard(
        title: 'Якість',
        value: quality.name,
      );
    },
  );
}

void makePrediction() {
  final b0 = provider.coefficients[0];
  final b1 = provider.coefficients[1];
  final b2 = provider.coefficients[2];
```

```
  final x1 = double.tryParse(x1Controller.text) ?? -1;
  final x2 = double.tryParse(x2Controller.text) ?? -1;
  if (x1 == -1 || x2 == -1) {
    _prediction = null;
    setState(() {});
    return;
  }
  _prediction = pow(10, b0) * pow(x1, b1) * pow(x2, b2);
  setState(() {});
}

Widget availableFactorsRange() {
  final minMaxFactors = provider.minMaxFactors();
  String formatNumber(double value) => value.toStringAsFixed(4);

  final minY = formatNumber(minMaxFactors.y.min);
  final maxY = formatNumber(minMaxFactors.y.max);

  final minX1 = formatNumber(minMaxFactors.x1.min);
  final maxX1 = formatNumber(minMaxFactors.x1.max);

  final minX2 = formatNumber(minMaxFactors.x2.min);
  final maxX2 = formatNumber(minMaxFactors.x2.max);

  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      const Text(
        'Допустимі значення:',
        style: TextStyle(fontSize: 16),
      ),
      const SizedBox(height: 8),
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Flexible(
            child: MetricsCard(
              title: 'Інтервали RFC',
              value: '[$minY - $maxY]',
            ),
          ),
          const SizedBox(width: 16),
          Flexible(
            child: MetricsCard(
              title: 'Інтервали CBO',
              value: '[$minX1 - $maxX1]',
            ),
          ),
```

```
          const SizedBox(width: 16),
          Flexible(
            child: MetricsCard(
              title: 'Інтервали WMC',
              value: '[$minX2 - $maxX2]',
            ),
          ),
        ],
      ),
    ],
  );
}

@override
Widget build(BuildContext context) {
  provider = context.watch<RegressionModelProvider>();
  return Column(
    children: [
      IntrinsicHeight(
        child: Row(
          children: [
            Expanded(
              child: _buildTextField(
                yController,
                'Введіть Y (RFC)',
              ),
            ),
            const SizedBox(width: 16),
            Expanded(
              child: _buildTextField(
                x1Controller,
                'Введіть X1 (CBO)',
              ),
            ),
            const SizedBox(width: 16),
            Expanded(
              child: _buildTextField(
                x2Controller,
                'Введіть X2 (WMC)',
              ),
            ),
          ],
        ),
      ),
      const SizedBox(height: 16),
      _buildPredictionOutput(),
      const SizedBox(height: 16),
      availableFactorsRange(),
```

```
        const SizedBox(height: 32),
      ],
    );
  }
}
```

regression_model.dart

```
import 'dart:math';

import 'package:flutter/widgets.dart';
import 'package:java_shop_metrics/constants.dart';
import 'package:java_shop_metrics/models/intervals/intervals.dart';
import 'package:java_shop_metrics/models/metrics/metrics.dart';
import 'package:java_shop_metrics/models/model_quality/model_quality.dart';
import 'package:java_shop_metrics/services/fisher.dart';
import 'package:java_shop_metrics/services/student.dart';
import 'package:ml_linalg/linalg.dart';

enum QualityTypes {
  high(name: 'Висока якість'),
  medium(name: 'Середня якість'),
  low(name: 'Низка якість'),
  unknown(name: 'Невідома якість');

  const QualityTypes({required this.name});

  final String name;
}

class RegressionModel {
  RegressionModel(this.metrics)
      : x1 = metrics.map((e) => e.cbo!).toList(),
        x2 = metrics.map((e) => e.wmc!).toList(),
        y = metrics.map((e) => e.rfc!).toList(),
        Zx1 = metrics.map((e) => log(e.cbo!) / log(10)).toList(),
        Zx2 = metrics.map((e) => log(e.wmc!) / log(10)).toList(),
        Zy = metrics.map((e) => log(e.rfc!) / log(10)).toList();

  List<Metrics> metrics;
  List<double> x1;
  List<double> x2;
  List<double> y;
  List<double> Zx1;
  List<double> Zx2;
  List<double> Zy;

  int get n => metrics.length;
```

```
double get x1Avg => x1.reduce((a, b) => a + b) / n;
double get x2Avg => x2.reduce((a, b) => a + b) / n;
double get yAvg => y.reduce((a, b) => a + b) / n;
double get Zx1Avg => Zx1.reduce((a, b) => a + b) / n;
double get Zx2Avg => Zx2.reduce((a, b) => a + b) / n;
double get ZyAvg => Zy.reduce((a, b) => a + b) / n;

List<List<double>> calculateCovarianceMatrix([List<List<double>>? data]) {
  List<List<double>> cov = List.generate(3, (_) => List.filled(3, 0));

  List<List<double>> values = data ??
      [
        Zy,
        Zx1,
        Zx2,
      ];

  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      double sum = 0;
      for (int k = 0; k < n; k++) {
        sum += (values[i][k] - values[i].reduce((a, b) => a + b) / n) *
            (values[j][k] - values[j].reduce((a, b) => a + b) / n);
      }
      cov[i][j] = sum / n;
    }
  }

  return cov;
}

List<List<double>> invertMatrix(List<List<double>> matrix) {
  int n = matrix.length;

  List<List<double>> identityMatrix =
      List.generate(n, (i) => List.generate(n, (j) => i == j ? 1.0 : 0.0));

  List<List<double>> a = matrix.map((row) => row.toList()).toList();

  for (int i = 0; i < n; i++) {
    double diagElement = a[i][i];
    if (diagElement == 0) {
      throw Exception('Matrix is not invertible');
    }
    for (int j = 0; j < n; j++) {
      a[i][j] /= diagElement;
      identityMatrix[i][j] /= diagElement;
    }
```

```
    for (int k = 0; k < n; k++) {
      if (k == i) continue;
      double factor = a[k][i];
      for (int j = 0; j < n; j++) {
        a[k][j] -= factor * a[i][j];
        identityMatrix[k][j] -= factor * identityMatrix[i][j];
      }
    }
  }

  return identityMatrix;
}

List<List<double>> inverseMatrix(List<List<double>> matrix) {
  final size = matrix.length;
  final identityMatrix = List<List<double>>.generate(
    size,
    (i) => List.generate(size, (j) => i == j ? 1.0 : 0.0),
  );
  final workingMatrix = matrix.map((row) => row.toList()).toList();

  for (var i = 0; i < size; i++) {
    final diagElement = workingMatrix[i][i];
    for (var j = 0; j < size; j++) {
      workingMatrix[i][j] /= diagElement;
      identityMatrix[i][j] /= diagElement;
    }

    for (var k = 0; k < size; k++) {
      if (k == i) continue;
      final factor = workingMatrix[k][i];
      for (var j = 0; j < size; j++) {
        workingMatrix[k][j] -= factor * workingMatrix[i][j];
        identityMatrix[k][j] -= factor * identityMatrix[i][j];
      }
    }
  }

  return identityMatrix;
}

List<double> calculateMahalanobisDistances(
  List<List<double>> covInv, [
  List<List<double>>? data,
]) {
  List<double> distances = List.filled(n, 0);
```

```
  List<List<double>> values = data ??
      [
        Zy,
        Zx1,
        Zx2,
      ];

  for (int i = 0; i < n; i++) {
    List<double> row = List.filled(3, 0);
    for (int j = 0; j < 3; j++) {
      for (int k = 0; k < 3; k++) {
        row[j] += covInv[j][k] *
            (values[k][i] - values[k].reduce((a, b) => a + b) / n);
      }
    }

    for (int j = 0; j < 3; j++) {
      distances[i] +=
          row[j] * (values[j][i] - values[j].reduce((a, b) => a + b) / n);
    }
  }

  return distances;
}

List<double> calculateTestStatistics(List<double> distances) {
  List<double> testStatistics = List.filled(n, 0);

  for (int i = 0; i < n; i++) {
    testStatistics[i] = (n - 3) * n / ((pow(n, 2) - 1) * 3) * distances[i];
  }

  return testStatistics;
}

Future<double?> calculateFisherFDistribution() {
  return Fisher.inv(
    alpha: alpha,
    df1: 3,
    df2: n - 3,
  );
}

Future<List<int>> determineOutliers(List<double> testStatistics) async {
  double? f = await calculateFisherFDistribution();
  if (f == null) {
    debugPrint('Failed to calculate Fisher F distribution');
    return [];
```

```
  }
  List<int> outliers = [];
  for (int i = 0; i < n; i++) {
    if (testStatistics[i] > f) {
      outliers.add(i);
    }
  }
  return outliers;
}

List<double> calculateRegressionCoefficients() {
  int n = y.length;

  Matrix X = Matrix.fromRows([
    for (int i = 0; i < n; i++) Vector.fromList([1, Zx1[i], Zx2[i]]),
  ]);

  Matrix yMatrix = Matrix.column(Zy);

  Matrix X_transpose = X.transpose();
  Matrix X_transpose_X = X_transpose * X;
  Matrix X_transpose_X_inv = X_transpose_X.inverse();
  Matrix X_transpose_y = X_transpose * yMatrix;
  Matrix b = X_transpose_X_inv * X_transpose_y;

  List<double> coefficients = b.getColumn(0).toList();

  return coefficients;
}

List<double> calculatePredictedValues(List<double> b) {
  double b0 = b[0];
  double b1 = b[1];
  double b2 = b[2];

  List<double> Zy_hat = List.filled(n, 0);
  for (int i = 0; i < n; i++) {
    Zy_hat[i] = b0 + b1 * Zx1[i] + b2 * Zx2[i];
  }

  return Zy_hat;
}

List<double> calculateYHat(List<double> predictedValues) {
  return predictedValues
      .map((value) => pow(10, value))
      .toList()
      .cast<double>();
```

```
}

ModelQuality calculateModelQuality(List<double> predictedValues) {
  final yHat = calculateYHat(predictedValues);

  final yDiffSquared =
      y.map((value) => pow(value - yHat[y.indexOf(value)], 2)).toList();
  final yAvgDiffSquared = y.map((value) => pow(value - yAvg, 2)).toList();

  final yDividedDiff = List.generate(y.length, (index) {
    return (y[index] - yHat[index]) / y[index];
  });

  final sy = yDiffSquared.reduce((value, element) => value + element);
  final rSquared =
      1 - (sy / yAvgDiffSquared.reduce((value, element) => value + element));

  final mmre = yDividedDiff
          .map((diff) => diff.abs())
          .reduce((value, element) => value + element) /
      y.length;

  final pred =
      yDividedDiff.where((diff) => diff.abs() < 0.25).length / y.length;

  return ModelQuality(
    rSquared: rSquared,
    sy: sy.toDouble(),
    mmre: mmre,
    pred: pred,
  );
}

List<List<double>> matrixMultiply(
  List<List<double>> a,
  List<List<double>> b,
) {
  final rows = a.length;
  final cols = b[0].length;
  final n = b.length;

  return List.generate(
    rows,
    (i) => List.generate(cols, (j) {
      double sum = 0;
      for (var k = 0; k < n; k++) {
        sum += a[i][k] * b[k][j];
      }
```

```
    return sum;
   }),
  );
}

List<List<double>> transposeMatrix(List<List<double>> matrix) {
  final rows = matrix.length;
  final cols = matrix[0].length;

  return List.generate(
    cols,
    (i) => List.generate(rows, (j) => matrix[j][i]),
  );
}

List<double> matrixVectorMultiply(
  List<List<double>> matrix,
  List<double> vector,
) {
  return List.generate(
    matrix.length,
    (i) => List.generate(vector.length, (j) => matrix[i][j] * vector[j])
        .reduce((a, b) => a + b),
  );
}

Future<Intervals> computeIntervals({
  required List<List<double>> zValues,
  required List<double> actualZy,
  required List<double> predictedZy,
}) async {
  final count = zValues.length;
  final paramCount = zValues.first.length;
  final degreesOfFreedom = count - (paramCount + 1);

  final tValue = await Student.inv2T(alpha: 0.05 / 2, df: degreesOfFreedom) ?? 0;

  try {
    final residuals = List.generate(count, (i) => actualZy[i] - predictedZy[i]);
    final residualStdDev =
        sqrt(residuals.map((r) => r * r).reduce((a, b) => a + b) / degreesOfFreedom);
    final zMeans = List.generate(
      paramCount,
      (j) => zValues.map((row) => row[j]).reduce((a, b) => a + b) / count,
    );
    final zCentered = List.generate(
        count, (i) => List.generate(paramCount, (j) => zValues[i][j] - zMeans[j]));
```

```
    final covarianceMatrix = matrixMultiply(
      transposeMatrix(zCentered),
      zCentered,
    );

    final invertedCovarianceMatrix = inverseMatrix(covarianceMatrix);

    final quadraticTerms = List.generate(count, (i) {
      final temp = matrixVectorMultiply(invertedCovarianceMatrix, zCentered[i]);
      return List.generate(paramCount, (j) => temp[j] * zCentered[i][j])
          .reduce((a, b) => a + b);
    });

    final lowerPrediction = List.generate(
      count,
      (i) => predictedZy[i] - tValue * residualStdDev * sqrt(1 + 1 / count + quadraticTerms[i]),
    );
    final upperPrediction = List.generate(
      count,
      (i) => predictedZy[i] + tValue * residualStdDev * sqrt(1 + 1 / count + quadraticTerms[i]),
    );
    final lowerConfidence = List.generate(
      count,
      (i) => predictedZy[i] - tValue * residualStdDev * sqrt(1 / count + quadraticTerms[i]),
    );
    final upperConfidence = List.generate(
      count,
      (i) => predictedZy[i] + tValue * residualStdDev * sqrt(1 / count + quadraticTerms[i]),
    );

    return Intervals(
      y: actualZy.map((x) => pow(10, x).toDouble()).toList(),
      yHat: predictedZy.map((x) => pow(10, x).toDouble()).toList(),
      predictionLower: lowerPrediction.map((x) => pow(10, x).toDouble()).toList(),
      predictionUpper: upperPrediction.map((x) => pow(10, x).toDouble()).toList(),
      confidenceLower: lowerConfidence.map((x) => pow(10, x).toDouble()).toList(),
      confidenceUpper: upperConfidence.map((x) => pow(10, x).toDouble()).toList(),
    );
  } catch (error) {
    return Intervals.empty();
  }
}

Future<QualityTypes?> evaluateQuality({
  double? predictedY,
  double? actualY,
  double? metric1,
  double? metric2,
```

```
}) async {
 if (predictedY == null || actualY == null || metric1 == null || metric2 == null) return null;
 try {
  final normalizedData = [
   [log(actualY) / log(10), ...Zy],
   [log(metric1) / log(10), ...Zx1],
   [log(metric2) / log(10), ...Zx2],
  ];

  final mahalanobisDistances = calculateMahalanobisDistances(
   inverseMatrix(
    calculateCovarianceMatrix(normalizedData),
   ),
   normalizedData,
  );
  const chiSquareThreshold = 12.83815647;
  for (final distance in mahalanobisDistances) {
   if (distance > chiSquareThreshold) {
    return QualityTypes.unknown;
   }
  }

  final intervals = await computeIntervals(
   predictedZy: [
    log(predictedY) / log(10),
    ...calculatePredictedValues(calculateRegressionCoefficients()),
   ],
   actualZy: normalizedData[0],
   zValues: List.generate(
    normalizedData.first.length,
    (i) => [normalizedData[1][i], normalizedData[2][i]],
   ),
  );

  final predLower = intervals.predictionLower.first;
  final predUpper = intervals.predictionUpper.first;
  final confLower = intervals.confidenceLower.first;
  final confUpper = intervals.confidenceUpper.first;


  if (actualY > predUpper) {
   return QualityTypes.low;
  } else if (actualY >= confLower && actualY <= confUpper) {
   return QualityTypes.medium;
  } else if ((actualY > confUpper && actualY <= predUpper) ||
     (actualY >= predLower && actualY < confLower)) {
   return QualityTypes.high;
  }
```

```
    } catch (error) {
      print(error);
      return QualityTypes.unknown;
    }

    return QualityTypes.unknown;
  }
}
```

ДОДАТОК В – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«JavaShopMetrics»

Програмне забезпечення «JavaShopMetrics» створене для: аналізу якості програмних застосунків, розроблених на Java, із застосуванням метрик якості коду. Це програмне забезпечення забезпечує автоматизований збір, обробку, аналіз та прогнозування на основі статистичних моделей, таких як лінійна та нелінійна регресія.

Функціональні можливості:

- Завантаження проектів Java у форматі CSV із метриками RFC, CBO, WMC.
- Побудова лінійних і нелінійних регресійних моделей.
- Розрахунок показників якості моделі.
- Прогнозування якості програмного забезпечення за метриками.

Мова розробки та технології:

- Мова програмування: Dart.
- Фреймворк: Flutter (версія не нижче 3.20).
- Використовуються пакети csv, ml_linalg, flutter_math_fork

Принципи роботи:

1. Користувач завантажує CSV-файл із метриками проектів.
2. Програмне забезпечення обробляє дані та нормалізує їх.
3. За допомогою статистичних методів будує моделі регресії.
4. Проводиться аналіз якості моделі на основі таких показників, як R^2, MMRE та PRED(0.25).
5. Прогнозування якості (висока, середня, низька) виконується за довірчими інтервалами та інтервалами прогнозування побудованої нелінійної регресійної моделі та за метриками RFC, CBO, WMC.

Технічні характеристики:

- Середовище виконання: Flutter Web, браузер Google Chrome.
- Платформи: Windows 7 і новіші версії.

Системні вимоги:

- Процесор: Intel Pentium 1 GHz.
- Оперативна пам’ять: 512 МБ.
- Дисковий простір: 1 ГБ.
- Роздільна здатність екрана: 800x600 пікселів.

Функціональні особливості:

- Інтуїтивно зрозумілий інтерфейс користувача.
- Підтримка завантаження даних у форматі CSV.
- Візуалізація результатів побудованої моделі.
- Можливість прогнозування якості застосунків.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JavaShopMetrics»

Програмне забезпечення «JavaShopMetrics» призначене для аналізу розміру програмних проектів інтернет-магазинів, реалізованих за допомогою мови програмування Java. Застосунок дозволяє завантажувати дані метрик, обробляти їх, аналізувати і здійснювати прогнозування на основі різних статистичних моделей.

1. Запуск застосунку

– Відкрийте веб-браузер та перейдіть за URL-адресою застосунка «JavaShopMetrics».

2. Головна сторінка

– На головній сторінці ви побачите кнопку для завантаження даних метрик із CSV файлу та компоненти меню.

– Натисніть кнопку "Завантажити дані", щоб імпортувати ваш CSV файл із метриками.

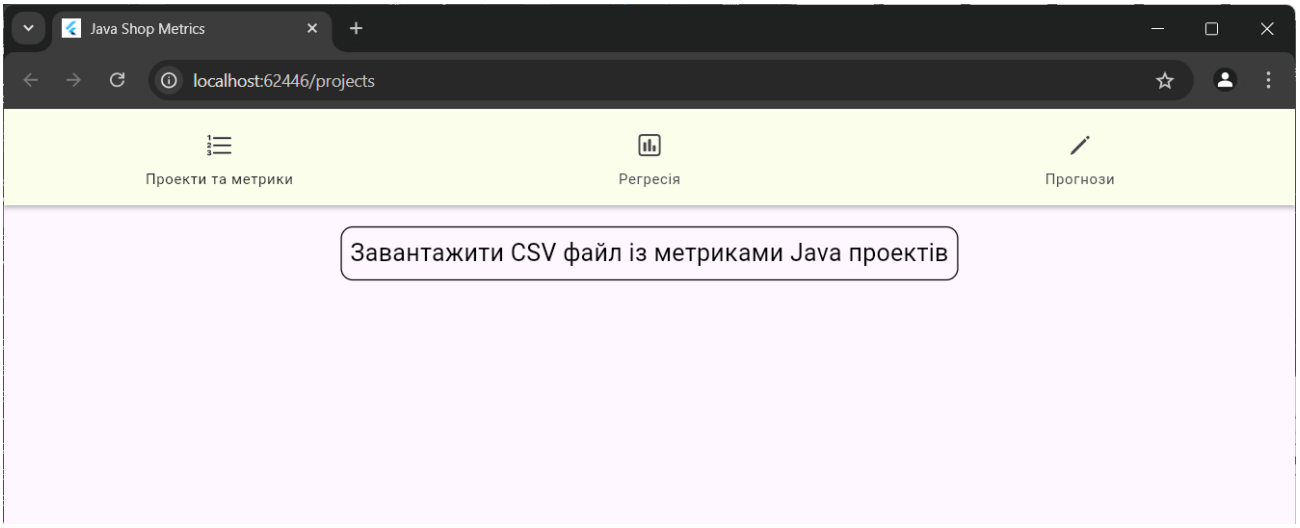

Рисунок Г.1 – Головна сторінка застосунку «JavaShopMetrics».

3. Проекти та метрики

– Після завантаження даних вони відобразяться у вкладці "Проекти та метрики".

– Тут ви можете переглянути список проектів та відповідні метрики.

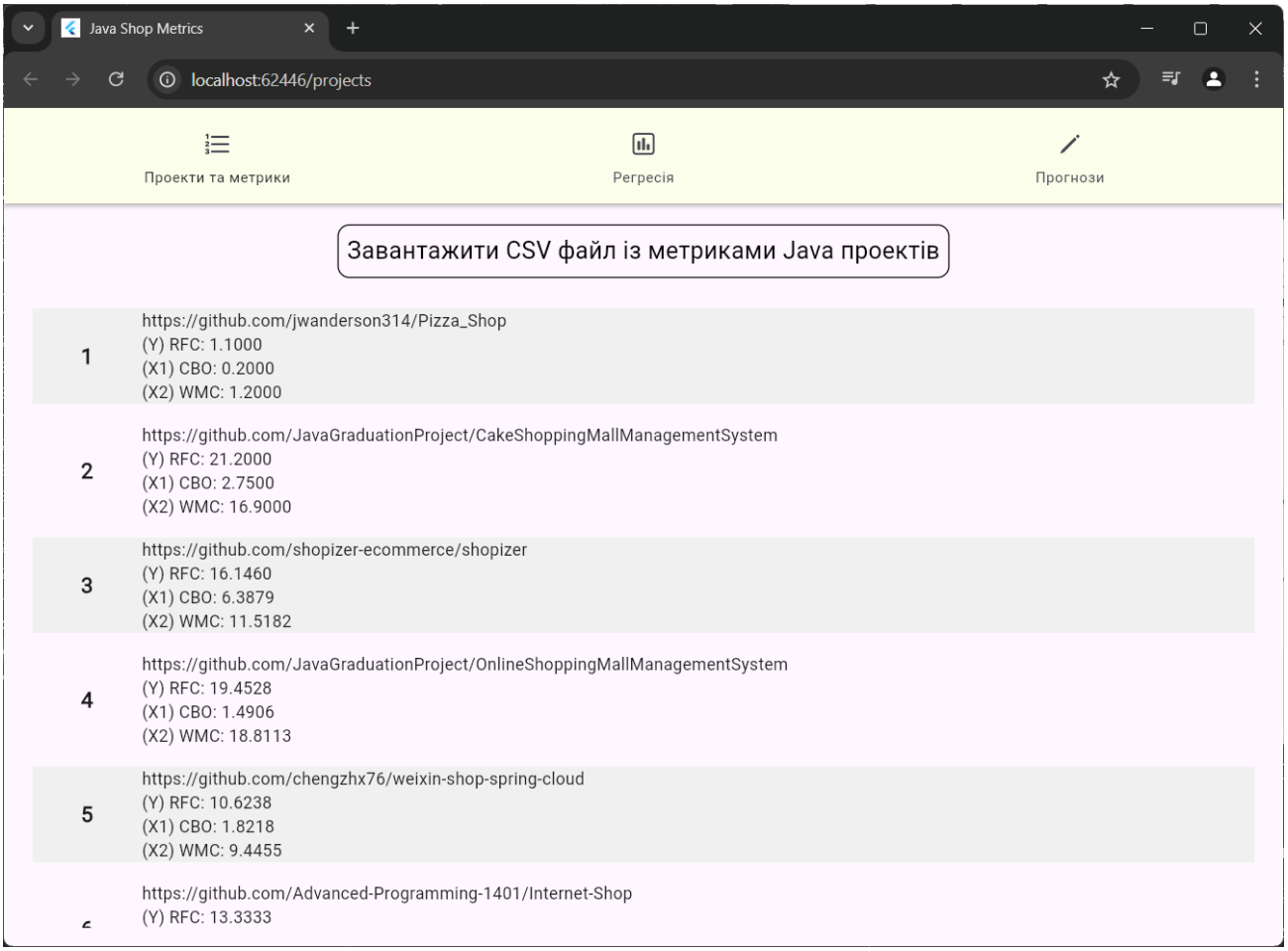

Рисунок Г.2 – Вкладка «Проекти та метрики» застосунку «JavaShopMetrics»

4. Регресія та якість моделі

– У вкладці "Регресія" здійснюється побудова лінійних та нелінійних регресійних моделей на основі даних проектів.

– Застосунок автоматично розрахує коефіцієнти регресії та оцінить якість моделі.

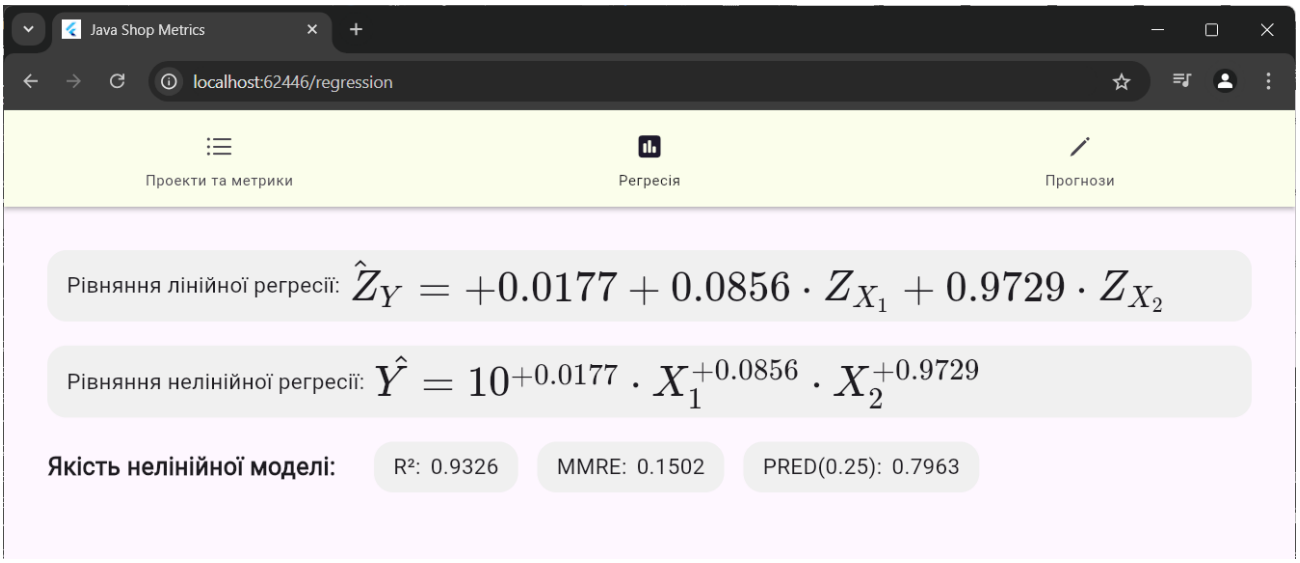

Рисунок Г.3 – Вкладка «Регресія» застосунку «JavaShopMetrics»

5. Прогнози

– У вкладці "Прогнози" здійснюється прогнози якості застосунків.

– Введіть бажане RFC, CBO, WMC у відповідні поля для прогнозу якості.

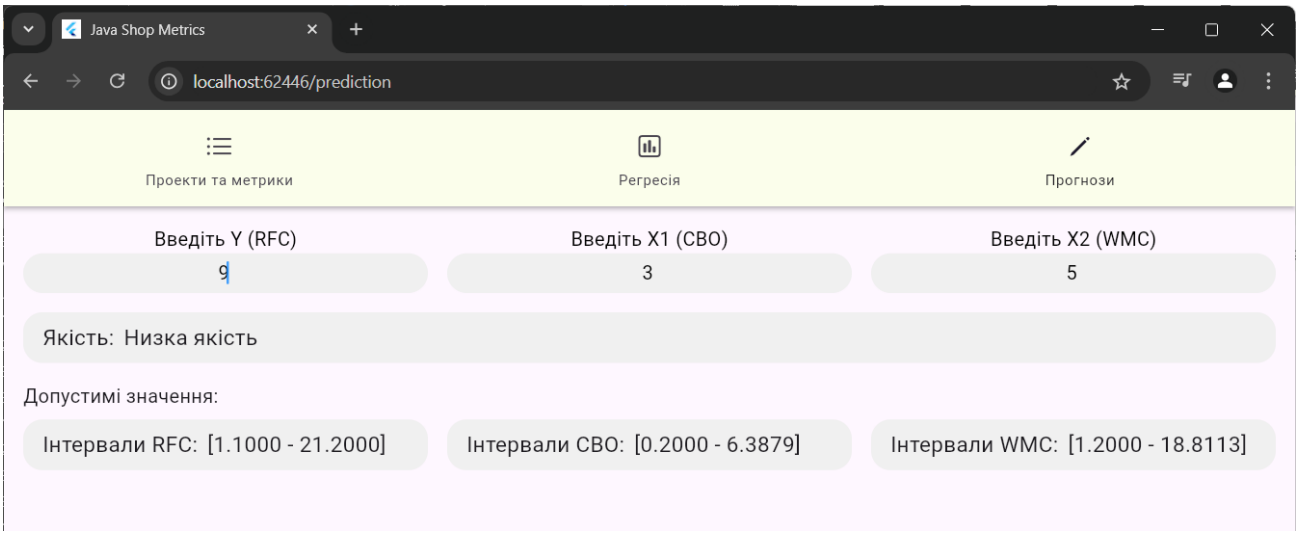

Рисунок Г.4 – Вкладка «Прогнози» застосунку «JavaShopMetrics»

Ця інструкція має на меті забезпечити швидкий старт та ефективне використання програмного забезпечення «JavaShopMetrics».

ДОДАТОК Д – ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «JavaShopMetrics»

1. Об’єкт випробування

Об'єктом випробування є програмне забезпечення «JavaShopMetrics» для оцінювання якості застосунків для онлайн-покупок з відкритим кодом на Java. Основним завданням програми є аналіз метрик коду (RFC, CBO, WMC) і застосування математичної моделі для оцінки якості.

2. Мета випробування

Метою випробування є перевірка відповідності програмного забезпечення «JavaShopMetrics» вимогам технічного завдання (Додаток А), оцінка працездатності основних функціональних можливостей та перевірка якості документальної підтримки.

3. Вимоги до програмного забезпечення

Програмне забезпечення повинно відповідати вимогам технічного завдання, розміщеного у Додатку А. Основні вимоги включають:

- Коректне обчислення метрик RFC, CBO, WMC.
- Побудова та валідація нелінійної регресійної моделі.
- Забезпечення графічного інтерфейсу для візуалізації результатів.

4. Вимоги до програмної документації

- Керівництво користувача.
- Опис архітектури системи.
- Інструкцію з розгортання та налаштування.
- Технічне завдання.

5. Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

– Комп’ютер з процесором AMD Ryzen 5 2600 Six-Core Processor 3,40 ГГц, ОЗУ 16 ГБ, SSD Диск KINGSTON SA400S37480G 447 ГБ

– Монітор з роздільною здатністю 1920x1080.

5.2 Програмні засоби, що використовуються під час випробувань

– Windows 11 Pro x64 24H2.

– Visual Studio Code 1.94.5.0.

– Flutter 3.23.4.

– Dart 3.4.5.

5.3 Порядок проведення випробувань

Випробування «JavaShopMetrics» включає два етапи: перевірку відповідності програмної документації та перевірку функціональних характеристик програмного забезпечення.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

– Перевірити наявність та відповідність документації технічним вимогам.

– Проаналізувати структуру, логічність викладення та повноту інформації.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Функціональні випробування проводитимуться шляхом виконання тестових сценаріїв, наведених у Таблиці Д.1.

Таблиця Д.1 – Перелік випробувань основних функціональних можливостей

№	Дія	Очікуваний результат
1	2	3
1	Завантаження вихідних даних (файл метрик)	Дані завантажено успішно, відображено у графічному інтерфейсі

Продовження таблиці Д.1

1	2	3
2	Побудова лінійної регресійної моделі	Модель побудована без помилок, параметри визначено
3	Побудова нелінійної регресійної моделі	Модель побудована без помилок, параметри визначено
4	Тестування якості моделі	Якість моделі відповідає заявленим показникам та відображено у графічному інтерфейсі
5	Розрахунок інтервалів	Визначено довірчий і прогнозний інтервали
6	Оцінка якості	Значення якості класифіковані як висока, середня або низька якість відповідно до інтервалів

У результаті проведення випробувань «JavaShopMetrics» необхідно підтвердити відповідність функціональних і нефункціональних характеристик програмного забезпечення технічним вимогам.