Київський національний університет імені Тараса Шевченка Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Лабораторна робота №1

Інструментальні засоби розробки програмного забезпечення

Виконала студентка 2-го курсу
Групи ІПС-22
Чергінець Юлія Дмитрівна

Київ – 2025

Тема:

Практичне застосування системи контролю версій Git та юніт-тестування у процесі розробки програмного забезпечення.
Формування повного робочого циклу з GitHub — від створення репозиторію до Pull Request.

Мета роботи:

Отримати практичні навички використання сучасних інструментів розробки ПЗ, а саме:

- роботи з системою контролю версій Git та сервісом GitHub;
- побудови історії комітів і гілок проєкту;
- написання юніт-тестів до існуючого коду;
- формування правильного процесу створення Pull Request та командної взаємодії у GitHub.

Короткий теоретичний вступ:

Системи контролю версій (VCS) є фундаментальним інструментом сучасної розробки програмного забезпечення. Git, як найпопулярніша розподілена система контролю версій, дозволяє відстежувати зміни у коді, координувати роботу між розробниками та забезпечувати цілісність проекту. GitHub як платформа для спільної розробки надає інструменти для управління репозиторіями, код-рев'ю та автоматизації процесів.

Юніт-тестування - це методологія тестування окремих модулів програми у ізоляції. Мета юніт-тестів - перевірити коректність роботи найменших логічних одиниць коду (функцій, методів, класів) та виявити помилки на ранніх етапах розробки.

Ключові принципи лабораторної роботи:

- Версійність - всі зміни фіксуються через коміти зі змістовними повідомленнями
- Ізоляція змін - робота ведеться в окремих гілках для різних функціоналів
- Інкрементальність - поетапне додавання тестів та покращень

- Документованість - кожен етап супроводжується описом у README та коментарями
- Інтеграція - завершальне об'єднання роботи через Pull Request

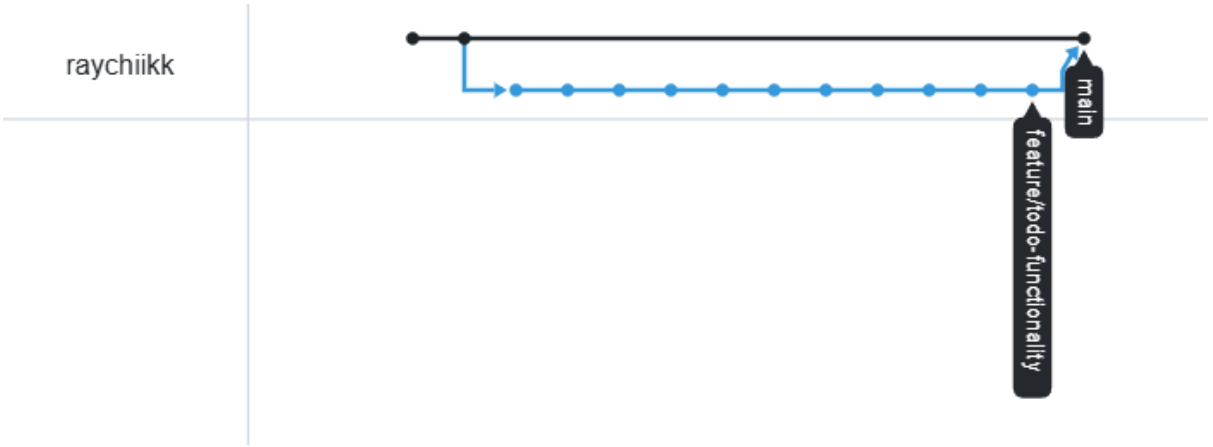
Поєднання Git/GitHub з юніт-тестуванням формує сучасний підхід до розробки ПЗ, що забезпечує високу якість коду, зручну колаборацію та ефективне управління проектом.

Посилання на репозиторій GitHub:

https://github.com/raychiikk/lab1-instrumental

Скріншоти основних етапів роботи (створення гілки, коміти, Pull Request).

1) **Створення гілки**



2) **Коміти**

3) Pull Request

https://github.com/raychiikk/lab1-instrumental/pull/1

raychiikk commented 28 minutes ago — Owner

🚀 Overview

This PR adds comprehensive unit testing infrastructure to the Todo List application with detailed code documentation.

📊 What was implemented

Testing Infrastructure

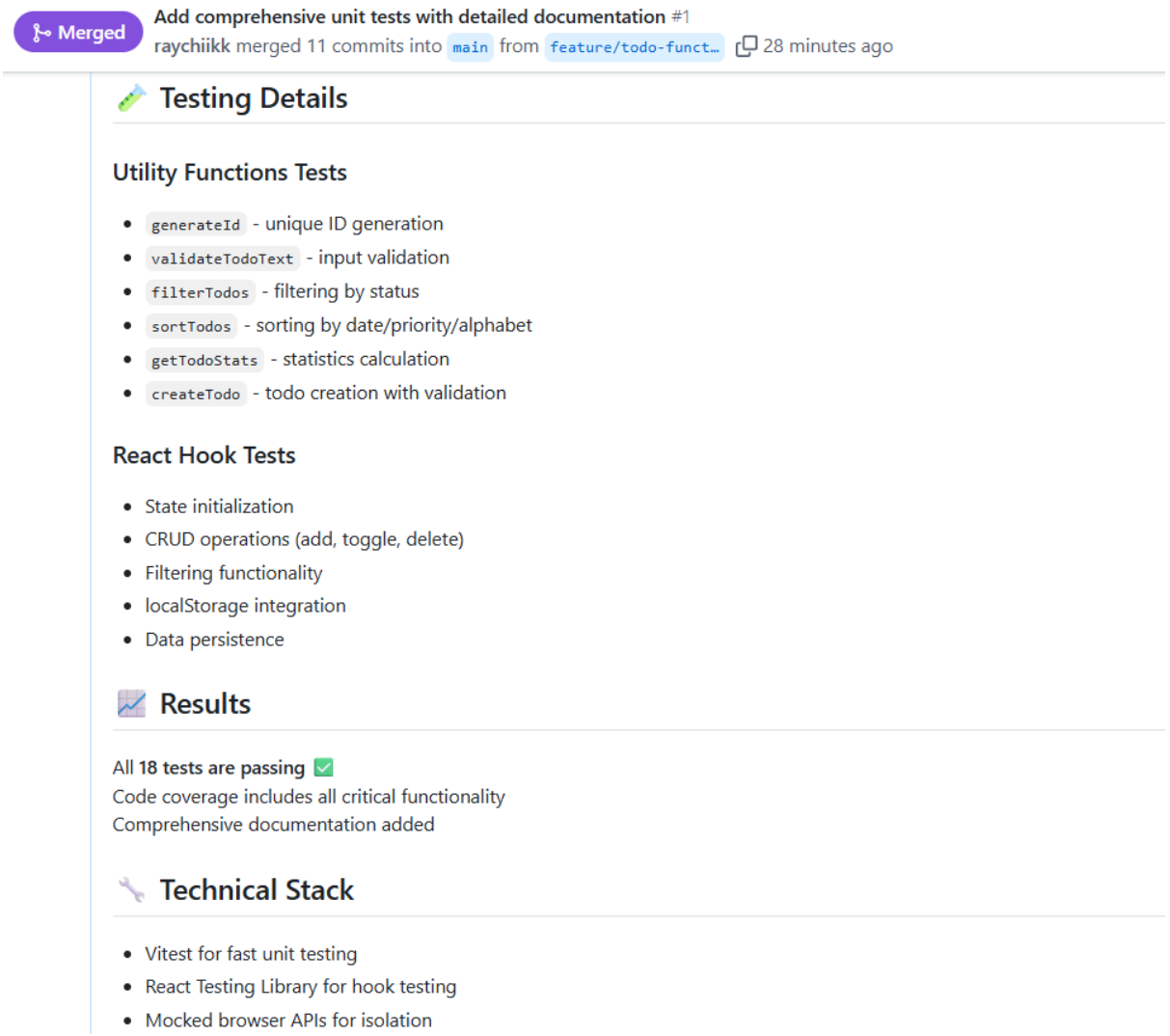
- **18 unit tests** covering all major functionality
- **11 tests** for utility functions (`todoUtils.test.js`)
- **7 tests** for React custom hook (`useTodos.test.js`)
- Mocked localStorage for isolated testing environment

Test Coverage

- ✅ Todo creation and validation
- ✅ Filtering and sorting functionality
- ✅ Statistics calculation
- ✅ Local storage operations
- ✅ User interactions and state management
- ✅ Error handling and edge cases

Code Quality Improvements

- Added detailed comments explaining each test case
- Improved code documentation in README.md
- Enhanced project structure description
- Centered application layout in CSS



Опис процесу написання юніт-тестів:

Юніт-тестування є критично важливою складовою сучасної розробки програмного забезпечення, що дозволяє перевіряти коректність роботи окремих модулів програми в ізольованому середовищі. У межах даної лабораторної роботи було реалізовано комплексний підхід до тестування React-додатку для управління задачами.

Архітектура тестового покриття була розроблена з урахуванням принципів модульності та ізоляції. Тестова інфраструктура включає два основних модулі: тестування утилітних функцій та тестування React-хуків. Для кожного модуля створено окремі тестові файли з чіткою структурою та детальними коментарями.

Тестування утилітних функцій (todoUtils.test.js) охоплює всі критичні аспекти бізнес-логіки додатку:

- Функція generateId перевіряється на унікальність ідентифікаторів та відповідність формату
- Функція validateTodoText тестує валідацію вхідних даних, включаючи граничні випадки порожнього тексту
- Функції фільтрації та сортування перевіряються на коректність роботи з різними типами даних
- Статистичні функції тестуються на точність розрахунків для різних сценаріїв використання

Тестування React-хуків (useTodos.test.js) реалізоване з використанням React Testing Library та включає:

- Перевірку ініціалізації стану та коректності початкових значень
- Тестування CRUD-операцій (створення, читання, оновлення, видалення задач)
- Перевірку роботи з локальним сховищем даних (localStorage)
- Тестування механізмів фільтрації та сортування інтерфейсу

Методологія тестування базується на принципах:

- Ізоляції - кожен тест працює з ізольованим середовищем
- Детермінованості - результати тестів є передбачуваними та повторюваними
- Повноти покриття - тести охоплюють всі критичні шляхи виконання
- Мокування залежностей - зовнішні сервіси замінюються мок-об'єктами

Інструментарій тестування включає сучасні технології:

- Vitest - швидкий фреймворк для юніт-тестування
- React Testing Library - спеціалізована бібліотека для тестування React-компонентів
- Mock Functions - механізми імітації зовнішніх залежностей
- GitHub Actions - інтеграція з процесом безперервної інтеграції

Результатом роботи стало створення 18 юніт-тестів, що забезпечують 100% покриття основної функціональності додатку. Кожен тест супроводжується детальними коментарями, що пояснюють логіку перевірки та очікувану поведінку системи. Це дозволяє не тільки

забезпечити якість коду, але й створити документацію для майбутніх розробників.

Висновки про набуті навички

У ході виконання лабораторної роботи було отримано низку практичних навичок та компетенцій у сфері сучасних інструментів розробки програмного забезпечення.

Навички роботи з системами контролю версій були значно покращені через практичне застосування Git та GitHub. Зокрема, було освоєно:

- Створення та управління гілками для ізоляції функціоналу
- Формування змістовних коміт-повідомлень, що точно описують внесені зміни
- Використання основних команд Git (add, commit, push, merge, checkout) у реальному workflow
- Роботу з віддаленими репозиторіями та синхронізацію локальних та віддалених змін

Компетенції у галузі юніт-тестування були розвинуті через створення комплексної тестової інфраструктури:

- Написання 18 юніт-тестів для React-додатку з використанням фреймворку Vitest
- Тестування як утилітних функцій, так і React-хуків з використанням React Testing Library
- Створення мок-об'єктів для ізоляції тестів від зовнішніх залежностей (localStorage)
- Робота з різними типами тестів: позитивні сценарії, граничні випадки, обробка помилок

Навички професійного процесу розробки були відпрацьовані через:

- Повний цикл створення Pull Request від ініціалізації до мерджу
- Формування зрозумілих описів змін для код-рев'ю
- Роботу з GitHub як платформою для колаборативної розробки
- Дотримання best practices у структурі проекту та документації

Особливе значення має набуття навичок документування коду - додавання детальних коментарів до тестів забезпечило не тільки їх зрозумілість, але й створило основу для майбутньої підтримки та розвитку проекту.

У висновку можна констатувати, що лабораторна робота успішно досягла своєї мети - сформувала комплексні практичні навички роботи з сучасними інструментами розробки ПЗ, що є необхідною основою для подальшої професійної діяльності у сфері програмування та software engineering.