Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Інструментальні засоби розробки програмного забезпечення

Лабораторна робота №1

Виконала студентка 2-го курсу

Групи ІПС-22

Порва Катерина 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.

**Теорія**

**Система контролю версій** – це система, що записує зміни у файл або набір файлів протягом певного часу, дозволяючи відновлювати попередні версії, порівнювати зміни та відстежувати історію роботи. Вона може бути централізованою або розподіленою, і особливо корисна для командної розробки, оскільки дозволяє кільком розробникам одночасно працювати над одним проєктом без конфліктів.

**Git** – це розподілена система керування версіями, створена для зручного відстеження змін у файлах, спільної роботи кількох розробників та надійного збереження історії проекту. Кожен учасник отримує повну локальну копію репозиторію з повною історією, тому робота залишається швидкою, стійкою до збоїв та гнучкою щодо організації процесів.

**GitHub** – це платформа, що поєднує можливості Git із зручним веб-інтерфейсом, яка дозволяє зберігати та керувати репозиторіями, відстежувати зміни в коді, працювати над проєктами в команді, вести документацію та обговорення, а також публікувати та розгортати проєкти.

**Юніт-тестування** – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. Основні принципи: ізоляція – тест перевіряє лише один елемент коду; повторюваність – тести можна запускати багаторазово з однаковим результатом; автоматизація – виконуються без ручного втручання.

**Посилання на репозиторій GitHub**

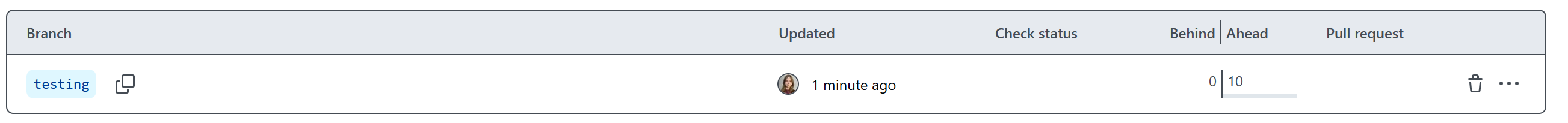
Для виконання лабораторної роботи я використала наявний власний проєкт. У межах нього я створила окрему гілку testing, де додала юніт-тести для існуючого коду, а також новий файл README.md, що містить короткий опис роботи та звіт по виконаних тестах. Усі юніт-тести зібрані в папці MiniBlogApp.Tests.

[KaterynaPorva/MiniBlogApp](https://github.com/KaterynaPorva/MiniBlogApp)

**Основні етапи роботи**

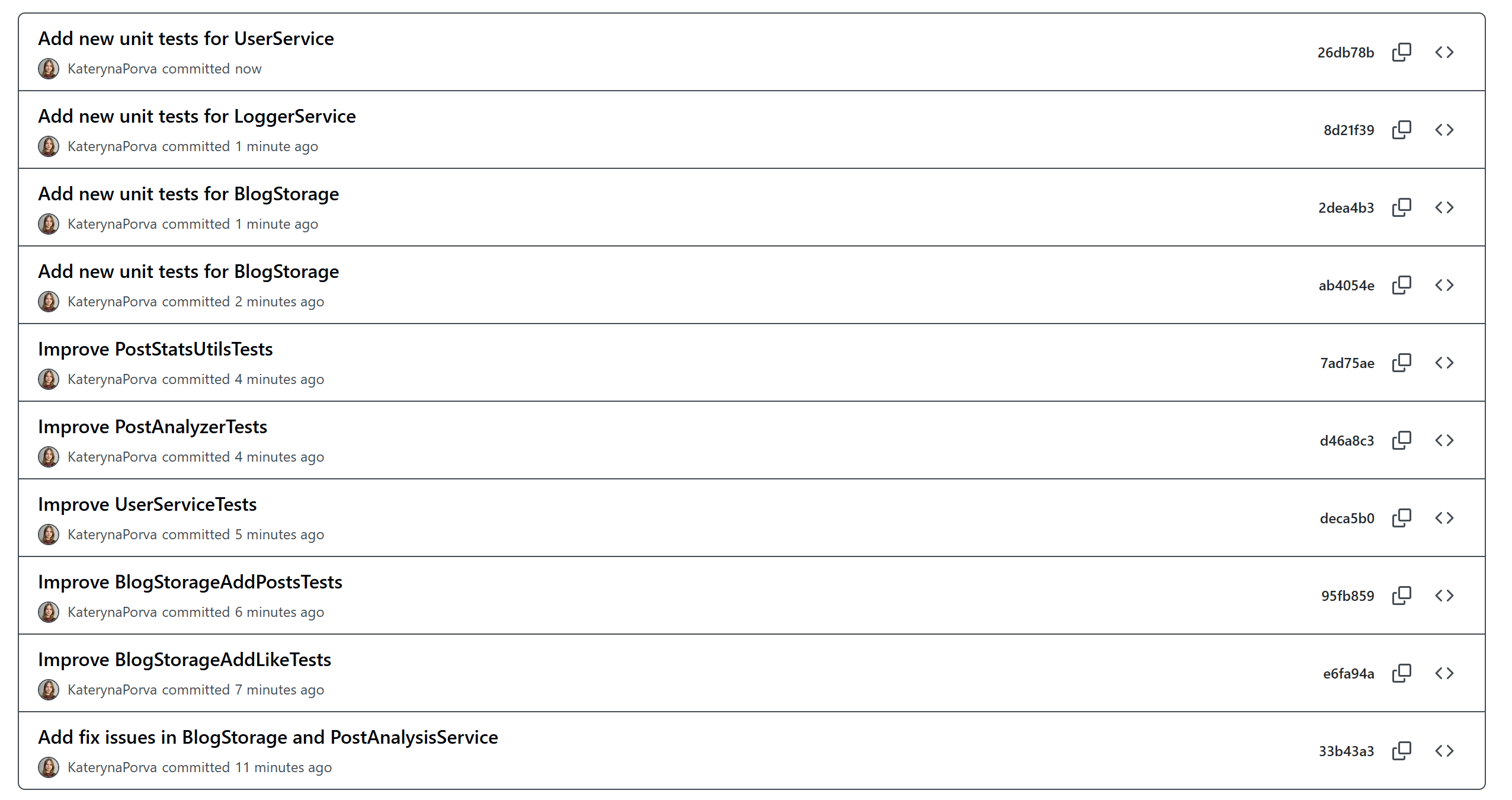
1. Створення гілки

Для юніт-тестування була створена окрема гілка testing

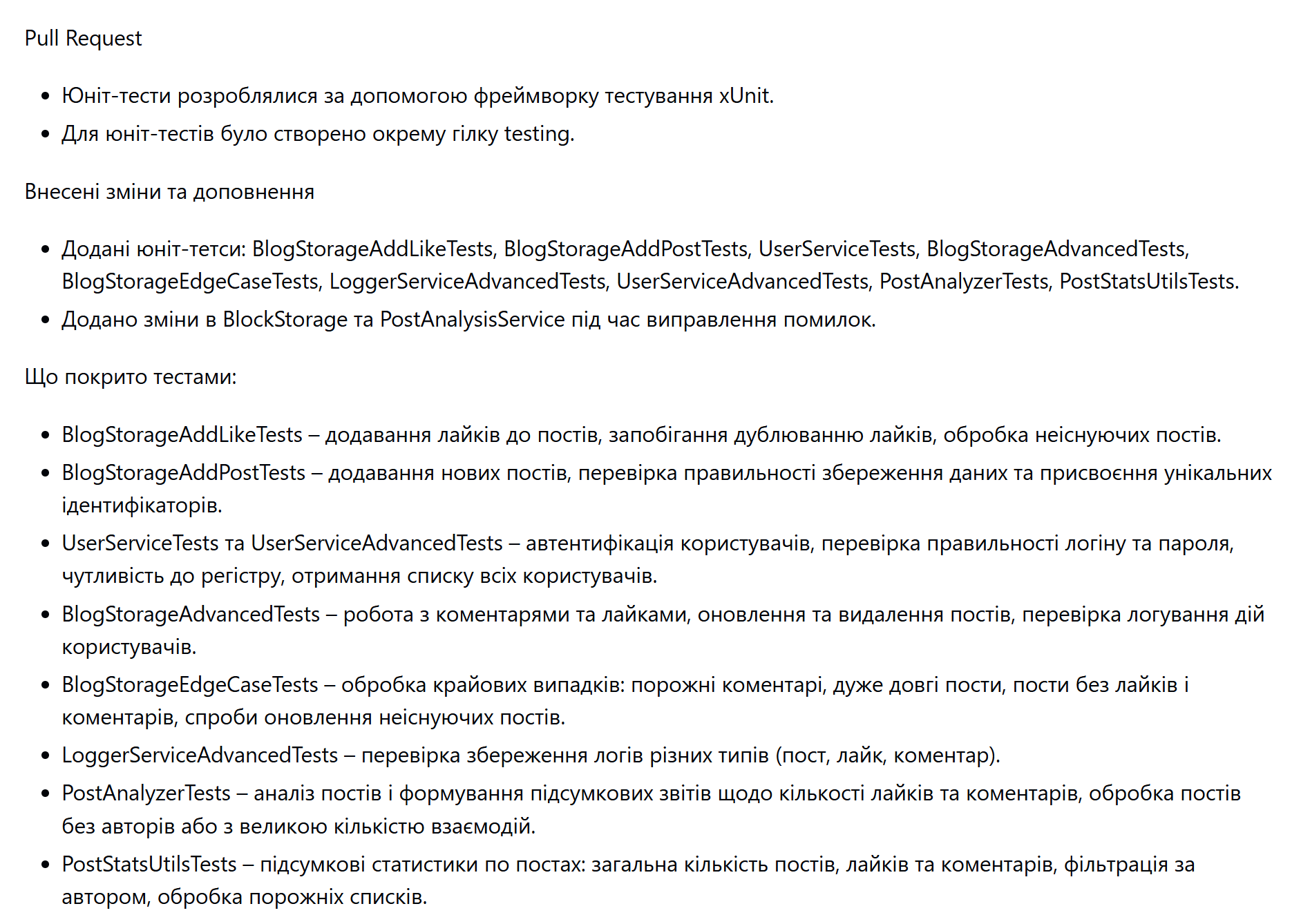


1. Історія комітів

Я послідовно додала окремо юніт-тести.



1. Pull Request



**Процес написання та опис юніт-тестів**

Було розпочато етап розробки юніт-тестів для перевірки коректності роботи ключових компонентів системи. Тестування здійснювалося з використанням бібліотеки xUnit, що забезпечило зручне структурування тестів і чіткий контроль очікуваних результатів. У процесі написання тестів було виявлено кілька недопрацювань у початковому коді - зокрема, неточності у перевірках існування постів, дублюванні лайків та обробці порожніх коментарів. Завдяки юніт-тестам основний код було доопрацьовано, що дозволило покращити стабільність роботи системи, підвищити її надійність і гарантувати коректну взаємодію між усіма сервісами застосунку.

1. **BlogStorageAddLikeTests** – перевіряє коректність роботи методу додавання лайків: унікальність дії користувача, неможливість додавання дубліката та поведінку при спробі лайкнути неіснуючий пост.
2. **BlogStorageAddPostTests** – тестує створення нового поста, перевіряючи правильність збереження автора, заголовка, контенту та збільшення загальної кількості постів.
3. **UserServiceTests** – охоплює базову автентифікацію користувачів: повернення коректного користувача при правильних даних, null при неправильних, а також отримання всіх користувачів системи.
4. **BlogStorageAdvancedTests** – відповідає за складні сценарії, зокрема додавання коментарів і лайків із логуванням, оновлення та видалення постів, перевірку унікальності їхніх ідентифікаторів і коректність записів у логах.
5. **BlogStorageEdgeCaseTests** – тестує граничні ситуації: додавання порожніх коментарів, реакцію на неіснуючі пости, роботу з великим обсягом контенту, перевірку нечутливості до регістру імен користувачів, а також поведінку при відсутності лайків чи коментарів.
6. **LoggerServiceAdvancedTests** – перевіряє роботу системи логування, зокрема правильне створення записів різних типів (створення поста, коментар, лайк) та ізольованість логів у спільному сховищі.
7. **UserServiceAdvancedTests** – зосереджується на додаткових аспектах автентифікації: чутливості до регістру, некоректних або порожніх паролях, а також наявності мінімального набору користувачів.
8. **PostAnalyzerTests** – тестує утиліту аналітики окремих постів, перевіряючи правильність формування підсумкового тексту, підрахунку лайків і коментарів, обробку null-полів і відсутності даних.
9. **PostStatsUtilsTests** – відповідає за статистичний аналіз усіх постів: підрахунок загальної кількості публікацій, лайків і коментарів, обробку фільтрації за автором і правильність роботи при порожніх або частково заповнених наборах даних.

**Висновок**

У ході виконання лабораторної роботи я здобула практичні навички роботи з системами контролю версій, розробки узагальнених класів та реалізації юніт-тестування. Я навчилася ефективно працювати з репозиторієм, створювати окремі гілки для реалізації функціоналу, фіксувати зміни за допомогою комітів і виконувати злиття через Pull Request.

Окрім цього, я закріпила навички тестування програмного забезпечення, перевіряючи роботу методів на стандартних і граничних випадках, що сприяло глибшому розумінню принципів надійності та якості коду.

**Використані джерела:**

* [Git - Про систему контролю версій](https://git-scm.com/book/uk/v2/%D0%92%D1%81%D1%82%D1%83%D0%BF-%D0%9F%D1%80%D0%BE-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%83-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8E-%D0%B2%D0%B5%D1%80%D1%81%D1%96%D0%B9)
* [Що таке Git — просте керівництво для початківців](https://fnx.com.ua/ua/articles/terms/chto-takoe-git-prostoe-rukovodstvo-dlya-nachinayushchih)
* [Модульне тестування — Вікіпедія](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
* [Що таке GitHub і як з ним працювати – GoIT Global](https://goit.global/ua/articles/shcho-take-github-i-iak-z-nym-pratsiuvaty/)