Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Інструментальні засоби розробки програмного забезпечення

Лабораторна робота

Виконала студентка 2-го курсу

Групи ІПС-22

Філоненко Дарина Дмитрівна

Київ - 2025

**Тема:** Використання системи контролю версій Git та написання юніт-тестів для забезпечення якості коду.

**Мета:** Набуття практичних навичок роботи з Git у повному циклі розробки (створення гілок, комітів, Pull Request) та освоєння методології тестування програмного забезпечення за допомогою юніт-тестів на мові Python для покращення надійності коду.

**Теорія:** *Git* — це розподілена система контролю версій, що дозволяє розробникам відстежувати зміни у файлах проєкту, повертатися до попередніх станів та ефективно співпрацювати. Ключовою концепцією є гілки (branches), які ізолюють розробку нових функцій, не впливаючи на стабільну версію коду в основній гілці (main).

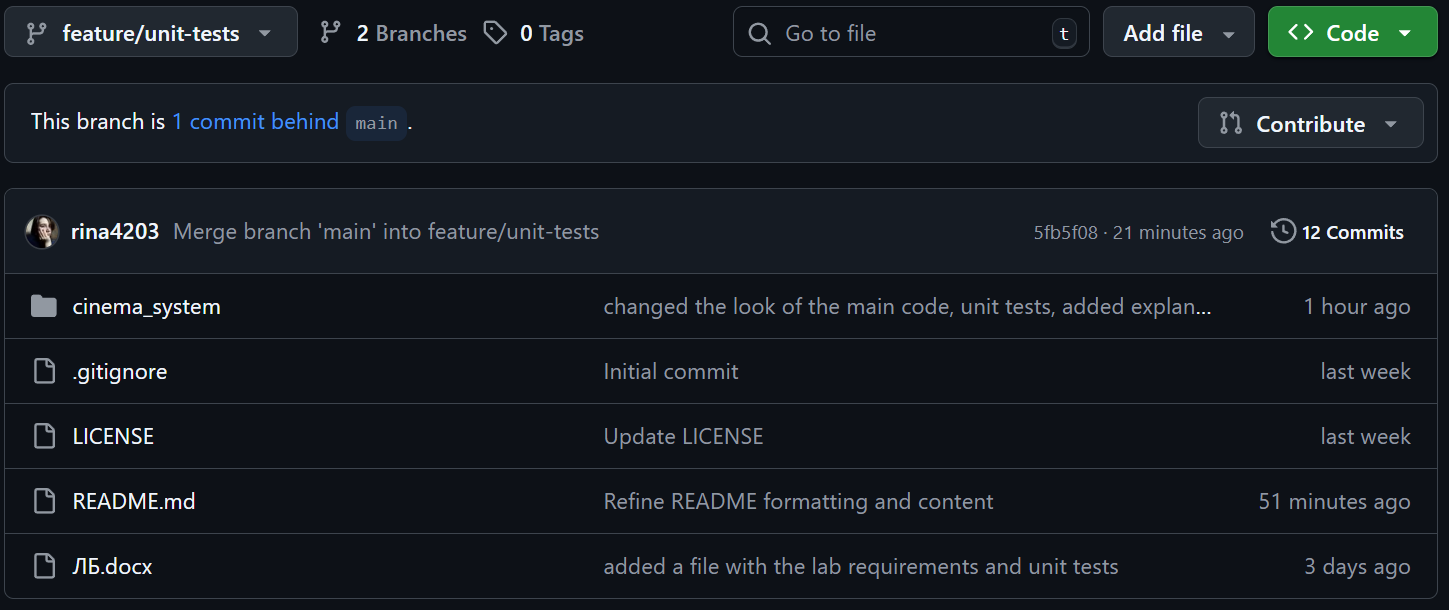
*Юніт-тестування* — це процес перевірки найменших ізольованих частин коду (функцій, методів), щоб переконатися, що вони працюють коректно. Це дозволяє виявляти помилки на ранніх етапах, спрощує внесення змін у майбутньому та служить живою документацією до коду.

[Посилання на репозиторій GitHub](https://github.com/rina4203/unit-testing-laba.git)

**Скріншоти основних етапів роботи**

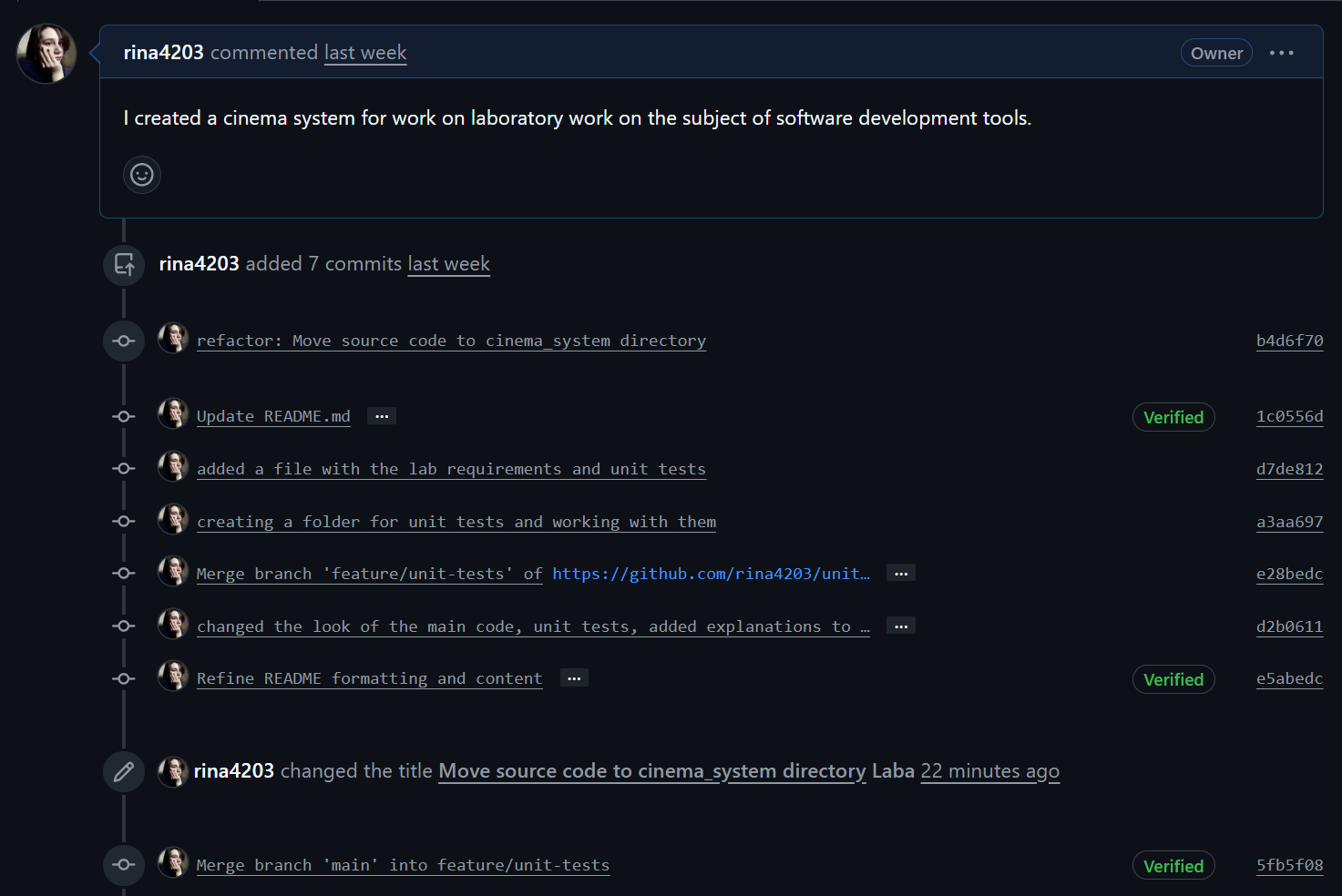
1. Створення нової гілки feature/unit-tests

Робота над новою функціональністю (написанням тестів) велася в окремій гілці, щоб ізолювати її від основної гілки main.



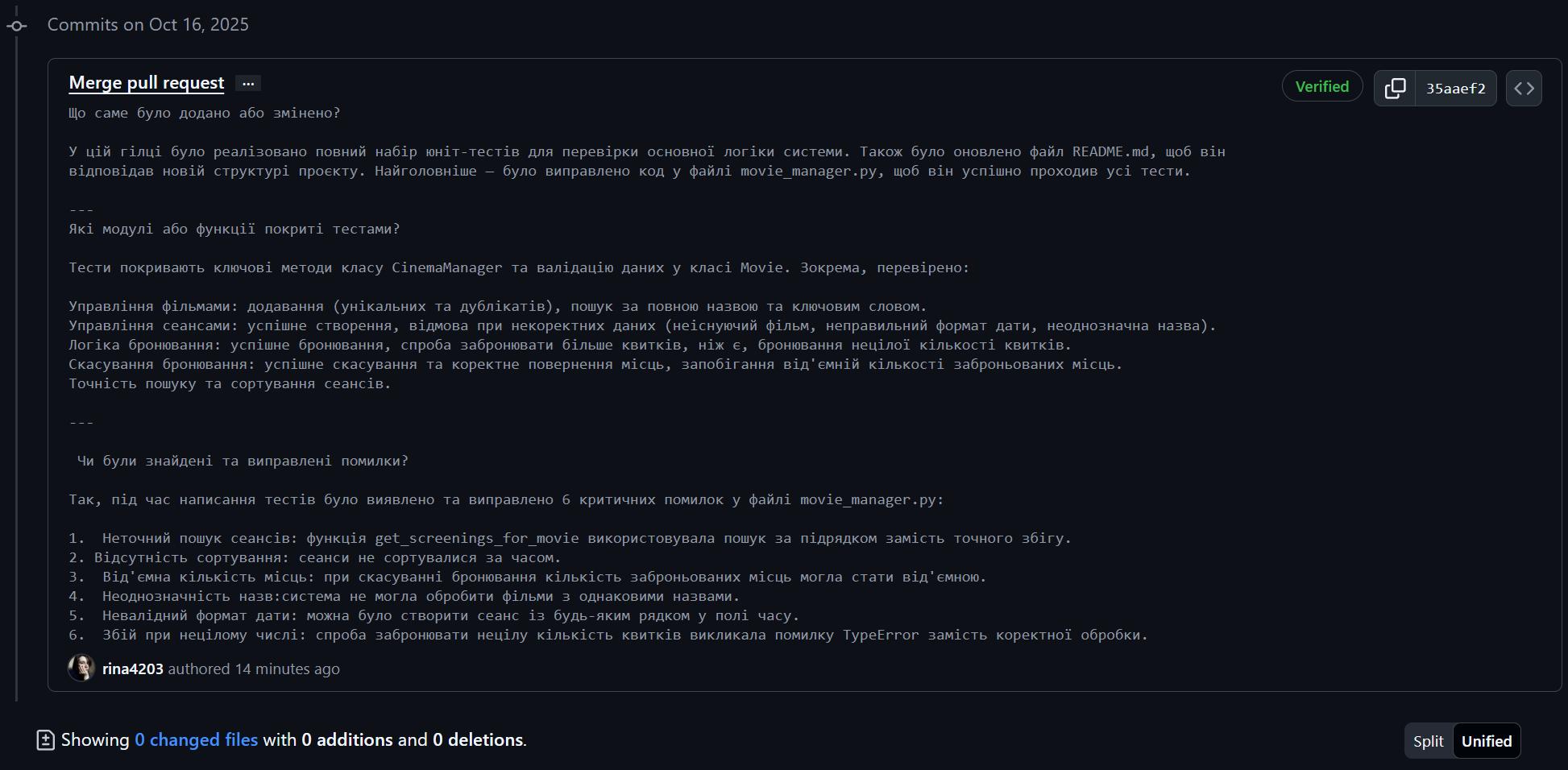
2. Серія логічних комітів

Усі зміни зберігалися через невеликі, логічні коміти з описовими повідомленнями, що відображають прогрес роботи.



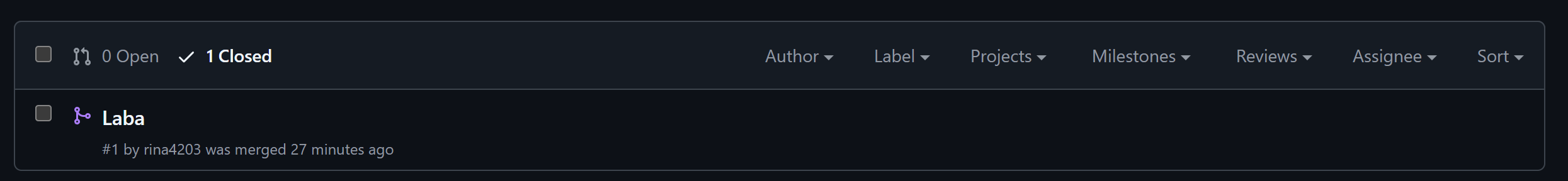
3. Створення Pull Request для перевірки та злиття

Після завершення роботи було створено Pull Request для злиття гілки feature/unit-tests у main. В описі було детально вказано, які зміни було внесено та які помилки виправлено.



4. Успішне злиття (Merge) Pull Request

Після перевірки коду та вирішення конфліктів, [Pull Request](https://github.com/rina4203/unit-testing-laba/pull/1) було успішно злито, а робоча гілка — видалена.



**Опис процесу написання юніт-тестів**

Для тестування було використано вбудований фреймворк Python unittest.

1. Структура тестів:

Було створено клас TestCinemaManagerBugs, що успадковує unittest.TestCase. У методі setUp створювався екземпляр класу CinemaManager для кожного тесту, що забезпечувало їхню незалежність.

2. Підхід до тестування:

Спочатку писався тест, який перевіряв певну функціональність або потенційну помилку. Цей тест очікувано падав (Red). Далі вносилися зміни в основний код (movie\_manager.py) для виправлення помилки, після чого тести запускалися знову доти, доки вони не починали проходити (Green).

3. Приклади виявлених помилок(детальніше в [цьому документі](https://github.com/rina4203/unit-testing-laba/blob/feature/unit-tests/cinema_system/tests/unit%20tests(discribe).docx)):

*Неточний пошук:* Тест test\_get\_screenings\_for\_movie\_uses\_substring виявив, що пошук сеансів за назвою "Батько" також повертав "Хрещений батько". Це було виправлено заміною пошуку за підрядком на точне порівняння.

*Відсутність сортування:* Тест test\_screenings\_are\_not\_sorted\_chronologically показав, що сеанси поверталися не в хронологічному порядку. Було додано сортування результатів за часом.

*Обробка некоректних даних:* Було додано тести, які перевіряли реакцію системи на невалідні дані (наприклад, спроба забронювати "два" квитки). Код було змінено так, щоб він не падав з помилкою TypeError, а коректно обробляв ситуацію, повертаючи None.

**Висновки про набуті навички**

Під час виконання лабораторної роботи я набула таких практичних навичок:

Освоїла нелінійний робочий процес за допомогою feature-гілок, що дозволяє безпечно розробляти нові функції.

Навчилась створювати інформативні коміти для відстеження історії змін.

Зрозуміла призначення Pull Request як інструменту для перевірки коду (code review) та командної взаємодії.

Отримала практичний досвід у вирішенні конфліктів злиття (merge conflicts), що є невід'ємною частиною роботи в команді.

**Написання юніт-тестів:**

Навчилась писати тести, що перевіряють не тільки успішні сценарії, але й крайні випадки (edge cases) та обробку помилок.

В результаті, я освоївла фундаментальні інструменти та практики, які використовуються в професійній розробці програмного забезпечення для забезпечення високої якості коду та ефективної командної роботи.