Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра систем автоматизованого проектування



Звіт

До лабораторної роботи №1

З дисципліни “Управління ІТ-проектами”

На тему “Шаблон проектування MVC”

**Виконав:**

cтуд. групи КН-408

Пагута В.О.

**Прийняв:**

Василишин Б. С.

Львів – 2024

**Мета роботи:** ознайомитись з системами контролю версій. Зрозуміти принципи синхронізації робочих груп у гнучких командах. Набути навичок використання технічних засобів та протоколів для роботи з системами контролю версій.

**Короткі теоретичні відомості**

GIT Git (вимов. «гіт») – розподілена система керування версіями файлів. Проект був створений Лінусом Торвальдсом для управління розробкою ядра Linux. На сьогоднішній день підтримується Джуном Хама. Система спроектована як набір програм, спеціально розроблених з урахуванням їх використання в скриптах. Це дозволяє зручно створювати спеціалізовані системи контролю версій на базі Git або користувацькі інтерфейси. Наприклад, Cogito є саме таким прикладом фронтенда до репозиторіїв Git, а StGit використовує Git для управління колекцією патчів . 9 Git підтримує швидке розділення і злиття версій, включає інструменти для візуалізації та навігації по нелінійній історії розробки. Git надає кожному розробнику локальну копію всієї історії розробки; зміни копіюються з одного репозиторію в іншій. Віддалений доступ до репозиторіїв Git забезпечується git-daemon, gitosis, SSH- або HTTPсервером. TCP-сервіс git-daemon входить в дистрибутив Git і є разом з SSH найбільш поширеним і надійним протоколом доступу. Метод доступу по HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використовувати існуючі конфігурації мережевих фільтрів.

**Лабораторне завдання**

1. Ознайомитись з принципами роботи систем контролю версій.

2. Отримати індивідуальне завдання у викладача.

3. Написати програму згідно з індивідуальним завданням.

4. Створити репозиторій та завантажити туди свій програмний код.

5. Внести зміни в програму, при необхідності провести вирішення конфліктів.

**Вимоги до програми**

7. Програму, яка виконує частотний аналіз тексту: визначає усі слова, які зустрічаються в ньому і кількість їх входження. Результати виводить у вікно графічного інтерфейсу.

**Реалізація програми**

import random

import string

def replace\_odd\_digits\_with\_random\_letter(text):

    result = []

    for c in text:

        if c.isdigit() and int(c) % 2 != 0:

            random\_letter = random.choice(string.ascii\_letters)

            result.append(random\_letter)

        else:

            result.append(c)

    return ''.join(result)

def count\_digits(text):

    return sum(c.isdigit() for c in text)

def count\_digits\_in\_file\_and\_replace\_odds(file\_path):

    try:

        with open(file\_path, 'r', encoding='utf-8') as file:

            text = file.read()

        # Рахуємо кількість цифр до зміни

        digit\_count\_before = count\_digits(text)

        # Замінюємо непарні числа на випадкові літери

        modified\_text = replace\_odd\_digits\_with\_random\_letter(text)

        # Рахуємо кількість цифр після зміни

        digit\_count\_after = count\_digits(modified\_text)

        # Записуємо змінений текст назад у файл

        with open(file\_path, 'w', encoding='utf-8') as file:

            file.write(modified\_text)

        return digit\_count\_before, digit\_count\_after

    except FileNotFoundError:

        print(f"Файл {file\_path} не знайдено.")

        return 0, 0

# Вказати шлях до вашого файлу

file\_path = 'text.txt'

digit\_count\_before, digit\_count\_after = count\_digits\_in\_file\_and\_replace\_odds(file\_path)

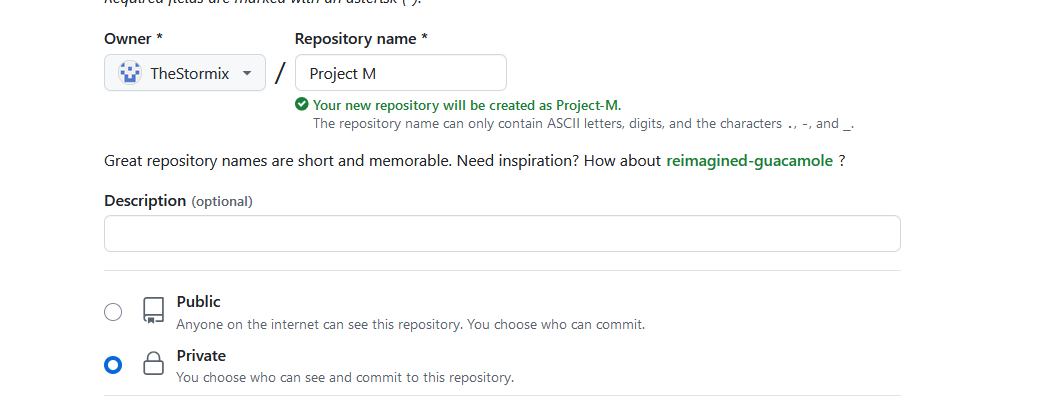
print(f"Кількість цифр у файлі до змін: {digit\_count\_before}")

print(f"Кількість цифр у файлі після змін: {digit\_count\_after}")

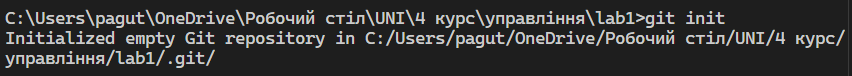
**Результат виконання**



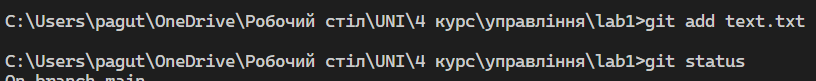
Після чого створюю GitHub репозиторій



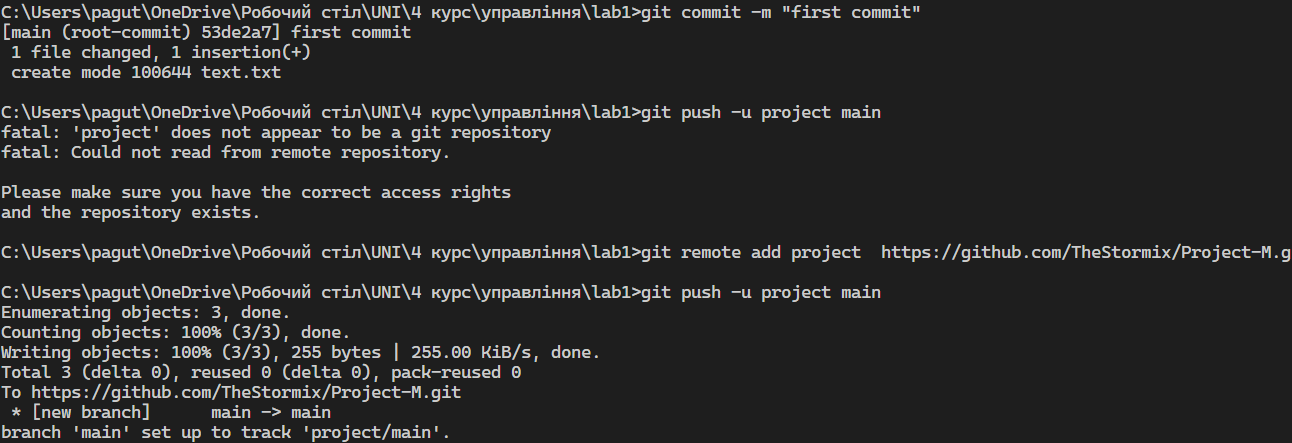
Ініціалізую локальний репозиторій



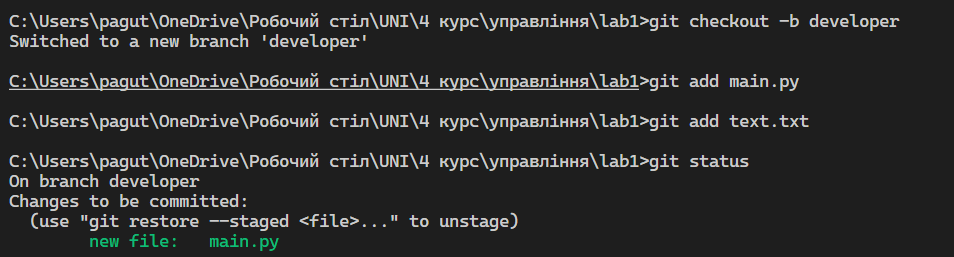
Створюю файл та роблю перший коміт.

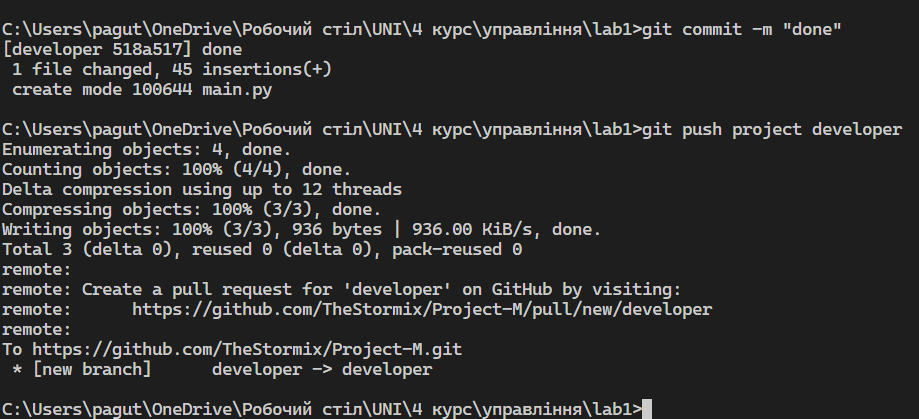


Виконую команду push на віддалений репозиторій.

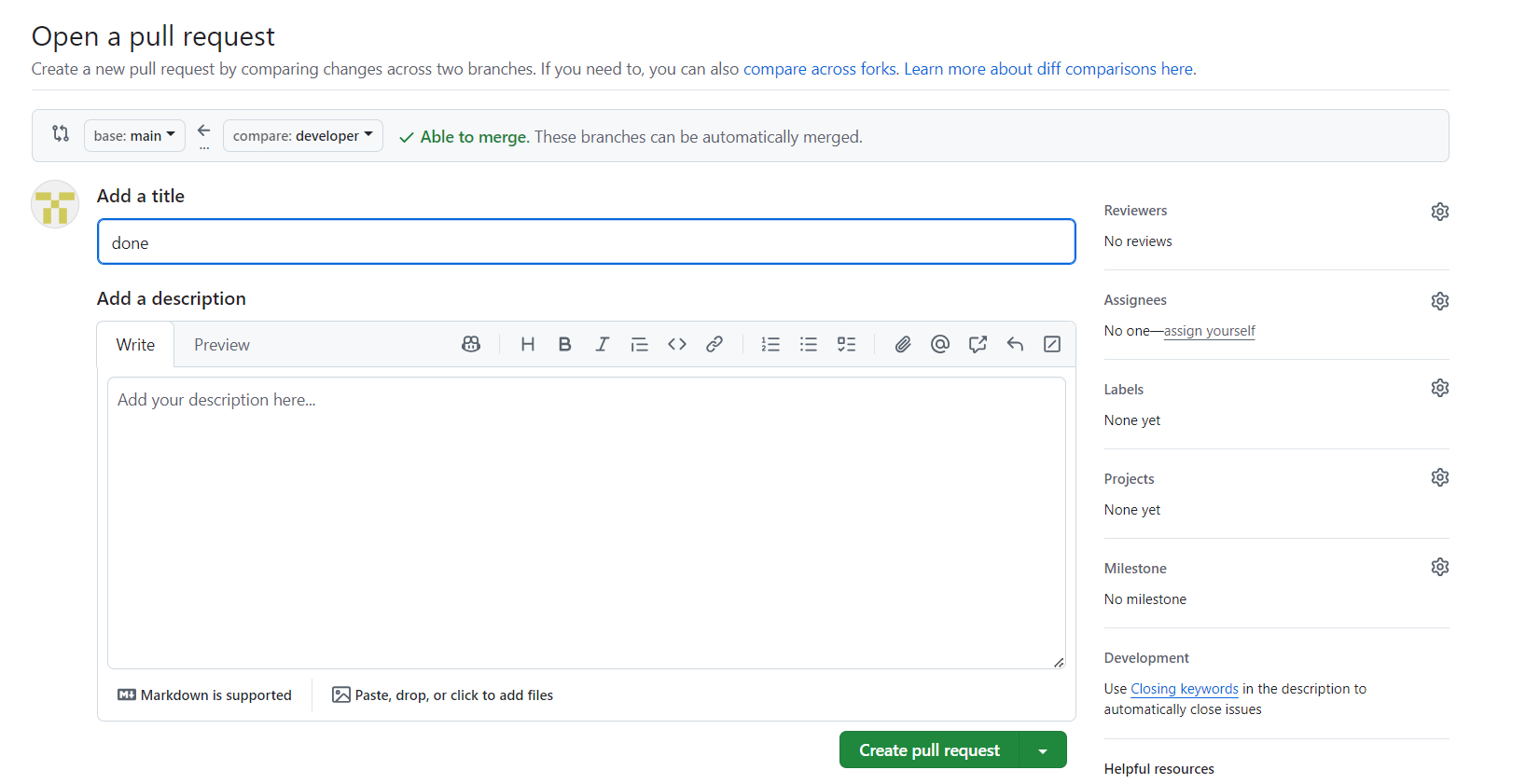


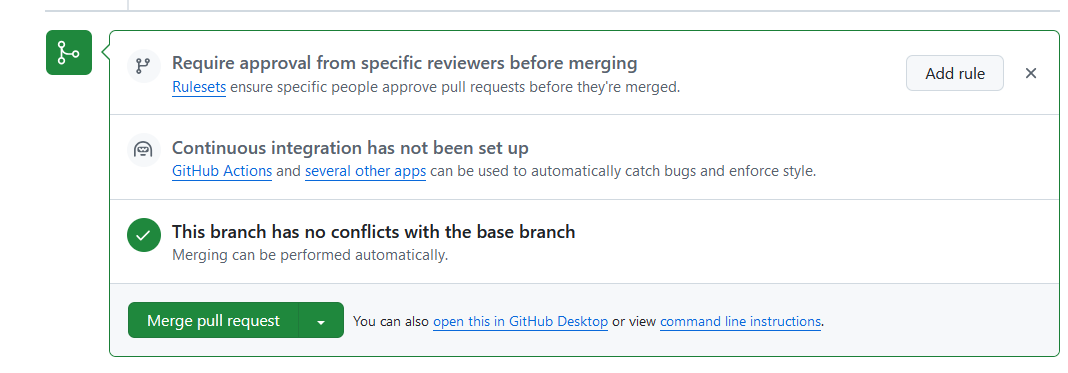
Створюю нову гілку “developer” та комічу туди свій код і пушу його на віддалений репозиторій.



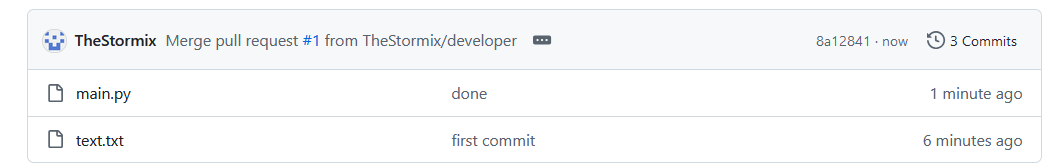


Створюю пул реквест для того, щоб змерджити зміни в гілку main.





Після вдалого мерджа отримав оновлену гілку main



**Висновок:** Отже, на даній лабораторній роботі я ознайомився з системами контролю версій. Зрозумів принципи синхронізації робочих груп у гнучких командах. Набув навичок використання технічних засобів та протоколів для роботи з системами контролю версій.