МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра системного аналізу та управління

Звіт

з практичної роботи №1 з дисципліни

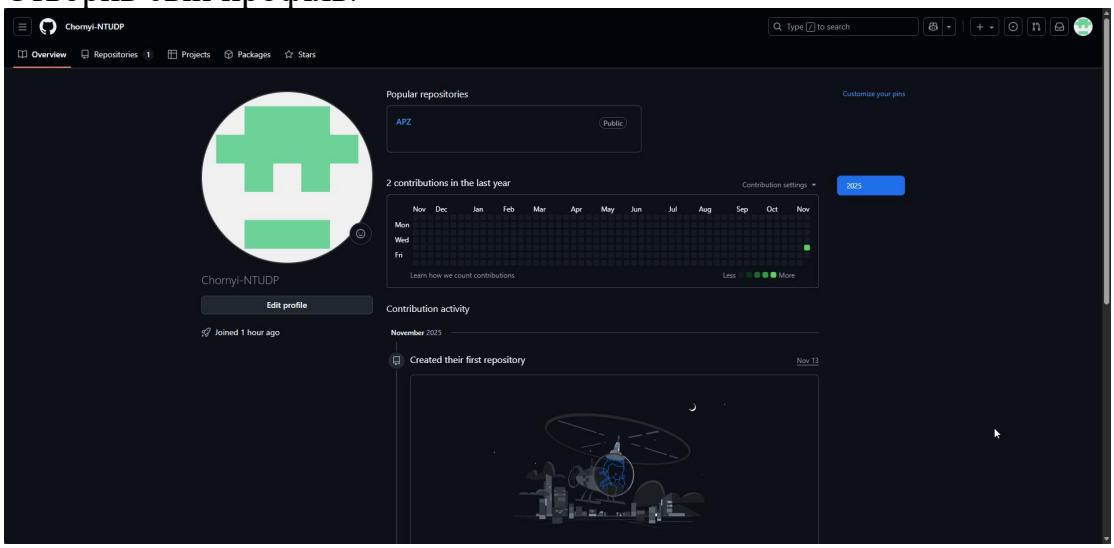
«Аналіз програмного забезпечення»

Виконала:
студент групи 121-22-2
Чорний М.Ю.
Перевірили:
доц. Мінєєв О.С.
ас. Шевченко Ю.О.

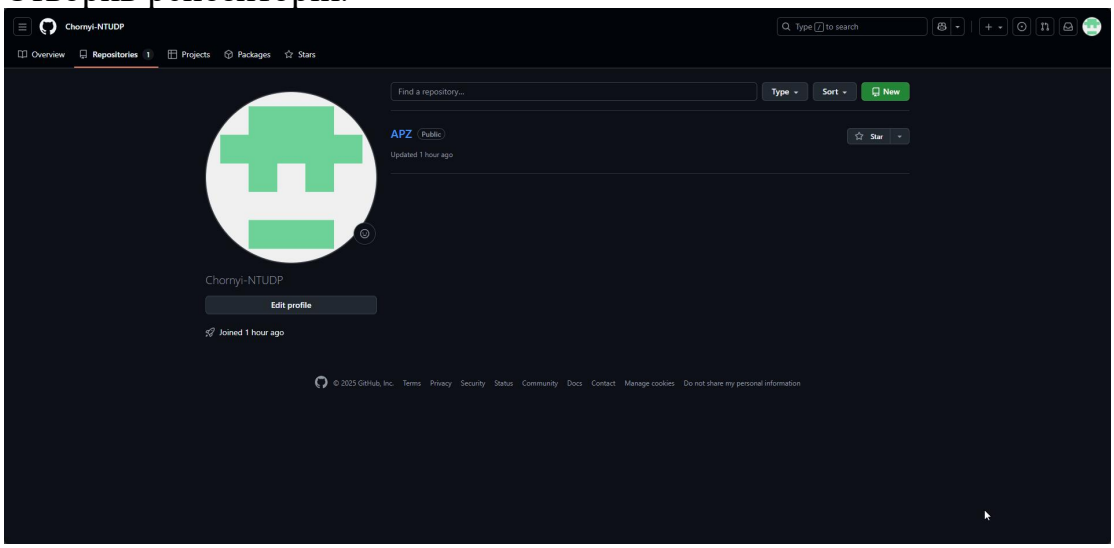
Дніпро
2025

Тема: Створення і налаштування профілю у системі Git.
Мета: Набування навичок при реєстрації та налаштуваню облікового запису (account) на хостінгу GitHub.

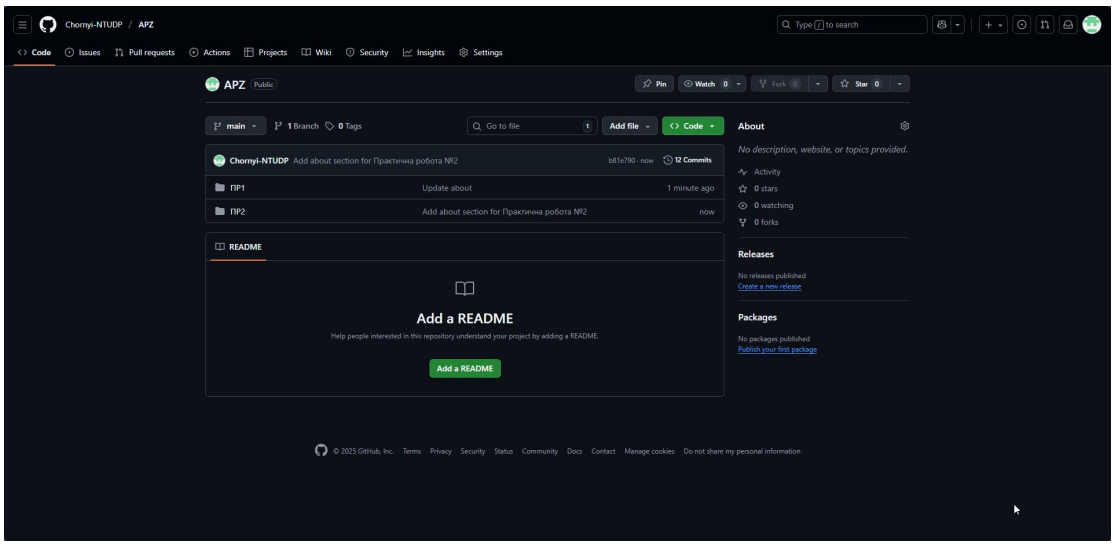
Виконання:
Створив свій профіль.



Створив репозиторій.



Створив папки з практичними роботами.



Контрольні запитання:
1. Що таке GIT?
GIT (Global Information Tracker) - це розподілена (децентралізована) система контролю версій (DVCS). Її фундаментальне призначення - це відстеження змін у наборі файлів, найчастіше - у вихідному коді програмних проектів. Система дозволяє фіксувати стан проекту у вигляді дискретних "знімків" (комітів), переглядати повну історію модифікацій, повертатися до будь-якої попередньої версії та, що ключове, забезпечувати ефективну паралельну роботу декількох розробників над одним проектом.

2. Що таке репозиторій у GIT?
Репозиторій (repository) в термінології GIT - це структура даних, що інкапсулює весь набір файлів проекту, а також повну історію їхніх змін та метадані. Технічно, це директорія проекту, яка містить приховану піддиректорію .git, де зберігається вся база даних системи контролю версій. Репозиторії бувають локальними (знаходяться безпосередньо на робочій станції розробника) та віддаленими (розміщені на сервері, наприклад, на платформах GitHub або GitLab, і слугують для синхронізації та спільної роботи).

3. Які переваги використання GIT?

Використання GIT надає низку суттєвих переваг у процесі розробки:
- Відстеження історії та цілісність даних: Кожна зміна фіксується, що дозволяє детально аналізувати історію та повертатися до стабільних станів.
- Нелінійна розробка (галуження): Система дозволяє створювати ізольовані "гілки" (branches) для розробки нових функціональностей або виправлення помилок, не впливаючи на основну кодову базу. Ці гілки згодом можуть бути безпечно інтегровані (злиті) назад.
- Розподілена архітектура: Кожен учасник проекту має повну локальну копію репозиторію, що уможливлює автономну роботу та підвищує надійність збереження даних.

- Ефективність командної роботи: GIT надає потужні механізми для злиття (merging) змін від різних розробників та інструменти для вирішення конфліктів.

4. Яка мова використовується в GIT?

Питання стосується двох аспектів. Саме ядро системи GIT написано переважно мовою програмування C для забезпечення максимальної швидкодії та крос-платформенності. Окрім ядра, багато допоміжних компонентів та скриптів написані з використанням Shell (зокрема Bash) та інших скриптових мов. Взаємодія ж користувача з системою відбувається через інтерфейс командного рядка (CLI) за допомогою команд, що інтерпретуються системною оболонкою (Shell).

5. Як можна створити репозиторій у Git?
Існує два основних методи для створення репозиторію:

- Ініціалізація нового репозиторію: Якщо проект вже існує локально, але ще не перебуває під контролем версій, використовується команда git init. Вона виконується в кореневій директорії проекту і створює новий локальний репозиторій (папку .git).
- Клонування існуючого репозиторію: Якщо репозиторій вже існує (зазвичай, на віддаленому сервері), його повна копія створюється локально за допомогою команди git clone <URL_репозиторію>.

6. Яка команда використовується для видалення гілки?
Для видалення локальної гілки використовується команда git branch -d <назва_гілки>. Цей прапор (-d) є "безпечним", оскільки GIT виконає перевірку, чи всі зміни з цієї гілки були повністю інтегровані (злиті) в іншу гілку. Якщо гілка містить неінтегровані зміни, для її примусового видалення необхідно використати прапор з великої літери: git branch -D <назва_гілки>.

7. Що таке контроль версій GIT?
Контроль версій - це систематичний процес управління та відстеження змін у документах, програмному коді чи інших наборах інформації. GIT, як система контролю версій (VCS), надає інструментарій для реалізації цього процесу. Вона дозволяє фіксувати дискретні стани проекту (коміти), переглядати відмінності між будь-якими двома версіями, повертатися до будь-якого збереженого стану та ефективно координувати одночасні зміни, що вносяться кількома учасниками.

8. Як можна виправити несправний комміт?
Метод виправлення залежить від того, чи був комміт опублікований у віддаленому репозиторії:

- Якщо комміт останній і не опублікований (не "push"-нутий): Найпростіший спосіб - це git commit --amend. Ця команда дозволяє додати нові зміни до

попереднього коміту або просто змінити його опис, фактично "перезаписуючи" його.

- Якщо комміт вже опублікований або знаходиться в глибині історії: Зміна опублікованої історії є поганою практикою. Натомість використовується "скасовуючий" комміт. Команда git revert <ID_коміту> створює новий коміт, який застосовує зміни, що є протилежними до вказаного "несправного" коміту, таким чином нівелюючи його ефект, але зберігаючи історію прозорою.

9. Як ви дізнаєтесь у GIT, чи гілку вже об’єднано в master?
Для верифікації, чи була певна гілка повністю інтегрована (злита) в гілку master, необхідно виконати наступну послідовність дій:

1) Переключитися на цільову гілку (в даному випадку master): git checkout master.
2) Виконати команду git branch --merged. Ця команда виведе список усіх гілок, чиї зміни вже повністю включені до поточного стану гілки master. Якщо шукана гілка присутня в цьому списку, це означає, що вона була успішно об'єднана.