Kyiv College of Communications

PERFORMANCE REPORT

Work Case №2

Discipline: "Operating Systems"

Topic: Git

Performed by students of Group CSN-33 *(Computer Systems and Networks)*.:

Kuznetsov Artur Serhiyovych

Finchuk Alina Oleksiivna

Checked by:

Sushanova V.S.

Kyiv – 2025

**Task 1** Finchuk Alina Oleksiivna

1. Опишіть для чого використовують git, які основні дії та команди в ньому виконують.

What is Git used for?

Git is a distributed version control system (VCS) used to track changes in files (usually source code) and coordinate work on projects between multiple developers. It allows you to:

* Keep a history of file changes (snapshots/commits).
* Roll back changes in case of errors.
* Work on multiple versions of a project simultaneously (branches).
* Enable collaboration by synchronizing changes between participants.
* Guarantee data integrity and performance.

Git is suitable not only for programming, but also for managing any files where changes need to be tracked.

**Basic actions and commands in Git**

1. *Initializing a repository*

Creating a new Git repository to track files in a folder.

Command: git init

Creates a .git folder with metadata for version control.

2. *Adding files to the index (preparing for commit)*

Selecting files whose changes will be saved in the next commit.

Command: git add <file> or git add . (for all files)

Prepares changes for commit.

3*. Creating a commit*

Saving a snapshot of the current state of the files in the repository with a description of the changes.

Command: git commit -m “Commit message”

Commits the changes with a brief description.

4. *Checking the repository status*

Shows which files have been modified, added to the index, or are untracked.

Command: git status

Helps you understand the current state of the project.

5*. Viewing commit history*

Displays a list of all commits with their authors and messages.

Command: git log

Allows you to study the history of changes (exit the log with the `q` key).

6*. Viewing changes*

Shows the differences between the current files and the last commit.

Command: git diff

Used before `git add` to see what has changed.

*7. Working with branches*

Allows you to create parallel versions of the project for experiments or new features.

Commands:

* `git branch <branch\_name>` — create a new branch.
* `git checkout <branch\_name>` — switch to a branch.
* `git merge <branch\_name>` — merge branches.

8. *Working with remote repositories*

Synchronize your local repository with a remote one (for example, on GitHub).

Commands:

* `git clone <URL>` — copy a remote repository.
* `git push` — sending changes to a remote repository.
* `git pull` — downloading and merging changes from a remote repository.

9. *Ignoring files*

Excluding files from tracking (e.g., passwords, temporary files).

File: gitignore

Specifies files or folders that Git should not track.

10. *ntegrity check*

Checking the repository for errors or corruption.

Command*:* git fsck

Checks the integrity of the data.

**Key features of Git**

* *Distributed*: each developer has a complete copy of the repository with history.
* *Speed*: fast operations such as commits and merges.
* *Flexibility*: Support for non-linear workflows (multiple branches).
* *Security*: Protection against accidental or malicious data corruption.

These commands and actions cover the basic scenarios for using Git for version control and collaboration.

**Task 2** Kuznetsov Artur Serhiyovych

2. Що таке "комміт", як він дозволяє відслідковувати зміни у файлах?

Комміт (англ. commit) — is the fundamental unit of change management in the Git version control system that underpins GitHub. When a user makes changes to files in a repository, they create a commit that records those changes along with metadata: author, date, commit message, and a unique hash. This allows you to track exactly what was changed, by whom, and when.

Commits create a chronological history of a project's development. Each commit is linked to the previous one, forming a chain that can be viewed, analyzed, compared, or even reverted. It is through commits that GitHub provides the ability to see the evolution of code, detect errors, restore previous versions, and collaborate with other developers without losing context.

GitHub, as a platform, has integrated commit support since its launch on April 10, 2008, as it is based on Git, a system created by Linus Torvalds in 2005. Commits were not just a part of GitHub—they were its heart. Over the years, GitHub has improved the commit viewing interface, adding new features for convenience.

For example, in March 2025, GitHub introduced an [updated commit viewer](https://github.blog/changelog/2025-03-12-refreshed-commit-details-page-now-generally-available/) page that allows you to:

1) see comments on changes directly in the file tree;

2) switch between different viewing modes (unified or split) without reloading the page;

3) use the keyboard to navigate through changed lines;

4)filter changes by filename or extension;

5) view the full commit message without hiding.

These updates make the process of reviewing changes even more intuitive, especially for large teams working on complex projects. A commit in GitHub is not just a record of changes, it is a tool for accountability and collaboration that allows each participant to see the contributions of others and understand the logic of code development.