WORK CASE #1

Міньков Ілля / Колотуша Микола

1.Git is a distributed version control system used for tracking changes in source code and collaborating on projects in a team. Its primary purpose is to store, track, and manage versions of files in a repository. Below are some key actions and commands you perform in Git:

Repository Initialization:

git init: Creates a new repository in the current directory.

Repository Cloning:

git clone <URL>: Clones a remote repository from the specified URL to the local computer.

Saving Changes:

git add <file>: Adds a file to the staging area for a future commit.

git add . or git add -A: Adds all modified or new files to the staging area.

git commit -m "Commit message": Saves the changes in the staging area with a comment.

Tracking Repository Status:

git status: Displays information about the repository's status, including uncommitted changes.

Viewing Commit History:

git log: Lists the commits in the repository with their hashes, authors, and comments.

Working with Branches:

git branch: Shows a list of local branches.

git branch <branch\_name>: Creates a new branch.

git checkout <branch\_name>: Switches to another branch.

git merge <branch\_name>: Merges changes from another branch into the current one.

git push origin <branch\_name>: Uploads a local branch to a remote server.

Updating and Synchronizing with Remote Repository:

git pull: Fetches changes from a remote repository and merges them into the current branch.

git push: Sends local changes to a remote server.

Deleting and Renaming Files:

git rm <file>: Removes a file from the repository.

git mv <old\_name> <new\_name>: Renames a file.

Fixing Mistakes in Commits:

Note: Fixing mistakes in commits can involve more advanced Git techniques like amending commits, squashing, or rebasing, which require a deeper understanding of Git.

This summary provides an overview of common Git commands and actions used for version control and collaborative development.

git commit --amend: Allows you to make changes to the last commit (useful for updating comments or adding a forgotten file).

These commands are just the basics, and Git has many other features and capabilities for version control of your code and collaborative project work.

A "commit" is a concept used in version control systems, such as Git, to record and save changes in code or files. Each commit represents a "snapshot" of the project's state at a specific point in time. When you make a commit, you save the changes you've made to your code and include a brief description of these changes to make it easier to understand what was modified or improved.

The main functions of commits include:

1.History of Changes: Each commit has a unique identifier and stores information about what was changed in a file or project. This history of changes allows you to track how the project has evolved over time and what specific changes were made in different commits.

2.Reverting to Previous States: Thanks to commits, you can revert to previous states of the project. If you discover errors or issues in the new code, you can switch back to a commit where everything was still working correctly.

3.Collaboration: Commits are also used for collaboration within a team of developers. Each developer can make their own commits, and these changes can be easily merged into a single shared development path.

4.Branches: Version control systems allow you to create branches, which represent separate development lines of a project. Each branch has its own commit history. This allows developers to work on different features or fixes in parallel without interfering with each other.

In summary, commits enable you to track and manage changes in files and projects, maintain a history of work on a project, and facilitate collaboration among developers.