WORK CASE #1

Греков Данііл / Богдан Раєв

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.  
In version control systems like Git, the term "commit" is used to describe the process of documenting and saving changes to code or files. Each commit serves as a "snapshot" of the state of the project at a certain moment in time. In order to make it clearer what was changed or improved, you save the modifications you've made to your code along with a brief description when you commit.

The primary purposes of commits are as follows:

1.Change History: Each commit contains a special identifier and details the changes made to a file or project. You may follow the project's development over time and identify the precise changes that were made in various commits by using the history of modifications.

3.cooperation: A development team may also utilize commits for cooperation. Every developer is allowed to create their own contributions, and it is simple to integrate all of these changes into one common development path.

4.Branches: Version control systems let you construct branches, which stand in for distinct project development lines. Each branch has a unique history of commits. Because of this, developers may work on various features or fixes concurrently without interfering with one another.

In conclusion, commits make it possible to keep track of and manage changes made to files and projects, preserve a record of previous work done on a project, and promote developer cooperation.