**Use of GitHub Repository**

**✔ Description of GitHub**

GitHub is a web-based platform built on top of Git, a version control system. It allows developers to collaborate, manage, and store their code repositories efficiently. Key features include:

* **Version Control**: Tracks changes to code, enabling teams to collaborate without overwriting each other's work.
* **Collaboration**: Provides features like pull requests, code reviews, and issue tracking.
* **Hosting**: Offers hosting for remote repositories, allowing projects to be stored in the cloud.
* **Integration**: Supports CI/CD pipelines, project management tools, and other developer utilities.

**✔ Create an Account on GitHub**

1. Go to [GitHub](https://github.com).
2. Click **Sign Up**.
3. Fill in the required details (email, username, and password).
4. Verify your email address.
5. Choose a plan (Free or Paid, depending on your needs).

**✔ Create a New Remote Repository**

1. Log in to your GitHub account.
2. Click the **+** icon at the top-right corner and select **New repository**.
3. Enter a **Repository Name** (e.g., "my-first-repo").
4. Add an optional **Description**.
5. Select repository visibility:
   * **Public**: Anyone can view.
   * **Private**: Only you (and collaborators you invite) can view.
6. (Optional) Add a README file, .gitignore, or license.
7. Click **Create repository**.

**✔ Apply Git Commands Related to Repository**

1. **Initialize Git in a Local Project:**

git init

This sets up a new Git repository in your local directory.

1. **Add Files to Staging:**

git add .

Stages all changes in the directory.

1. **Commit Changes:**

git commit -m "Initial commit"

Saves changes to the local repository with a descriptive message.

1. **Link to a Remote Repository:**

git remote add origin <repository-URL>

Connects your local repository to the remote GitHub repository.

1. **Push Changes to Remote:**

git push -u origin main

Pushes changes from the local repository to the remote main branch.

**● Git Clone**

The git clone command is used to copy an existing repository (remote or local) into a new directory.

**Command Syntax:**

git clone <repository-URL>

**Example:**

git clone https://github.com/username/my-first-repo.git

This creates a local copy of the remote repository.

**● Git Remote**

Git remote commands manage the connection between your local repository and a remote repository.

1. **View Current Remote Connections:**

git remote -v

1. **Add a New Remote:**

git remote add origin <repository-URL>

1. **Remove a Remote:**

git remote remove <name>

1. **Change a Remote URL:**

git remote set-url origin <new-repository-URL>

**Learning outcome 2: Manipulate files**

**Definition of General Key Terms**

**✔ Status**

The git status command provides an overview of the current state of your working directory and staging area. It shows:

* **Tracked files**: Modified, staged, or committed.
* **Untracked files**: New files not yet staged.
* **Deleted files**: Removed files that haven't been staged or committed.

**✔ Branch**

A branch in Git is a pointer to a specific commit. Branches allow developers to work on isolated features or fixes without affecting the main codebase.

**✔ Commit**

A commit is a snapshot of changes made to the files in the repository. It acts as a record in the version history, enabling you to revert or review changes later.

**Add File Changes to Git Staging Area**

**✔ Operation on git status Command**

The git status command helps track the status of your working directory and staging area. It displays:

1. **Untracked files** (new files not staged).
2. **Modified files** (changed files not staged).
3. **Deleted files** (files removed but not staged).
4. Files staged for the next commit.

**Examples:**

git status

* **Untracked Files:**

Untracked files:

(use "git add <file>..." to include in what will be committed)

newfile.txt

* **Modified Files:**

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

modifiedfile.txt

* **Deleted Files:**

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

deletedfile.txt

**✔ Operation on git add Command**

The git add command moves file changes to the staging area.

1. **Stage All Files:**

git add .

1. **Stage a Specific File:**

git add filename.txt

1. **Stage a Folder:**

git add foldername/

**✔ Operation on git reset Command**

The git reset command removes files from the staging area.

1. **Unstage a File:**

git reset filename.txt

1. **Unstage All Files:**

git reset

**✔ Operation on git rm Command**

The git rm command is used to remove files from the working directory and stage the changes for the next commit.

1. **Remove and Stage a File:**

git rm filename.txt

1. **Remove and Stage a Folder:**

git rm -r foldername/

**Commit File Changes to Git Local Repository**

**✔ Best Practices for Creating a Commit Message**

1. **Short and Descriptive Title:** Limit the first line to 50 characters or less.
2. **Separate Description:** Provide a detailed description if necessary, separated by a blank line.
3. **Use the Imperative Mood:** Start with verbs like "Add," "Fix," or "Update."
4. **Reference Issues:** If relevant, include issue numbers (e.g., Fixes #42).

**Example:**

Add login functionality

- Implemented user authentication.

- Added error handling for failed logins.

- Updated database schema for user roles.

**✔ Operation on git commit Command**

1. **Commit All Staged Changes:**

git commit -m "Add feature X"

1. **Commit a Specific File:**

git commit filename.txt -m "Update filename.txt with bug fixes"

1. **Edit the Last Commit Message:**

git commit --amend -m "Updated commit message"

**✔ Operation on git log Command**

The git log command displays the commit history of a repository.

1. **View Basic Commit History:**

git log

1. **View One-Line Summaries:**

git log --oneline

1. **Include File Changes in Logs:**

git log -p

1. **Filter Logs by Author:**

git log --author="Author Name"

**Git Commands for Managing Commits and Branches**

**Commit History Commands**

**● To See a Simplified List of Commits**

The git log command can display a simplified list of commits using the --oneline option.  
**Command:**

git log --oneline

* Displays each commit as a single line with:
  + Commit hash (shortened).
  + Commit message.

**Example Output:**

d2c3e21 Add user authentication feature

a1b4f67 Fix typo in README

c4a5b29 Initial commit

**● To See a List of Commits with More Detail**

The default git log command provides detailed commit information, including:

* Commit hash
* Author name
* Date
* Commit message

**Command:**

git log

For even more details, you can use:

git log -p

* Shows the full commit diff.

**Managing Branches**

**✔ Operations on Branches**

**● Create a Branch**

To create a new branch:

git branch <branch\_name>

**Example:**

git branch feature-login

This creates a branch named feature-login.

**● List Branches**

To see all branches in the repository:

git branch

* The active branch will be marked with \*.

To list all remote and local branches:

git branch -a

**● Delete a Local Branch**

To delete a local branch:

git branch -d <branch\_name>

* Use -D to force deletion if the branch hasn’t been merged.

**Example:**

git branch -d feature-login

**● Delete a Remote Branch**

To delete a branch from the remote repository:

git push origin --delete <branch\_name>

**Example:**

git push origin --delete feature-login

**● Switch Branch**

To switch to another branch:

git checkout <branch\_name>

OR (preferred for modern Git versions):

git switch <branch\_name>

**Example:**

git switch feature-login

**● Rename a Branch**

To rename the current branch:

git branch -m <new\_branch\_name>

To rename a branch from a different branch:

git branch -m <old\_branch\_name> <new\_branch\_name>

**Example:**

git branch -m old-name new-name

To push the renamed branch to the remote:

git push origin -u new-name

And delete the old branch:

git push origin --delete old-name

**Learning outcome 3: Ship codes**

**Definition of General Key Terms**

**● Pull**

The git pull command is used to fetch changes from a remote repository and merge them into the current branch. It is equivalent to performing git fetch followed by git merge.

**Syntax:**

git pull <remote> <branch>

**● Fetch**

The git fetch command retrieves changes (commits, branches, tags) from the remote repository but does not merge them into your local branch. It allows you to review changes before integrating them.

**Syntax:**

git fetch <remote>

**● Push**

The git push command uploads your local repository changes (commits) to the remote repository. It is used to share code with collaborators.

**Syntax:**

git push <remote> <branch>

**● Pull Request**

A pull request (PR) is a GitHub feature that allows developers to propose changes from one branch to another (usually from a feature branch to the main branch). It enables code review, discussion, and collaboration before merging the changes.

**● Merge**

The git merge command is used to integrate changes from one branch into another. For example, you can merge a feature branch into the main branch.

**Syntax:**

git merge <branch>

**Fetch File from GitHub Repository**

**✔ Operation on git fetch Command**

The git fetch command retrieves remote changes without altering the local working directory.

1. **Fetch All Changes:**

git fetch

1. **Fetch Changes from a Specific Remote Repository:**

git fetch origin

**● Fetch the Remote Repository**

To fetch all updates (commits, branches, and tags) from the remote repository:

git fetch <remote>

**Example:**

git fetch origin

**● Fetch a Specific Branch**

To fetch updates from a specific branch only:

git fetch <remote> <branch>

**Example:**

git fetch origin feature-login

**● Fetch All Branches Simultaneously**

To fetch updates for all branches in the remote repository:

git fetch --all

**● Synchronize the Local Repository**

To ensure your local repository is up-to-date with the remote repository:

1. **Fetch Changes:**

git fetch origin

1. **Merge Changes into Your Branch:**

git merge origin/<branch\_name>

OR

1. **Pull Changes Directly:**

git pull origin <branch\_name>

**Git Commands: Pull, Push, Rebase, and Merge**

**✔ Operation on git pull**

**● Default git pull**

The default git pull command fetches changes from the remote repository and merges them into the current active branch.  
**Command:**

git pull

**● Git Pull Remote Branch**

To pull changes from a specific branch in a remote repository:

git pull <remote> <branch>

**Example:**

git pull origin feature-login

**● Git Force Pull**

To discard local changes and force the remote branch content into your current branch:

git fetch origin

git reset --hard origin/<branch>

**Example:**

git fetch origin

git reset --hard origin/main

**● Git Pull from Origin Master**

To pull changes specifically from the master branch of the origin remote:

git pull origin master

**✔ Tags Used on git push Command**

**Commonly Used Tags:**

1. **--tags**: Push all tags to the remote repository.

git push origin --tags

1. **--force or -f**: Force push changes to overwrite remote changes.

git push origin <branch\_name> -f

1. **--set-upstream**: Link the local branch to a remote branch and push.

git push --set-upstream origin <branch\_name>

1. **--verbose**: Shows detailed output while pushing.

git push origin <branch\_name> --verbose

**✔ Operation on git push**

**● Push to Origin Master**

To push the changes from your current branch to the master branch on the remote:

git push origin master

**● Git Push Force**

To overwrite the remote branch with your local changes:

git push origin <branch\_name> --force

**● Git Push Verbose**

To display additional information while pushing:

git push origin <branch\_name> --verbose

**● Delete a Remote Branch**

To delete a branch from the remote repository:

git push origin --delete <branch\_name>

**Example:**

git push origin --delete feature-login

**Merge Branches on Remote Repository**

**✔ Operation on git rebase Command**

The git rebase command integrates changes from one branch onto another by replaying commits.

1. **Rebase Current Branch onto Another:**

git rebase <branch\_name>

1. **Rebase onto Master:**

git rebase master

**✔ Create Pull Request**

To create a pull request:

1. Push your branch to the remote repository.

git push origin <branch\_name>

1. Go to the remote repository (e.g., GitHub).
2. Create a pull request by comparing your branch with the target branch (e.g., main).

**✔ Operation on git merge**

**● Merge the Specified Commit to Current Active Branch**

1. Find the commit hash:

git log

1. Merge the specific commit:

git cherry-pick <commit\_hash>

**● Merge Commits into the Master Branch**

To merge changes from a feature branch into the master branch:

1. Switch to master:

git checkout master

1. Merge the feature branch:

git merge feature-login

**● Git Merge Branch**

To merge one branch into another:

git merge <branch\_name>

**Example:**

git merge development