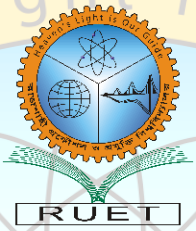
Heaven's light is our guide.

Rajshahi University of Engineering and Technology, Rajshahi.



Department of Electrical and Computer Engineering.

Course Code:

ECE 3118

Course Title:

Software Engineering & Information System Design Sessional

Submitted To:	Submitted By:
Oishi Jyoti	Md. Towhidul Alam
Assistant Professor	Roll: 2010016
Department of	Department of Electrical & Computer Engineering
ECE RUET	

Introduction

Git is a version control system that enables developers to track and manage changes to files over time, facilitating collaboration and allowing for quick recovery from accidental deletions or bugs. This lab focuses on essential Git commands, their usage, and how they contribute to effective project management.

Objectives

- Understand fundamental Git commands.
- Learn how to create, modify, and track changes within a Git repository.
- Familiarize with best practices in version control for collaborative development.

Git Commands and Descriptions

Command	Description
git init	Initializes a new Git repository in the current directory.
git clone <repository_url>	Clones a remote repository to the local machine.
git status	Displays the current state of the working directory and staging area.
git add <file>	Adds specified files to the staging area, preparing them for commit.
git commit -m "message"	Commits staged changes with a descriptive message.
git log	Shows a list of all previous commits in the repository.
git branch	Lists all branches in the repository; highlights the current branch.
git branch <branch_name>	Creates a new branch with the specified name.
git checkout <branch_name>	Switches to the specified branch.

Command	Description
git merge <branch_name>	Merges changes from the specified branch into the current branch.
git pull	Fetches and integrates changes from a remote repository into the current branch.
git push	Uploads commits from the local branch to the corresponding remote repository.
git remote -v	Shows a list of remote repositories associated with the local repository.
git fetch	Retrieves updates from a remote repository without integrating them into the working branch.
git diff	Displays differences between files in the working directory and the last committed version.
git reset <file>	Removes specified file from the staging area but keeps changes in the working directory.
git rm <file>	Deletes a file from both the working directory and the staging area.
git stash	Temporarily stores changes in the working directory for later use.
git stash pop	Reapplies stashed changes to the working directory.
git rebase <branch_name>	Reapplies commits from the current branch on top of another specified branch, streamlining history.
git tag <tag_name>	Creates a tag at the current commit, useful for marking release versions.
git show <commit_hash>	Displays information about a specific commit, including changes made.

Command	Description
git cherry-pick <commit_hash>	Applies changes from a specific commit onto the current branch.
git reflog	Lists recent changes and allows recovery of commits that were removed from the branch history.
git config --global user.name	Sets the global username for Git commits.
git config --global user.email	Sets the global email address associated with Git commits.
git mv <old_file> <new_file>	Renames or moves a file within the repository and stages the change.
git clean -f	Removes untracked files from the working directory.

Discussion

These basic Git commands are integral to managing a codebase effectively. Commands like git init, git clone, and git add form the foundation of initializing and adding files to a repository. Branching commands such as git branch, git checkout, and git merge enable effective feature isolation and collaboration. Additionally, git stash and git rebase are valuable tools for managing code in complex workflows without losing work in progress.

Using Git not only improves project management but also enhances teamwork by allowing developers to work on separate branches, contribute to the main codebase without conflicts, and track historical changes for accountability.

Conclusion

Understanding and using Git commands effectively is essential for maintaining a clean, organized, and collaborative code environment. This lab report covers key commands that are beneficial for beginners and essential for advanced project management. Through continuous practice, Git commands become an invaluable asset for developers, enhancing productivity and ensuring seamless collaboration in team environments.