**ARCHIS GOKHALE**

**1032233755**

**TY - CSE BTECH**

**PANEL I-2 (52)**

**FSD Laboratory 01**

Aim: Version control with Git.

Objectives:

1. To introduce the concepts and software behind version control, using the example of Git.
2. To understand the use of 'version control' in the context of a coding project.
3. To learn Git version control with Clone, commit to, and push, pull from a git repository.

Theory:

**1. What is Git? What is Version Control?**  
Git is a distributed version control system created by Linus Torvalds in 2005. It's designed for fast and efficient handling of projects, both small and large.

#### Key Features:

* Distributed: Every developer has a full copy of the repository.
* Speed: Optimized for quick operations like commits and merges.
* Branching: Easy to create and manage branches for different features.
* Integrity: Ensures data integrity with SHA-1 hashing.
* Staging: Allows reviewing changes before committing.

**2. How to use Git for version controlling?**

### **What is Version Control?**

Version control is a system that records changes to files over time, allowing you to recall specific versions later. It facilitates collaboration and tracks changes.

#### Types:

1. Local: Simple databases on the local disk.
2. Centralized (CVCS): Single central repository (e.g., SVN).
3. Distributed (DVCS): Each user has a full repository copy (e.g., Git).

#### Benefits:

* Collaboration: Multiple people can work simultaneously.
* History: Tracks changes and allows reversion.
* Branching: Separate branches for features, easily merged.
* Backup: Acts as a backup system.
* Traceability: Clear project evolution record.

Git is a powerful version control tool, perfect for managing project files and team collaboration efficiently.

FAQ:

**1. What is branching in Git?**  
Branching in Git is a fundamental concept that allows you to diverge from the main line of development and continue to work without affecting that main line. It enables multiple developers to work on different features or bug fixes simultaneously in a collaborative environment.

**Branches**: A branch in Git is essentially a lightweight movable pointer to one of these commits. The default branch name in Git is master (now often renamed to main). When you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.

**2. How to create and merge branches in Git? Write the commands used.**

### Creating and Merging Branches in Git

#### 1. Creating a Branch

To create a new branch, use the git branch command followed by the name of the branch.

git branch new-branch

#### 2. Switching to a Branch

Or using the newer git switch command:

git switch new-branch

#### 3. Merging a Branch

To merge a branch into your current branch, first switch to the branch you want to merge into (usually main or master), then use the git merge command:

git switch main

git merge new-branch

#### 4. Deleting a Branch

If the branch hasn't been merged yet and you want to force delete it:

git branch -D new-branch

### Example Workflow

1. Create and switch to a new branch:

git checkout -b feature-branch

1. Work on your changes and commit them:

git add .

git commit -m "Add new feature"

1. Switch back to the main branch:

git checkout main

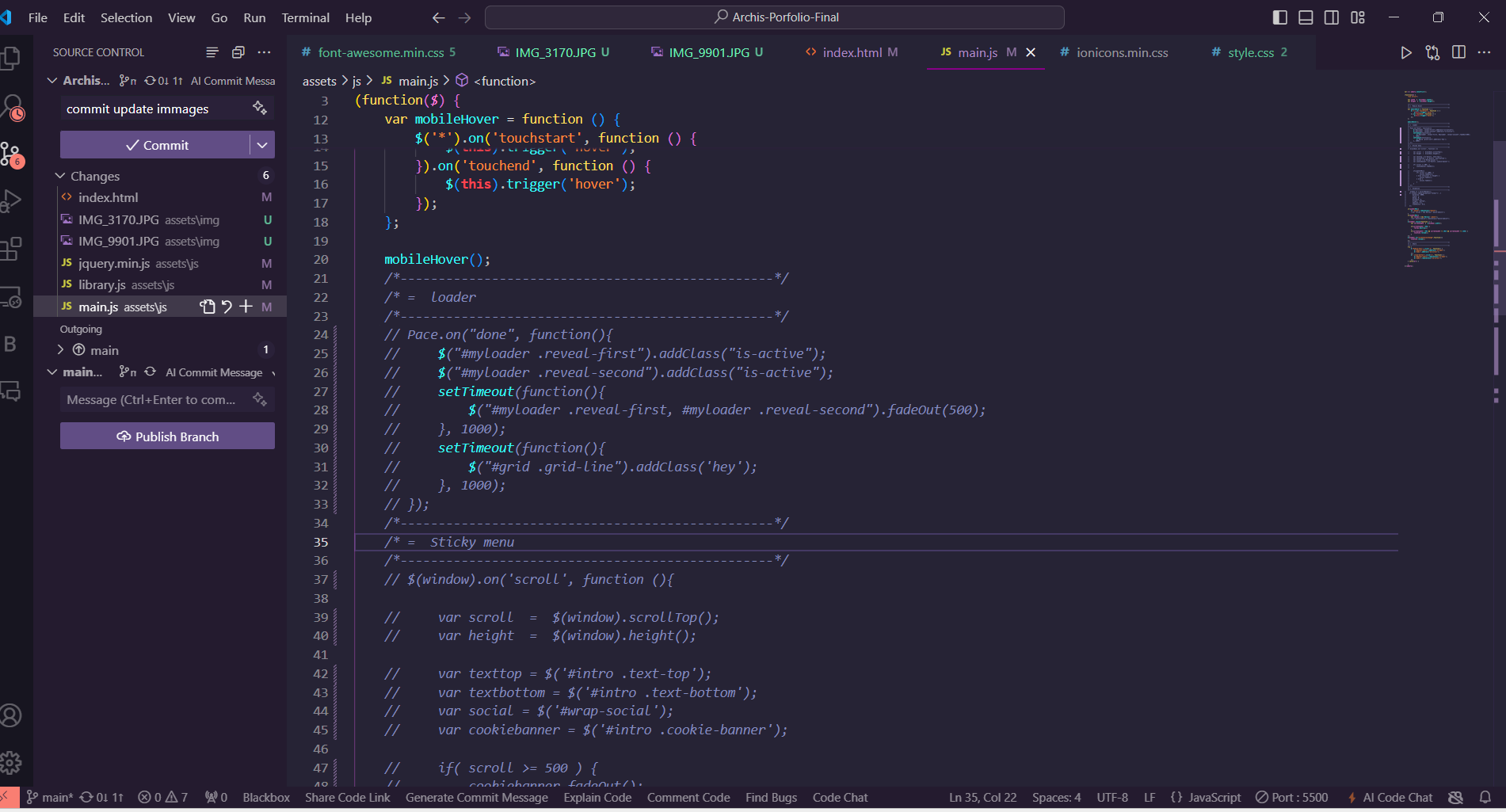
1. Merge the feature branch into the main branch:

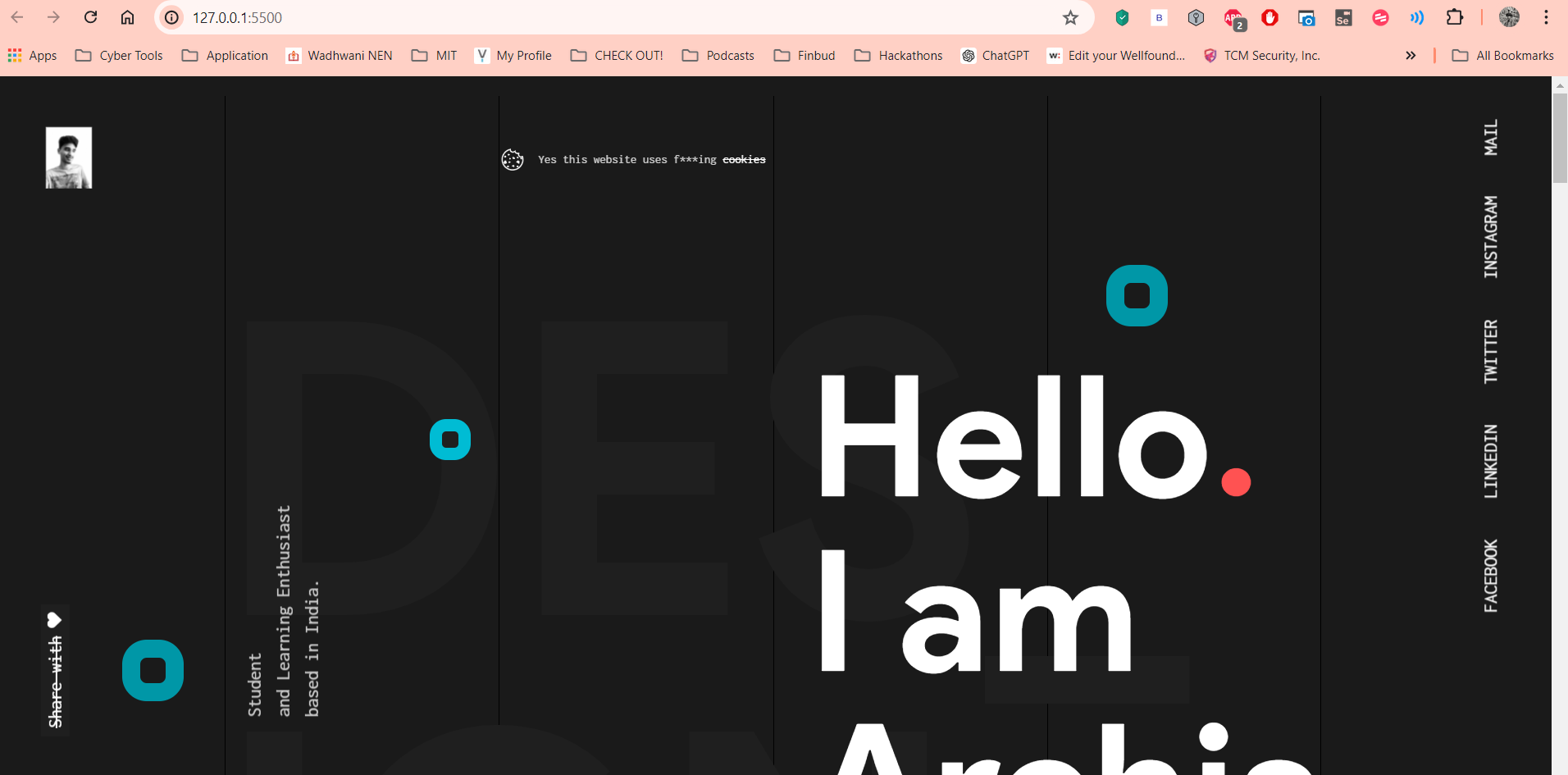
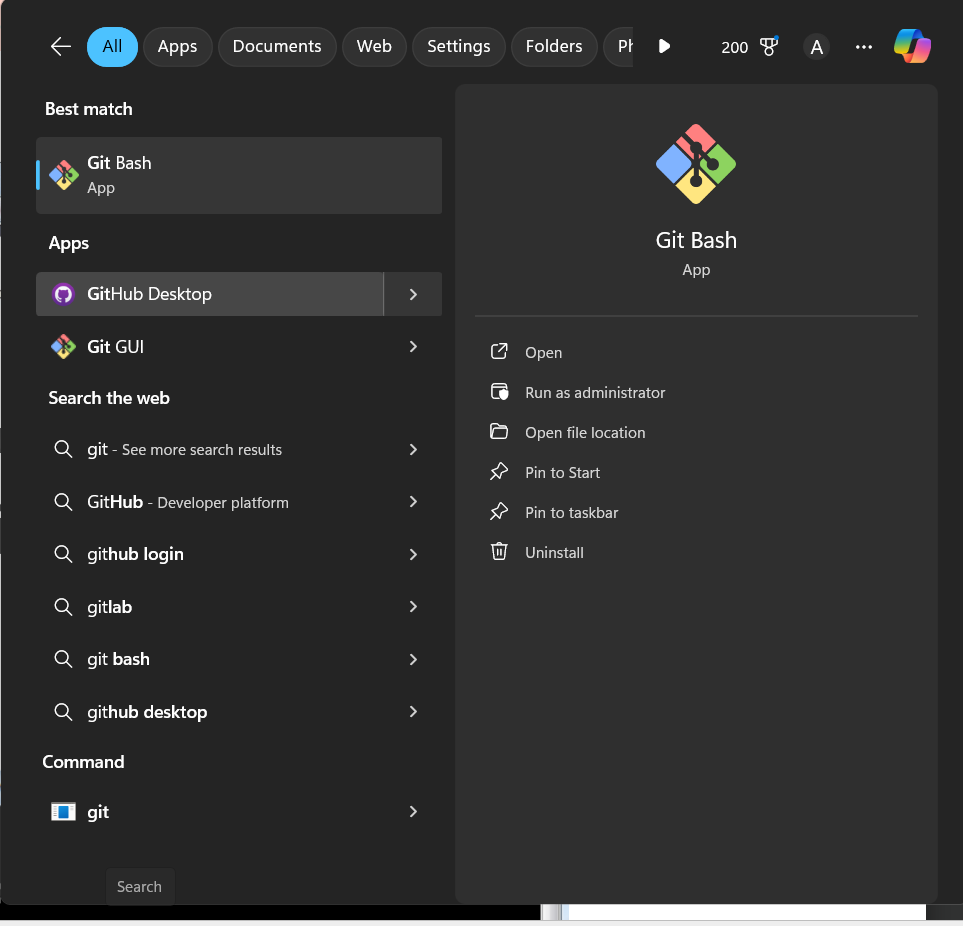
git merge feature-branch

1. Delete the feature branch:

git branch -d feature-branch

Output: Screenshots of the output to be attached.



****

**Problem Statement:**

Create a public git repository for your team and submit the repo URL as a solution to this assignment, Learn Git concept of Local and Remote Repository, Push, Pull, Merge and Branch.  
