VIT Vidyalankar Institute of Technology ACCREDITED A+ BY NAAC	**Department of Information Technology**

Semester	T.E. Semester VI – INFT
Subject	DevOps Lab
Lab Teacher:	Prof. Bushra Shaikh
Laboratory:	CC02

Student Name	Sahil Shangloo	
Roll Number	22101A0027	
Grade and Subject Teacher's Signature		

Experiment Number	2	
Experiment Title	Experiment 2: Automating Version Control and Code Collaboration of TE Mini Project with Git	
Problem Statement	To Perform Code Collaboration of TE Mini Project using GIT workflow	
Resources / Apparatus Required	Hardware: Desktop/Laptop	Software: Git & Github

Implementation:

- Set up a Git repository for a project (either create a new project or use an existing one).
- Collaborate with a team member by creating multiple branches for different features.
- Implement the following:
  - Create a feature branch and make changes to the code.
  - Push changes to the remote repository.
  - Use a Git cheat sheet to commit, merge branches, and resolve conflicts.
  - Create a pull request (PR) and ensure code review processes are followed.

Step 1: Initialize a Git Repository and Push It to GitHub

1.1 Initialize a Git Repository Locally

- Navigate to your project directory on your local machine (or create a new project folder).
- Open your terminal and run the following command to initialize a Git repository:
  `git init`

1.2 Add Files to the Repository

- Add a new file or make changes to existing files. For example, create a `README.md` file in the project folder:
  `echo "# BG Changer" > README.md`

1.3 Stage and Commit Files

- Stage the files for commit using:
  `git add .`

- Commit the staged files with a message:
  git commit -m "Initial commit"

1.4 Create a GitHub Repository

- Go to GitHub and create a new repository. You can name it something like "BgChanger".
- Copy the URL of the GitHub repository, e.g., https://github.com/Anuj-Gill/yuva-setu

1.5 Push to GitHub

- Set the remote repository URL:
  `git remote add origin https://github.com/Anuj-Gill/yuva-setu.git`

- Push the local repository to GitHub:
  `git push -u origin main`

Step 2: Create and Merge Multiple Feature Branches

2.1 Create a New Feature Branch

- Create a new branch:

  `git checkout -b sahil-ui`

2.2 Make Changes in the Feature Branch

- Add a new file or modify an existing file:
  Created many component files.

2.3 Stage and Commit Changes

- Stage the changes and commit them:

  `git add .`

```
git commit -m "UI made for NGO"
```

2.4 Push the Feature Branch to GitHub

- Push the feature branch to GitHub:
  `git push origin sahil-ui`

2.5 Create Another Feature Branch

- Create another branch for a different feature, for example, `feature-button`:

  ```
  git checkout -b sahil-ui
  ```

2.8 Push the `feature-button` Branch to GitHub

- Push the button branch to GitHub:

  ```
  git push origin feature-button
  ```

Step 3: Simulate a Merge Conflict and Resolve It

3.1 Switch to `main` Branch

- Now switch back to the `main` branch:

  ```
  git checkout main
  ```

3.2 Merge `feature-color` into `main`

- Merge the `feature-color` branch into the `main` branch:

  ```
  git merge sahil-ui
  ```

- Since the changes are in different parts of the file, the merge will succeed without conflict.

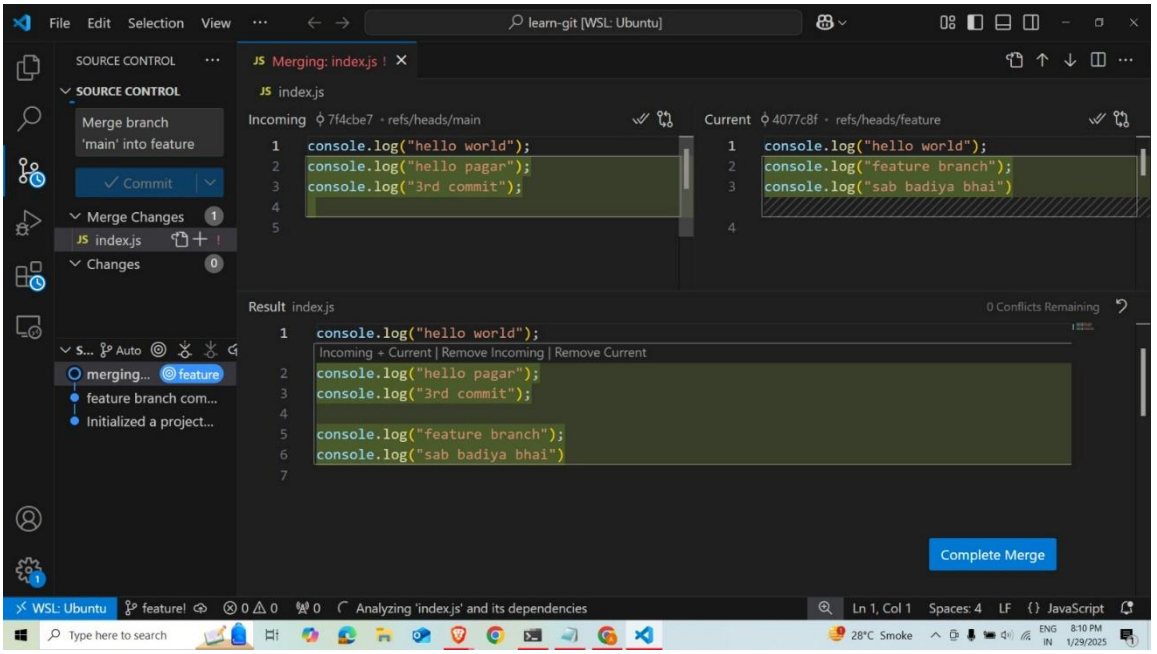
3.3 Merge `feature-button` into `main`

- Now, try to merge the sahil-ui branch:

  ```
  git merge main
  ```

3.4 Resolve the Merge Conflict

- Open the file structure. You will see some updated lines of code like this:



- Edit the file to resolve

3.5 Stage the Resolved Conflict

- Once the conflict is resolved, stage the file:
  `git add .`

3.6 Commit the Merge

- Commit the merge with a message:
  `git commit -m "Resolve merge conflicts"`

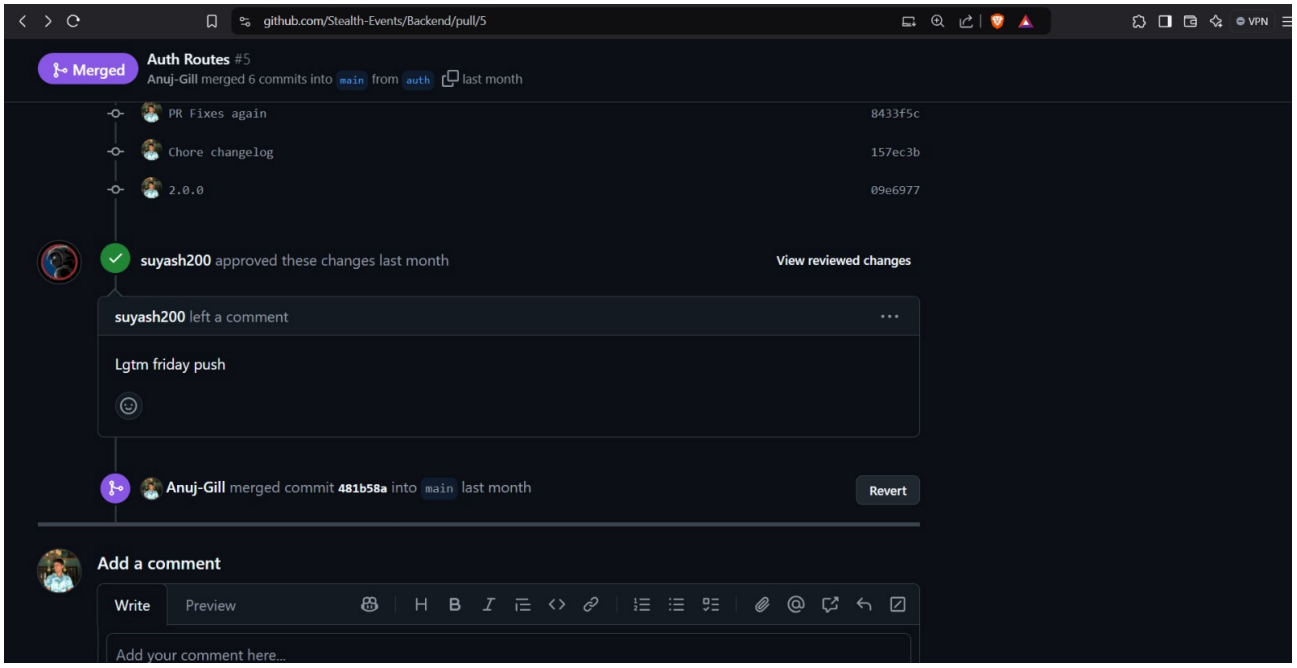
Step 4: Perform a Pull Request Review and Handle the Integration Process

4.1 Create a Pull Request (PR) on GitHub

- Go to your repository on GitHub.
- You should see a button to create a Pull Request for the branches you've pushed.
- Click on "New Pull Request" and select the Auth Routes branch and compare it with `main`.
- After reviewing the changes, click "Create Pull Request."
- Add a description of the changes and create the PR.

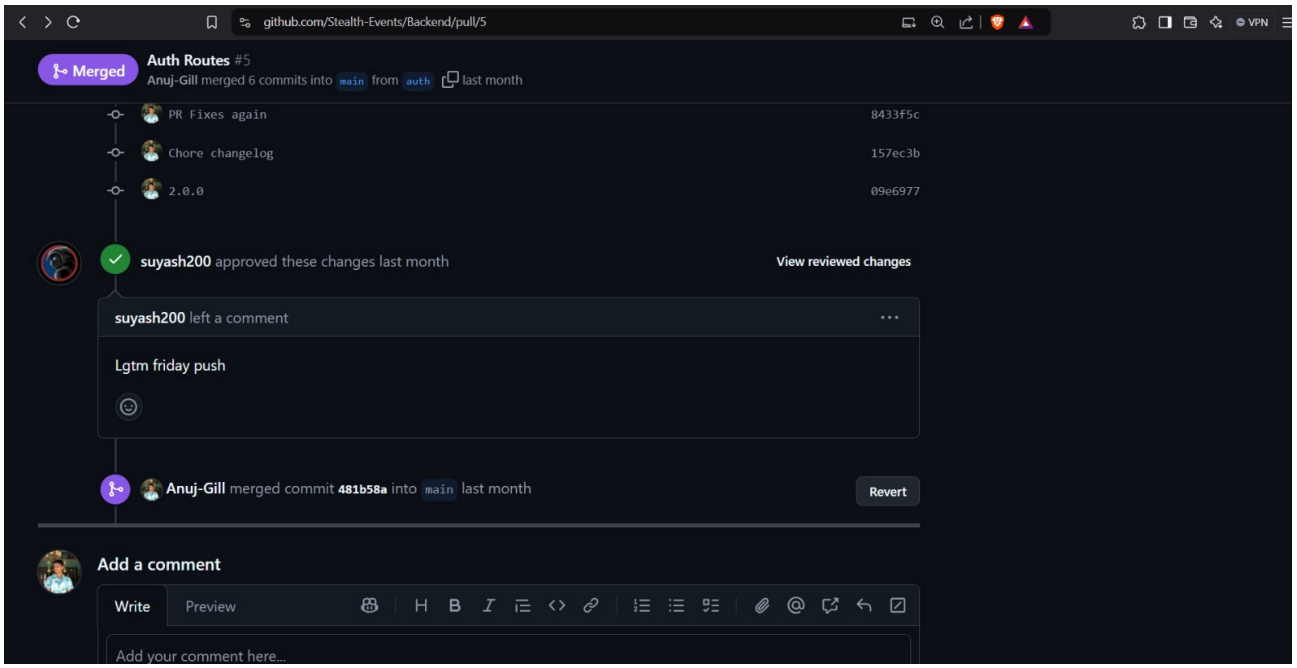
4.2 Review the Pull Request

- Review the changes in the pull request.
- You can see the changes made and leave comments on specific lines of code if necessary.
- You can ask a team member to review it as well or approve it yourself.



4.3 Merge the Pull Request

- After the PR review, click the "Merge pull request" button.
- Choose "Confirm merge" to integrate the changes from Auth Routes into `main`.



4.4 Clean Up the Branches

- After successfully merging the feature branches, delete them from GitHub (optional):

  ```
  git push origin --delete sahil-ui / Auth Routes
  ```