1. **Demonstrate how to create GitHub account**.

Steps of create a GitHub account:

Step 1: Visit the GitHub Website

- Open a browser.
- Go to https://github.com.

Step 2: Click "Sign up"

- On the GitHub homepage, click the **"Sign up"** button in the top right corner.

Step 3: Enter Your Details

You'll be prompted to fill in:

- Email address
- Create a password
- Choose a username
- Verify your account (GitHub may ask for a puzzle or CAPTCHA)
- Click **"**Continue**"** after filling in each section.

Step 4: Choose Account Preferences

- Choose whether you want to receive updates via email.
- Choose free or paid plan (you can select **"**Free**"** to start).
- Click **"**Continue**"** or **"**Complete setup**"**.

Step 5: Verify Your Email

- Go to your email inbox.
- Open the verification email from GitHub.
- Click on the **"**Verify email address**"** link.

Done!

You now have a GitHub account. You can:

- Create repositories
- Share code
- Collaborate on projects

2. Exploring Git Commands through Collaborative Coding

- **Setting Up Git Repository**
- **Creating and Committing Changes**
- **Branching and Merging.**

1. Setting Up Git Repository

Step 1.1: Install Git (if not installed)

Download Git and install it.

Step 1.2: Configure Git with your identity

```
git config --global user.name "Vrushali Tambe"
git config --global user.email "vrusha08@gmail.com"
```

Step 1.3: Create a new directory and initialize a Git repo

```
mkdir collaborative-project
cd collaborative-project
git init
```

2. Creating and Committing Changes

Step 2.1: Create a new file
Echo “welcome” > index.txt
Step 2.2: Track the new file
git add index.txt
Step 2.3: Commit your change
git commit -m "Initial commit with index.txt"
Step 2.4: Push to GitHub (first time)
git remote add origin https://github.com/vrusha8/collab-git-practical.git
git push origin master

3. Branching and Merging
Step 3.1: Create a new branch (e.g., for a new feature)
git checkout -b feature-login
Step 3.2: Make changes and commit
Create a new file
Echo “welcome to imrd” > login.txt
Stage and commit the change
git add login.txt
git commit -m " Add login functionality with login.txt"
Step 3.3: Switch back to the main branch
git checkout master
Step 3.4: Merge the feature branch
git merge feature-login
Step 3.5: Push the updated main branch
git push origin master

3. Implement GitHub Operations using Git
• Cloning a Repository
• Making Changes and Creating a Branch
• Push/Pull Changes to GitHub

1. Cloning a Repository
#Cloning a repository allows you to create a local copy of a remote repository from GitHub.
git clone https://github.com/vrusha8/collab-git-practical.git
2. Making Changes and Creating a Branch
Step 1: Create a new branch
git checkout -b feature-update
Step 2: Make a change using terminal:
#Create a new file
echo "This is a new feature" > feature.txt
#Add the file:
git add feature.txt
#Commit the change:
git commit -m "Added feature.txt file"
3. Push/pull Changes to GitHub
#Push the branch to remote GitHub:
git push origin feature-update
# Pull latest changes from remote branch
git pull origin feature-update
4. Create account on docker step by steps.

Step-by-Step: Create Docker Account

1. Go to: https://hub.docker.com/signup OR sign in Docker Desktop
2. Open the link above in your browser.
3. Fill in the signup form**:**
    - Username: unique Docker ID (e.g., vrushali123)
    - Email address
    - Password
4. Click "Sign Up"**.**
5. Verify your email**:**
    - Open your inbox and click the verification link Docker sends you.
6. Once verified, you can log in at: https://hub.docker.com/login
7. Docker Account Created Successfully

5. Demonstrate a practical on Version Control Tools.

1. Initialize a Git Repository

```
mkdir my-git-practical
cd my-git-practical
git init
```

This creates a new local Git repository in the directory.

2. Create a File and Add Content

```
echo " My First Git Repo" > index.txt
```

3. **Check Git Status**

```
git status
```

Shows that index.txt is untracked.

4. Stage and Commit the File

```
git add index.txt
git commit -m "Initial commit with index"
```

You've now recorded your first snapshot.

Part 2: Making Changes and Tracking History

5. **Edit the File**

Add more content:

```
echo "This is a Git practical session." >> index.txt
```

6. **View Changes**
git status
git diff

7. Commit the Changes
git add index.txt
git commit -m "Updated index with session info"

8. **View Commit History**
git log
Shows a list of all commits made.

9. **Create a GitHub Repository*(Optional)***

- Go to GitHub, log in, and click **New Repository**.
- Name it my-website and keep it public or private.

10. **Push Your Local Repository*(Optional)***
git remote add origin https://github.com/vrusha08/ my-website.git
git branch -M main
git push -u origin main
project is now version-controlled

6. Create a merge request on gitlab and Review the merge request.
Create a Merge Request on GitLab

Step 1: Create a Repository on GitLab

1. Log in to your GitLab account.
2. Click New Project → Create Blank Project.
3. Give it a name (e.g., `merge-request-demo`).
4. Choose **Private** or **Public** visibility.

`Click` **`Create Project`**

Step 2: Clone the Repository to Your Local System

git clone https://gitlab.com/username/merge-request-demo.git
cd merge-request-demo

Step 3: Create a New Branch

git checkout -b feature-branch

Create a File and Add Content
echo " My First Git Repo" > index.txt

Step 5: Commit and Push Changes

```
git add .
git commit -m "Added index.txt"
git push origin feature-branch
```

Step 6: Create a Merge Request in GitLab

1. Go to your project on GitLab.
2. You'll see a banner suggesting to create a merge request — click Create Merge Request**.**
3. Select source branch = `feature-branch`, target branch = `main`.
4. Add a title and description.

Click Submit Merge Request**.**

Step 7: Review a Merge Request

- As another developer (or using a second account), go to **Merge Requests**.
- Open the MR and review the code changes(click on changes).
- Add **comments**, **suggestions(click on start review button)**
- click on overview option and Add reviewer, or **approve** the merge request.
- If approved, click **Merge**.

7. To study docker file intructions, build an image for a sample web application using docker file.

1. Making Directory and create index.html file and docker file

```
mkdir docker-webapp

cd docker-webapp

notepad index.html
```

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Docker Web App</title>
</head>
<body>
  <h1>Hello, Docker!</h1>
  <p>This is a sample web application running in Docker.</p>
```

```
</body>
</html>
```

In same directory create dockerfile (I use that link for image + dockerfile coding)
Create **Dockerfile** (no extension, just Dockerfile):

```
notepad Dockerfile
```

Step 2: create Dockerfile in same folder
Dockerfile

```
# 1. Use an official Nginx image as base
FROM nginx

# 2. Copy your HTML file into Nginx's default web directory
COPY index.html /usr/share/nginx/html

EXPOSE 80

# 4. Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

Step 3: Build Docker Image

```
docker build -t mywebapp .
```

If successful, you'll see Successfully tagged mywebapp

Step 4: Run Container

Now run your web app:

```
docker run -d -p 8080:80 mywebapp
```

Step 5: Test Web App

1. Open browser and go to:
2. http://localhost:8080
3. You should see **Hello, Docker!**