Experiment: 02

Aim: To understand version control system, install Git and GitHub account

What is Version Control?

Version control is a system that allows developers to track and manage changes to software code over time. It enables collaboration, ensures code integrity, and allows multiple versions of the code to be stored and retrieved as needed. The main goal of version control is to keep track of modifications to a project, making it easier to collaborate, manage different versions of files, and track changes over time.

There are two types of version control systems:

1. **Local Version Control**: This is the simplest form, where a developer keeps track of changes on their own computer.
2. **Distributed Version Control**: This is more advanced and is used by systems like Git, where each developer has their own local copy of the entire project repository (including its history), and changes are synchronized with others.

What is Git?

Git is a distributed version control system created by Linus Torvalds (the creator of Linux). Git helps developers manage the source code history by tracking changes and enabling multiple developers to work on a project without stepping on each other's toes. With Git, you can:

- **Track changes**: See what was modified and by whom.
- **Branching and merging**: Work on different parts of a project in parallel, then merge those parts back together.
- **Collaboration**: Work with others by pushing and pulling changes from remote repositories.

How Git Works:

1. **Repository (repo)**: A directory or storage space where Git keeps all the files, history, and versions of a project.
2. **Commit**: A snapshot of your project at a particular point in time. It records changes made to the project files.
3. **Branch**: A parallel version of the repository. You can create a new branch to work on a feature without affecting the main project.

4. **Merge**: The process of combining changes from different branches back into the main branch.

What is GitHub?

GitHub is a web-based platform that hosts Git repositories. It provides a user interface for managing Git repositories, collaborating with others, and sharing your code. GitHub makes it easier for developers to work together on a project, track bugs, and manage project releases. Key features of GitHub:

- **Remote repositories**: You can upload your local Git repositories to GitHub to back them up or collaborate.
- **Pull requests**: A way of proposing changes to a repository. A developer can submit a pull request to request that their changes be merged into another branch or the main codebase.
- **Issues and projects**: Track bugs, feature requests, and manage the workflow of development using tools integrated into GitHub.
- **Collaboration**: GitHub makes it easy for multiple people to work on a project by allowing them to push, pull, and merge changes from others.

Installing Git and Setting Up GitHub Account

Step 1: Install Git

1. **Download Git** from Git's official website.
2. **Install Git** by following the installation prompts specific to your operating system (Windows, Mac, or Linux).
    - On Windows, during installation, it is recommended to choose the default options.
    - On Mac/Linux, you can install Git using package managers like Homebrew (Mac) or apt (Linux).
3. Once Git is installed, you can verify it by opening a terminal or command prompt and running:

bash Copy

git --version

This will display the version of Git installed.

Step 2: Set Up Git

Before you start using Git, you should configure your identity:

bash Copy git config --global user.name "Your Name" git

config --global user.email "your-email@example.com"

This ensures that your commits are properly attributed to you.

Step 3: Create a GitHub Account

1. Go to GitHub and sign up for an account.
2. After signing up, you'll be able to create repositories and start collaborating with others.
3. You can create a new repository by clicking the **New** button on your dashboard or through the "Repositories" tab.

Step 4: Link Git with GitHub (SSH Keys)

To allow Git to communicate with GitHub, you need to set up SSH keys for authentication (instead of using your password every time).

1. **Generate an SSH key**:

bash Copy ssh-keygen -t rsa -b 4096 -C "your-

email@example.com" This will generate a key pair

(public and private keys).

2. **Add the SSH key to your GitHub account**:
    - Copy the public key using:

bash Copy

cat ~/.ssh/id_rsa.pub

    - Go to **GitHub** > **Settings** > **SSH and GPG Keys** > **New SSH Key**, then paste the key into the field provided.

3. **Test the connection**:

bash

Copy ssh -T

git@github.com

If successful, GitHub will confirm the connection.

Step 5: Clone a GitHub Repository

To get a project from GitHub onto your local machine, you can **clone** the repository. In your terminal, run:

bash Copy

git clone https://github.com/username/repository-name.git or,

if using SSH:

bash Copy git clone

git@github.com:username/repository-name.git

Step 6: Basic Git Commands

- **Check the status of your repository**:

bash Copy

git status

- **Add changes** to the staging area:

bash Copy

git add .

- **Commit changes** to your local repository:

bash Copy git commit -m "Your

commit message"

- **Push changes** to GitHub:

bash

Copy

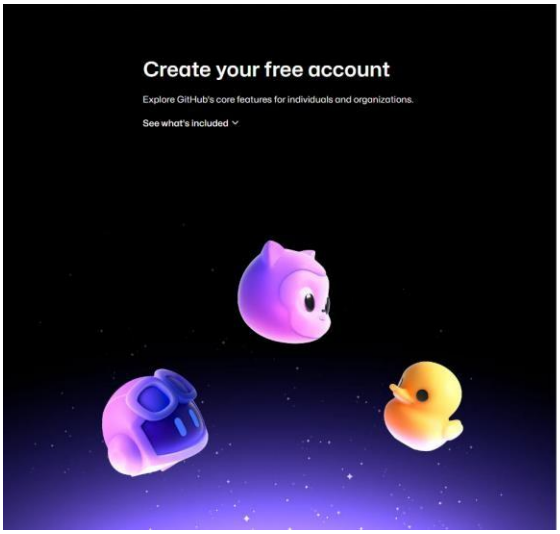
git push origin main

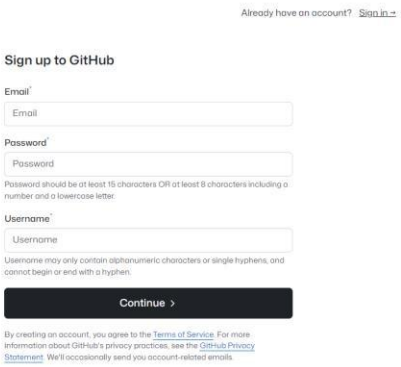
- **Pull changes** from GitHub to your local repository:

bash Copy

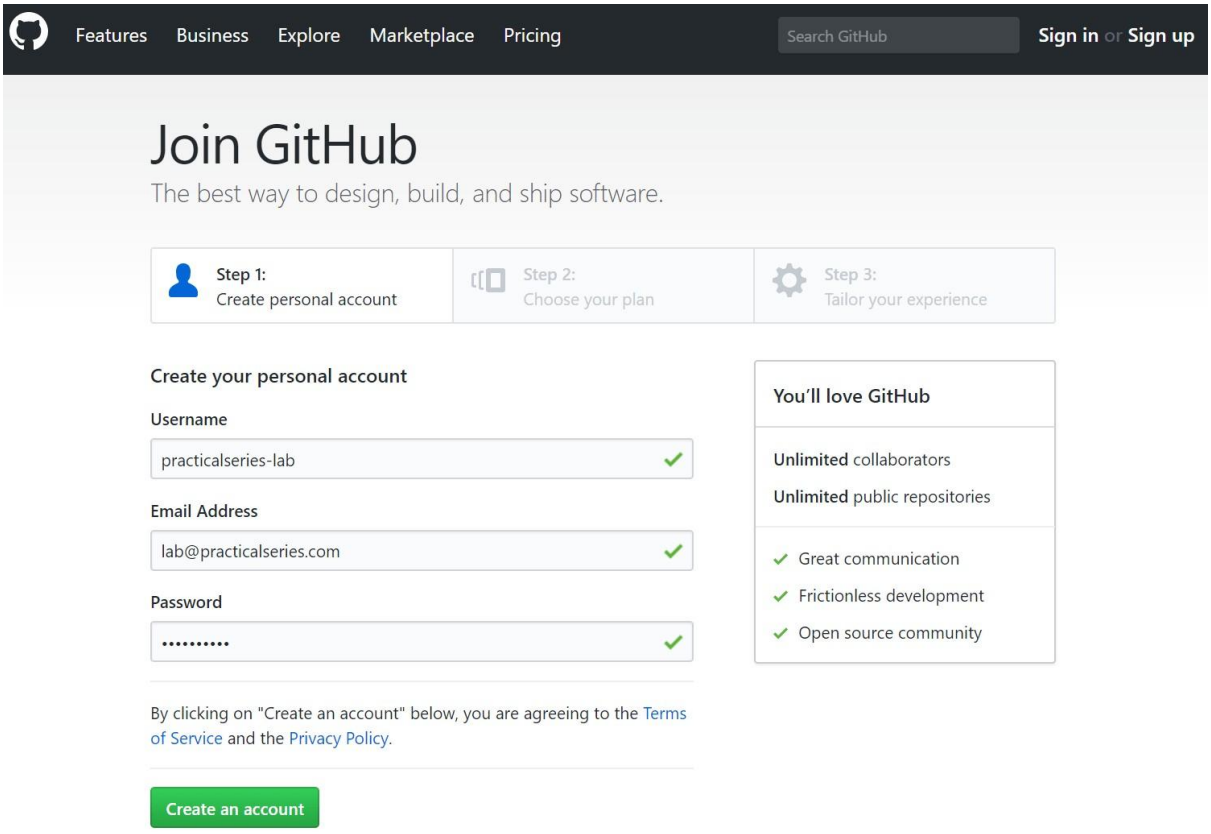
git pull origin main

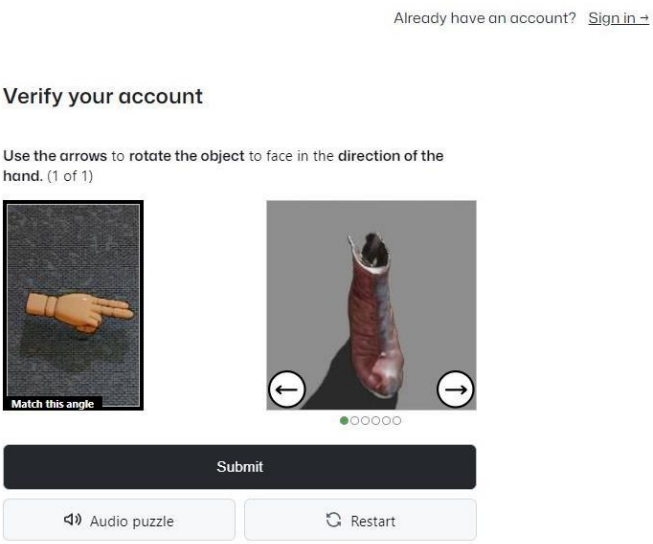
Step 1: Go to https://github.com/join in a web browser.



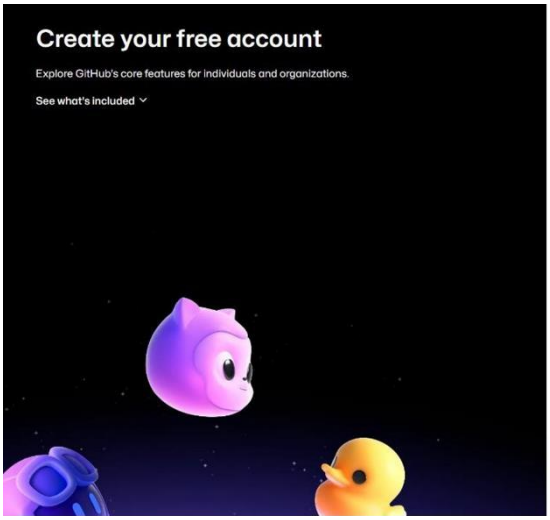


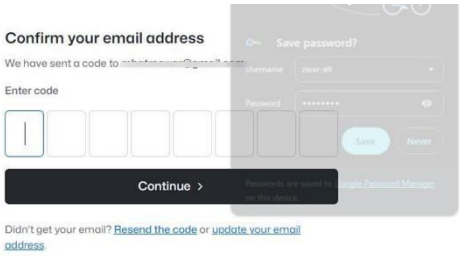
Step 2: Personal Information

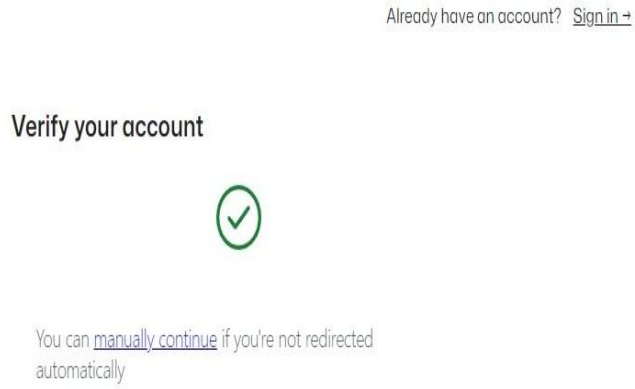




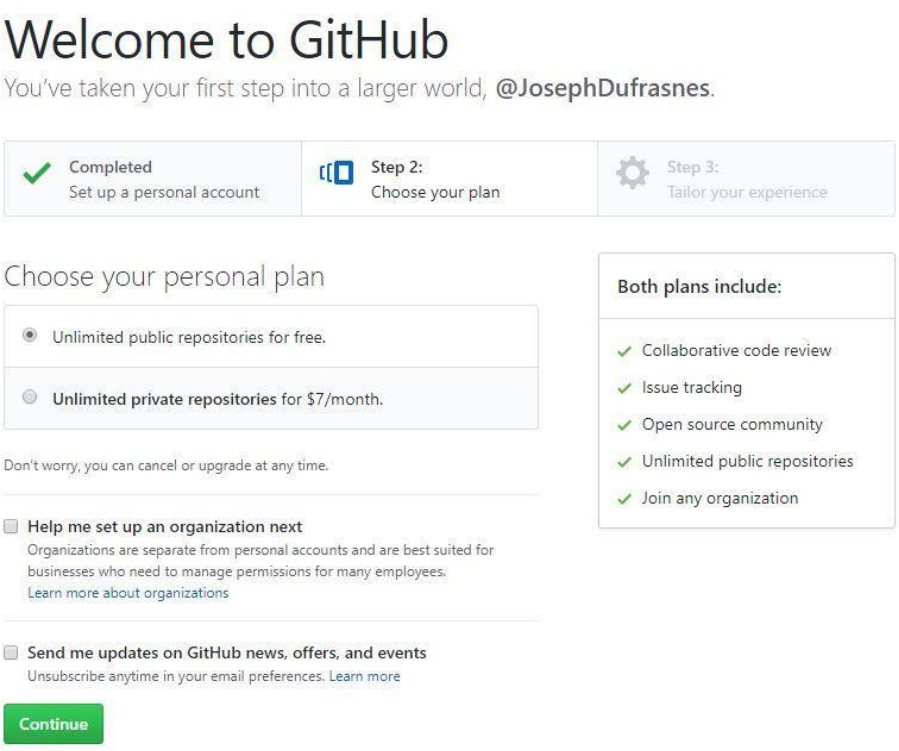
Step 3:Verify Email By Code

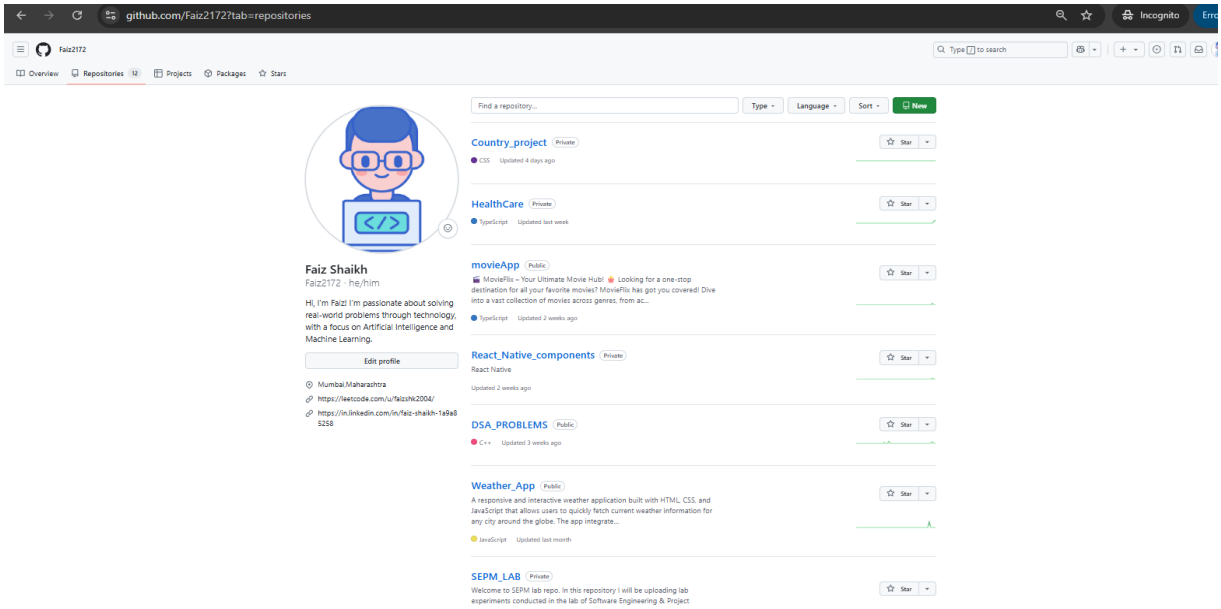






Step 4: Select your preference and Free plan





Conclusion:

In summary, Git is a powerful version control system that helps you track changes, collaborate with others, and maintain the integrity of your codebase. GitHub is an online platform that makes it easy to host and share Git repositories, enabling collaboration and project management. Setting up Git and GitHub allows you to contribute to open-source projects, manage your own code, and work effectively with others in a development environment.