Lab 14: Introduction to GitHub Cloning Repository

Lab 13: Introduction to GitHub and Cloning Repository

Objective

1. Understand the fundamentals of Git as a version control system.
2. Learn how to connect local projects to GitHub.
3. Practice committing changes, pushing code, pulling remote changes, and resolving conflicts.
4. Understand repository structure, staging, committing, branching, and remote synchronization.
5. Explore collaborative workflows used in software development teams.

Reference Material

- Lecture Notes: Introduction to Version Control
- Textbook: *Practical Git and GitHub*
- Classic Reference: *Pro Git* (Scott Chacon

Latest References (2024–2025):

- GitHub Docs: *Getting Started with GitHub*
- Atlassian Git Tutorial
- Git Documentation – Command Reference

Activity Timeline

Activity Name	Time
Introduction & Overview (Git, GitHub, Workflow)	30 minutes
Walkthrough Task 1: Cloning + Project Setup	40 minutes
Walkthrough Task 2: Commit, Push & Pull Changes	50 minutes
Practice Tasks (Git Workflow Scenarios)	40 minutes
Conclusion & Discussion (Collaboration & CI/CD)	20 minutes
Total	180 minutes

Introduction:

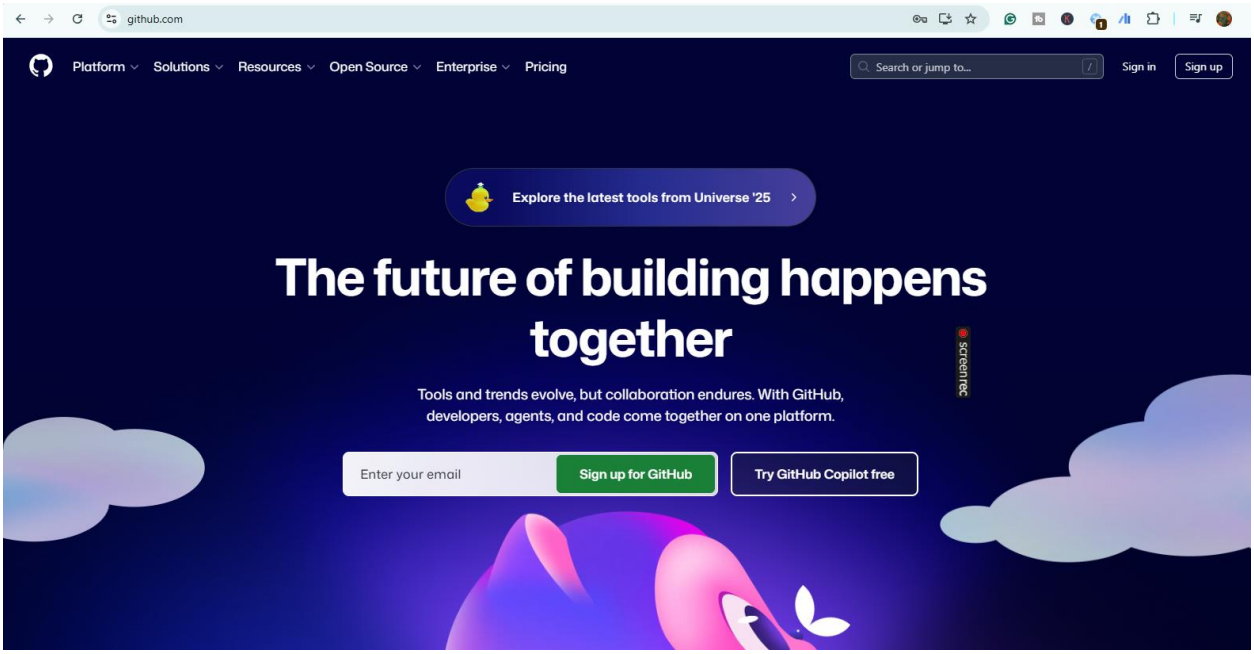
Version control is an essential component of modern software development. Git allows developers to track changes, maintain revisions, and collaborate efficiently across distributed environments.

GitHub is the most widely used hosting platform for Git repositories. It provides online repository management, collaboration features, pull requests, issue tracking, reviews, and integration pipelines.

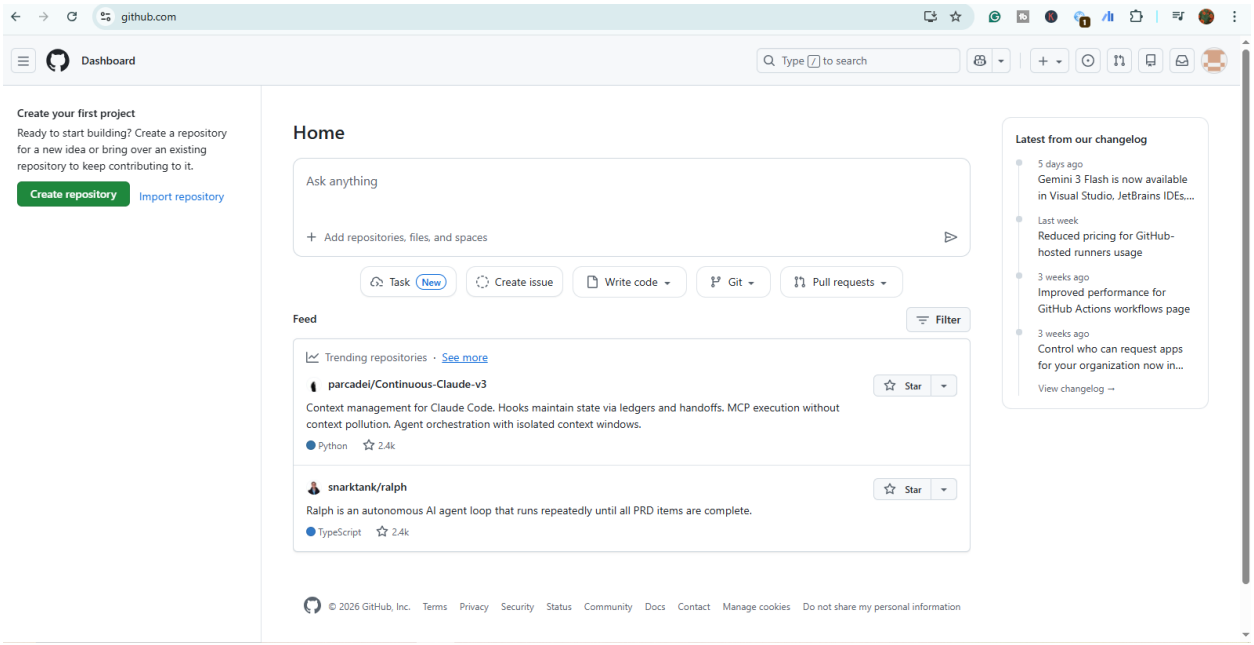
In this lab, students will learn how to work with Git and GitHub, manage code versions, commit changes, and synchronize local projects with remote repositories.

Walkthrough Task 1:

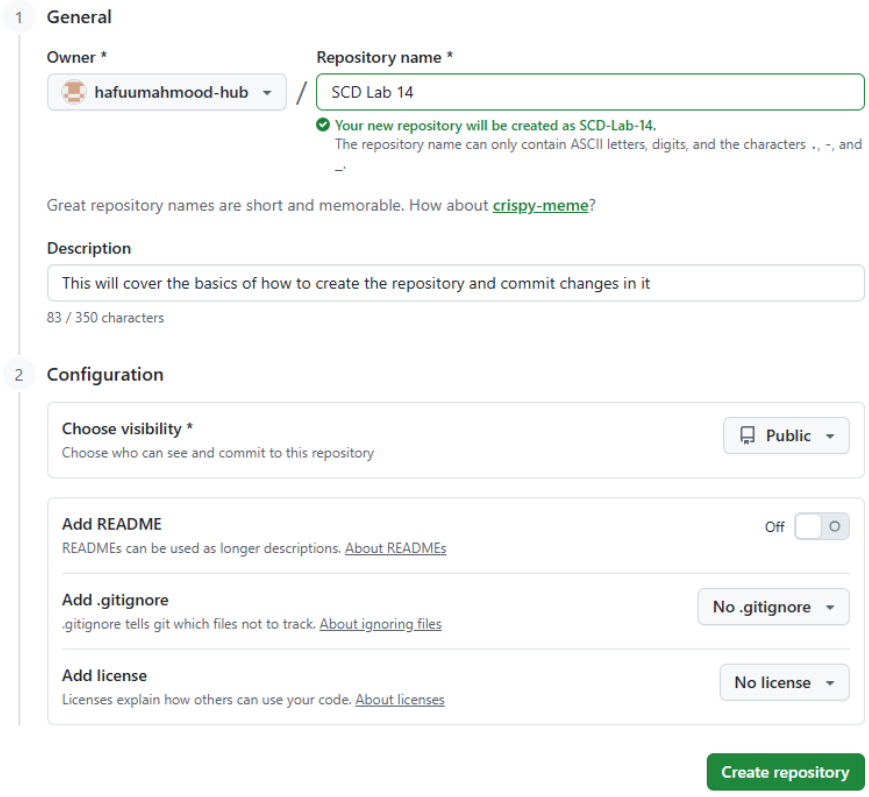
Create account on Github



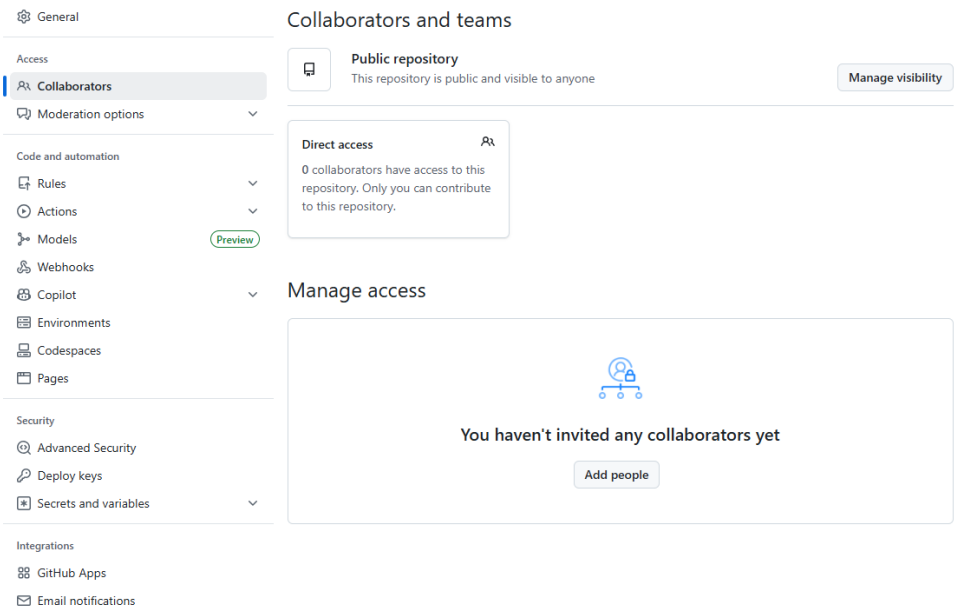
The dashboard of github will look like below



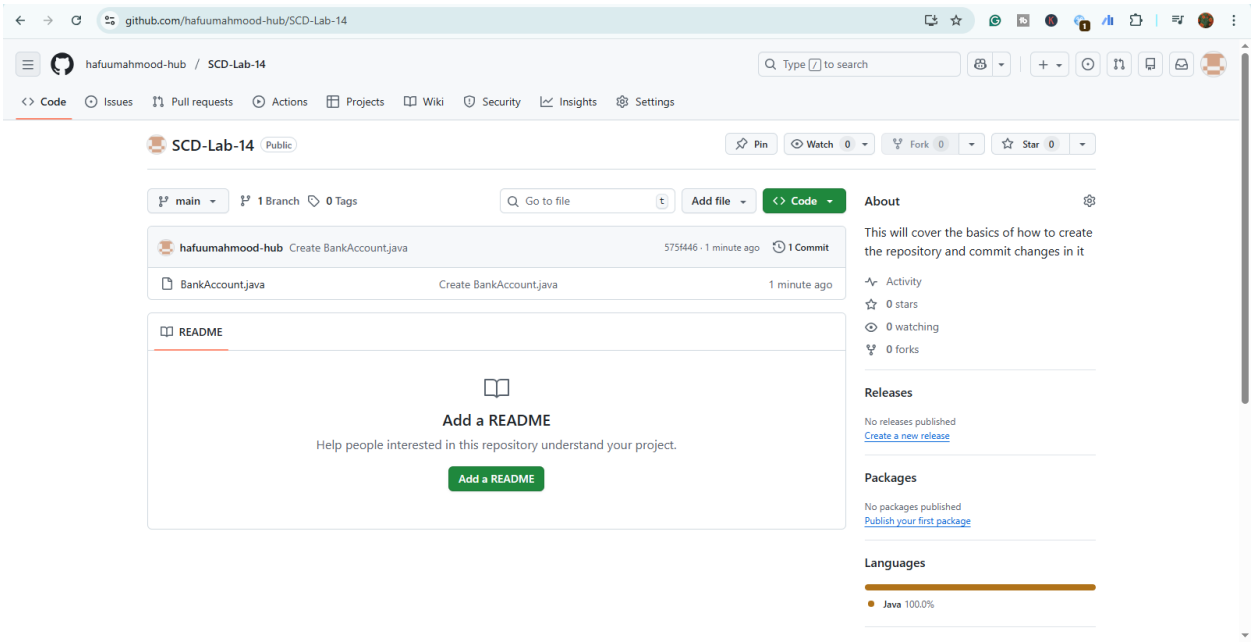
Create a repository



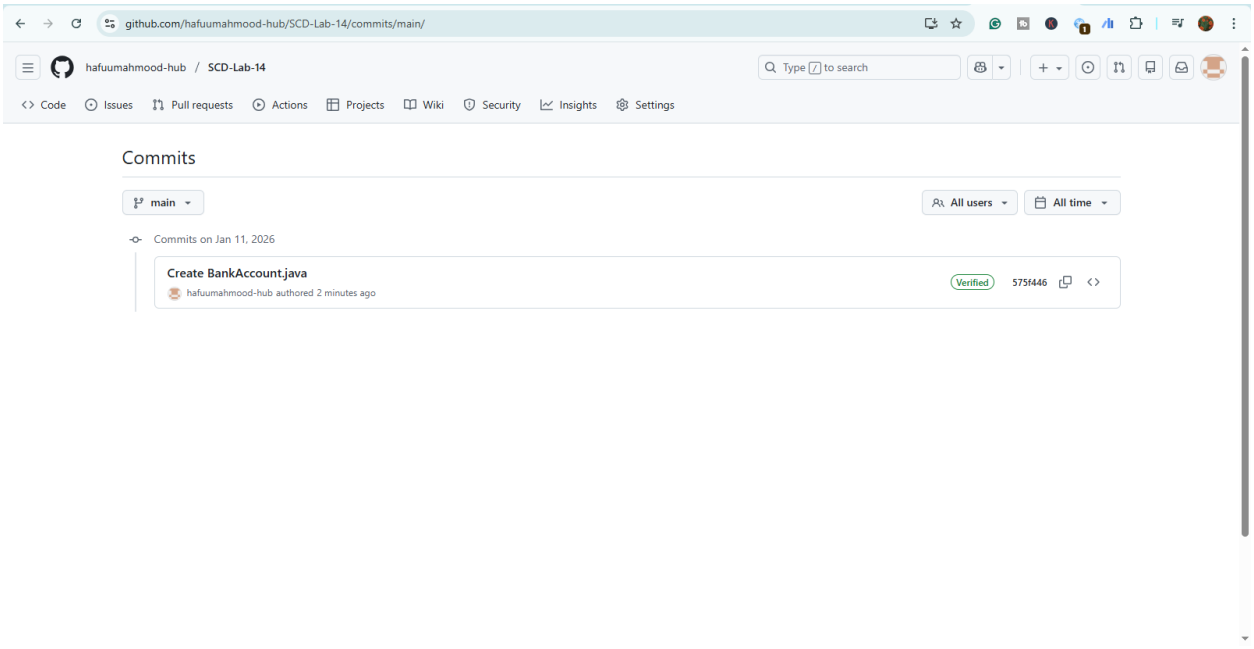
After that, invite the collaborators (teammates that can make changes to your project)



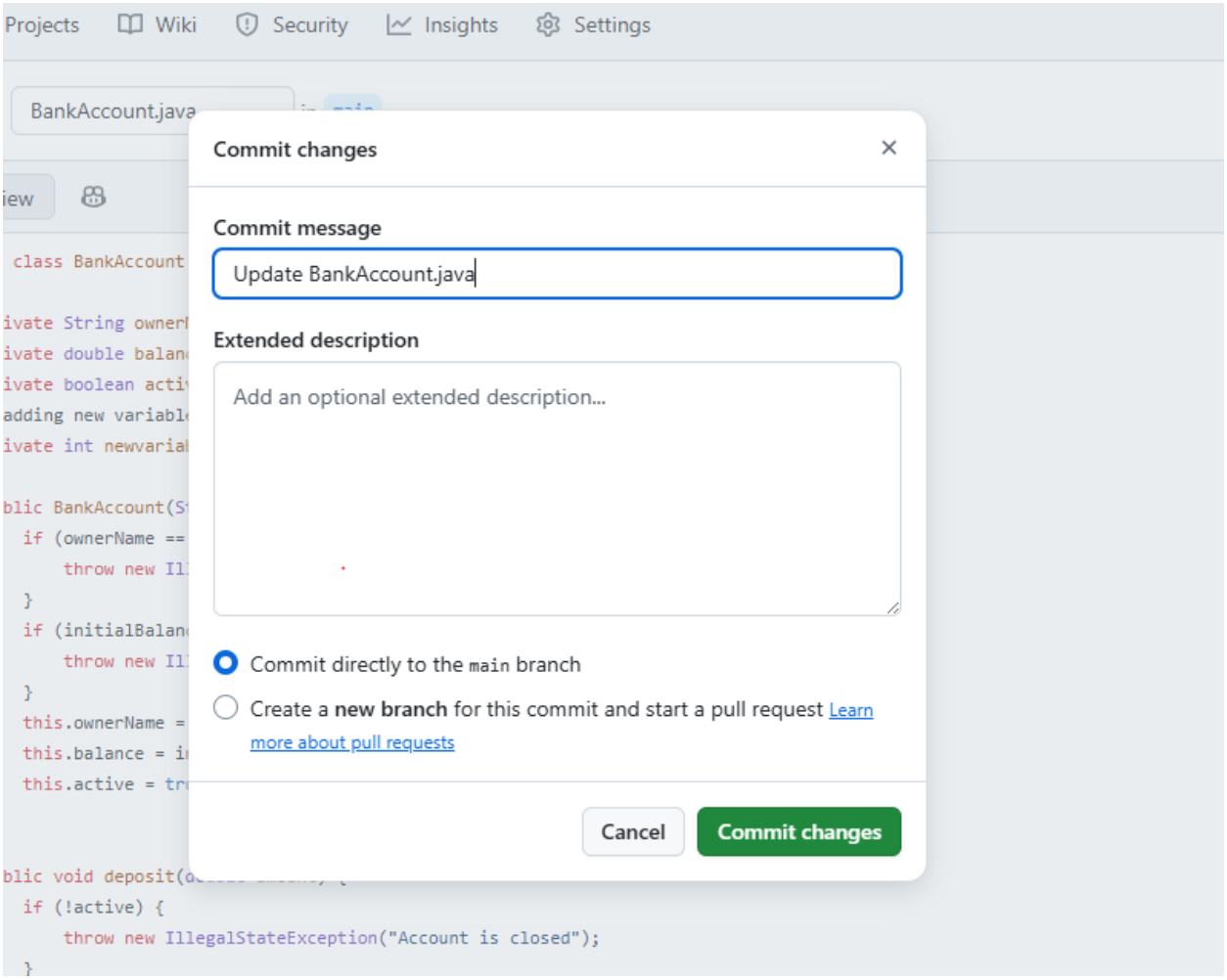
Add your code files inside the project.



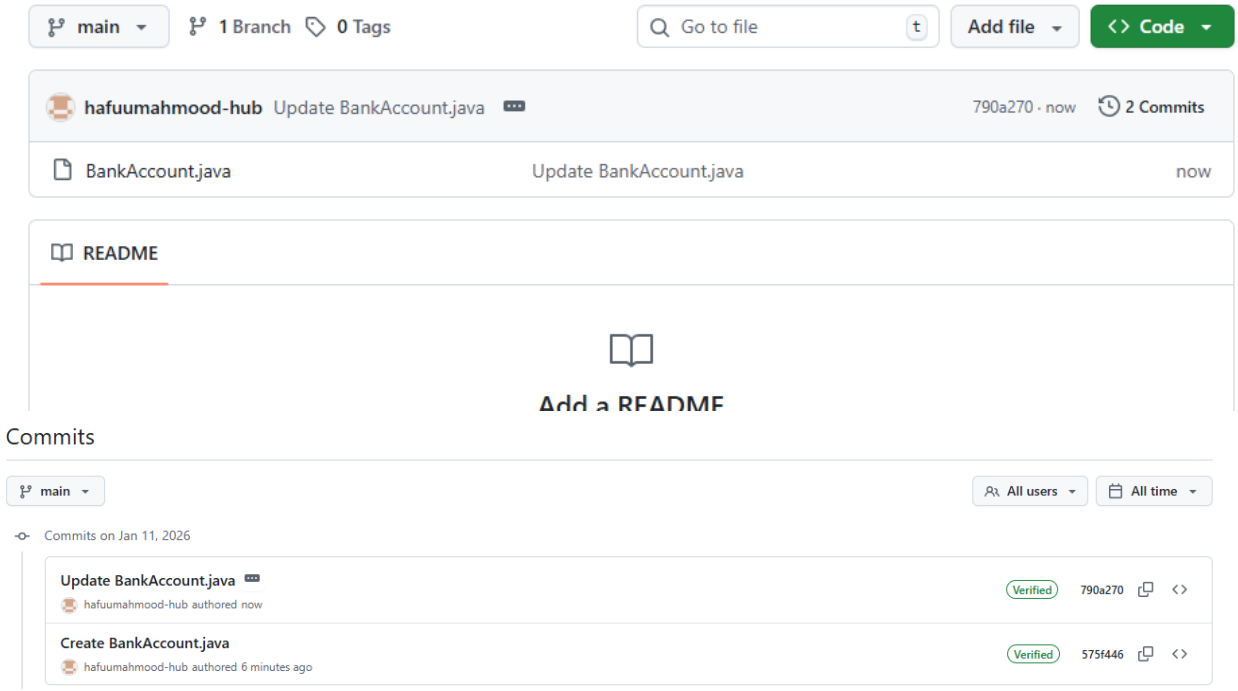
Click on commits to see the commit history



Now do some changes in the code



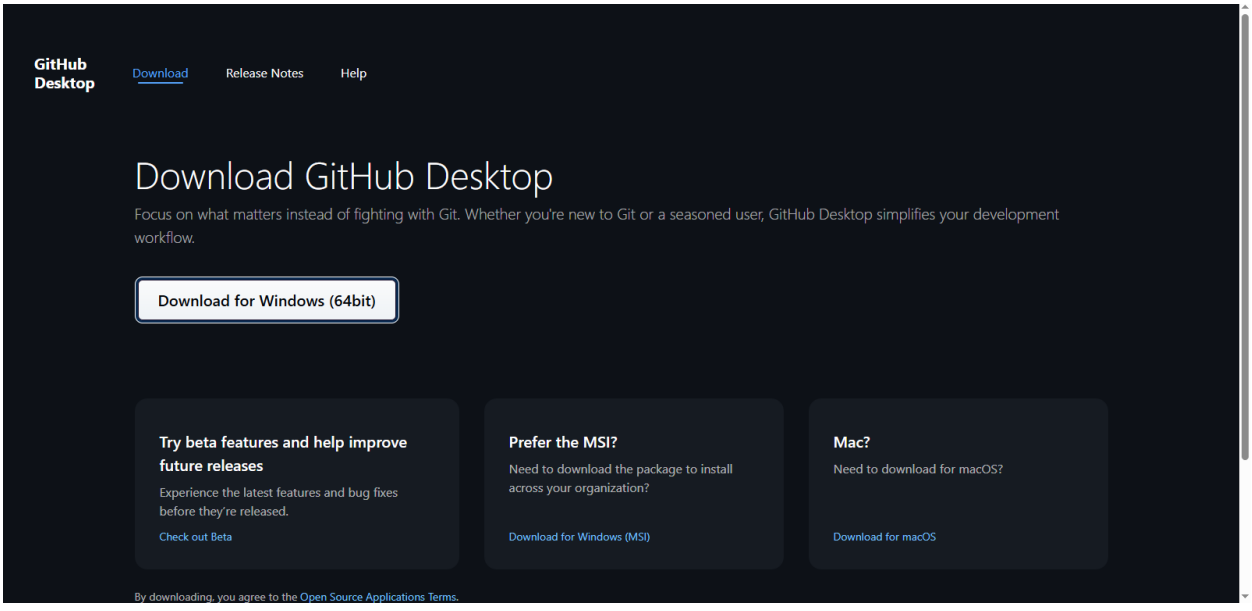
Now you will see two commits on the main code page



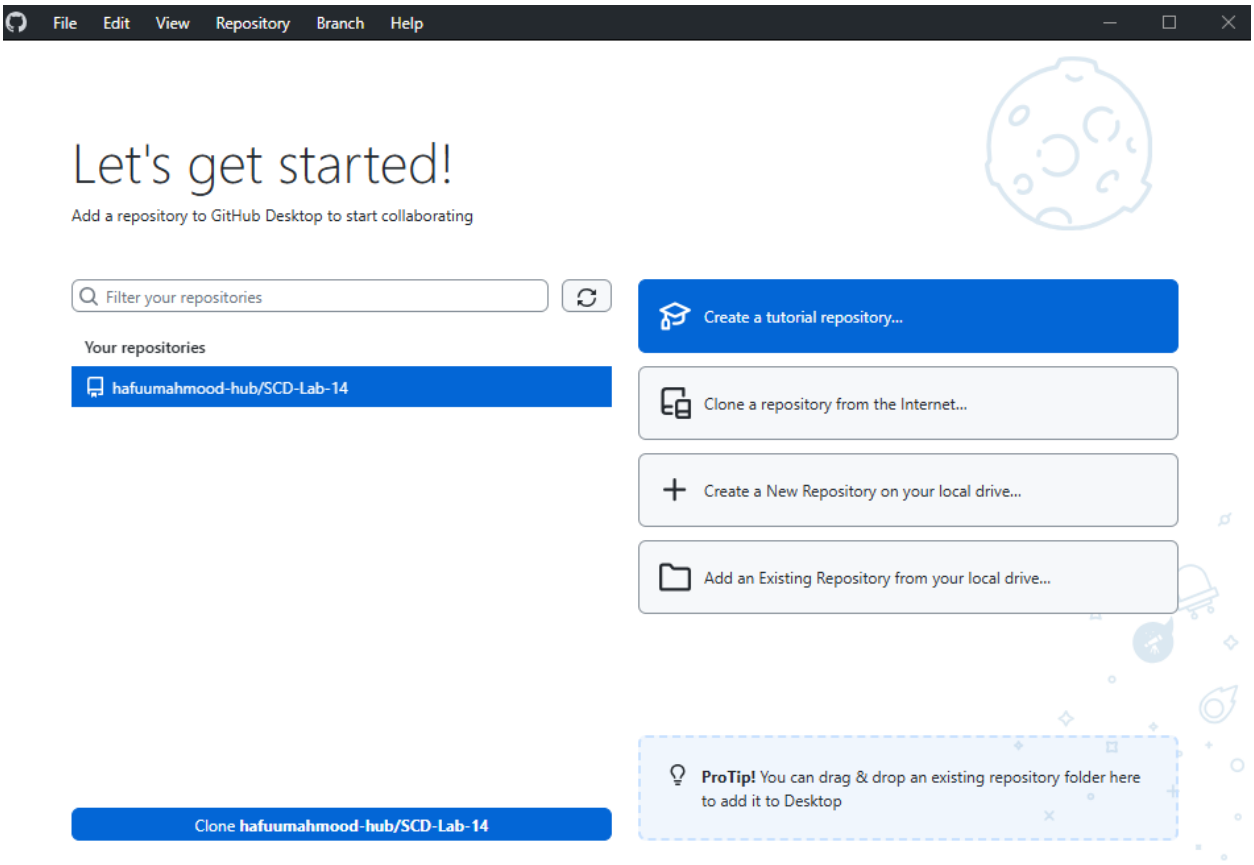
Walkthrough Task 2:

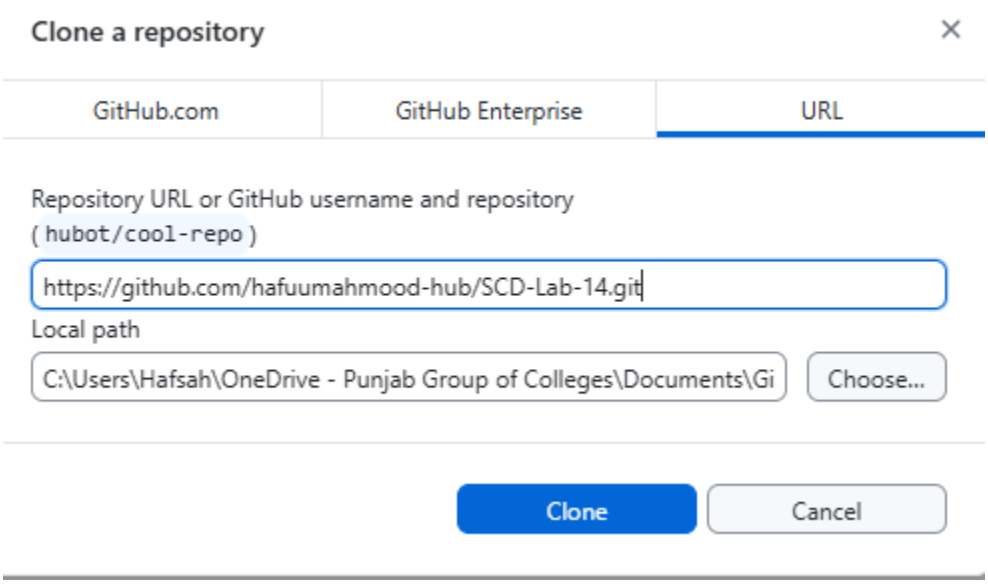
Using GitHub Desktop

Download GitHub Desktop

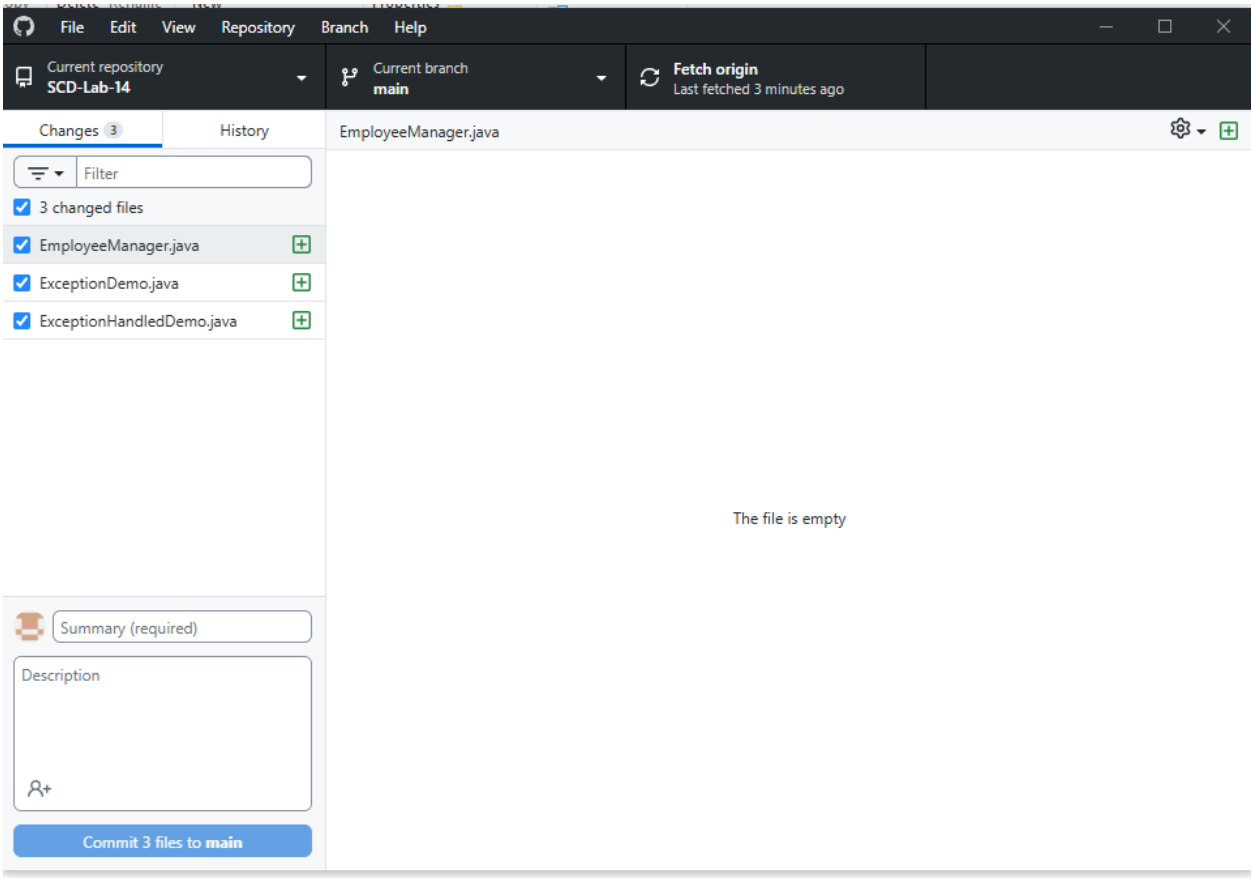


Clone the repository that you created in your online github account to your laptop/PC

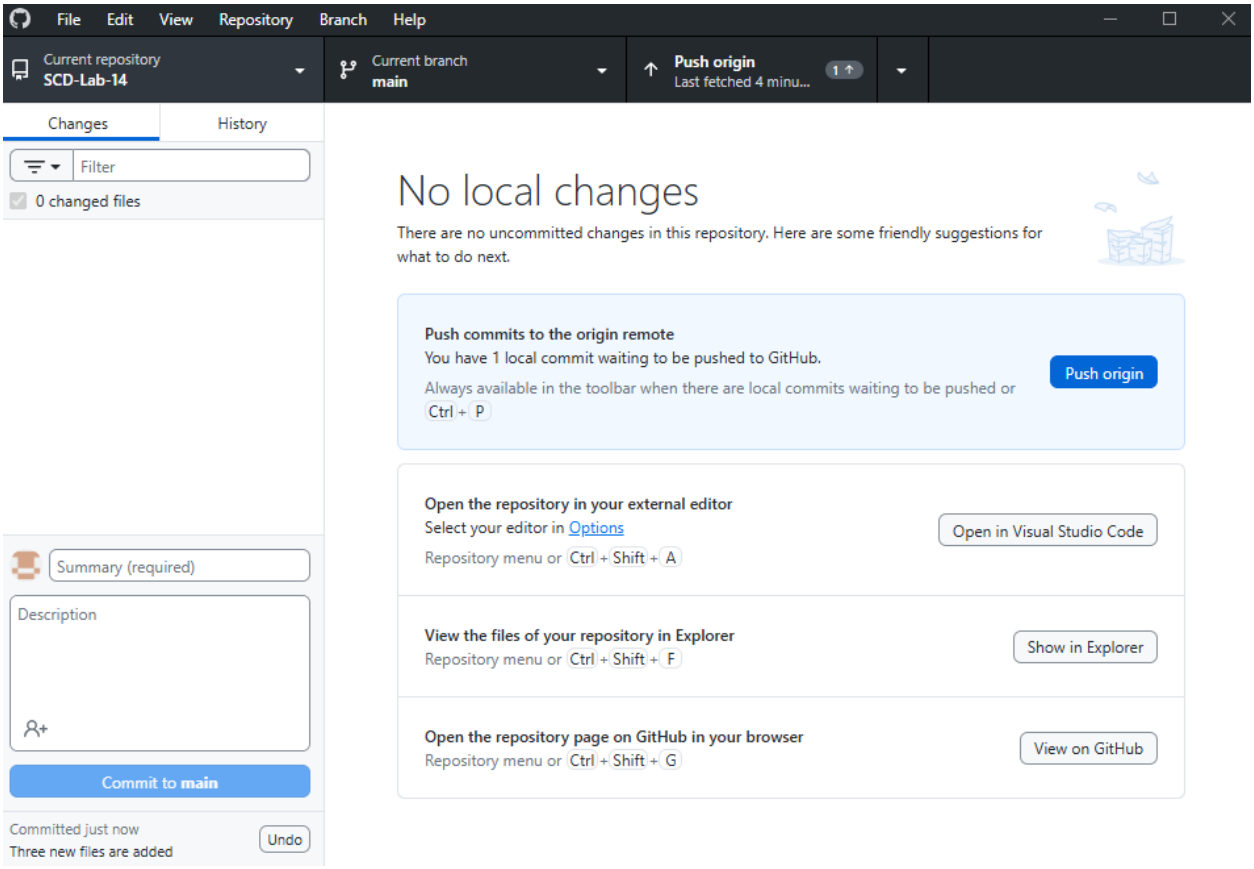




Now copy some of the files in your project folder

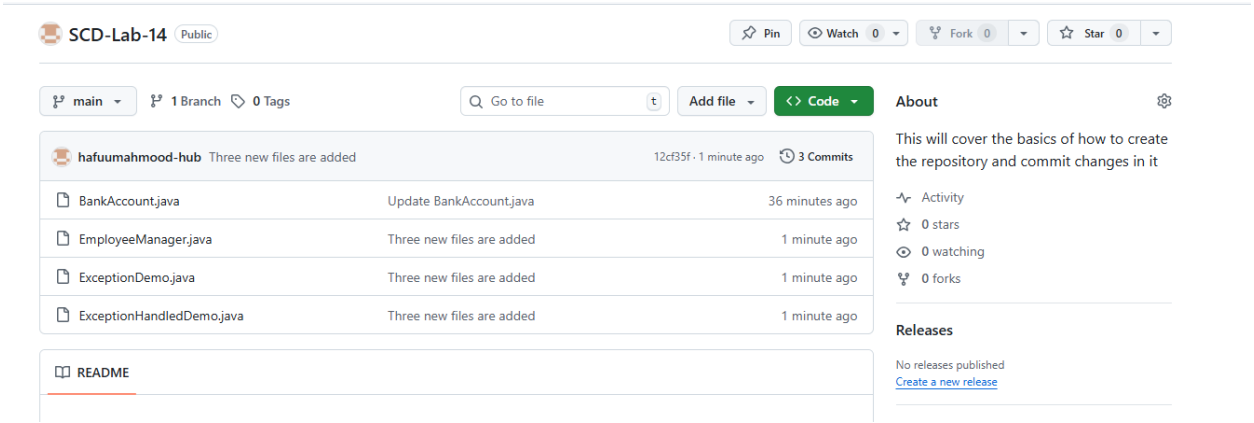


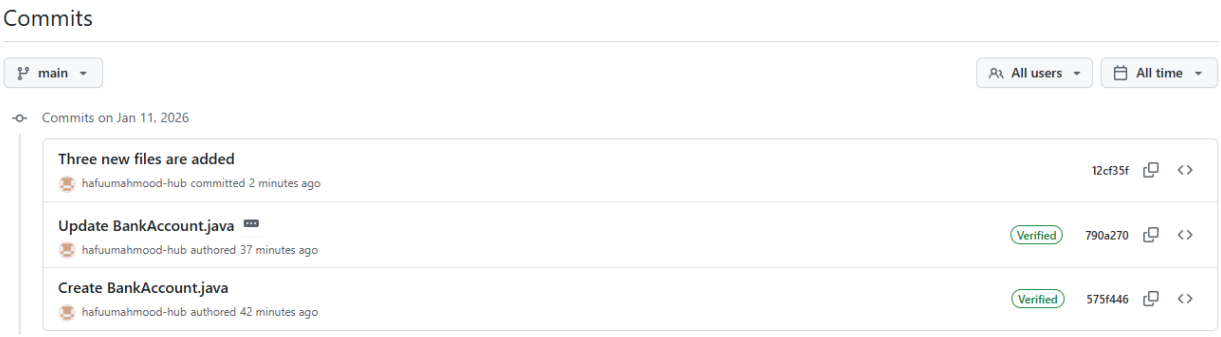
Now commit these new added files to main



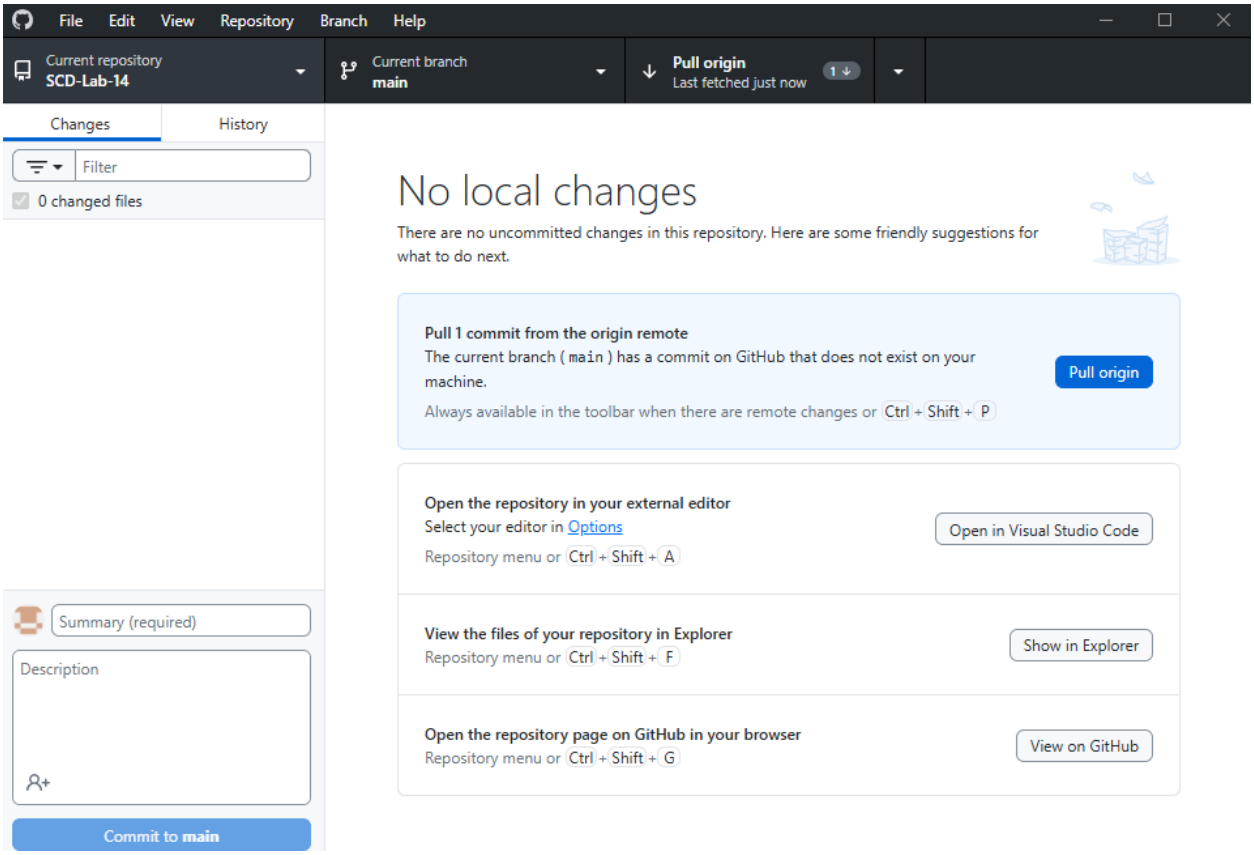
Push the commits to origin remote

After that your original repository on the internet will also reflect the changes that you made inside your system.





Now if you make any changes on the script on your web GitHub you will need to pull those changes into your desktop app.



Click on pull origin and your both directories will be synced.

Practice Tasks:

Practice Task 1: Team-Based Git Collaboration on a Mini Application

Objective

To enable students to collaboratively build different modules of a small management system while using GitHub as the shared version control platform for integration, conflict handling, and synchronization

Task Description

Students are required to perform the following steps:

1. **Form teams of 3–4 members.**
2. **Select one mini application to develop**, such as (examples):
    - Library Management System
    - Inventory Management System
    - Course Registration System
    - Hostel/Room Allocation System
    - Clinic Appointment System
    - Vehicle Service Tracking System
    - Simple Payroll System

3. **Assign modules to each member**, such as:
    - Member A → User/Student/Employee Module
    - Member B → Resource/Inventory/Asset Module
    - Member C → Records/Transactions/Bookings
    - Member D (optional) → Reporting / Authentication / Admin Panel
4. **Create a single GitHub repository** for the team and invite all members as collaborators.
5. Each member must:
    - Clone the repository
    - Develop their assigned module locally
    - Commit changes with meaningful messages
    - Push changes to the main project
6. Teams must create **at least one separate branch per member**, such as:

a) module-user
b) module-inventory
c) module-transactions
d) module-report

7. System must include **at minimum**:
    - Basic CRUD operations for each module
    - Input validation & error handling
    - Simple console or GUI interface
8. Final submission artifacts:

- Repository link
- Screenshots of commits
- Screenshot of merges

Outcomes:

- Students will understand version control concepts using Git.
- Students will be able to stage, commit, push, and pull code using GitHub.
- Students will handle real-world synchronization issues and merge conflicts.
- Students will collaborate through GitHub repositories.
- Students will appreciate GitHub as a professional development, deployment, and teamwork tool.