ASSIGNMENT 1

Computer Networks

Tejas Lohia, Umang Shikarvar
23110335, 23110301

TABLE OF CONTENTS

Contents

Chapter 1

DNS Resolver

1.1 Introduction

The aim of the laboratory session was to provide a hands-on introduction to version control using Git and GitHub. The activities focused on core concepts—initializing repositories, staging and committing changes, and synchronizing work—while also highlighting good practices for collaboration and project organization.

1.2 Tools

- **Programming Language:** Python 3.12.9 — Used for code and using pylint library.
- **Editor/IDE:** Visual Studio Code — Used for coding, debugging and execution.
- **Version Control:** Git and GitHub — Used to track and the changes in the code, and to improve maintainability of the codebases.
- **Linting Tool:** Pylint — Used to ensure that the code follows PEP8 standards.
- **Automation Platform:** GitHub Actions — used to automate the linting workflow and provide continuous integration (CI) feedback.
- **Virtual Environment:** venv — To prevent library version conflicts by isolating working environments.

1.3 Setup

To configure GitHub for this project, I had to setup git on my machine and Visual Studio Code (VS Code) as the code editor. For github setup I had to sign up using email ID and password;
To maintain isolated coding environment, created a new vevn **lab1** with python version **3.12.9**.

1.4 Methodology and Execution

The following methodology was followed execute tasks.

Part A: Initial Compilation and Git Setup

1. **Git initialization:**

   - Installed git on MacOS, and checked git version:

     ```
     git --version
     ```

2. **Version Control Initialization:**

   - Configured Git global username and email:

     ```
     git config --global user.name "TejasLohia21"
     git config --global user.email 23110335@iitgn.ac.in
     ```

   - Verified configurations:

     ```
     git config --list
     git config user.name
     git config user.email
     ```

3. **Repository Setup in a New Folder:**

   - A new folder named `testing_lab1` was created and entered:

     ```
     cd testing_lab1
     ```

   - A Git repository was initialized inside the folder:

     ```
     git init
     ```

   - A `README.md` file was created and initialized with content:

     ```
     echo "# read_test" > README.md
     ```

   - The file was staged and committed with a message:

     ```
     git add README.md
     git commit -m "add readme"
     ```

4. **Repository Setup in a New Folder:**

   - A new folder named `testing_lab1` was created and entered:

     ```
     cd testing_lab1
     ```

   - A Git repository was initialized inside the folder:

     ```
     git init
     ```

   - A `README.md` file was created and initialized with content:

     ```
     echo "# read_test" > README.md
     ```

   - The file was staged and committed with a message:

     ```
     git add README.md
     git commit -m "add readme"
     ```

5. **Pushing Code to GitHub:**

   - The local commits were pushed to the remote repository using:

     ```
     git push -u origin main
     ```

   - The `-u` flag sets the upstream branch so that subsequent pushes can simply use `git push`.

6. **Checking Commit History:**

   - The commit history was viewed using:

     ```
     git log
     ```

   - Output:

     ```
     commit 7e4d7da93c58274df903e73f653de9e9fe8e84fb (HEAD -> main)
     Author: TejasLohia21 <23110335@iitgn.ac.in>
     Date:   Sat Sep 6 01:00:00 2025 +0530

         add readme
     ```

Part C: Working with Remote Repositories

1. **Connecting to GitHub:**
   - A new repository was created on GitHub named `TejasLohialab1`.
   - The local repository was linked to GitHub using:

```
git remote add origin git@github.com:TejasLohia21/TejasLohialab1.git
git branch -M main
git push -u origin main
```

2. **Pushing Changes to GitHub:**
   - The committed changes were pushed to GitHub using:

```
git push -u origin main
```

3. **Cloning a Repository:**
   - An existing repository was cloned from GitHub to the local machine using:

```
git clone git@github.com:TejasLohia21/datascience-HNSW.git
```

4. **Pulling Changes:**
   - Updates from the remote repository were pulled using:

```
git pull origin main
```

Part D: Setting up Pylint Workflow with GitHub Actions

In this part, a continuous integration (CI) workflow was created using **GitHub Actions** to automatically check the Python code using `pylint`. The steps followed were:

1. **Python Script Creation:** A Python file `code.py` was created with more than 30 lines of code. The script implemented functions for factorial calculation, Fibonacci sequence generation, and prime number detection.
2. **Workflow Configuration:** A workflow file was created at the path:

```
.github/workflows/pylint.yml
```

   The content of the workflow file is shown below:

```
name: Pylint Check

on: [push, pull_request]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.12'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pylint

      - name: Run pylint
        run: |
          pylint code.py
```

3. **Commit and Push:** The workflow file and the Python script were staged, committed, and pushed to the GitHub repository:

```
git add main.py .github/workflows/pylint.yml
git commit -m "Add Python script and pylint workflow"
git push
```

4. **Verification:** After pushing, the GitHub Actions workflow was triggered. The Python script was linted using `pylint`, and all errors were resolved until a green tick (✓) appeared, confirming successful execution.

This ensured that the code followed Python coding standards and passed linting checks automatically on every push.

1.5 Results and Analysis

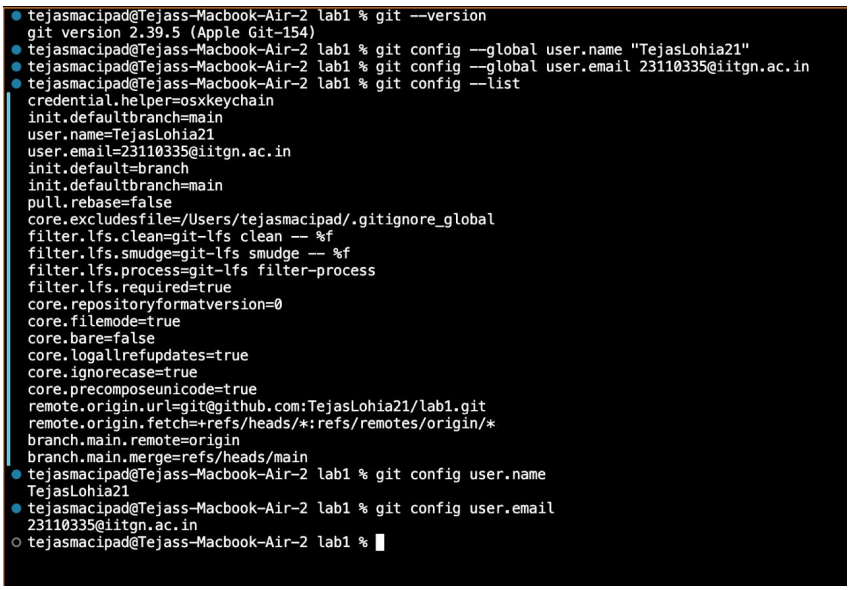

Figure 1.1: Setting up git

Commands to check the version and verify initialization gave the following outputs.

- Git version initialized: 2.39.5 (Apple Git - 154)
- Git username: TejasLohia21
- Git user email: 23110335@iitgn.ac.in

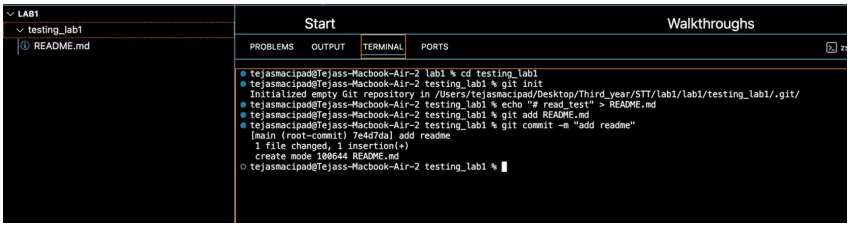

Figure 1.2: Init git Repo and addition of README file

Upon committing the staged files in the main branch of the initialized local git repository:

```
[main (root-commit) 7e4d7da] add readme
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

Figure 1.3: Checking commit history

Command git log generated the commit history along with Metadata.

```
Author: TejasLohia21 <23110335@iitgn.ac.in>
Date: Sat Sep 6 01:00:00 2025 +530

    add readme
```

Figure 1.4: Linking local repositories with github

Created a new repository on github named lab1. Using the command, the local repository was linked to Online Repo. The repository was then pushed to the online repository on github in the main branch.

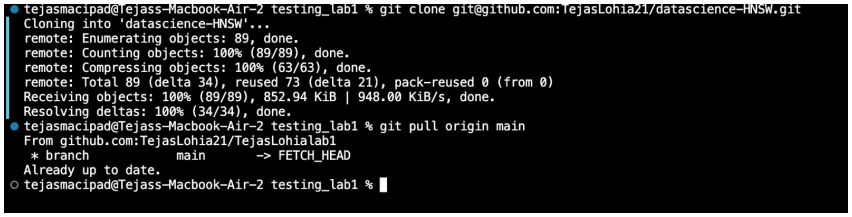

Figure 1.5: Cloning existing repositories and initiating pull

To get a hands on experience of pulling repositories, I cloned an existing repository and pulled that. As the repository did not have any changes, the output was 'Already upto date'.

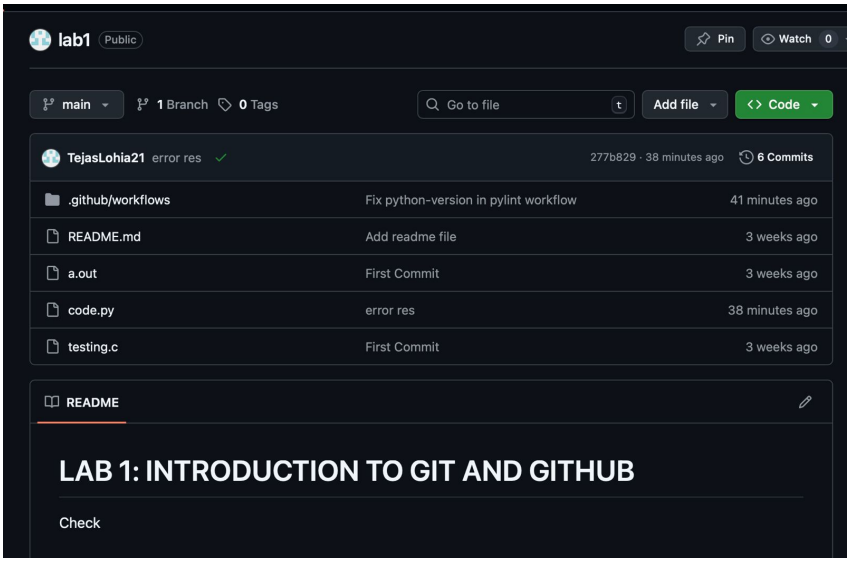

Figure 1.6: Pylint workflow

Initially there was a cross mark after pushing to the online repository. THe error was because of no blank empty line in the end.
After rectifying this, there was a tick symbol in the github repository.

1.6 Discussion and Conclusion

This lab was quite important to understand and learn git in a systematic way. It introduced us to the very basics of git init, till using github workflows, making us familiar to use and maintain repositories using git and github.