MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Software Tools And Technology

Lab Notebook

Assignment Details

- Create a Git Repository Containing a Lab Notebook in LaTeX Format.
- Group No. : 28
- Github Repository: `https://github.com/ThisIsSidam/SEC-Group-28-Project`

Name	**Dept.**	**Roll No.**
Anshu Kumar Singh	BSc in AI	30054623011
Aniket Gupta	BSc in AI	30054623015
Arup Debnath	BCA	30001223052
Debyani Sarkar	BSc in Forensic Sciences	30059223033
Jeet Majumdar	BCA	30001223079

Acknowledgment

I would like to express my heartfelt gratitude to Ayan Sir for his guidance and assignments related to GitHub, which have greatly enhanced my understanding of version control and collaboration in software development. I also extend my sincere thanks to Pabitra Sir for introducing me to LaTeX and guiding me through the process of learning the use of Latex.

This document was created in LaTeX, and the project involved teamwork on GitHub, where each team member contributed to the successful completion of this assignment.

Signature

Table of Contents

1 Introduction To Git

1.1 Objective

Git is a distributed version control system designed to manage and track changes in source code during software development. It was created by Linus Torvalds in 2005 to support the development of the Linux kernel. Git is known for its efficiency, performance, and reliability in handling projects of all sizes, from small personal projects to large-scale enterprise applications.

1.2 Key Features of Git

1. **Distributed Version Control**: Every developer has a local copy of the entire project history, including all branches and commits. This allows for fast access to project data and enables developers to work offline.
2. **Branching and Merging**: Git allows developers to create branches to work on different features or bug fixes independently. Branches can be merged back into the main codebase once the work is complete, enabling a smooth integration process.
3. **Commit History**: Changes in Git are recorded as commits. Each commit represents a snapshot of the project at a specific point in time and includes a unique identifier and a message describing the changes. This history is crucial for tracking the evolution of the project and understanding the rationale behind changes.
4. **Staging Area**: Git uses a staging area to prepare changes for a commit. Developers can add specific changes to the staging area, allowing them to group related changes into a single commit.
5. **Collaboration**: Git supports collaborative workflows through features like pull requests and code reviews. Pull requests allow contributors to propose changes, which can be reviewed and discussed before being merged into the main codebase.
6. **Conflict Resolution**: Git provides tools to help manage and resolve conflicts that arise when multiple developers make changes to the same part of the codebase.
7. **Performance**: Git is optimized for performance, handling large projects and repositories efficiently. It uses compression techniques to minimize the storage required for the project history.

1.3 Using Git and GitHub with GitHub Desktop

1.3.1 Materials Required

- A computer with internet access
- GitHub Desktop installed (Download GitHub Desktop)
- A GitHub account (Create a GitHub account)

1.3.2 Experiment Steps

Installing GitHub Desktop

1. Download and install GitHub Desktop from here.

Setting Up GitHub Desktop

1. Open GitHub Desktop and sign in with your GitHub account.

Cloning a Repository

1. Select ”Clone Repository” from the ”File” menu.
2. Choose a repository from GitHub or enter the repository URL.

Creating a New Repository

1. Select ”New Repository” from the ”File” menu.
2. Enter repository details and click ”Create.”

Making Changes and Committing

1. Edit files in your preferred editor.
2. Stage and commit changes in GitHub Desktop.

Pushing Changes to GitHub

1. Click ”Push origin” to upload changes to GitHub.

Creating and Switching Branches

1. Click the branch name to create or switch branches.

Creating a Pull Request

1. Push the branch, then click ”Create Pull Request” in GitHub Desktop.

Merging Pull Requests

1. Review and merge the pull request on GitHub’s website.

1.4 Observations and Results

- Successfully installed and set up GitHub Desktop.
- Cloned existing repositories and created new repositories.
- Made and committed changes to local repositories.
- Pushed changes to remote repositories on GitHub.
- Created and managed branches.
- Created and merged pull requests.

1.5 Conclusion

Using Git and GitHub through GitHub Desktop simplifies version control and collaborative development by providing a user-friendly interface for managing repositories, making commits, and handling pull requests.

2 Introduction to Latex

2.1 Objective

Latex is a high-quality typesetting system used for the creation of technical and scientific documents. It separates content from formatting, allowing users to focus on writing while Latex handles the layout. Its ability to handle complex documents, such as theses or research papers, makes it the go-to choice for many academic and professional writers.

2.2 Why Use Latex?

Some key benefits of using Latex include:

- High-quality typesetting
- Perfect for documents containing mathematical symbols and formulas
- Automatic management of cross-references, citations, and tables of contents
- Free and cross-platform
- Excellent for large documents, such as books or reports

2.3 Basic Commands in Latex

Here are some fundamental Latex commands:

- `\documentclass{article}`: Defines the document type.
- `\usepackage{package}`: Adds packages for extra functionality.
- `\title{Title}`, `\author{Author}`, `\date{}`: Define title, author, and date.
- `\section{Section Title}`, `\subsection{Subsection Title}`: Adds sections and subsections.
- `\textbf{bold}`, `\textit{italic}`: For bold and italic text.

2.4 Mathematical Typesetting

Latex excels at handling mathematical formulas, such as the well-known Einstein equation:

$$E = mc^2 \tag{1}$$

Inline mathematical expressions can be written like this: $a^2 + b^2 = c^2$.

2.5 Steps to Compile a Latex Document

Compiling a Latex document is the process of transforming the `.tex` source file into a formatted document, typically a PDF. Here's how you can compile your document:

2.5.1 Writing the Document

Write your document in a plain text editor like `Notepad`, `TeXShop` (Mac), or a dedicated LaTeX editor like `Overleaf`, `TeXworks`, or `Kile`. Save the file with a `.tex` extension.

2.5.2 Using an Editor or Compiler

- Overleaf: Upload your `.tex` file to Overleaf, an online LaTeX editor. The document will compile automatically.

2.5.3 Compiling the Document

To compile the `.tex` file into a PDF:

- In Overleaf: The compilation happens automatically. You can download the final PDF once it compiles.

2.5.4 Viewing the PDF

After compiling, your LaTeX editor or Overleaf will usually display the resulting PDF. If you're working locally, you can open the generated PDF using any PDF viewer.

2.6 Conclusion

This document provided an introduction to Latex and detailed steps on how to compile a document. With the right tools and a bit of practice, we will be able to create professional documents with ease.

3 Calculator in C

3.1 Objective

The objective of this lab is to develop a basic calculator program using the C programming language. The calculator will perform simple arithmetic operations like addition, subtraction, multiplication, and division based on user input.

3.2 Program Overview

The calculator program is designed to:

- Accept two numbers from the user.
- Prompt the user to select an arithmetic operation (Addition, Subtraction, Multiplication, Division).
- Perform the selected operation.
- Display the result of the operation to the user.

The program includes error handling to manage division by zero and other invalid inputs.

3.3 Code Implementation

The following is the C code for the calculator program:

Listing 1: Calculator Program in C

```c
#include <stdio.h>

int main() {
    char operator;
    double num1, num2, result;

    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator);

    printf("Enter two operands: ");
    scanf("%lf %lf", &num1, &num2);

    switch(operator) {
        case '+':
            result = num1 + num2;
            break;
        case '-':
            result = num1 - num2;
            break;
        case '*':
            result = num1 * num2;
            break;
        case '/':
            if (num2 != 0)
                result = num1 / num2;
            else {
                printf("Error! Division by zero.\n");
                return -1;
            }
            break;
        default:
            printf("Error! Operator is not correct\n");
            return -1;
    }

    printf("Result: %.2lf\n", result);
    return 0;
}
```

3.4 Initialize a Local Git Repository

1. Open **GitHub Desktop**.
2. Click on **File New Repository**.
3. Fill in the repository details:
   - : Name the repository.
   - : Choose or create a directory where the **calculator.c** file is located.
   - Select a template to ignore certain files.
4. Click **Create Repository**. This initializes the repository and opens it in GitHub Desktop.

3.5 Add the File to the Repository

1. In **GitHub Desktop**, see the `calculator.c` file listed under the **Changes** tab on the left side.
2. Add a summary and optional description for your commit in the **Summary** and **Description** fields at the bottom left.

3.6 Commit the Changes

1. After reviewing the changes, click **Commit to main** (or **master**, depending on the default branch name) in the lower-left corner.

3.7 Push the Changes to GitHub

1. After committing the changes, then publish the repository:
   - Click on the **Publish repository** button in the upper-right corner.
   - If this is a new repository, GitHub Desktop will prompt to enter a name for the remote repository and optionally a description.
   - Click **Publish repository** to push the local commits to GitHub.

3.8 Verify the Upload

1. Open the web browser and navigate to GitHub repository URL.
2. Verify that the `calculator.c` file is listed and accessible in the repository.

By following these steps in GitHub Desktop, one can be able to perform the same actions as in the GitHub CLI, but with a graphical user interface.

4 Mind Reader JAVA Program

4.1 Objective

The task is to change the button name from ”Submit” to ”Chin Tapak Dum Dum” in the *SymbolApp.java* program into a GitHub repository.

4.2 Program Overview

Process to Change the Button Name:

- To rename the button in the code, the label of the button needs to be changed from ”Submit” to the desired new label. In the code, the following line:

```
submitButton = new Button("Submit");
```

- Can be updated to reflect the new name for the button. The updated code would be:

```
submitButton = new Button("Chin Tapak Dum Dum");
```

- The result of the updated button name will be displayed to the user.

4.3 Code Implementation

```
import java.awt.*;
import java.awt.event.*;
import java.util.Random;

public class SymbolApp extends Frame implements ActionListener {
    private Label[] symbolLabels = new Label[99];
    private Button submitButton;
    private String specialSymbol;
    private String selectedSymbol;

    public SymbolApp() {
        // Generate a random special symbol
        Random rand = new Random();
        specialSymbol = Character.toString((char) (rand.nextInt(94)
        + 33)); // Random ASCII character from 33 to 126
        selectedSymbol = specialSymbol;

        // Setting up the main frame
        setLayout(new BorderLayout());
        setSize(800, 700);
        setTitle("Symbol App");

        // Adding instruction message
        TextArea instruction = new TextArea(
            "Think of any two digit number. Now reverse it and find
            the difference of them.\n" +
            "Now find the number you got and remember the symbol from
            the panel below.\n" +
            "Don’t tell me, I’ll read your mind! Hit the below button
            when you are ready to see the magic!",
            5, 60, TextArea.SCROLLBARS_NONE);
        instruction.setEditable(false);
        instruction.setFont(new Font("Arial", Font.PLAIN, 16));
        add(instruction, BorderLayout.NORTH);

        // Panel for symbols
        Panel symbolPanel = new Panel(new GridLayout(11, 9));
        for (int i = 0; i < 99; i++) {
            String symbol = (i % 9 == 0) ? specialSymbol : Character
            .toString((char) (33 + (i % 94)));
            symbolLabels[i] = new Label(i + ": " + symbol);
            symbolLabels[i].setAlignment(Label.CENTER);
            symbolPanel.add(symbolLabels[i]);
        }
        add(symbolPanel, BorderLayout.CENTER);

        // Panel for submit button
        Panel controlPanel = new Panel(new FlowLayout());
        submitButton = new Button("Chin Tapak Dum Dum");
        submitButton.addActionListener(this);
        controlPanel.add(submitButton);
        add(controlPanel, BorderLayout.SOUTH);

        // Setting up the window close event
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        // Clear the current content and display the selected symbol
        if (ae.getSource() == submitButton) {
            removeAll();
            setLayout(new BorderLayout());
            Label resultLabel = new Label(selectedSymbol, Label.CENTER);
            resultLabel.setFont(new Font("Arial", Font.BOLD, 50));
            add(resultLabel, BorderLayout.CENTER);
            validate();
            repaint();
        }
    }

    public static void main(String[] args) {
        new SymbolApp();
    }
}
```

4.4 Compiling and Running the Program

To run the Java Program:
Prerequisites:

- Ensure that the Java Development Kit (JDK) is installed. The JDK can be downloaded from Oracle's official website or installed using a package manager for the operating system in use.
- While not mandatory, it may be easier to work with an Integrated Development Environment (IDE) such as IntelliJ IDEA, Eclipse, or NetBeans.

Steps to Run the Program:

- **Install Java**

  On Windows:

  - Download the JDK from Oracle's website.
  - Install the JDK by following the instructions.
  - Set the `JAVA_HOME` environment variable and update the `Path` environment variable with the JDK's bin directory.
  - To verify the installation, open Command Prompt and run `java -version`.
- **Save the Program**
  - Open a text editor (e.g., Notepad, VSCode, or any Java IDE).
  - Save the file as `SymbolApp.java`.
- **Compile the Program**
  - Open a terminal or command prompt.
  - Navigate to the directory where the `SymbolApp.java` file is saved.

    ```
    cd path/to/your/java/file
    ```
  - Compile the program with:

    ```
    javac SymbolApp.java
    ```

    This will generate a `SymbolApp.class` file if there are no compilation errors.
- **Run the Program**
  - After successful compilation, run the program using:

    ```
    java SymbolApp
    ```
  - The application window will open, allowing interaction as described in the program.

4.5 Adding the JAVA Program to GitHub Repository via GitHub Desktop

To add this Java program to a GitHub repository using GitHub Desktop, follow these steps:

1. Open **GitHub Desktop**.
2. Clone your repository:
   (a) In the top-left corner, click **File** → **Clone Repository**.
   (b) Choose your repository from the list or paste the repository URL.
   (c) Select the local path where you want to clone the repository and click **Clone**.
3. Add the `SymbolApp.java` file to your local repository folder.
4. Return to **GitHub Desktop**.
5. In the **Changes** tab, you should see the `SymbolApp.java` file listed.
6. Commit the changes:
   (a) Enter a commit message such as `Rename button name in the JAVA program`.
   (b) Click the **Commit to master** button.
7. Push the changes to GitHub:
   (a) Click **Push origin** to upload the commit to your GitHub repository.
8. Verify the upload:
   (a) Visit your GitHub repository URL in a web browser.
   (b) Verify that the `SymbolApp.java` file is listed and accessible in the repository.

4.6 Conclusion

- The button's text has been changed from "Submit" to "Chin Tapak Dum Dum".
- The functionality of the button remains unchanged while updating its displayed label in the user interface.

5 Creating a LaTeX Repository on GitHub

5.1 Objective

The purpose of this lab notebook entry is to document the process of creating a LaTeX repository on GitHub using Overleaf for document creation and Git CLI for version control and repository management.

5.2 Requirements

- GitHub account
- Overleaf account
- Git installed locally on your system
- A LaTeX project created in Overleaf
- Command-line interface (CLI) access

5.3 Procedure

5.3.1 Step 1: Create and Edit LaTeX Project on Overleaf

1. Go to `https://www.overleaf.com` and create a new project.
2. Edit your LaTeX files using Overleaf's online editor.
3. Ensure your project has all necessary files, such as:
   - `main.tex` (main LaTeX document)
   - Images (stored in a folder like `images/`)
4. Once you are satisfied with the project, download it as a zip file:
   - Click on `Menu` → `Download` → `as Zip`.

5.3.2 Step 2: Extract and Organize Files Locally

1. Extract the downloaded zip file to a folder on your local machine.
2. Open a terminal (CLI) and navigate to the folder where the project is extracted.

5.3.3 Step 3: Initialize Git and Create a GitHub Repository

1. Log in to your GitHub account and create a new repository.
   - Go to `https://github.com`, click on the `Repositories` tab, and then click `New`.
   - Name the repository (e.g., `latex-project`) and choose whether to make it public or private.
   - Optionally, initialize the repository with a `README.md`, a `.gitignore` and a license.
2. In the terminal, initialize Git in your project folder:
   ```
   git init
   ```
3. Add the remote GitHub repository to your local Git repository:
   ```
   git remote add origin github_repo_url
   ```

5.3.4 Step 4: Push LaTeX Files to GitHub

1. Stage all the files for the first commit:

   ```
   git add *
   ```

2. Commit the changes with a message:

   ```
   git commit -m "Initial commit with LaTeX files"
   ```

3. Push the changes to the GitHub repository:

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

5.4 Updating LaTeX Files

5.4.1 Update LaTeX Files on Overleaf and Push Changes

1. Make necessary updates to your LaTeX files on Overleaf until you are satisfied with the changes.
2. Once the updates are complete, download the updated project as a zip file from Overleaf.
3. Extract the zip file locally, replacing the existing files in your project folder.
4. Open the terminal (CLI) and navigate to the project folder.
5. Stage the updated files for a new commit:

   ```
   git add *
   ```

6. Commit the changes with a descriptive message:

   ```
   git commit -m "Commit Message"
   ```

7. Push the updated files to the GitHub repository:

   ```
   git push origin main
   ```

5.5 Conclusion

This procedure successfully creates a LaTeX repository on GitHub using Overleaf for LaTeX project creation and the Git CLI for version control and repository management. By following these steps, LaTeX files can be easily managed in a Git repository, enabling version control, collaboration, and access to project files from any system.

6 Latex File Recreation

6.1 Objective

The task was to create a LaTeX document formatted to replicate the provided example. The final submission required the following components:

- Source code (`.tex`)
- Output document in PDF format (`.pdf`)
- An image file (`.png` or `.jpg`)

These files were to be zipped and named according to the format: `Rollno_DeptName_Firstname.zip`.

6.2 Steps Taken

6.2.1 Document Setup

The LaTeX document was structured to match the provided template, including sections such as:

- Introduction with subsections on personal background and interests.
- A section for favorite quotations.
- A mathematics section, detailing mathematical expressions, tables, and formulas.

6.2.2 Mathematical Typesetting

Mathematical notation was included as per the requirements:

- Superscripts, subscripts, and Greek letters.
- Fractions, roots, and complex display math.
- Tables and equation arrays were implemented, adhering to the provided mathematical structures.

6.2.3 Image Inclusion

A placeholder for an image was added in the document, simulating the space for a profile photo. An image was uploaded in the required `.png` or `.jpg` format, to ensure full compliance with the task requirements.

6.2.4 Output Generation

The source code (`.tex`) was compiled to generate the final output document in PDF format (`.pdf`). The files were verified to ensure that the output matched the provided format.

6.2.5 Final Packaging

The source code, PDF output, and image files were compressed into a single zip file following the naming convention: `Rollno_DeptName_Firstname.zip`.

6.3 Conclusion

The LaTeX document was successfully created and compiled in accordance with the provided instructions. The output matched the formatting and structural requirements of the example, and the necessary files were packaged and submitted as per the guidelines.

7 Git Branching and Merging

7.1 Objective

The objective of this assignment was to demonstrate proficiency in Git branching, merging, and conflict resolution using the GitHub Desktop application.

7.2 Steps Taken

7.2.1 Create a New Repository

I opened GitHub Desktop and created a new repository called `git-advanced`.

7.2.2 Clone the Repository

Cloned the repository to my local machine by selecting the repository and clicking on the "Clone" button.

7.2.3 Create and Switch to New Branch

I created a new branch called `feature-1` by clicking on the "Current Branch" dropdown and selecting "New Branch."

7.2.4 Create and Edit File

I created a new file called `shared.txt` in my text editor with the following content:

```
This is a shared file.
Line 1: Original text.
Line 2: Original text.
```

7.2.5 Stage and Commit the File

In GitHub Desktop, I staged the file by checking the box next to `shared.txt` and added a meaningful commit message before clicking the "Commit to feature-1" button.

7.2.6 Push the Branch to GitHub

I pushed the `feature-1` branch to GitHub by clicking on the "Push origin" button in the top-right corner.

7.2.7 Create Another Branch

I created another branch called `feature-2` using the same method as before: by selecting "New Branch" from the "Current Branch" dropdown.

7.2.8 Checkout File from Main Branch

I returned to the main branch and checked out the `shared.txt` file to ensure I had the original version.

7.2.9 Modify the File

I modified the `shared.txt` file to change the second line to:

```
Line 2: Modified text in feature-2.
```

7.2.10 Stage and Commit Changes

In GitHub Desktop, I staged the changes and committed them with a meaningful message.

7.2.11 Push the Branch to GitHub

I pushed the `feature-2` branch to GitHub using the "Push origin" button.

7.2.12 Switch Back to Feature-1

I switched back to the `feature-1` branch by selecting it from the "Current Branch" dropdown.

7.2.13 Modify the File Again

I modified the `shared.txt` file again, changing the second line to:

```
Line 2: Modified text in feature-1.
```

7.2.14 Stage and Commit the Changes

I staged and committed the changes in GitHub Desktop with a meaningful message.

7.2.15 Push the Branch to GitHub

I pushed the `feature-1` branch to GitHub.

7.2.16 Merge Feature-1 into Main

I merged `feature-1` into the main branch by switching to the main branch and using the "Merge into Current Branch" option.

7.2.17 Merge Feature-2 into Main

I then attempted to merge `feature-2` into the main branch, which introduced a conflict.

7.2.18 Resolve the Conflict

I resolved the conflict by editing `shared.txt` directly in my text editor, and then committed the resolution in GitHub Desktop.

7.2.19 Push the Updated Main Branch

I pushed the updated main branch to GitHub using the "Push origin" button.

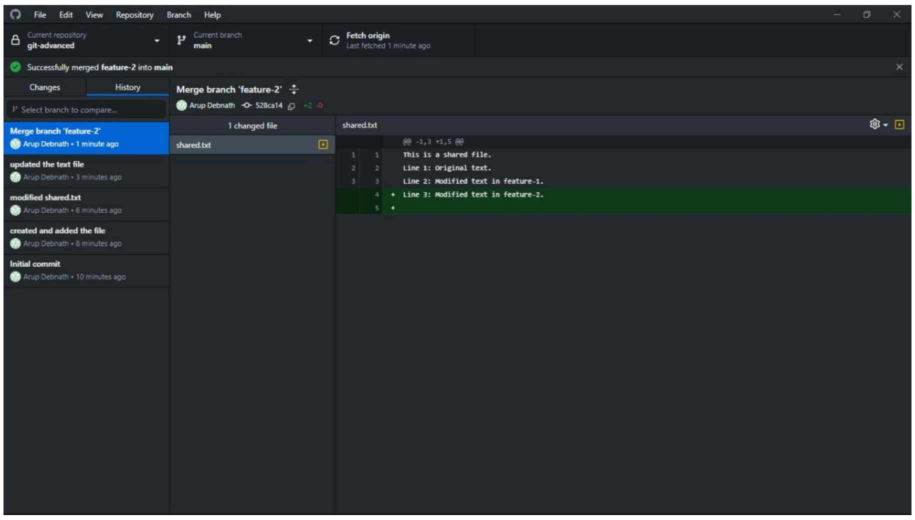
7.2.20 Delete the Feature Branches

Finally, I deleted the `feature-1` and `feature-2` branches from the repository in GitHub Desktop.

7.3 Deliverables

The final submission includes:

- A screenshot of the GitHub repository showing the commit history and branching.


File Edit View Repository Branch Help
Current repository git-advanced
Current branch main
Fetch origin Last fetched 1 minute ago
Changes History
Select branch to compare...
1 changed file
shared.txt

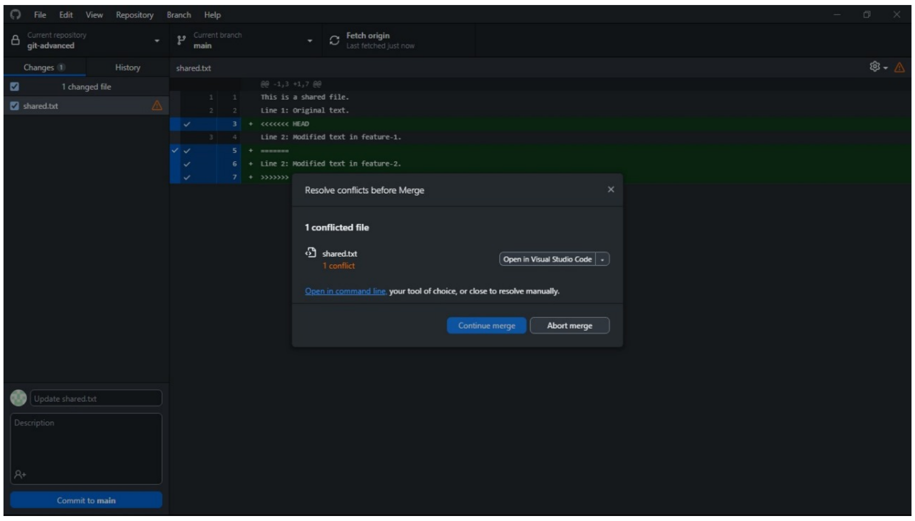
Successfully merged feature-2 into main

Merge branch 'feature-2'
Arup Debnath 528ca14 +2 -0

Merge branch 'feature-2' — Arup Debnath • 1 minute ago
updated the text file — Arup Debnath • 3 minutes ago
modified shared.txt — Arup Debnath • 6 minutes ago
created and added the file — Arup Debnath • 8 minutes ago
Initial commit — Arup Debnath • 10 minutes ago

```
@@ -1,3 +1,5 @@
 This is a shared file.
 Line 1: Original text.
 Line 2: Modified text in feature-1.
+Line 3: Modified text in feature-2.
+
```

- A screenshot of the local machine showing the Git log and conflict resolution.


File Edit View Repository Branch Help
Current repository git-advanced
Current branch main
Fetch origin Last fetched just now
Changes 1
History
1 changed file
shared.txt
Update shared.txt
Description
Commit to main
Open in Visual Studio Code
Continue merge
Abort merge

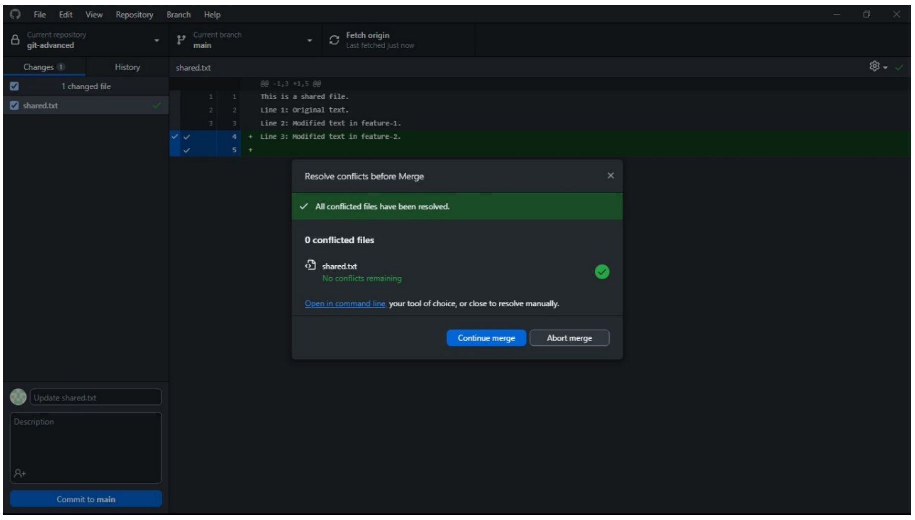
```
@@ -1,3 +1,7 @@
 This is a shared file.
 Line 1: Original text.
+<<<<<<< HEAD
 Line 2: Modified text in feature-1.
+=======
+Line 2: Modified text in feature-2.
+>>>>>>>
```

Resolve conflicts before Merge

1 conflicted file

shared.txt
1 conflict

Open in command line, your tool of choice, or close to resolve manually.



```
@@ -1,3 +1,5 @@
 This is a shared file.
 Line 1: Original text.
 Line 2: Modified text in feature-1.
+Line 3: Modified text in feature-2.
+
```

Resolve conflicts before Merge

All conflicted files have been resolved.

0 conflicted files

shared.txt
No conflicts remaining

Open in command line, your tool of choice, or close to resolve manually.

- A brief write-up of my experience with Git branching and merging.

7.4 Conclusion

This lab notebook entry documents the steps taken to successfully complete the assignment, showcasing the processes of branching, merging, and conflict resolution using the GitHub Desktop application.

8 Creating a CV in LaTeX

8.1 Objective

To create a professional CV using LaTeX as per the assignment requirements.

8.2 Requirements

- Create CV using LaTeX
- Include all source files in a single zip file
- Name the zip file as Rollno_DeptName_Firstname_CV.zip

8.3 Process

8.3.1 Setup

- Installed LaTeX distribution (TeXLive/MiKTeX) on my computer
- Chose a LaTeX editor (TeXstudio/Overleaf)

8.3.2 CV Content Preparation

- Reviewed existing CV content from the provided PDF
- Organized information into sections:
  - Personal Information
  - Summary
  - Projects
  - Courses
  - Education
  - Skills

8.3.3 LaTeX Document Structure

- Created a new .tex file
- Set up document class and necessary packages
- Defined custom commands for consistent formatting

8.3.4 Content Implementation

- Implemented each section of the CV using appropriate LaTeX environments and commands
- Ensured proper formatting and layout for each section

8.3.5 Styling and Formatting

- Applied consistent formatting throughout the document
- Adjusted margins, font sizes, and spacing for optimal presentation

8.3.6 Review and Refinement

- Compiled the document multiple times to check for errors
- Made necessary adjustments to improve layout and readability

8.3.7 Final Compilation

- Performed a final compilation to generate the PDF output

8.3.8 File Preparation

- Created a zip file named according to the specified format:
  Rollno_DeptName_Firstname_CV.zip
- Included all source files (.tex, any custom .sty files, images if used) in the zip file

8.4 Final Output

The resulting PDF generated in prior step is attached on the next page.

Credits

This Lab Notebook has been collaboratively created by **Group 28**, with contributions from the following group members:

- **Anshu Kumar** (Leader, BSc in AI, Roll No. 30054623011):
  - Cover Page
  - Table of Contents
  - Topic 5: Creating a Latex Repository in Github
  - Topic 8: Creating a CV in LaTeX
- **Debyani Sarkar** (BSc in Forensic Sciences, Roll No. 30059223033):
  - Topic 3: Calculator in C
- **Arup Debnath** (BCA, Roll No. 30001223052):
  - Topic 4: Mind Reader JAVA Program
  - Provided Images for deliverables in section 7
- **Jeet Majumder** (BCA, Roll No. 30001223079):
  - Topic 2: Introduction to Latex
  - Topic 6: Latex File Recreation
- **Aniket Gupta** (BSc in AI, Roll No. 30054623015):
  - Topic 1: Introduction to Git
  - Topic 7: Git Branching and Merging