# SOURCE CODE MANAGEMENT LABORATORY RECORD

Name : Yashaswini S

Enrollment No. : A86605224210

Program: B tech CSE

Submitted to : Monit Kapoor

| SL.No. | Index |
| --- | --- |
| 1. | Introduction to Gitbash |

2. Introduction to GitHub

3. Gitbash and GitHub

4. File creation with commit

and push command

5. Branches Creation

6. Git commit and Merge

7. Open and Close

Pull

Request

## Lab Exercise 1 :

### Introduction to Gitbash

Git Bash

- Git Bash is an application for Microsoft Windows that provides a command-line interface to use Git, the version control system.
- It emulates a Bash (Bourne Again

Shell) environment, allowing users to run Linux-style commands on Windows.

- It is mainly used by developers to:

1. Execute Git commands (e.g., git init, git commit)
2. Manage code repositories
3. Interact with remote repositories

- Key Features:

1. Supports Git version control operations.
2. Provides Unix-style command-line tools (like ssh, scp, ls,

etc.).

3. Helps users practice command-line Git workflows.

- Steps to Install Git Bash:

1. Go to the official Git website:
   - https://git-scm.com

2. Download Git for Windows:

- Click the “Download for Windows” button.

3.Run the Installer :

- Double-click the downloaded .exe file.

4. Follow the Setup Wizard:

- Click Next on the welcome screen.
- Choose the default options (recommended for beginners).
- Select the text editor (e.g., Notepad or VS Code).
- Choose “Git from the command line and also from 3rd-party software”.
- Continue clicking Next until you reach Install.

## 5. Click Install:

- Wait for the installation to complete.

## 6. Finish Setup:

- Click Finish and leave “Launch Git Bash” checked.

## 7. Start Using Git Bash:

- Git Bash will open in a terminal window.

## • Basic Git Commands in Git Bash:

1. Git init

Initializes a new Git repository in your folder.

2. Git clone <repo-url>

Copies (clones) a remote repository to your local machine.

3. Git status

Shows the status of files (tracked, modified, staged).

4. Git add <filename>

Adds a specific file to the staging area.

Use git add . to add all files.

5. Git commit -m “Your message”

Saves the staged changes with a message.

6. Git push

Uploads your commits to the remote repository (GitHub).

7. Git pull

Downloads changes from the remote repository and merges them.

8. Git remote add origin <repo-url>

Connects your local repo to a GitHub repository.

9. Git log

Shows the commit history.

10. Git branch

Lists all branches.

Use git branch <name> to create a new branch.

11. Git checkout <branch-name>

Switches to a different branch.

12. Git merge <branch-name>

Merges changes from one branch into the current one.

## Lab Exercise : 2

# Introduction to GitHub

Git Hub.

- GitHub is a web-based platform for hosting and sharing Git repositories.
- It is widely used for collaboration, version control, and open-source development.
- GitHub allows users to:

1. Store code in repositories
2. Track changes using Git

3. Collaborate with others through pull requests and issues
4. Manage projects with built-in tools like GitHub Projects and Actions

- Key Features:

1. Cloud-based hosting for Git repositories.
2. Social coding features (followers, stars, forks)
3. Integration with CI/CD, project management, and automation tools
4. Access control and team collaboration

- Steps to Install GitHub Desktop:

1. Go to: https://desktop.github.com
2. Click “Download for Windows” (or Mac).
3. Open the downloaded file and run the installer.
4. Follow the setup wizard.
5. After installation, open GitHub Desktop.
6. Sign in with your GitHub account.
7. You can now clone repositories, make commits, and push changes using a user-friendly interface.

# Lab Exercise 3 :

## Gitbash and GitHub

Git.

Git is a version control system that helps developers track and manage changes to code over time. It allows multiple people to work on a project at the same time without overwriting each other's work.

• Key Features of Git:

1. Version Tracking: Keeps a history of

changes made to files.

2. Branching: Lets you work on new features or fixes in isolation.

3.Merging: Combines changes from different branches.

4.Collaboration: Enables teams to

work

together using services like GitHub, GitLab, or Bitbucket.

- Step-by-Step Workflow :

Step 1: Initialize Git

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ git init
Initialized empty Git repository in C:/Users/Admin/.git/
```

- Creates a new Git repository in your project folder.
- Git starts tracking changes to files in this folder.

## Step 2: Add Files to Staging

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git add main.c
warning: in the working copy of 'main.c', LF will be replaced by CRLF the next t
ime Git touches it
```

- Adds all files to the staging area (preparing them to be committed).
- You can also use git add filename to add specific files.

## Step 3: Commit Changes

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git commit -m "first commit"
[master (root-commit) 919117e] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 main.c
```

- Records your changes in Git history with a message.

• This saves your code locally (not yet on GitHub).

## Step 4: Link to GitHub Repository

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git remote add origin https://github.com/syashaswini2006/l3-l4.git
```

• Connects your local Git project to a remote GitHub repository.

## Step 5: Push to GitHub

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 233 bytes | 233.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/syashaswini2006/l3-l4/pull/new/master
remote:
To https://github.com/syashaswini2006/l3-l4.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

• Uploads your committed code to

GitHub.

• After this, your project appears online in the linked GitHub repo.

## Lab Exercise 4 :

### File Creation with commit and push command

### PART 1: Create the GitHub repository

1. Go to GitHub (https://github.com) and sign in.

2. Click “New repository”.

3. Enter a repository name (e.g., my-repo).

4. Choose Public or Private.

5. Don’t add README, .gitignore, or license here — we will push an existing repo.

6. Click “Create repository”.

7. Copy the HTTPS URL shown (e.g., https://github.com/yourusername/my-repo.git).

This URL is where you will push your code.

## PART 2: Using Git Bash

Step 1: Open Git Bash

This is your command line tool for running Git commands.

Step 2: Create a new local folder and navigate into it

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ mkdir yashaswini

Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ cd yashaswini
```

- Mkdir my-repo: makes a new directory (folder) called my-repo.
- Cd my-repo: changes into that folder so your commands affect this directory.

Step 3: Initialize Git in this folder

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ git init
Initialized empty Git repository in C:/Users/Admin/yashaswini/.git/
```

- This creates a hidden .git folder that tracks changes to files here.
- Now this folder is a Git repository.

## Step 4: Create a file

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ vi main.c
```

- This writes the text This is a sample file. Into a file named sample.txt.
- You can also create files using any editor.

## Step 5: Stage the file

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git add main.c
warning: in the working copy of 'main.c', LF will be replaced by CRLF the next t
ime Git touches it
```

- This tells Git to “stage” (prepare) the file sample.txt for committing.
- Staging means you are telling Git what changes to include in the next commit.

## Step 6: Commit the staged file

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git commit -m "first commit"
[master (root-commit) 919117e] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 main.c
```

- This creates a commit, which is like a snapshot of your project.
- The -m flag adds a message describing the commit (“Add sample.txt”).

• Commits save your work history.

Step 7: Add the remote repository URL

Git remote add origin

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git remote add origin https://github.com/syashaswini2006/l3-l4.git
```

• This links your local Git repository to the remote one on GitHub.

• Origin is the default name for your remote repository.

• Replace the URL with your actual GitHub repo URL.

Step 8: Push your commit to GitHub

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch
* master

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git push -u origin master
Everything up-to-date
branch 'master' set up to track 'origin/master'.
```

- Git branch -M main: renames your branch to main (modern default branch name).
- Git push -u origin main: uploads your local commits to GitHub.
- The -u flag sets the remote origin/main as the default upstream branch.

## PART 3: Verify

- Go to your GitHub repository page

in a browser.

- You should see sample.txt uploaded there.

## Lab Exercise : 5

### Branches Creation

Branches in Git allow you to work on different features, bug fixes, or experiments without affecting the main codebase. Here's how to create and manage branches using Git Bash:

1. View Current Branches

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch
* master
  test
```

- Shows all local branches.
- The currently active branch is

highlighted with *.

## 2. Create a New Branch

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch test
```

- This creates a new branch called new-feature, but does not switch to it.

## 3. Switch to the New Branch

- Changes your working directory to the new-feature branch.

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git checkout test
Switched to branch 'test'

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ git branch
  master
* test
```

OR create and switch in one step:

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git checkout test
Switched to branch 'test'

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ git branch
  master
* test
```

## 4.Make Changes and Commit (on the new branch)

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git checkout test
Switched to branch 'test'

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ vi main.c
```

- Git add feature.txt
- Git commit -m “Add feature.txt in new-feature branch”

## 5. Push the New Branch to GitHub

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git push -u origin test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 384 bytes | 384.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/syashaswini2006/l3-l4.git
   919117e..78b163a  test -> test
branch 'test' set up to track 'origin/test'.
```

- The -u flag sets the remote new-feature branch as the default upstream for this branch.

## 6. Switch Back to Main Branch

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ git checkout master
Switched to branch 'master'
```

## 7. Merge New Branch into Main (Optional)

- First, make sure you're on the master branch:

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git merge test
Updating 919117e..78b163a
Fast-forward
 l3-l4              | 1 +
 main.cpp           | 1 +
 yashaswini/main.c  | 1 +
 3 files changed, 3 insertions(+)
 create mode 160000 l3-l4
 create mode 100644 main.cpp
 create mode 100644 yashaswini/main.c
```

- This merges the new-feature branch into master.

## Lab Exercise : 6

### Git commit and Merge (Merge Request)

STEP 1: Create a Local Git Repository (if not already done)

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ mkdir yashaswini

Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ cd yashaswini
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ vi main.c

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ git init
Initialized empty Git repository in C:/Users/Admin/yashaswini/.git/
```

## STEP 2: Add a Remote GitHub Repository

- Create a new repo on GitHub, then connect:

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git remote add origin https://github.com/syashaswini2006/l3-l4.git
```

## STEP 3: Create and Switch to a New Branch

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch
* master
  test
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git checkout test
Switched to branch 'test'

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ git branch
  master
* test
```

- This is your feature branch — where you make changes.

## STEP 4: Add or Modify Files

Example:

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ vi main.cpp
```

## STEP 5: Stage and Commit Changes

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git status
On branch test
Your branch is up to date with 'origin/test'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   l3-l4
        new file:   main.cpp
        new file:   yashaswini/main.c


Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git commit -m "second commit"
[test 78b163a] second commit
 3 files changed, 3 insertions(+)
 create mode 160000 l3-l4
 create mode 100644 main.cpp
 create mode 100644 yashaswini/main.c
```

# STEP 6: Push Feature Branch to GitHub

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git push -u origin test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 384 bytes | 384.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/syashaswini2006/l3-l4.git
   919117e..78b163a  test -> test
branch 'test' set up to track 'origin/test'.
```

- This uploads your branch to GitHub.

STEP 7: Create a Merge Request (Pull Request) on GitHub

1. Go to your repo on GitHub

2. GitHub will show a “Compare & pull request” button

— click it

(or go to the Pull Requests tab > click New pull request)

3. Set:

• Base branch: main (target)

• Compare branch: feature-1 (your work)

4. Add a title and description

5. Click “Create pull request”

STEP 8: Review and Merge

• You or a teammate reviews the

code

- If everything looks good, click “Merge pull request”
- Confirm the merge

STEP 9: Delete the Feature Branch (Optional)

- GitHub will offer an option to delete the branch. Or do it locally:

Git branch -d feature-1

STEP 10: Update Local Main Branch

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git pull origin master
From https://github.com/syashaswini2006/l3-l4
 * branch            master     -> FETCH_HEAD
Already up to date.
```

# Lab Exercise 7 :

## Open and Close Pull Request

## STEP-BY-STEP WORKFLOW :

### 1.Initialize Local Repo (Git Bash)

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ mkdir yashaswini

Admin@DESKTOP-TIAO4C4 MINGW32 ~
$ cd yashaswini
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ vi main.c
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini
$ git init
Initialized empty Git repository in C:/Users/Admin/yashaswini/.git/
```

## 2. Connect to GitHub Repository

On GitHub:

- Create a new repo: my-project
- Back in Git Bash:

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git remote add origin https://github.com/syashaswini2006/l3-l4.git
```

## 3. Create a File, Commit It

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello.c


Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (master)
$ git commit -m "first commit"
[master 7817c81] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello.c
```

## 4.Push to Master Branch

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch
* master

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git push -u origin master
```

# 5. Create a Feature Branch and Switch to It

## Git checkout test

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git branch
* master
  test
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (master)
$ git checkout test
Switched to branch 'test'

Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ git branch
  master
* test
```

# 6.Add Another File and Commit

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini/yashaswini (test)
$ vi main.c
```

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git status
On branch test
Your branch is up to date with 'origin/test'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   l3-l4
        new file:   main.cpp
        new file:   yashaswini/main.c


Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git commit -m "second commit"
[test 78b163a] second commit
 3 files changed, 3 insertions(+)
 create mode 160000 l3-l4
 create mode 100644 main.cpp
 create mode 100644 yashaswini/main.c
```

# 7.Push the Feature Branch to GitHub

```
Admin@DESKTOP-TIAO4C4 MINGW32 ~/yashaswini (test)
$ git push -u origin test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 384 bytes | 384.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/syashaswini2006/l3-l4.git
   919117e..78b163a  test -> test
branch 'test' set up to track 'origin/test'.
```

Git push -u origin test

8.Open a Pull Request (on GitHub)

1. Go to your repo on GitHub.
2. GitHub will show “Compare & pull request” button — click it.
3. Set:

• Base: main

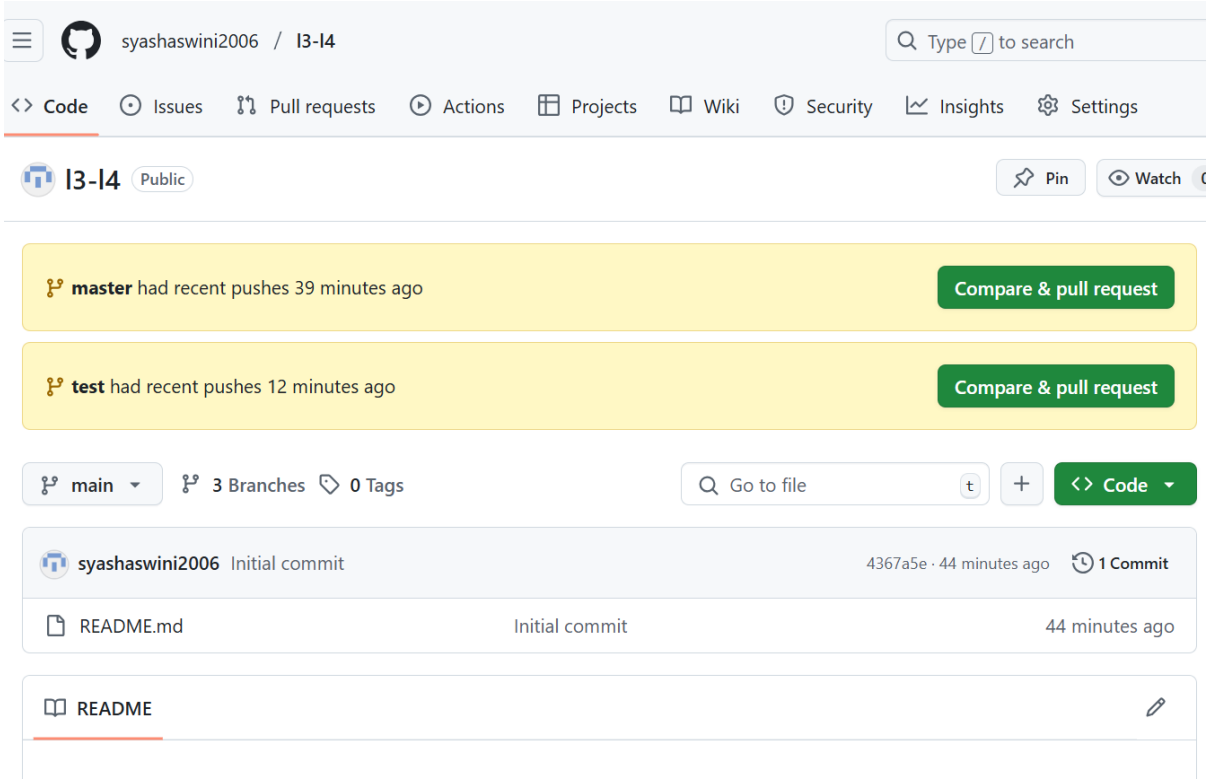
• Compare: add-about-page

4. Add:

• Title: Add About Page

• Description: This PR adds about.html with basic content

# 5.Click “Create pull request”




- Pull request is now open!

syashaswini2006 / shreya-4051

Type / to search

Code | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

**shreya-4051** Public

forked from shreyanmshreya310/shreya-4051

Pin | Watch

master | 1 Branch | 0 Tags | Go to file t | + | Code

This branch is 1 commit ahead of `shreyanmshreya310/shreya-4051:master`.

Contribute | Sync fork

syashaswini2006 added a heading

main.c added a heading 1 minute ago

2 Commits

README

This branch is 1 commit ahead of `shreyanmshreya310/shreya-4051:master`.

Open a pull request to contribute your changes upstream.

Open pull request

## Add a README

# added a heading #1

Open | syashaswini2006 wants to merge 1 commit into `shreyanmshreya310:master` from `syashaswini2006:master`

Conversation 0 | Commits 1 | Checks 0 | Files changed 1

syashaswini2006 commented now

@shreyanmshreya310

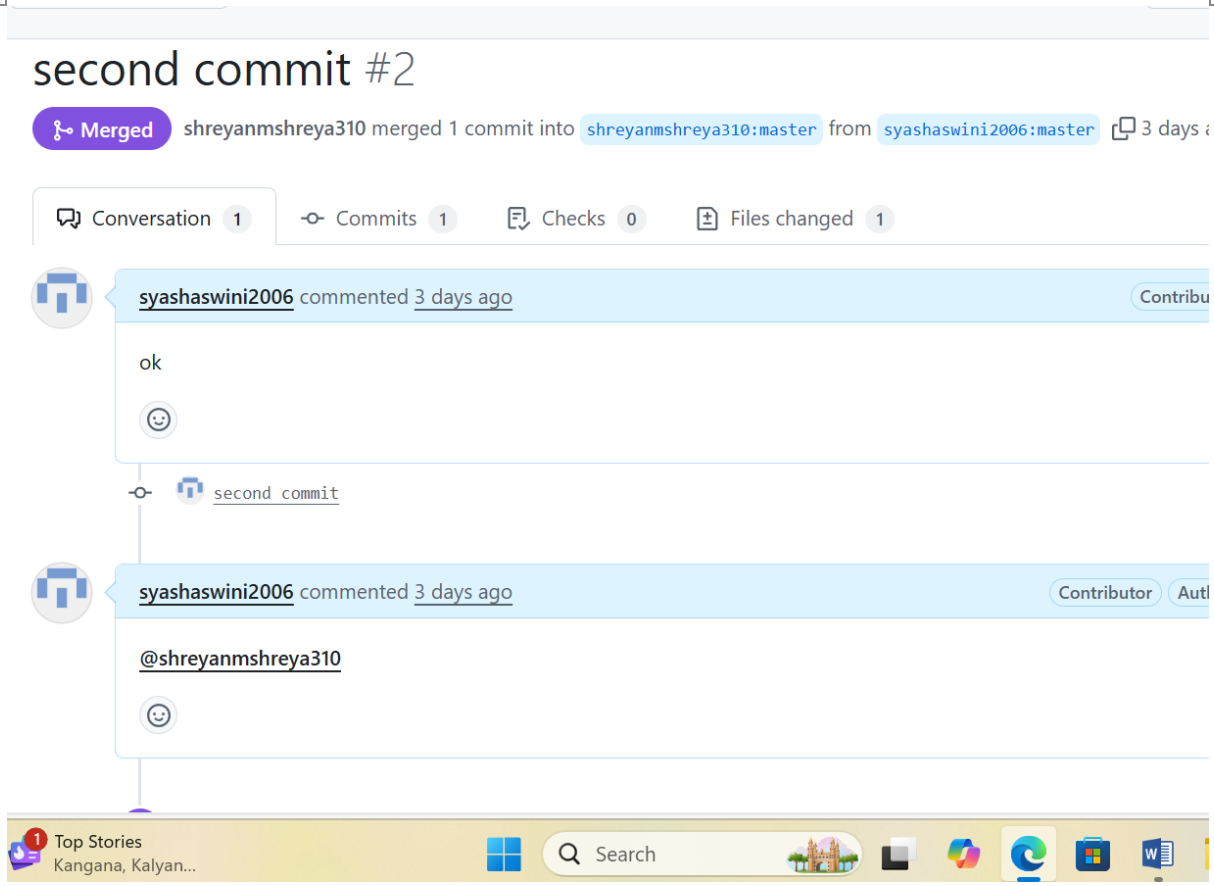
added a heading 9193a42

syashaswini2006 commented now Author

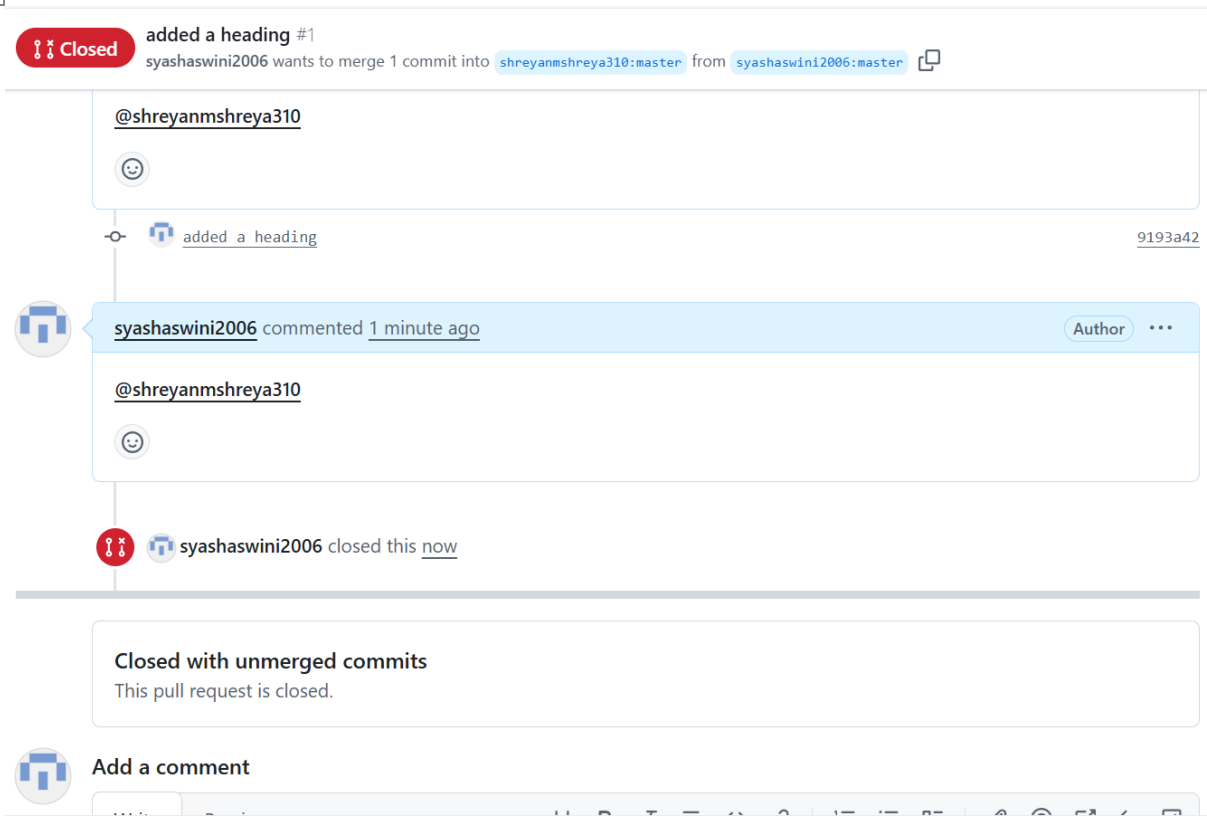
@shreyanmshreya310

**No conflicts with base branch**

Changes can be cleanly merged.




# 9.Close a Pull Request (Without Merging)




1. Scroll to the bottom of the pull request page.
2. Click “Close pull request”
3. (Optional) Add a comment like “Work in progress” or “Not needed”
4. Click “Close”

Pull request is now closed and not merged.

## 10.Reopen (Optional)

- Go to the closed pull request
- Click “Reopen pull request”