

# Mini Project – Git Branching and Merging

---

This project demonstrates how multiple developers (Tom and Jerry) can collaborate on a project using Git branching and merging strategies. It outlines the process of creating and reviewing Pull Requests (PRs) before merging into the main branch.

---

## Workflow Overview

The project simulates collaboration between two developers, each working on their own branch:

- Tom: update-navigation
- Jerry: edit-contact (formerly add-contact-info)

Both developers push their changes, create Pull Requests, and merge into the main branch.

---

## Understanding Pull Requests

A Pull Request (PR) is a GitHub feature that enables developers to:

- Propose code changes,
  - Request code reviews,
  - Collaborate on improvements,
  - Merge only approved contributions into shared branches.
- 

## Developer 1: Tom – update-navigation

### Step-by-Step Workflow

1. Create and switch to feature branch:

```
git checkout -b update-navigation
```

The screenshot shows the VS Code interface with two tabs open: 'index.html' (DevOps-Projects • 03. Mini Project - Basic Git Commands) and 'index.html' (DevOps-Projects • 04. Mini Project - Git Branching and Merging). The code editor displays the following HTML and CSS:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Innovate - Transforming the Future</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: Arial, sans-serif;
        }
        body {
            ...
        }
    </style>

```

The terminal window shows the command \$ git checkout -b update-navigation being run, followed by a message indicating the switch to a new branch.

## 2. Edit navigation content.

The screenshot shows the VS Code interface with the same two tabs open. The code editor now includes a new CSS rule for a .navbar:

```

/* Navigation */
.navbar {
    background-color: #3b3b44;
    padding: 1rem 2rem;
    position: fixed;
    width: 100%;
    top: 0;
    z-index: 1000;
}

```

The terminal window shows the command \$ git checkout -b update-navigation being run again, confirming the switch to the 'update-navigation' branch.

## 3. Stage and commit changes:

```
git add .
git commit -m "Update navigation menu"
```

## 4. Push the feature branch to GitHub:

```
git push origin update-navigation
```

Screenshot Placeholder: Push confirmation in terminal

5. Open GitHub Repository and switch to update-navigation branch



6. Create a Pull Request

7. Review and approve the PR



8. Merge the PR into main



9. Confirm merge success



Status: update-navigation successfully merged into main

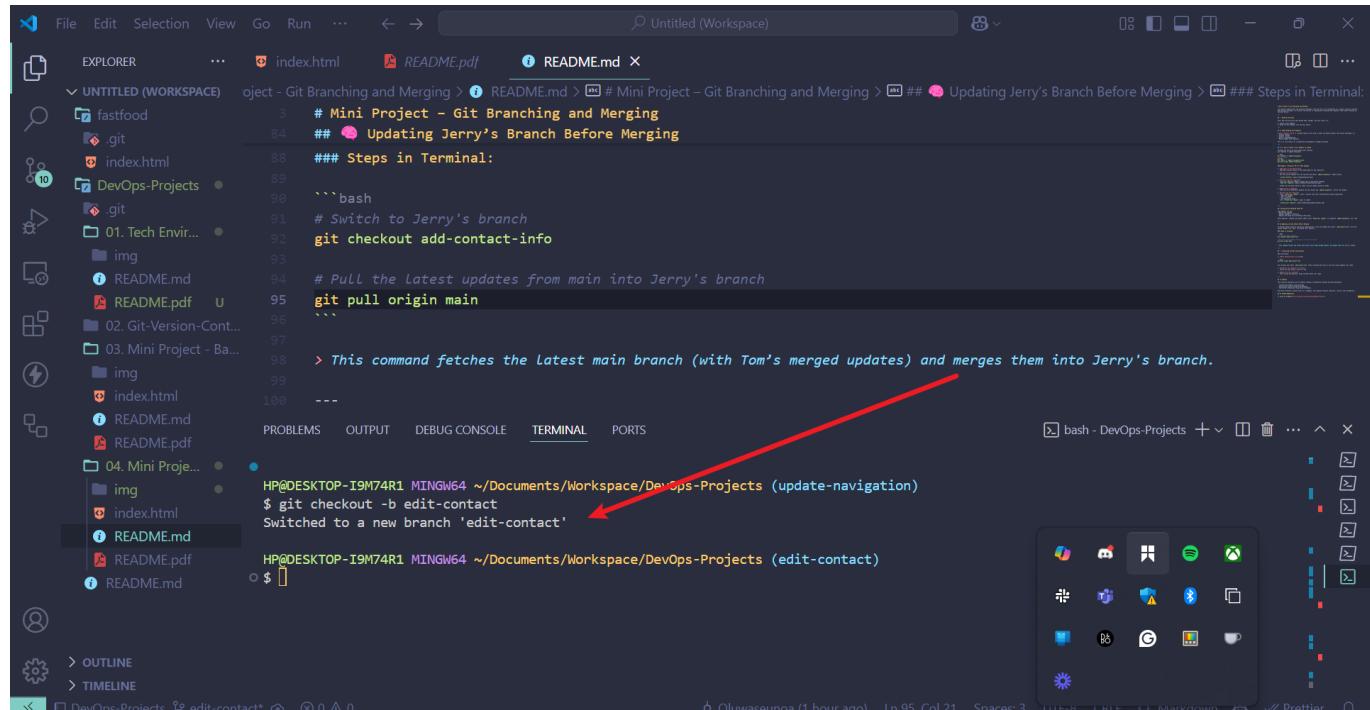
---

## Developer 2: Jerry – edit-contact

### Step-by-Step Workflow

1. Create and switch to new feature branch:

```
git checkout -b edit-contact
```



```
git checkout -b edit-contact
Switched to a new branch 'edit-contact'
```

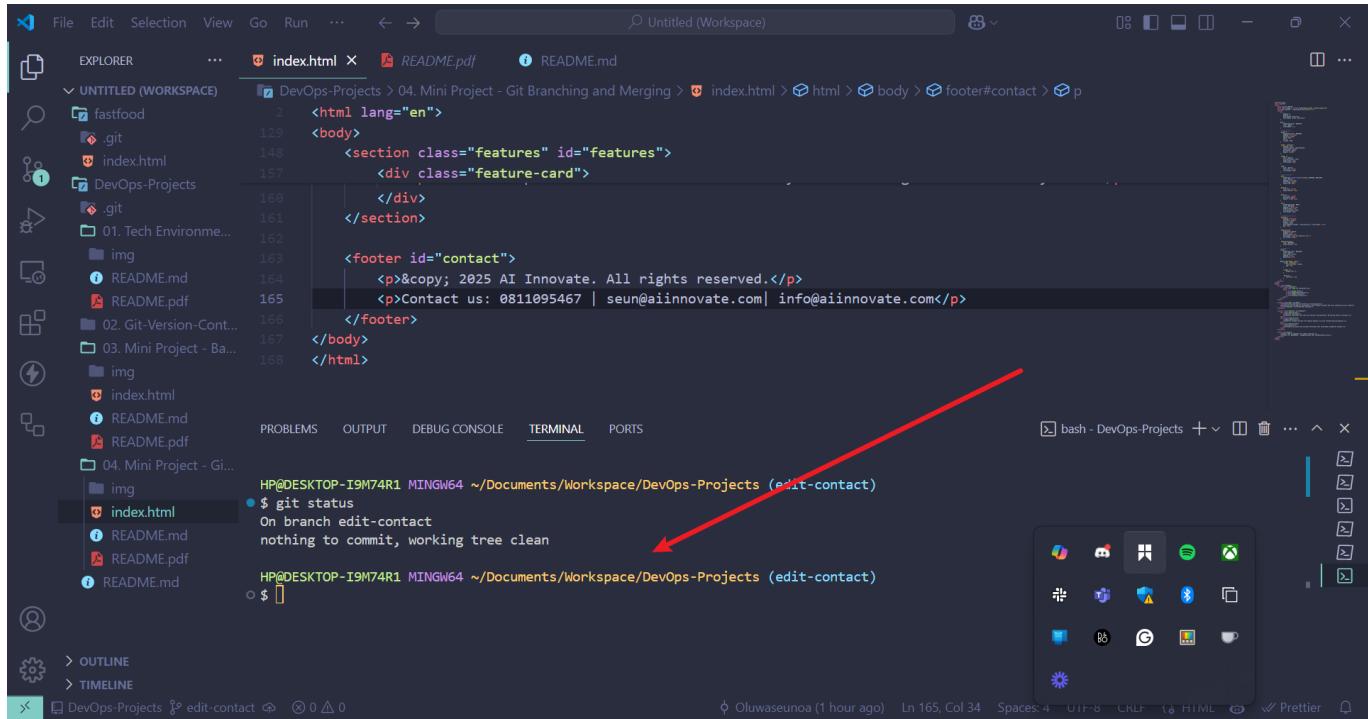
2. Pull the latest updates from main:

```
git pull origin main
```

 Screenshot Placeholder: Pull success with merge confirmation

## 3. Confirm branch status:

```
git status
```



The screenshot shows a Windows desktop environment with the Visual Studio Code application open. The terminal tab is active, displaying the command \$ git status and its output: "On branch edit-contact" and "nothing to commit, working tree clean". The code editor shows an HTML file with some footer content. The file explorer sidebar shows various project files and folders. A red arrow points from the terminal output area towards the status message.

## 4. Push updated branch to GitHub:

```
git push origin edit-contact
```

The screenshot shows a DevOps project workspace in VS Code. The Explorer sidebar lists files like index.html, README.md, and README.pdf. The terminal shows a GitHub pull request creation process:

```

Compressing objects: 100% (22/22), done.
Writing objects: 100% (22/22), 6.26 MiB | 100.00 KiB/s, done.
Total 22 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.
remote:
remote: Create a pull request for 'edit-contact' on GitHub by visiting:
remote:   https://github.com/Oluwaseunoa/DevOps-Projects/pull/new/edit-contact
remote:
To https://github.com/Oluwaseunoa/DevOps-Projects.git
 * [new branch] edit-contact -> edit-contact

```

The GitHub browser interface shows the 'edit-contact' branch selected in the dropdown menu. The repository page displays recent commits and the 'About' section.

## 5. Open a Pull Request on GitHub

- Screenshot Placeholder: New Pull Request for edit-contact
- Screenshot Placeholder: New Pull Request for edit-contact

## 6. Review changes and approve

- Screenshot Placeholder: Reviewer review and approval
- Screenshot Placeholder: Reviewer review and approval

## 7. Merge the PR into main

- Screenshot Placeholder: Merge confirmation

## 8. Confirm successful merge

- Screenshot Placeholder: GitHub PR shows "Merged" status

- Status: edit-contact successfully merged into main

## 📁 GitHub Repository Validation

🔗 Repository URL: [DevOps Projects GitHub Repo](#)

---

### 📋 Summary

This Git collaboration workflow demonstrates:

- Feature isolation through branching.
- Controlled integration with Pull Requests.
- Conflict prevention via branch synchronization.
- Traceable contributions through commits, screenshots, and PR history.

It highlights DevOps best practices for collaborative software development using Git and GitHub.

---