🛠️ 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:

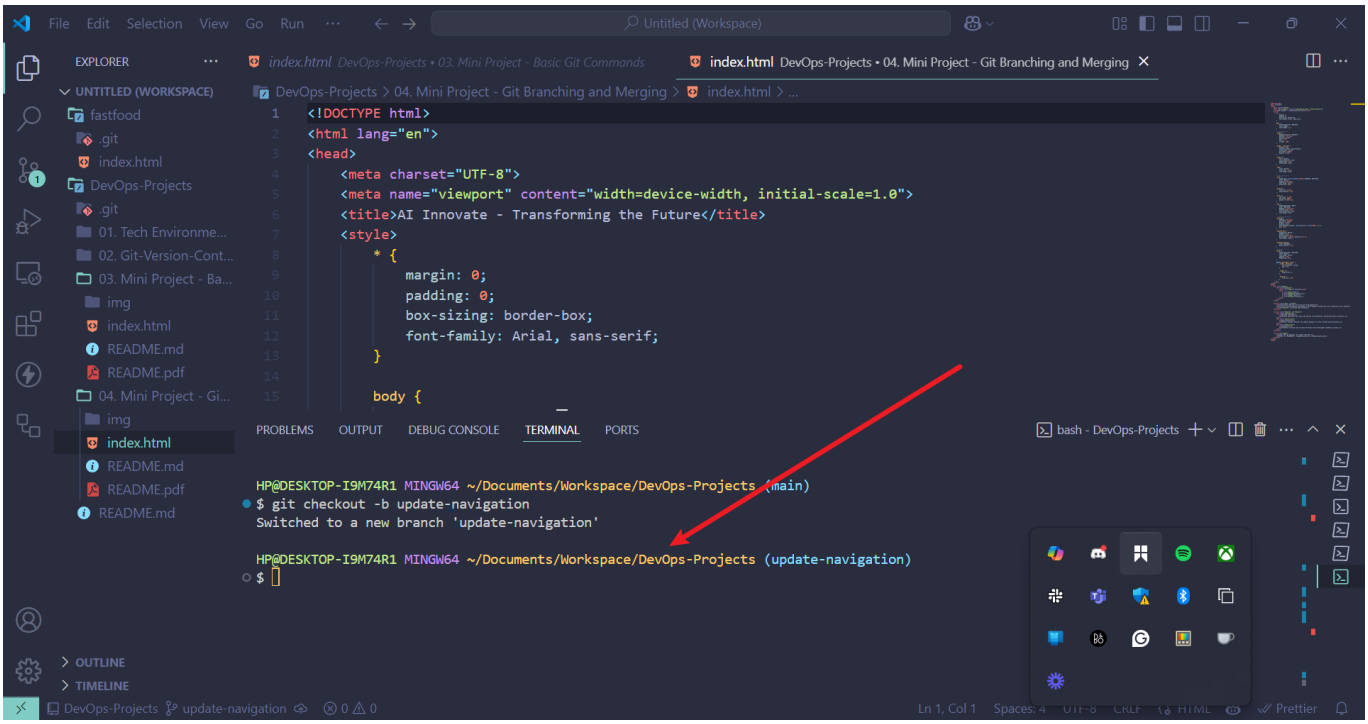
- Propose code changes,
- Request code reviews,
- Collaborate on improvements,
- Merge only approved contributions into shared branches.

👨‍💻 Developer 1: Tom – update-navigation

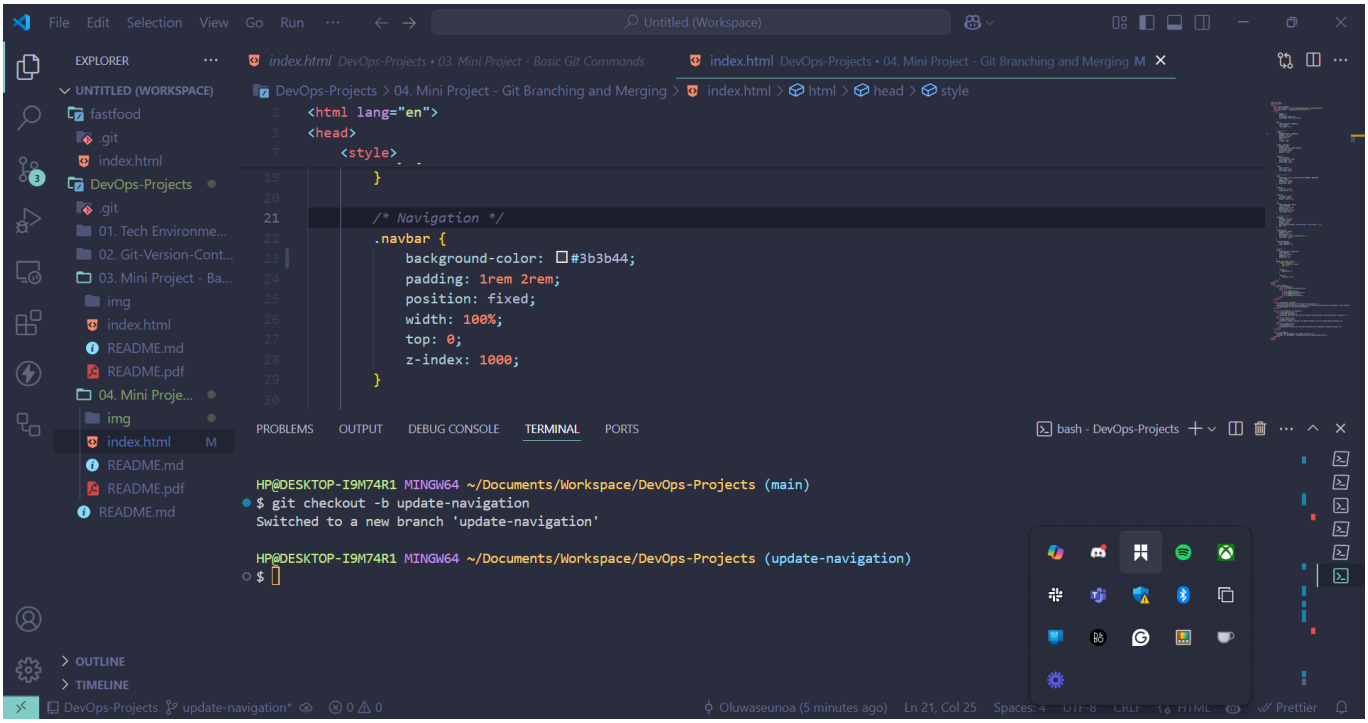
Step-by-Step Workflow

1. Create and switch to feature branch:

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



2. Edit navigation content.



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
   Screenshot Placeholder: GitHub branch switch

6. Create a Pull Request
   Screenshot Placeholder: New Pull Request creation Screenshot Placeholder: Opened Created Pull Request creation

7. Review and approve the PR
   Screenshot Placeholder: Reviewer approval screen

8. Merge the PR into main
   Screenshot Placeholder: Merged PR confirmation

9. Confirm merge success
   Screenshot Placeholder: GitHub PR shows "Merged" status

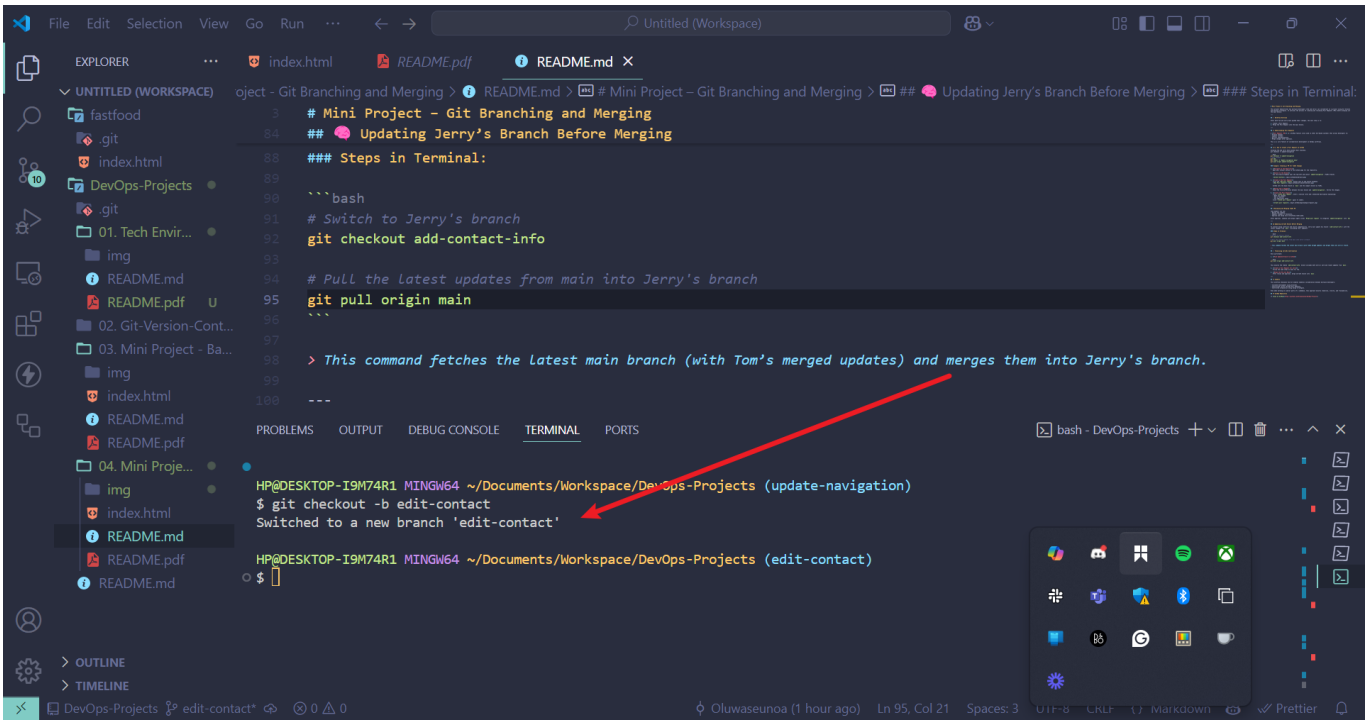
☑ Status: update-navigation successfully merged into main

Developer 2: Jerry – edit-contact

Step-by-Step Workflow

1. Create and switch to new feature branch:

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



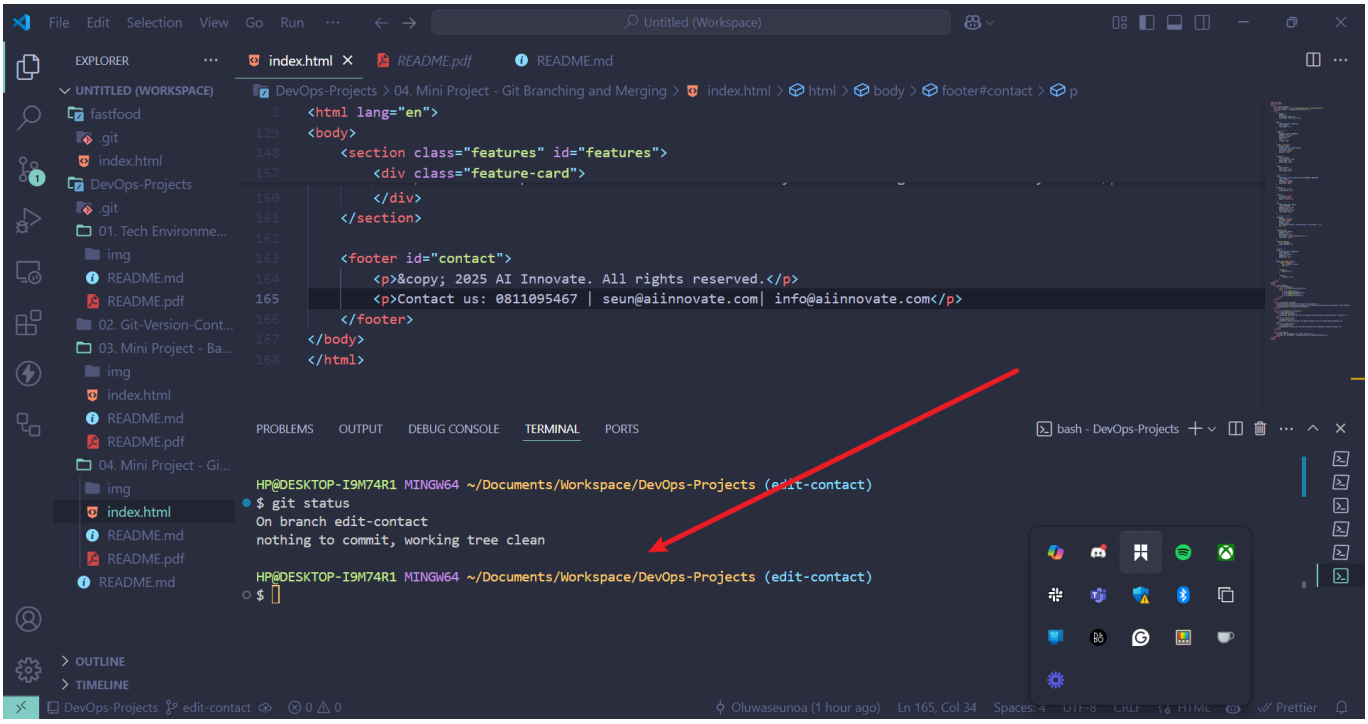
2. Pull the latest updates from main:

```
git pull origin main
```

Screenshot Placeholder: Pull success with merge confirmation

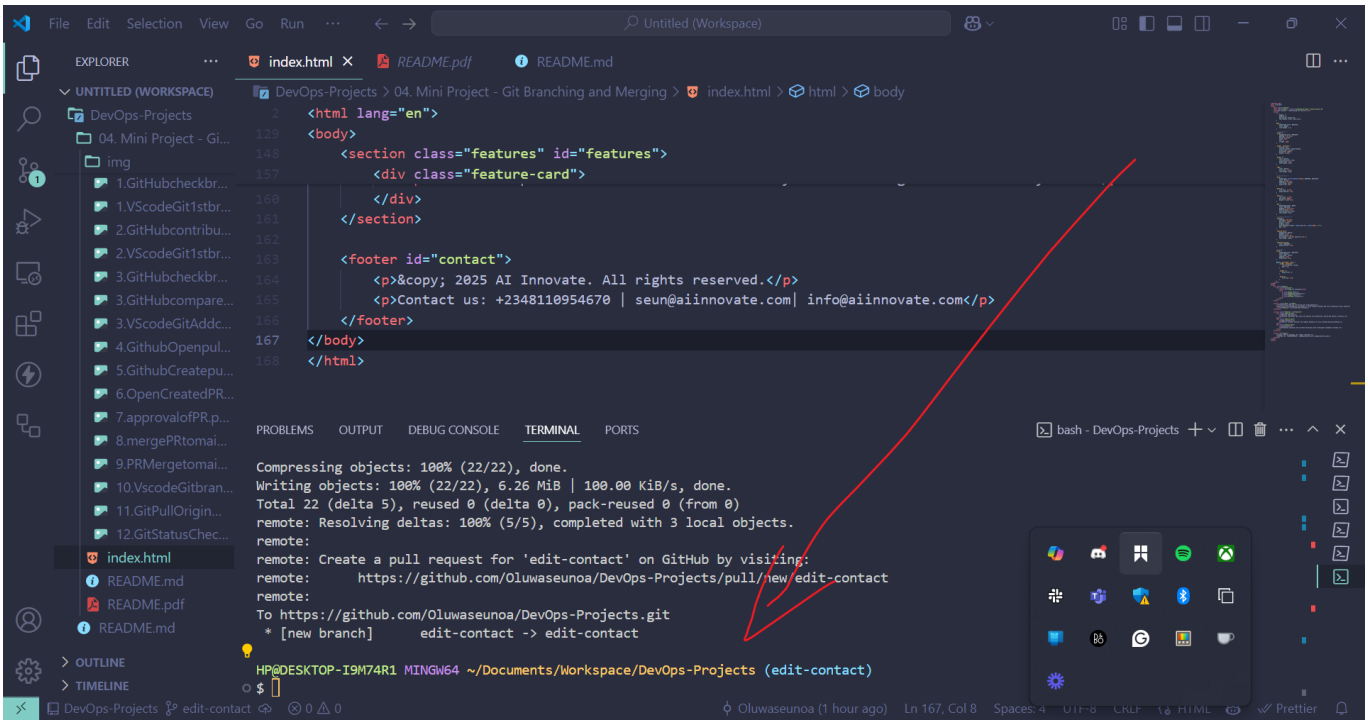
3. Confirm branch status:

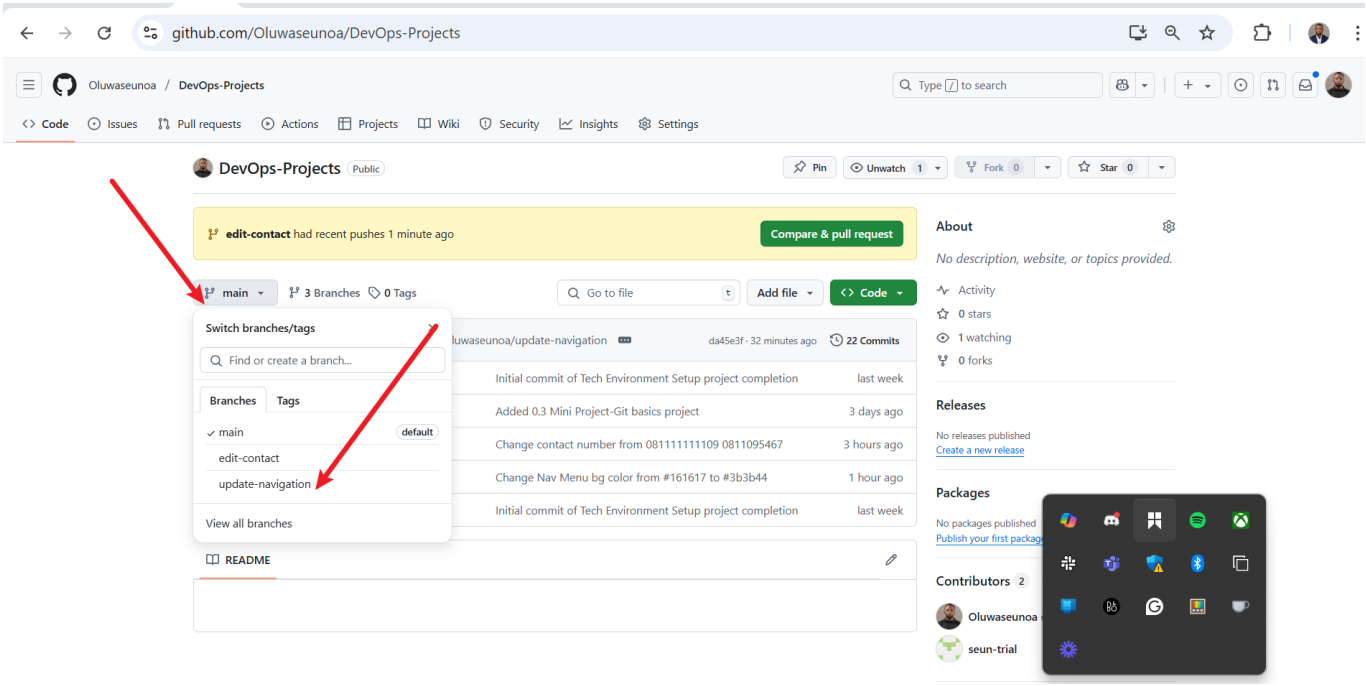
```
git status
```



4. Push updated branch to GitHub:

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





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.
