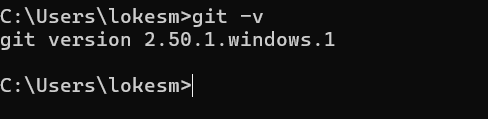
#### ✅ 1. \*\*Install and verify Git\*\*

\* Use Homebrew to install Git.

\* Confirm installation by checking the Git version.



#### ✅ 2. \*\*Configure Git\*\*

\* Set your global Git username and email to your real or test identity.

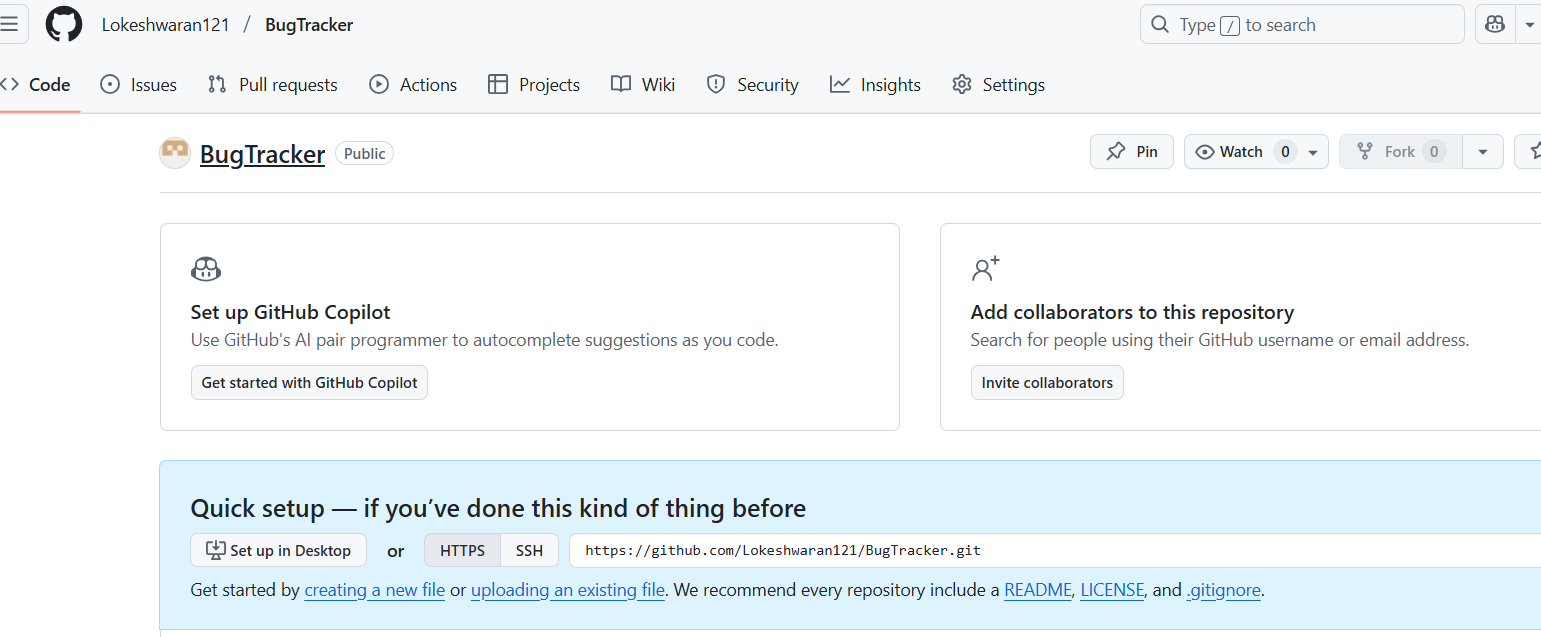


#### ✅ 3. \*\*Create GitHub Repository\*\*

\* Go to GitHub and create a new repo named: `BugTracker`

\* Do \*\*not\*\* initialize with README or license.

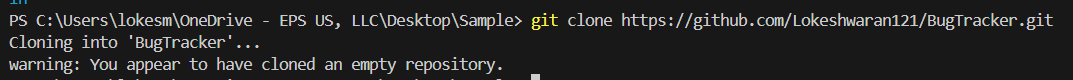
\* Keep the repo \*\*private\*\* or \*\*public\*\* as needed.



#### ✅ 4. \*\*Clone the Repository\*\*

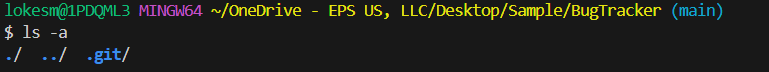
\* Clone the repo to your local system using HTTPS.

\* Use your \*\*GitHub token\*\* to authenticate.



#### ✅ 5. \*\*Verify `.git` folder\*\*

\* Inside the cloned folder, use the terminal to check if `.git` is present.



#### ✅ 6. \*\*Create Basic Files\*\*

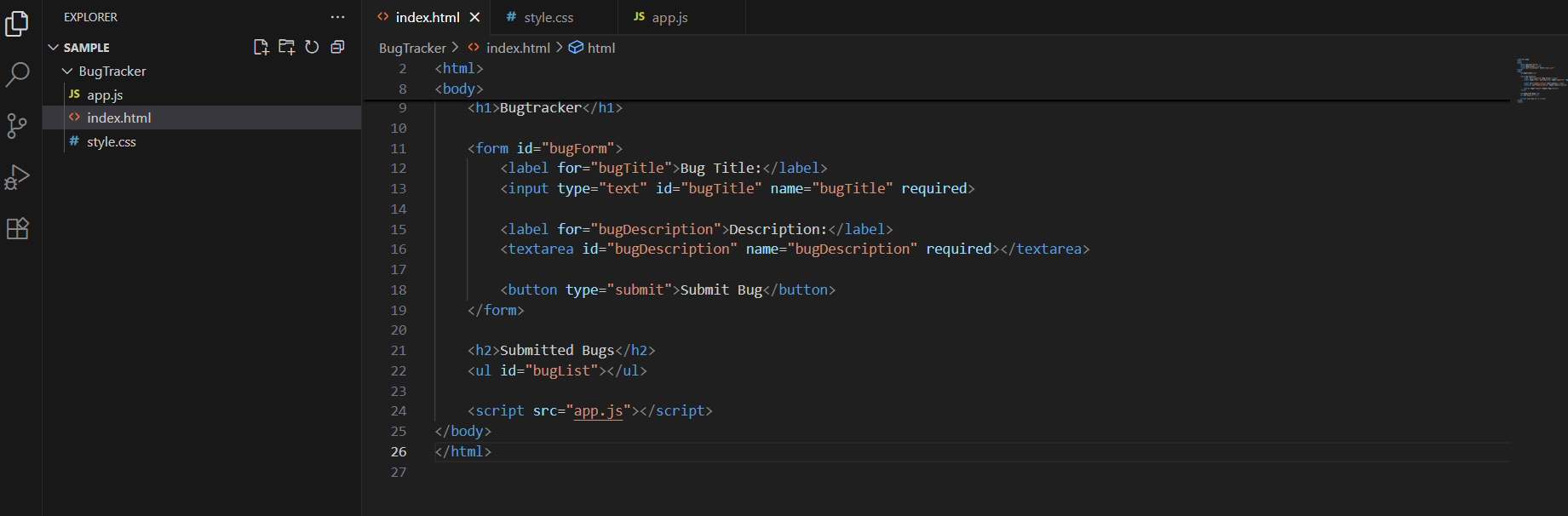
\* Create the following in the root of your repo:

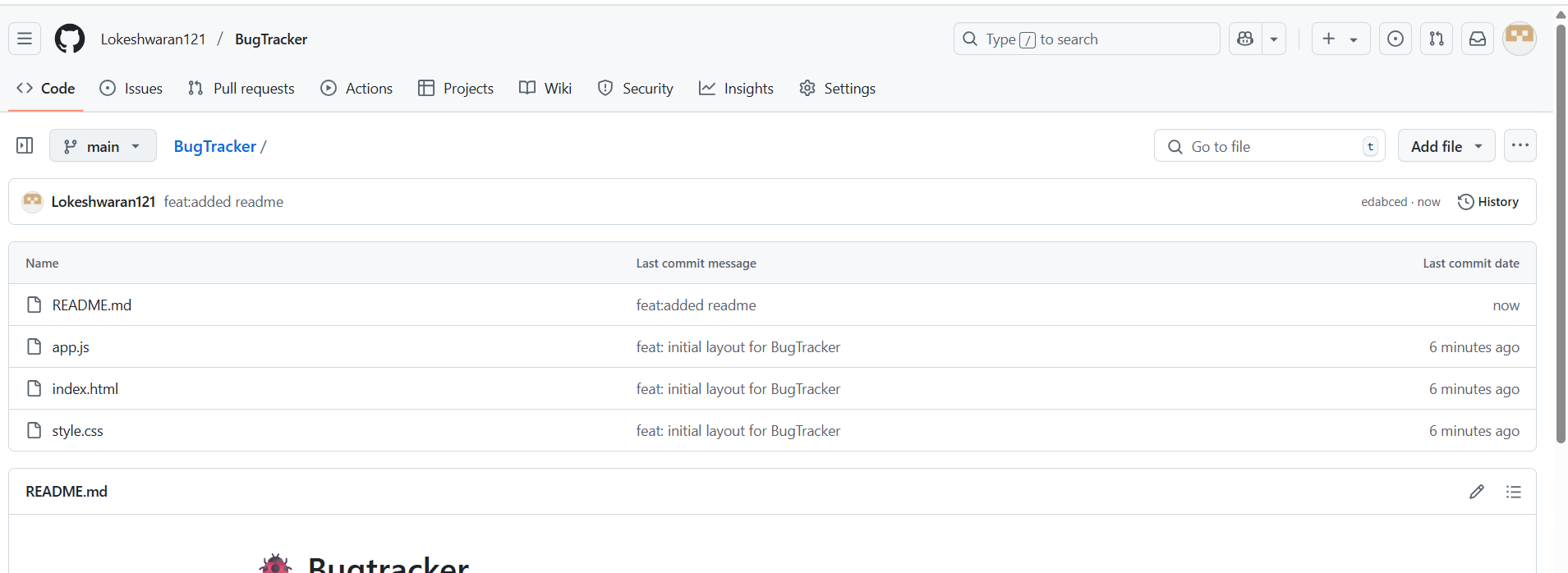
\* `README.md`

\* `index.html` (basic BugTracker layout)

\* `style.css` (simple styles)

\* `app.js` or `main.java` or `Program.cs` (based on backend choice)



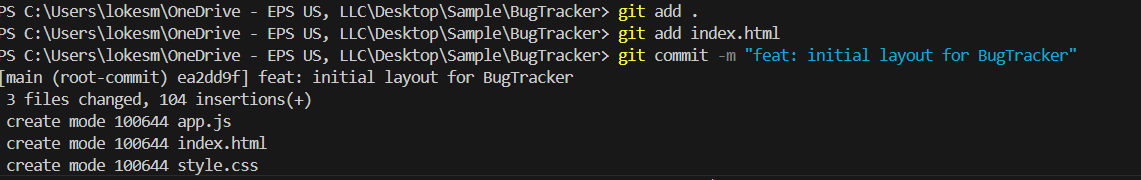


#### ✅ 7. \*\*Track Your Work\*\*

\* Add and commit the above files using \*\*conventional commits\*\*:

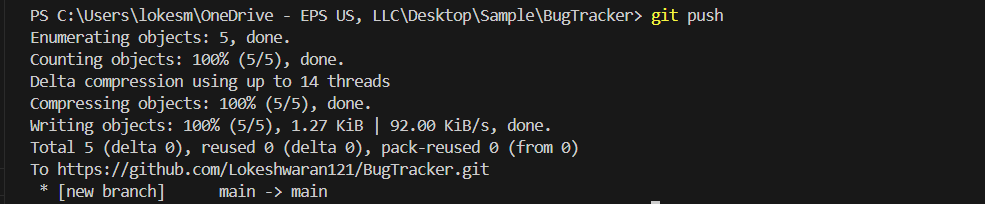
\* `feat: initial layout for BugTracker`

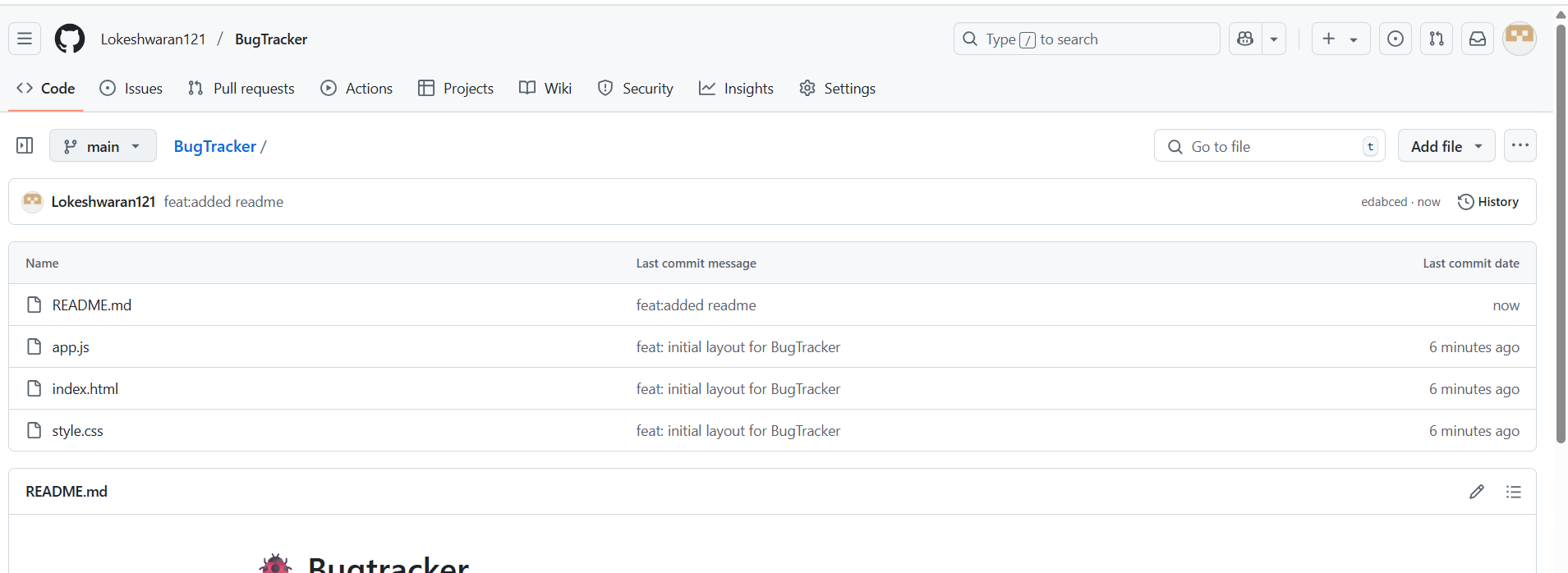
\* `style: basic styles for homepage`



#### ✅ 8. \*\*Push Changes to GitHub\*\*

\* Push to the `main` branch using GitHub token if needed.

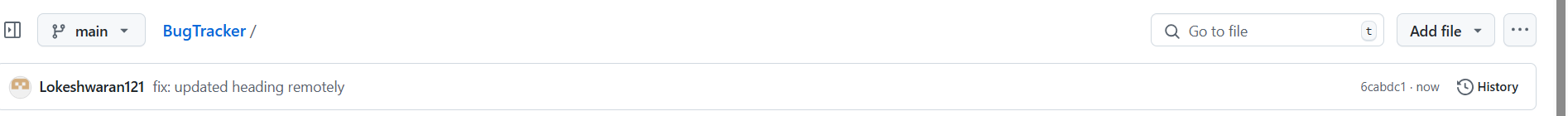




#### ✅ 9. \*\*Edit File Remotely (Conflict Setup)\*\*

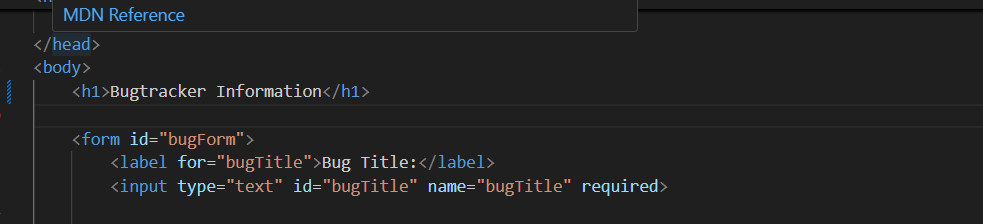
\* On GitHub, open `index.html` and change an `<h1>` tag.

\* Commit the change with: `fix: updated heading remotely`



#### ✅ 10. \*\*Change Same File Locally\*\*

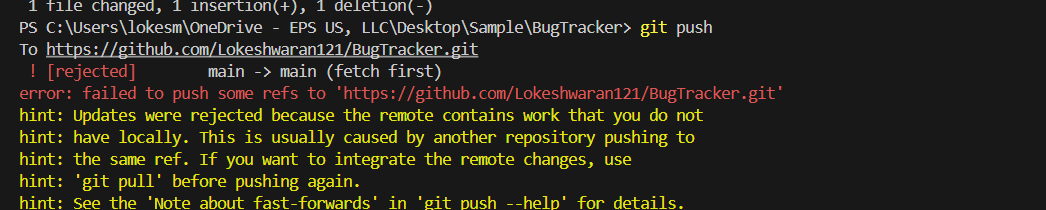
\* Locally, change the \*\*same line\*\* in `index.html`.



#### ✅ 11. \*\*Trigger Merge Conflict\*\*

\* Try to push your local changes without pulling first.

\* Observe the \*\*merge conflict error\*\*.

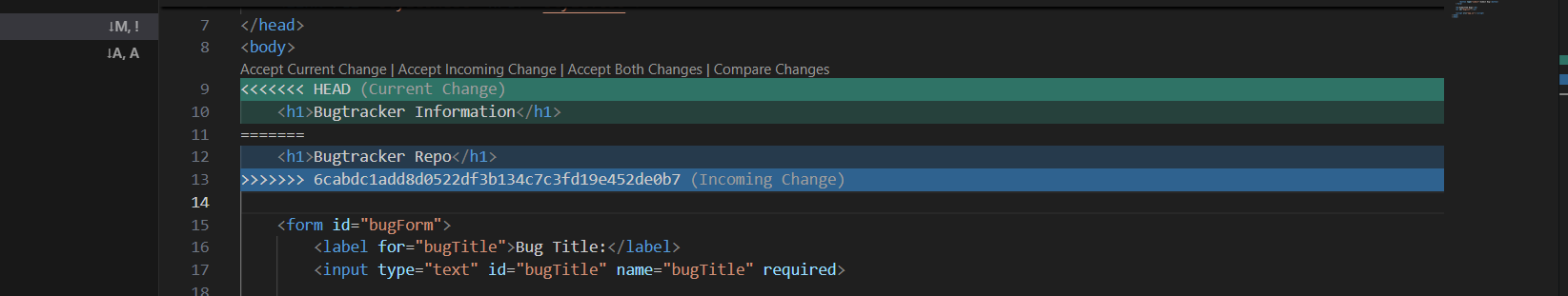


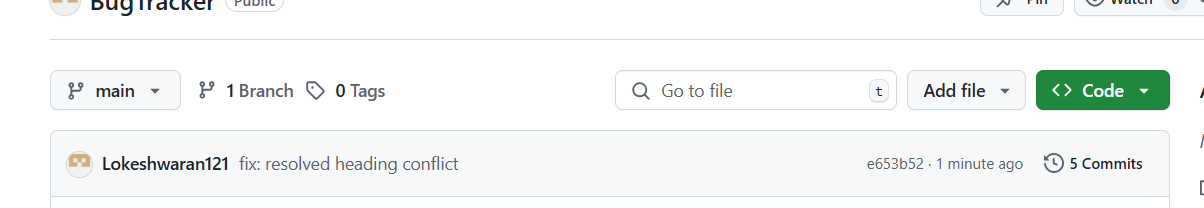
#### ✅ 12. \*\*Resolve Conflict\*\*

\* Pull from GitHub (`git pull origin main`)

\* Open conflict in VS Code and manually resolve it.

\* Commit the resolved file: `fix: resolved heading conflict`





#### ✅ 13. \*\*Branch for Frontend\*\*

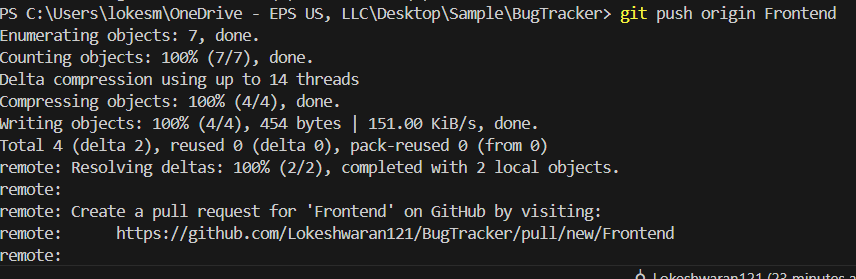
\* Create a new branch `frontend`

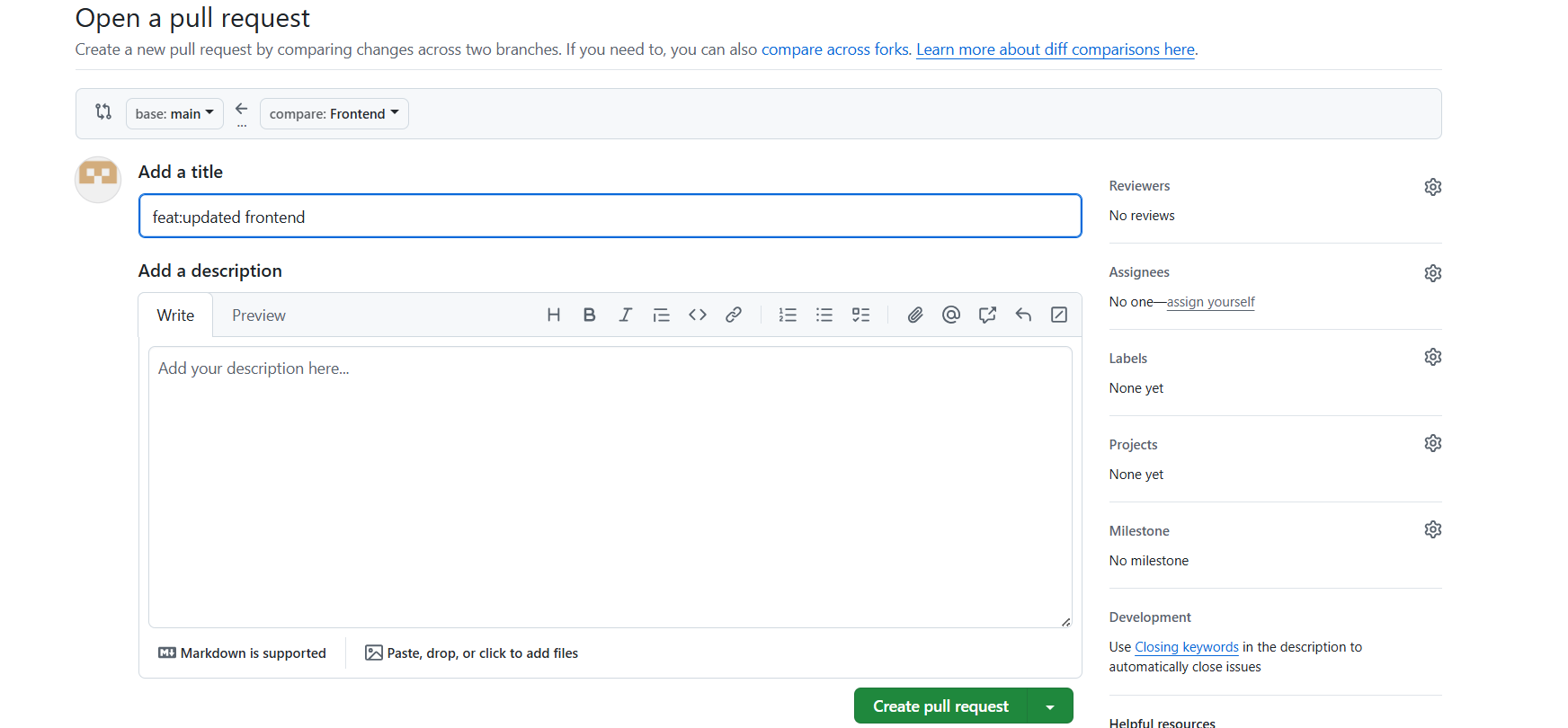


\* Modify/add to `index.html` and `style.css`

\* Commit your changes with a conventional commit.

\* Push the branch and create a \*\*pull request\*\* to `main`.





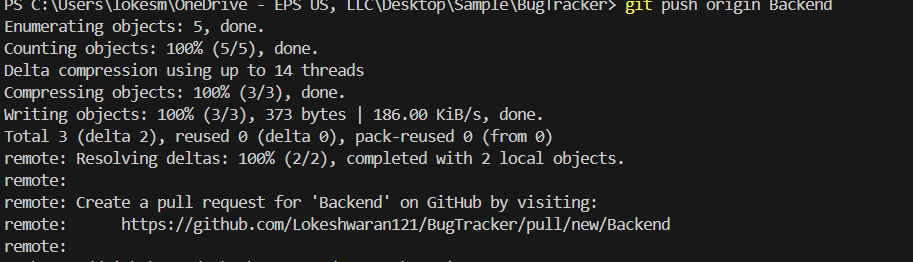
#### ✅ 14. \*\*Branch for Backend\*\*

\* Switch to `main`, then create a new branch `backend`

\* Create/modify your backend logic file (`main.java`, `Program.cs`, etc.)

\* Commit and push.

\* Raise a PR to `main`.



#### ✅ 15. \*\*Create `.gitignore`\*\*

\* Add the following:

```

node\_modules/

\*.log

.env

```

\* Commit it: `chore: add .gitignore file`

---

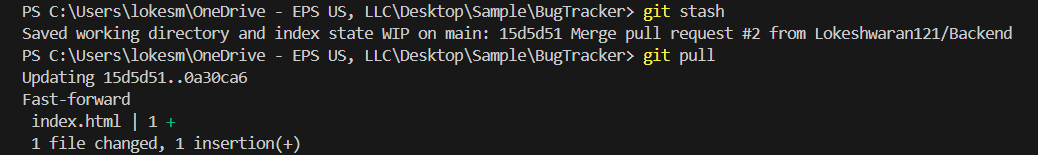
#### ✅ 16. \*\*Stash Scenario\*\*

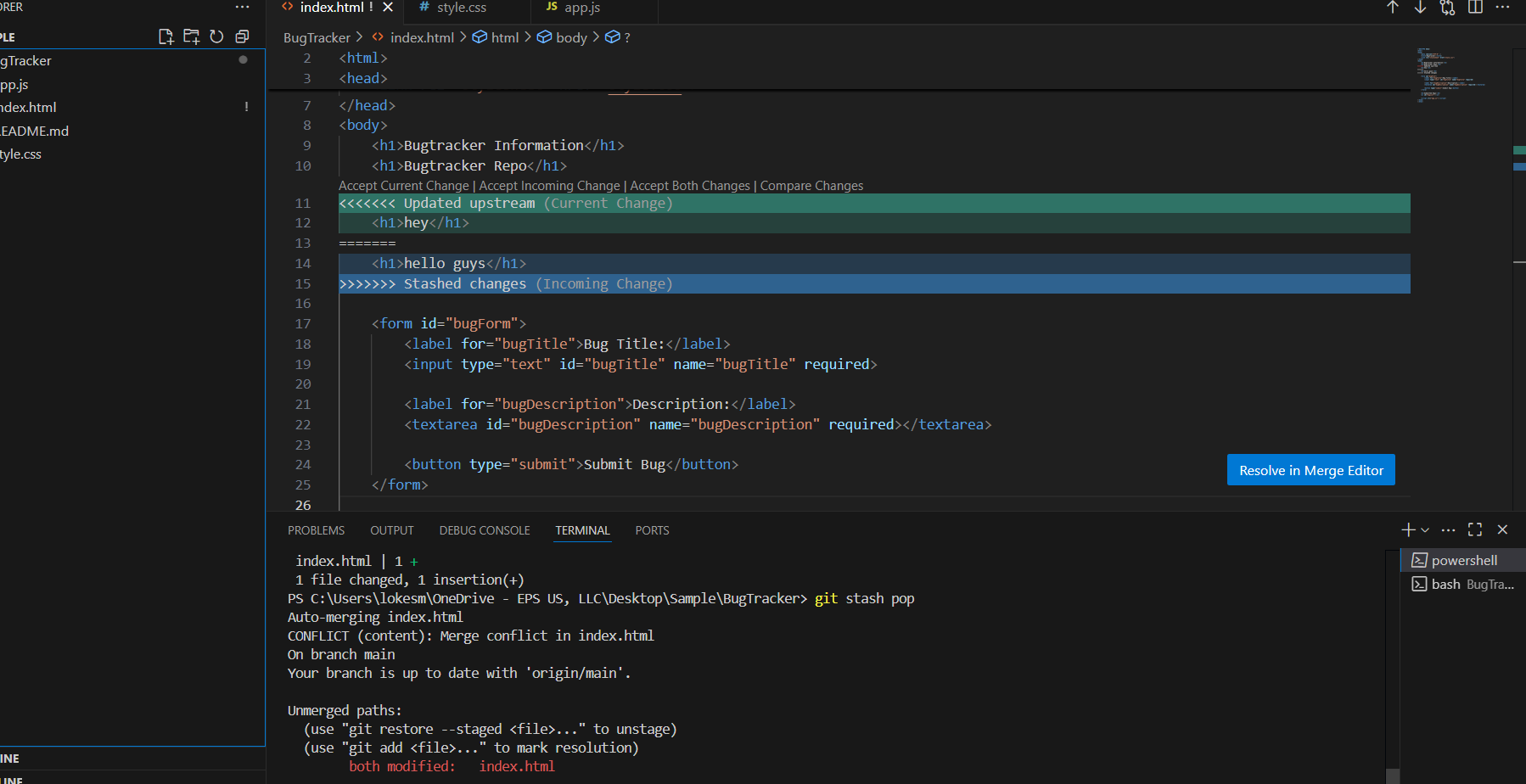
\* Edit any file (e.g., `index.html`) but \*\*don’t commit\*\*

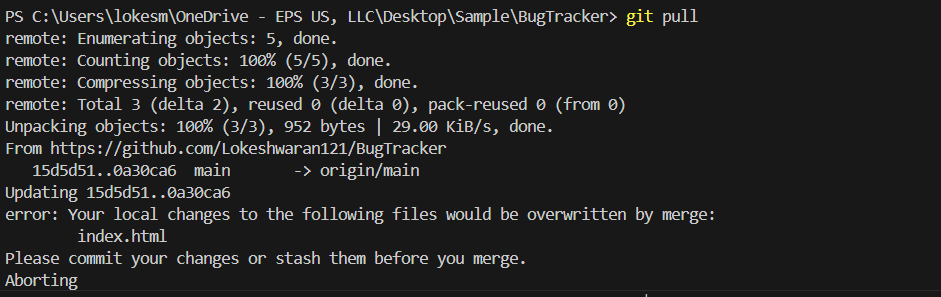
\* Try pulling new changes from GitHub — Git should block you

\* Use `git stash`, then `git pull`, then `git stash pop`

\* Resolve any conflict if it appears





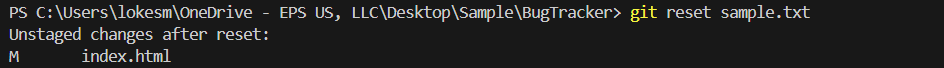


#### ✅ 17. \*\*Reset and Restore\*\*

\* Add and stage a new dummy file (e.g., `debug.txt`)

\* Then \*\*unstage it\*\* using `git reset`

\* Restore any accidental local change using `git restore`





#### ✅ 18. \*\*Rename Branch\*\*

\* Rename the current `main` branch to `dev-main` using Git



---

#### ✅ 19. \*\*View Commit History\*\*

\* Use `git log` and take a screenshot showing at least 5 commits with:

\* different commit types (`feat`, `fix`, `chore`, etc.)

\* merge conflict resolved commit

