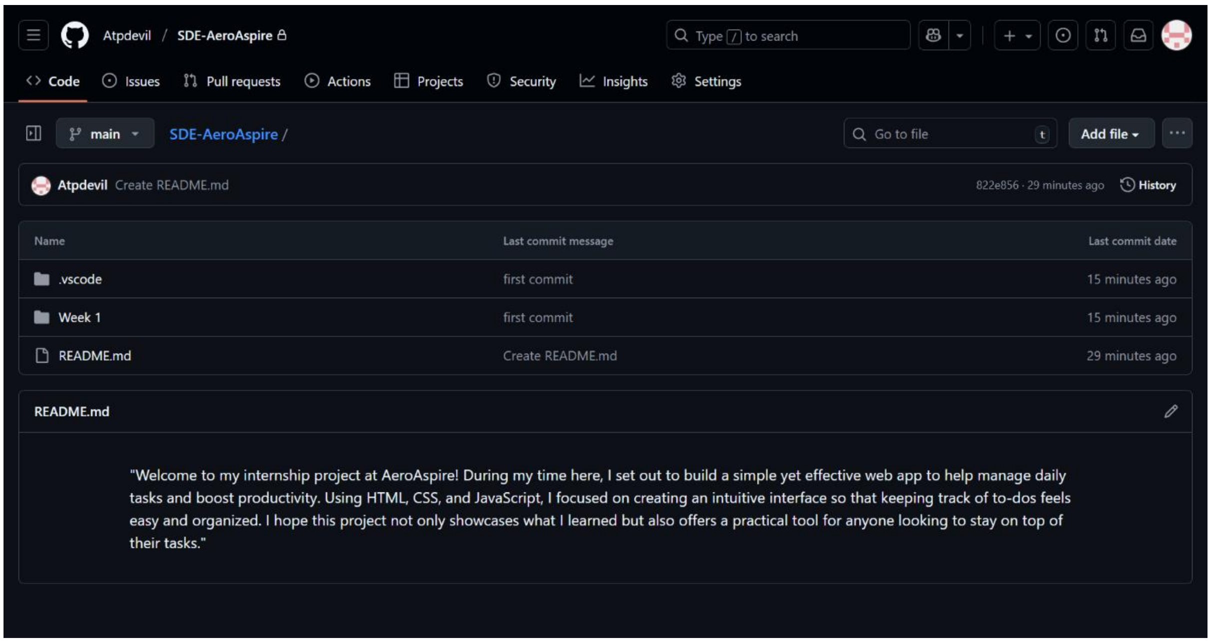
AeroAspire-SDE Intern
Gokul Krishna S

Week 1 – Day 3 (September 24)

Task/Assignment :

- Initialize repo; commit daily work; create feature branch; merge after review



Atpdevil Create README.md 822e856 · 29 minutes ago

Name	Last commit message	Last commit date
.vscode	first commit	15 minutes ago
Week 1	first commit	15 minutes ago
README.md	Create README.md	29 minutes ago

README.md

"Welcome to my internship project at AeroAspire! During my time here, I set out to build a simple yet effective web app to help manage daily tasks and boost productivity. Using HTML, CSS, and JavaScript, I focused on creating an intuitive interface so that keeping track of to-dos feels easy and organized. I hope this project not only showcases what I learned but also offers a practical tool for anyone looking to stay on top of their tasks."

Steps: For Commit in a new repository

- Create repository
- Name repository
- Initialize files and folders (git init)
- Add files to be pushed (git add ./filename)
- Add a commit message (git commit -m "Message")
- Add a url to act on(git remote add originhttps://github.com/reponame.git)
- Push Files(git push -u origin main)

Questions/Reflections :

1. **What is the workflow from making changes → staging → commit → push?**

- Work typically begins by making changes in a working directory (editing, adding, or deleting files).
- These changes are then staged using git add, which prepares them for the next commit.
- After staging, changes are saved (“committed”) to the local repository with git commit.
- Once committed, the changes are pushed to a remote repository using git push.
- This is often called “GitHub Flow,” and forms the core of most collaborative workflows.

2. **What is a merge conflict: what causes it, and how do you resolve?**

- A merge conflict occurs when two branches have competing changes to the same line in a file or when one branch modifies a file that another branch deletes.
- Conflicts happen most often during a merge or a pull from remote if others have pushed changes.
- To resolve: open the conflicted files, look for conflict markers (e.g., <<<<<<<, =======, >>>>>>>), and choose/merge the appropriate content. Save and stage the resolved file, then complete the merge with a git commit. There are GitHub Skills modules and interactive exercises that teach conflict resolution practically.

3. **Describe what happens under the hood with git commit: what objects are stored?**
   - When committing, Git creates a snapshot of staged changes and saves it as a new commit object in the repository history.
   - This commit object points to tree objects (representing directory content/layout) and blob objects (containing file data).
   - Each commit object also references its parent commit(s)—this forms the foundation for Git’s branching and history structure.
   - Visualization tools like Learn Git Branching demonstrate the internal relationships and storage of these objects interactively.