GitHub Usage Report-Software Development 2025

1. Introduction

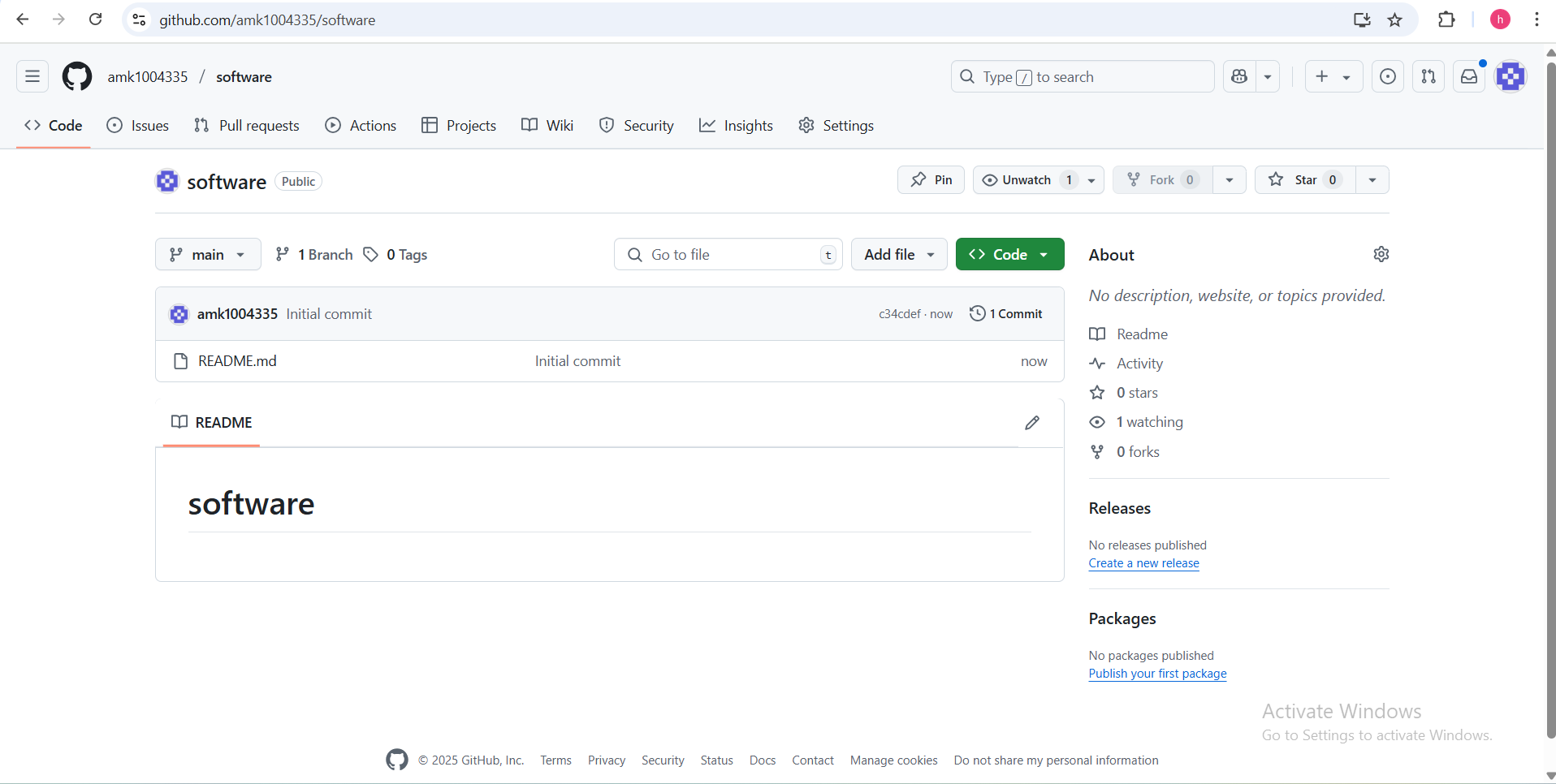
In modern software development, version control is the practice of tracking and managing changes to code over time. It allows multiple people to work on the same codebase simultaneously without overwriting each other’s work, provides a complete history of who changed what and when, and makes it easy to revert to earlier versions if bugs are introduced. Git is a distributed version control system that’s fast, flexible, and widely adopted; each developer has a full copy of the repository, enabling offline commits and branching. GitHub builds on Git by adding a centralized, cloud-hosted platform for collaboration: it provides an online repository where teams can share code, use pull requests for peer review, manage issues, and automate workflows. By using GitHub, our project gains centralized backup, clear audit trails, streamlined code reviews, and integrated tools for project management—making team development more efficient, transparent, and reliable.

1. Repository creation

We began by creating a new GitHub repository to host our project code.

1. Log in to GitHub and click the + icon in the top right corner.
2. Select new repository.
3. Enter the repository name.
4. Choose public so teacher can see our project.
5. Check initialize this repository with README to create the first file.
6. Click create repository.

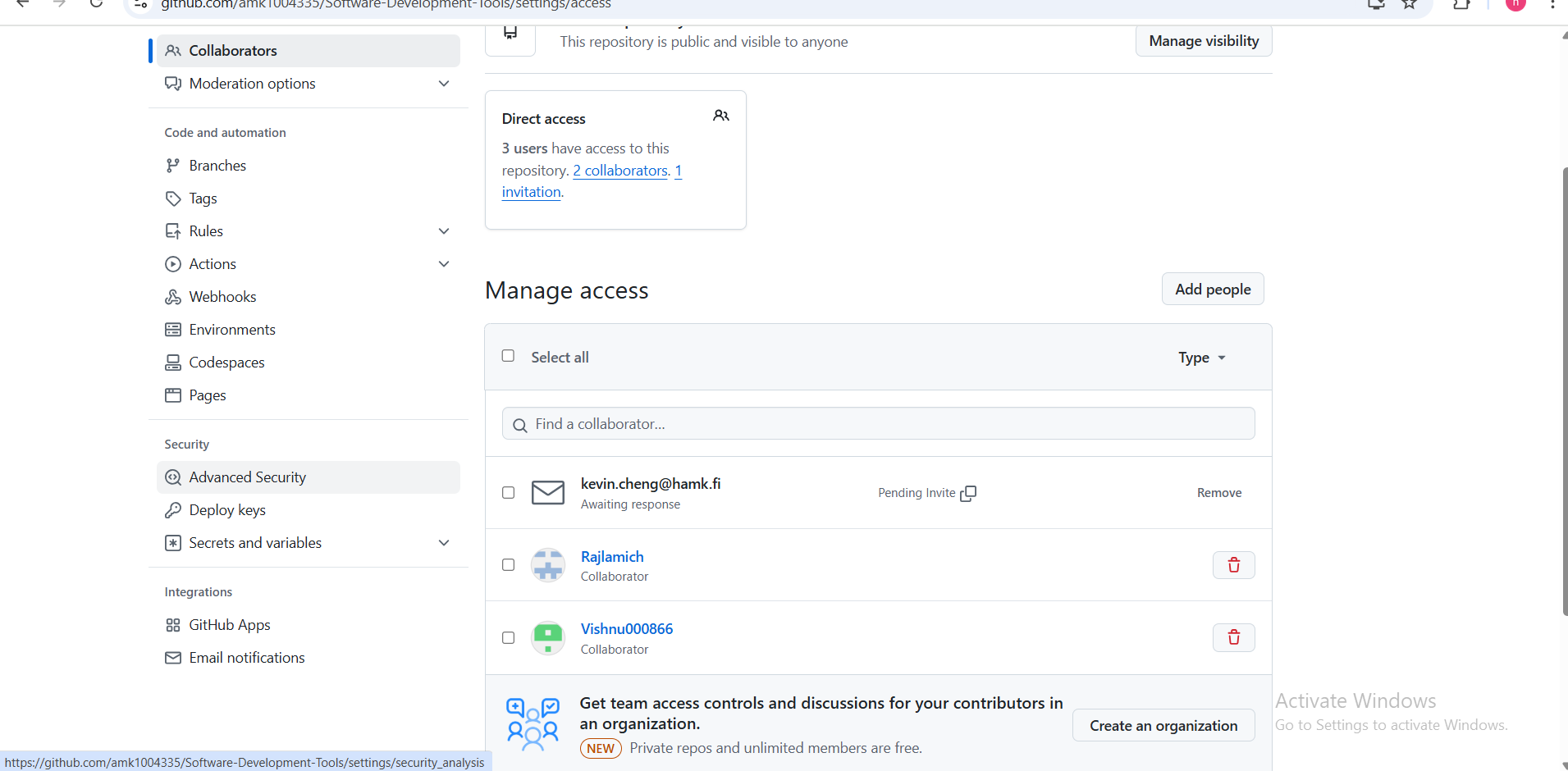
Fig. 1: Repository creation page and README.md file created on GitHub page



1. Managing Access

After creating the repository, we went to setting → manage access and clicked invite a collaborator. We then entered each team member’s GitHub username and granted them write permissions. This ensured everyone could, push changes, and review code directly in the repository.

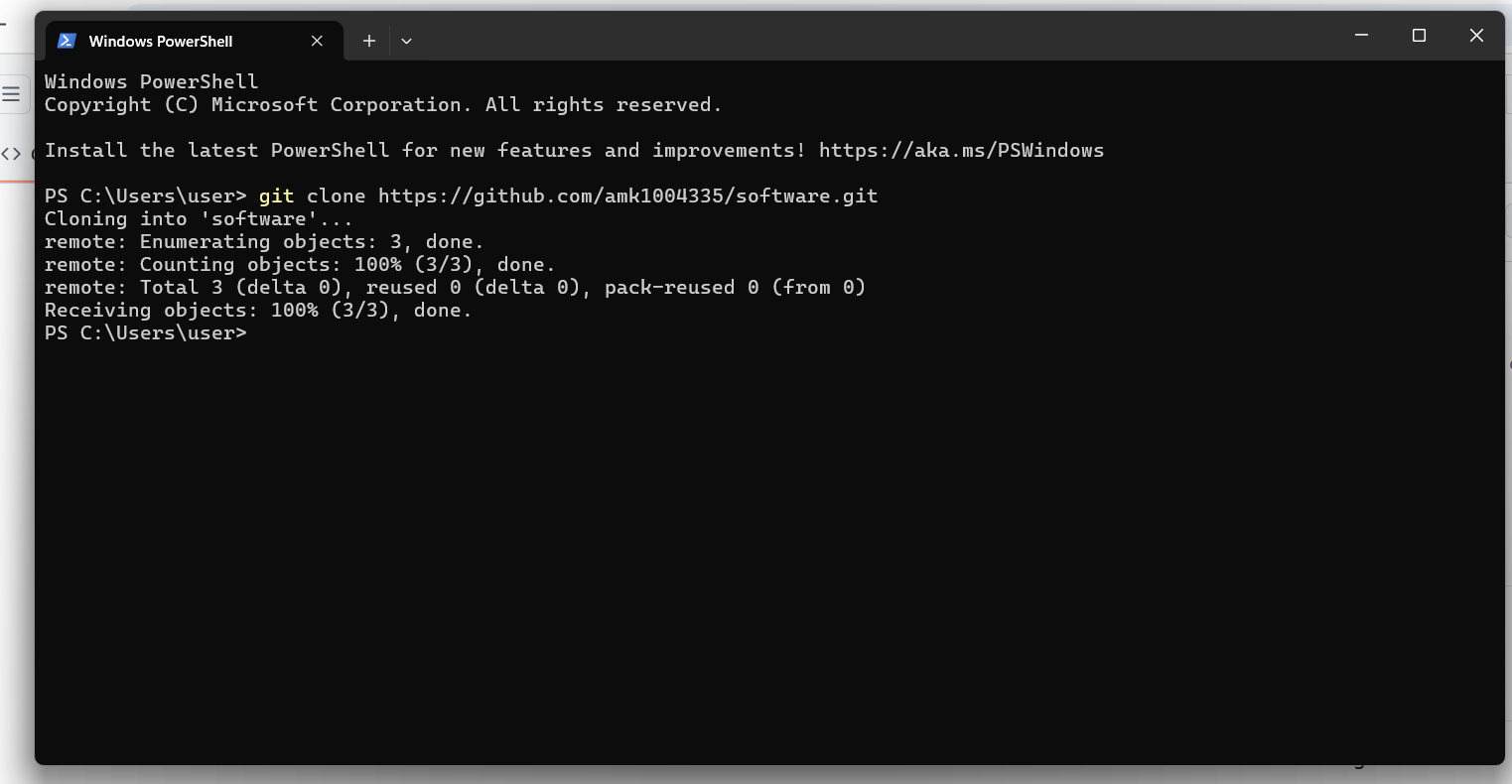
Fig. 2: Collaborators page showing invited team members



1. Cloning and setup

Each team member cloned the repository locally, navigated into the project folder, and configured their git identity .This established the local workspace for development and ensured commits were correctly attributed.

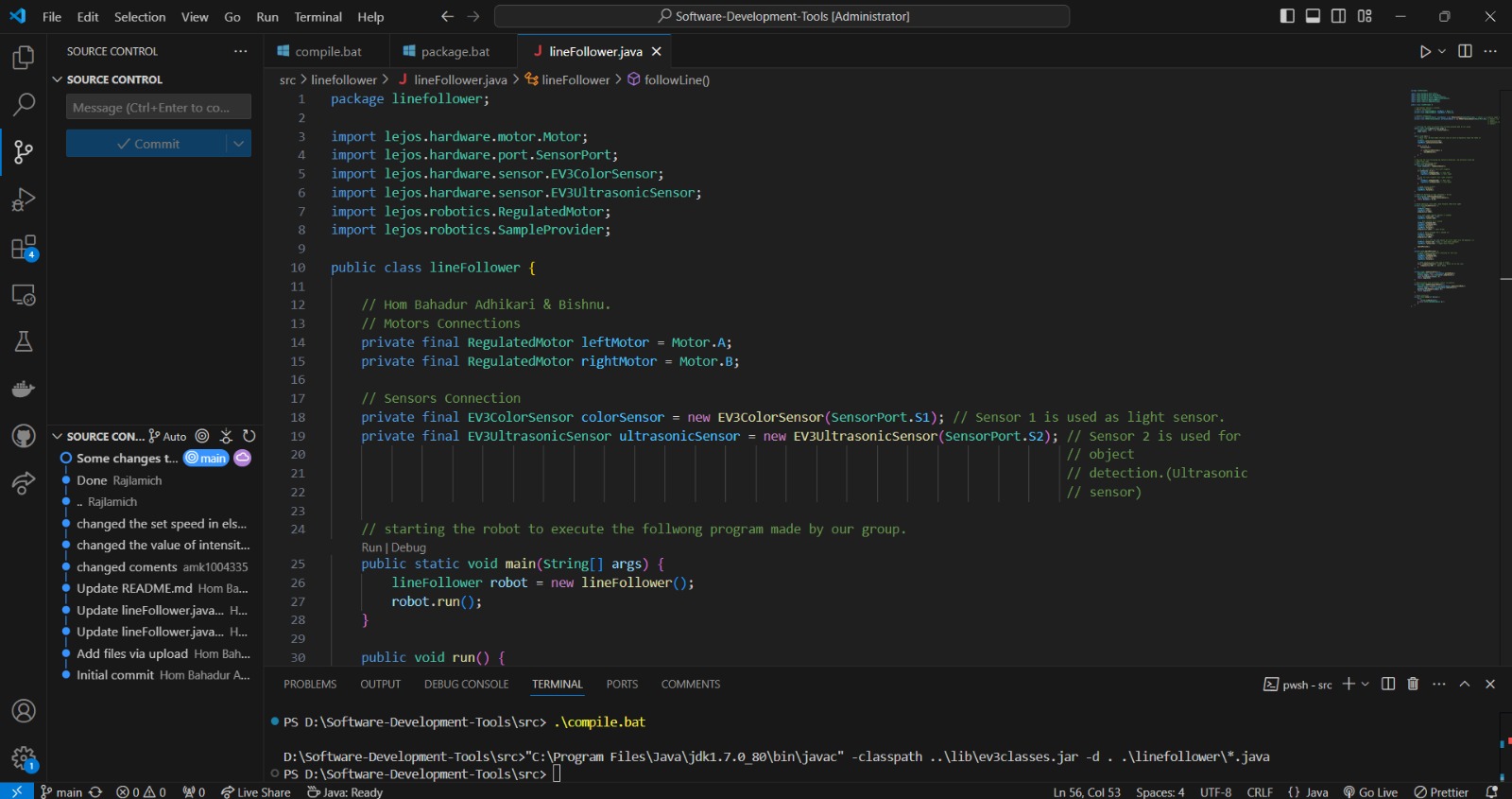
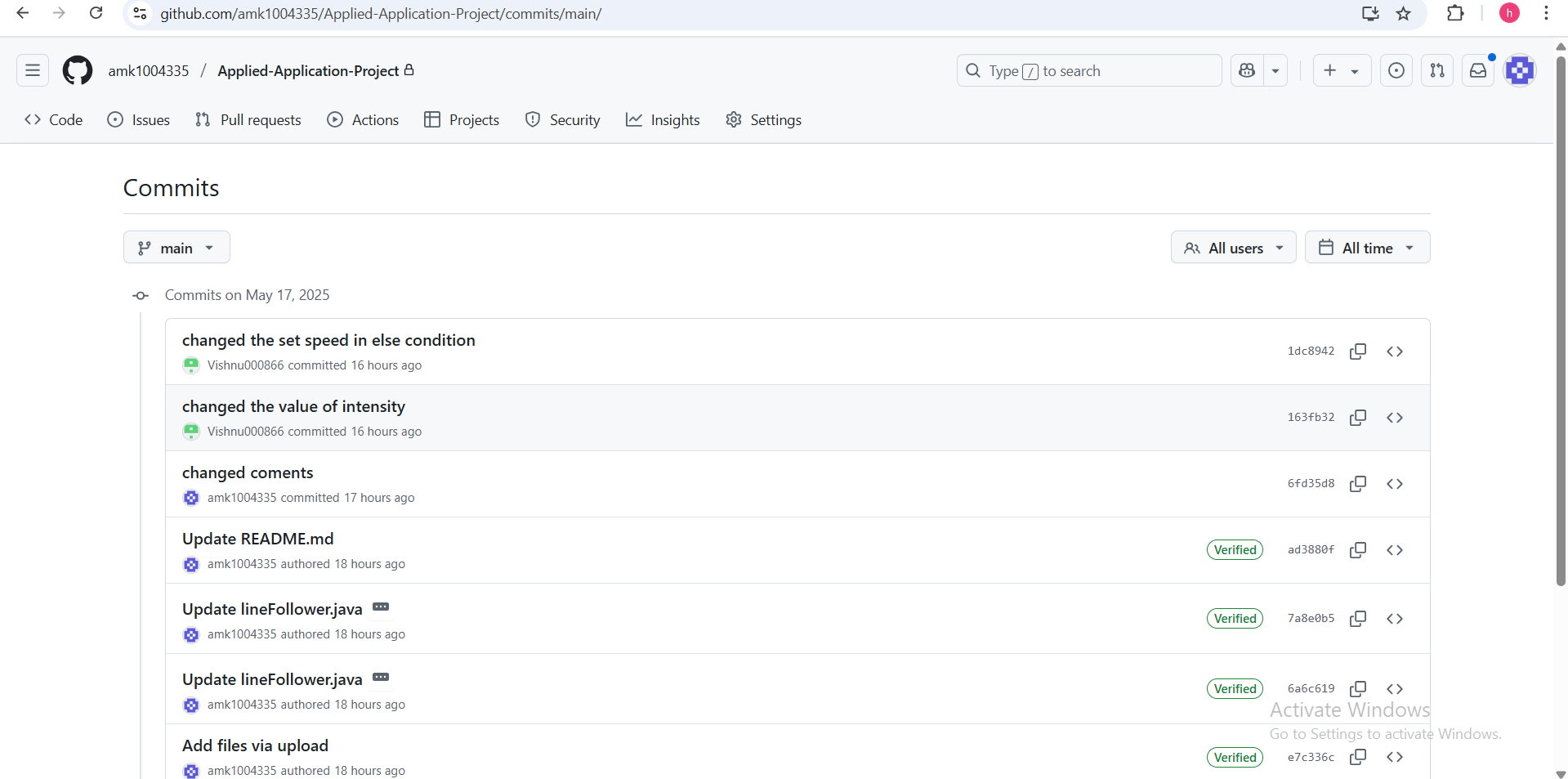
Fig. 3: Cloning and setup



1. Committing Code

Team members successfully committed the changes according to the requirements.

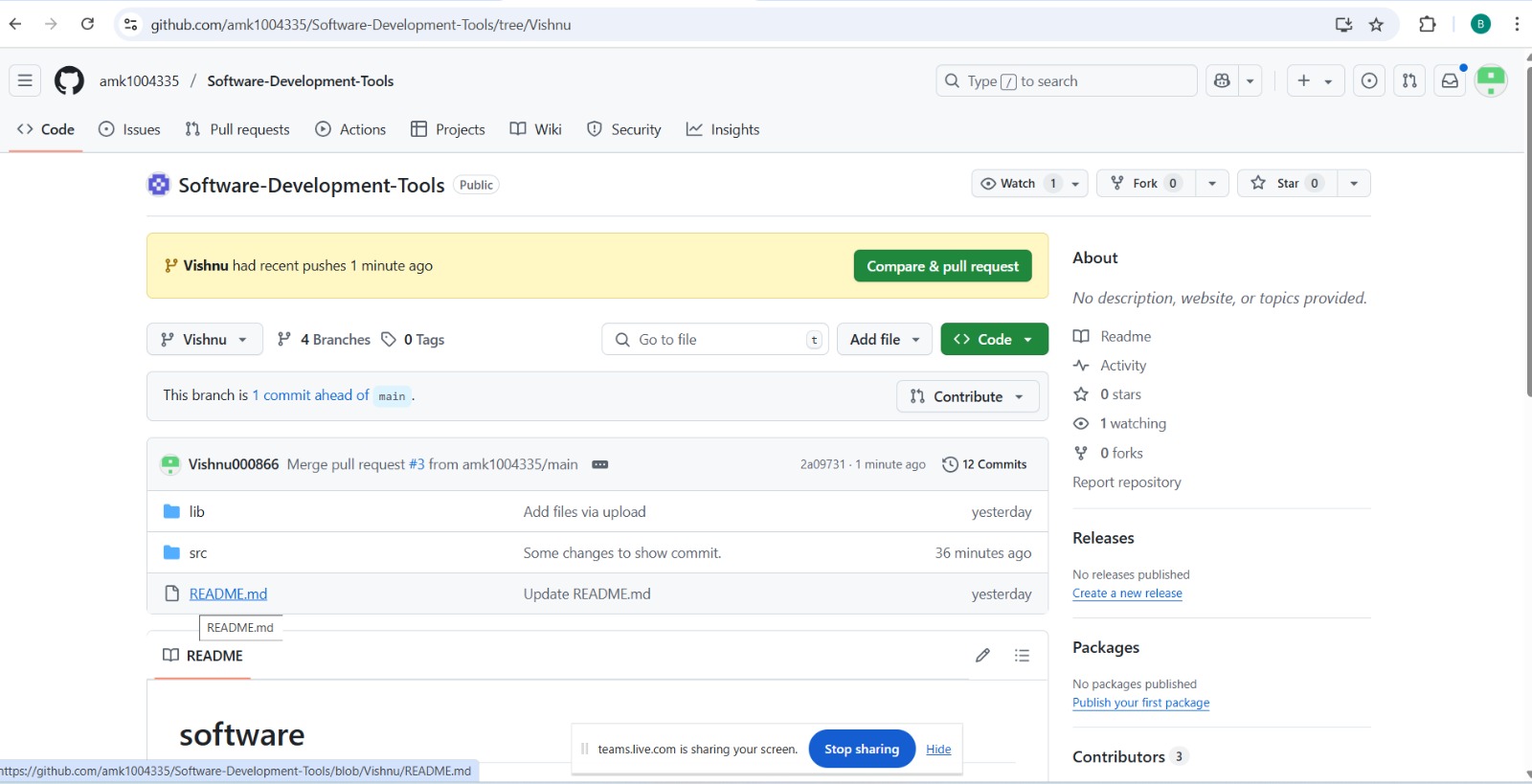
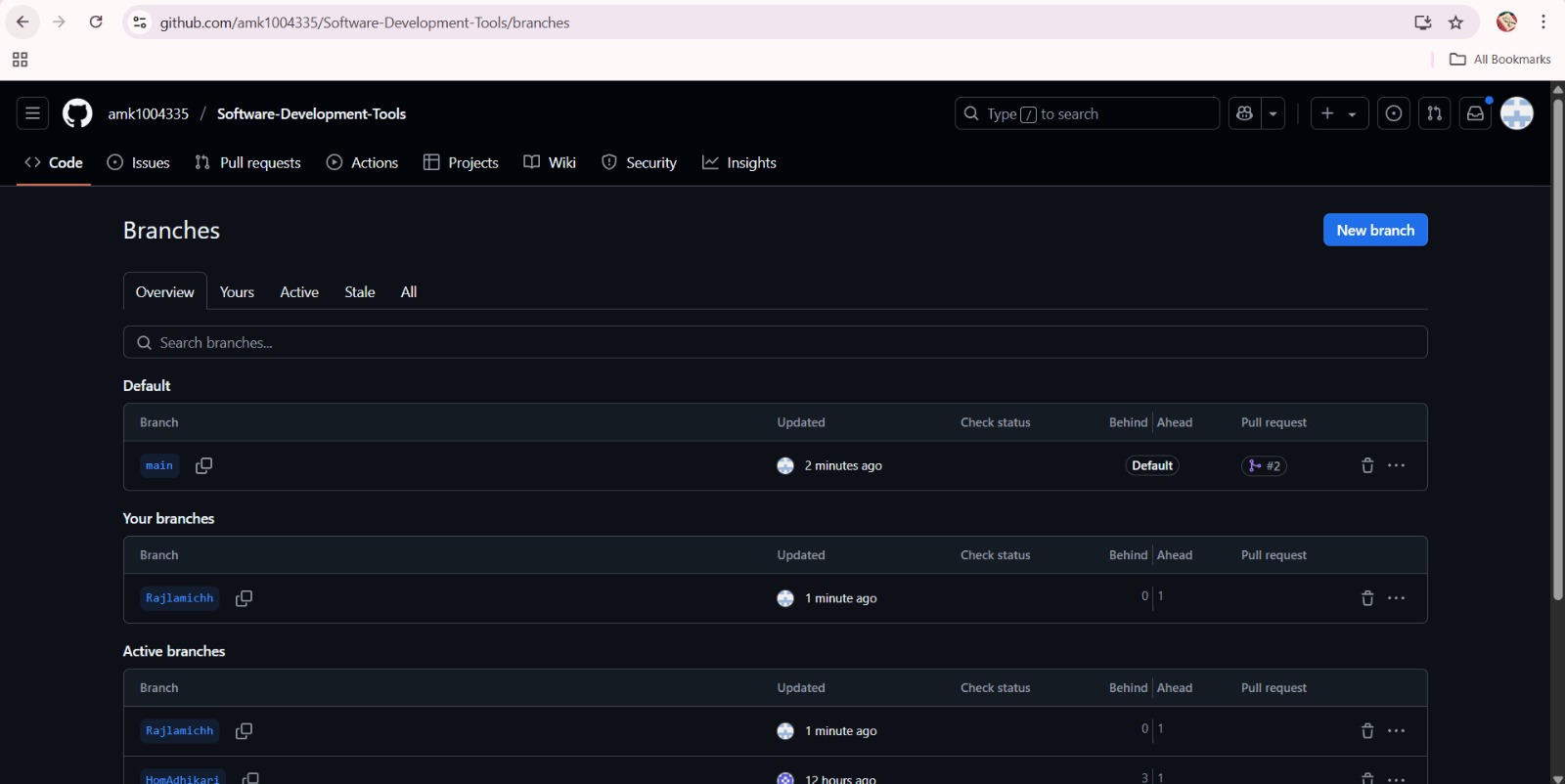
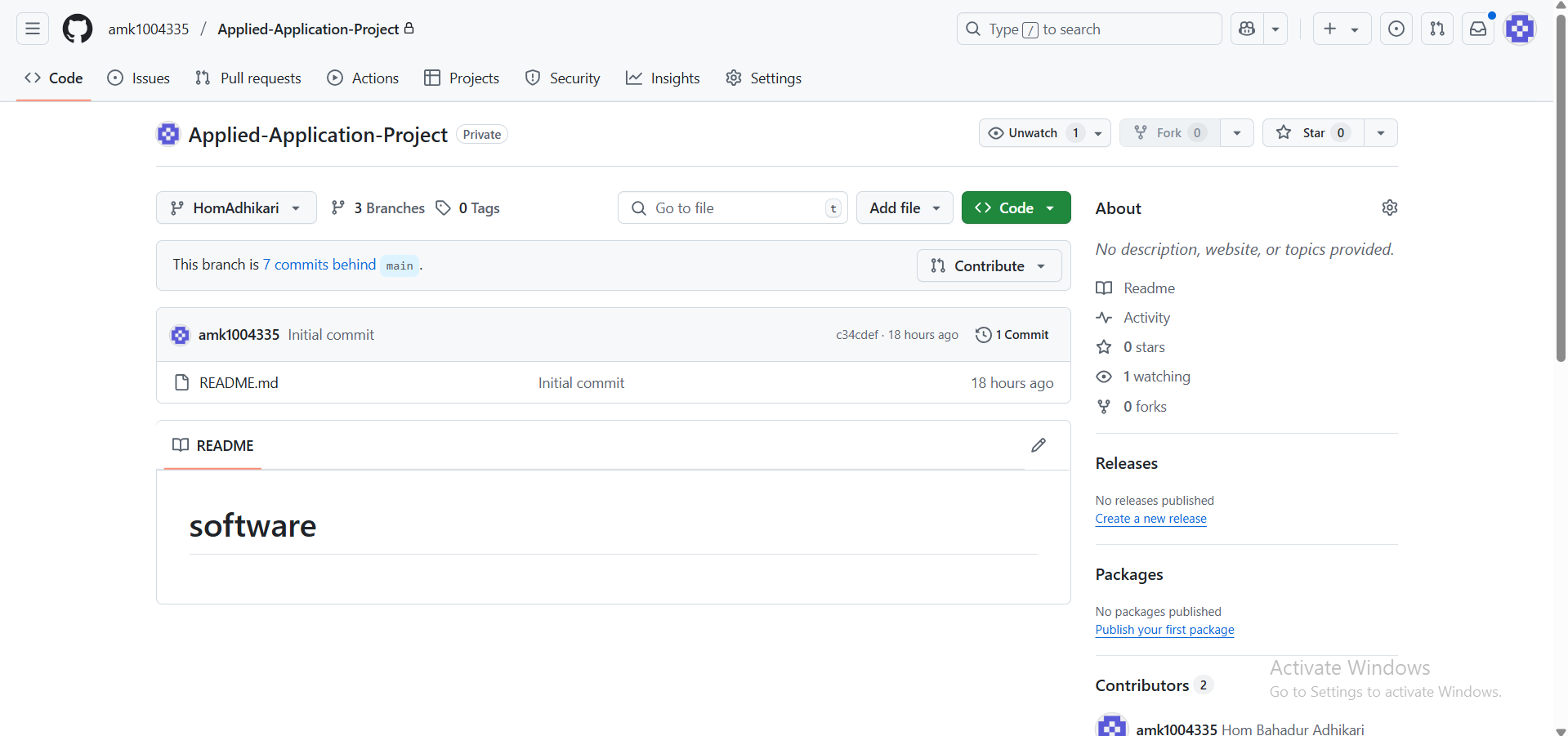
Fig. 4: Committing code



1. Branch Management

To keep our codebase clean and organized, we used branches to separate features. For example, separate branches were created for UI design, backend logic, and integration.

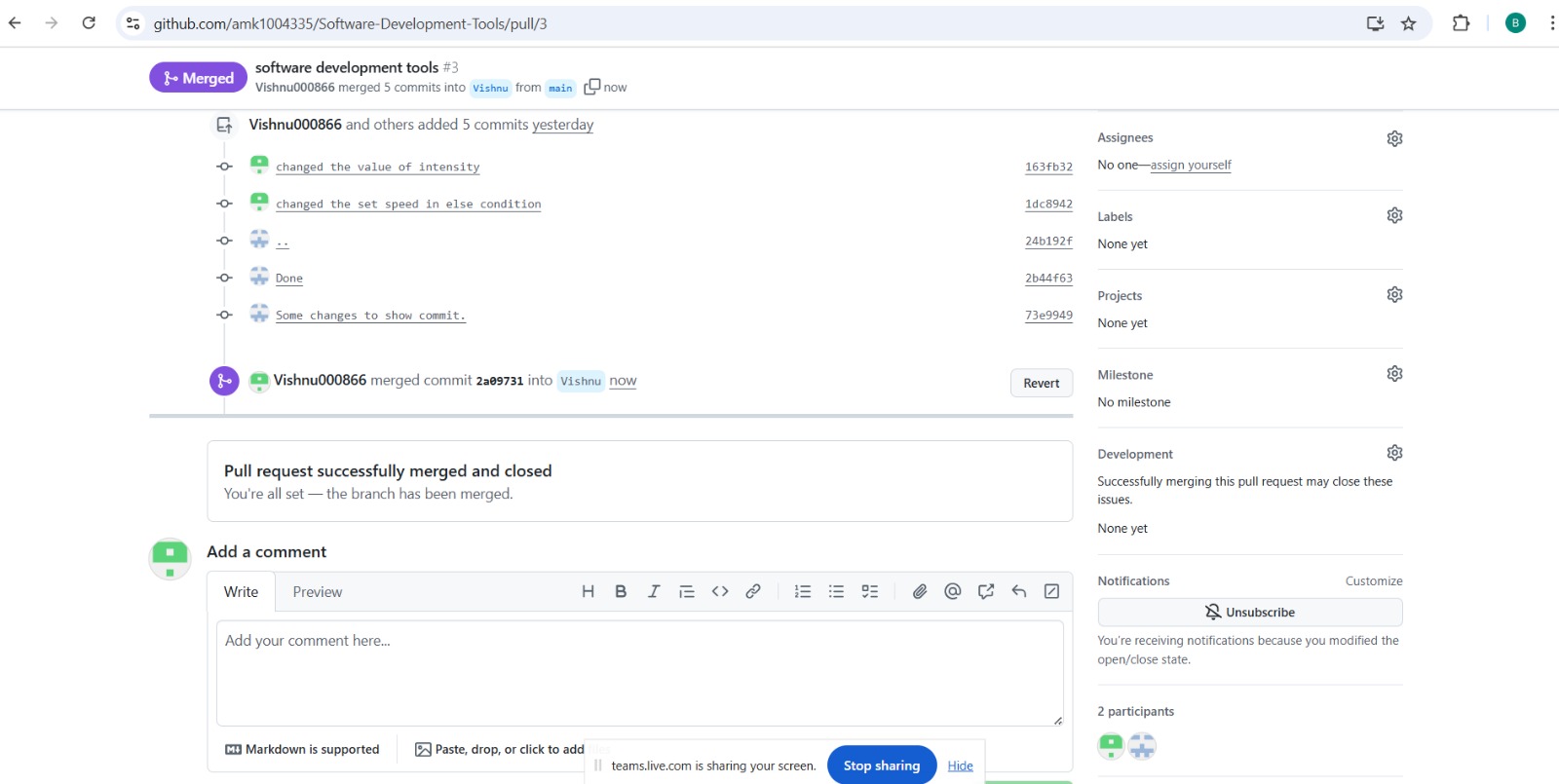
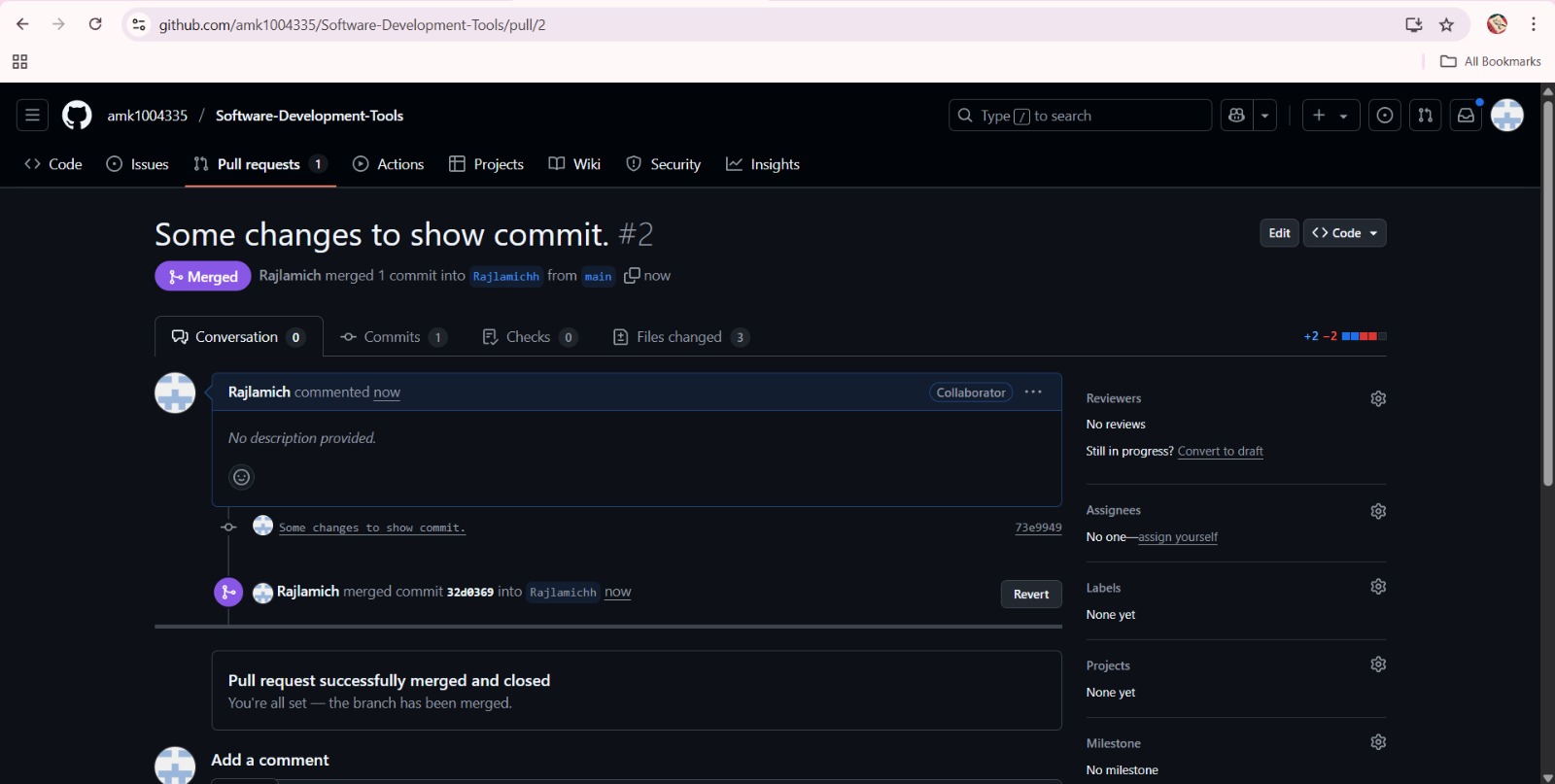
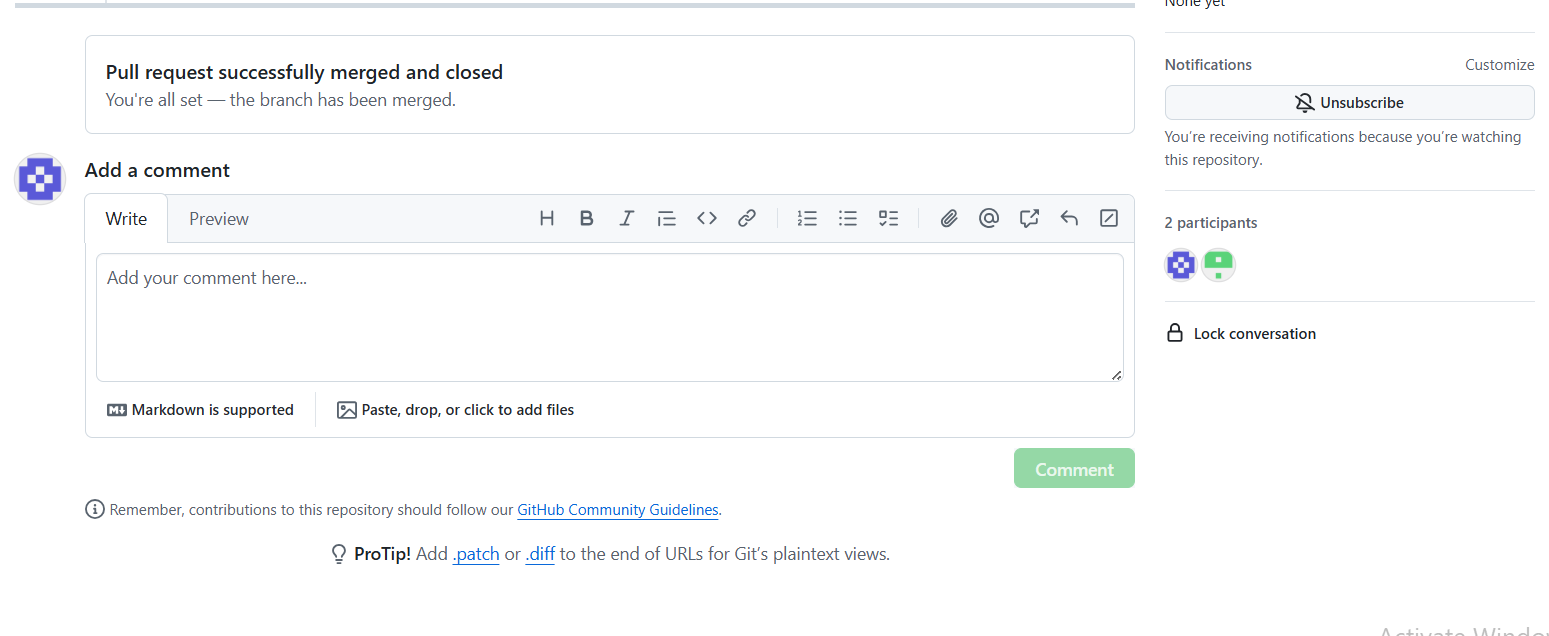
Fig. 5: Created branch



1. Pull Requests and merges

Pull requests (PRs) were created to propose changes from feature branches to the main branch. Each pull request allowed team members to review the code, suggest improvements, and ensure the changes were error-free before merging. This process helped maintain code quality and facilitated clear communication among collaborators.

Fig. 6: Pull request successfully merged and closed



1. Collaboration

Our team effectively collaborated by dividing tasks and working on different branches simultaneously. We used GitHub issues to track bugs and features requests, and discussions to communicate ideas and resolve doubts. This structured approach allowed everyone to contribute without overwriting each other’s work, ensuring smooth progress and timely updates throughout the project.

1. Conclusion

Using GitHub for version control significantly improved our project collaboration by providing a clear workflow for managing code changes. The ability to create branches, make commits, and review pull requests ensured code quality and minimized conflicts .Overall, GitHub helped keep our project organized, enhanced communication among team members, and increased the safety and reliability of our application development process.