Effective Collaboration on GitHub

What is GitHub Collaboration?

GitHub is a platform that can be used to collaborate on the development of software projects effectively. When it comes to this web application project, working in small teams on the same functionality of the web application using GitHub requires effective collaboration and version control practices to avoid conflicts.

Here’s the suggested workflow for the teams that can be used to effectively collaborate on the project.

Suggested Workflow

1. Branching Strategy

Branches in Git enable parallel development within a software project. Each team should create a branch dedicated to the specific functionality they are working on. Creating branches allows each team to isolate their changes and work independently on the functions that are assigned to them without interfering with each other’s codes or the main codebase until they are ready to be integrated.

2. Regular Pulls and Updates

Teams should pull the changes from the main branch regularly to keep the branches up to date with the changes that have been made in the codebase. This is a best practice in Git that also helps to minimise the chances of merging conflicts that arise later on when merging the changes later on.

3. Communication

Communication among the team members about the changes that are being done is important in order to avoid conflicts. The team members should have a clear idea about the tasks that they have been assigned. Members should discuss the changes that they are making, and any potential conflicts or dependencies beforehand to maintain a smooth workflow without disrupting each other’s work. Software such as Slack, Microsoft Teams, Discord or even WhatsApp can help team members stay coordinated.

4. **Pull Request Workflow**

The team members can create a pull request to merge their changes into the main branch when a team completes a part of the functionality or reaches a milestone. The pull requests should be reviewed by another team member to ensure the quality of the code and the compatibility. It allows reviewing and discussion and guarantees that the code complies with the project's requirements before merging the changes.

5. **Conflict Handling**

It is essential to resolve the conflicts that might occur when merging branches. Teams can avoid these merging conflicts by effective communication within the team and understating the changes made by all the other teams that work within the project.

6. **Code Reviews**

Thorough code review within the teams and across teams is compulsory to maintain the quality of the code. It also ensures the consistency of the project.

7. **Git Workflow Best Practices**

Frequently commit the new changes that have been made to the code with clear commit messages. Also, the teams should utilise Git's features effectively such as branching, tagging (used to capture a point in history), and reverting commits when needed.

8. **Extra Tools**

Project management software such as GitHub Projects and Trello can be used to organise tasks, set priorities, and track the progress of the team.

Advantages of Using a Version Control System

1. History tracking – VCS such as GitHub allows one to view the changes made to the project over time, who made them, and why smoothing out the collaboration and troubleshooting process.

2. Collaboration – The branches and merging ensure that multiple team members can work on the same project concurrently without having to face any conflicts.

3. Ability to retrieve previous versions – If any issues arise in the project, it is possible to revert to a previous functioning version of the codebase and retrieve old data.

References

https://code.tutsplus.com/how-to-collaborate-on-github--net-34267t

https://adiati.com/how-to-use-github-for-project-collaboration-based-on-agile-method

https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-tag#:~:text=Tagging%20is%20generally%20used%20to,no%20further%20history%20of%20commits.