Name: Melissa Johnson

Term: Group 15

Previous Team Projects

I have worked on group projects for two other courses at Oregon State University: Introduction to Usability Engineering and Introduction to Databases. In the Usability Engineering course, my group consisted of two people, including a friend from a previous class. We chose to work as a smaller group to minimize coordination and planning efforts. We each took on two roles in the project based on our strengths and interests, which complemented each other well. For example, my roles were UX Design and Prototyping and User Research and Communication, while my partner's roles were Leadership and Management and Writing and Deliverables. In this course, we designed a prototype for a mobile gardening app using Figma for prototyping. Although we used Figma's collaboration mode, which allowed us to work on the same file, we had to rely heavily on external communication to keep each other updated on the document's status.

In Introduction to Databases, all groups were composed of two students. We built a front-end UI to interact with the backend database we built throughout the term. Although we both used Git for version control and GitHub as a remote repository, my partner had not taken this course yet and preferred sending files back and forth with the piece each of us had worked on. We essentially branched and merged changes manually, which resulted in a lot of duplication and redundancy.

In both courses, my group and I had clear communication, set expectations early on, and were organized throughout the projects. However, the main difficulties we encountered stemmed from a lack of continuous integration or using version control systems. In the Usability Engineering project, we had to communicate frequently outside of the design tool to keep each other updated on the status of the document, and in the Databases project, we had to manually merge changes.

Working with Continuous Integration

When I first started working with continuous integration (CI) in this course's group project, I was excited about the prospect of a more streamlined workflow for a shared codebase. In my professional experience, collaborating on projects with colleagues can be frustrating as we don’t have the same version control tools and practices in place for editing reports and presentations. We often end up with multiple versions of the same file, with confusing naming conventions that make it difficult to keep track of changes and merge them together.

While I didn't have any major reservations about using continuous integration (CI) specifically, I did have concerns about how our group dynamic would work out. Fortunately, my group was highly motivated and began working on the project soon after it was assigned. However, we had not yet established clear expectations for how often we would commit changes or the timeline for completing code reviews. This initially led to some confusion and a hectic pace, but we were able to quickly address these issues and establish a more effective workflow as a team.

As a computer science student, we’re often reminded to write clear code- give variables and functions meaningful names, don’t repeat code, test your code, comment appropriately – it wasn't until I started using continuous integration that I truly appreciated the value of doing all these things. Continuous integration provided me with a valuable opportunity to see how other students went about solving a problem, and it allowed me to review and assess my comments and their comments in a more concrete way. In this program, we don't have many opportunities to see other students' code and we often don’t receive in-depth feedback on our own code, especially when there aren’t errors in an assignment, so using continuous integration to review code and provide feedback was a useful exercise to gain more experience in evaluating code. This experience reinforced the importance of clear and concise comments and serves as a reminder that good commenting can greatly enhance the readability and maintainability of code, and it's a practice that should be prioritized throughout the development process.

Using continuous integration in a team project helped me grow as a team member in several ways. It reminded me that clear communication, transparency, and expectation setting are essential in ensuring that group projects run smoothly. Continuous integration provided a way for everyone on the team to see the progress of the project and the contributions of each team member, which promoted accountability and transparency. Additionally, it allowed us to catch errors and bugs early, minimizing wasted effort and time. Continuous integration in a team project reminded me of the importance of clear communication and expectation setting. I realized that I can take on a leadership role to ensure that these aspects are effectively addressed, and that transparent communication promotes accountability and a shared understanding of project goals.

Lessons for the Future

Continuous integration has been instrumental in facilitating better software development practices, and part of that was done by having mandatory code reviews. Besides being able to identify potential bugs and logic errors, it also provided an opportunity to learn from others' coding styles and techniques. This not only improves the quality of the code but also helps to build a culture of collaboration and learning within a team.

Another important aspect of software development that continuous integration helps facilitate is the need for a solid test suite. When working on a shared code repository, it's crucial to ensure that changes made by one developer don't adversely affect the work of another. Continuous integration allows for automated testing, which helps to ensure that the code is functioning correctly after every change is made. This reduces the risk of introducing bugs into the codebase and also saves time by catching issues early in the development process.