Name: Paris Zhou

Term: Fall 2024

Previous Team Projects

Throughout my programming career, I have had the opportunity to work on both solo and group projects. One notable group project involved developing a database-driven website, where my primary responsibility was backend implementation. Specifically, I focused on writing SQL queries and ensuring the database schema adhered to the Normal Form 3. This was crucial in maintaining data integrity and optimizing the performance of the website. Additionally, I took steps to ensure that the schema was correctly formatted, which contributed to the project's overall success.

Despite the positive outcomes, the project did not come without its challenges. One significant issue we faced was communication, particularly when it came to coordinating tasks and aligning on project goals. This was compounded by punctuality problems, which caused delays in progress and led to occasional misunderstandings about deliverables. We eventually addressed these difficulties by improving our use of collaboration tools and setting clearer deadlines, but the experience taught me the importance of consistent communication and time management in team-based projects.

Working with Continuous Integration

When I first encountered Continuous Integration (CI) as part of this project, I had some reservations due to my lack of experience with GitHub and pull requests. I had never worked within a CI workflow, and the idea of automating builds and managing pull requests was new to me. To overcome this, I took the time to familiarize myself with GitHub's functionality and studied how pull requests worked within the CI pipeline. Gradually, I became more comfortable with the process, and it helped streamline our team's workflow.

Our team had a mandatory code review process, which was also unfamiliar territory. To make the reviews more effective, we primarily used Discord for communication. This helped in discussing the rationale behind certain code changes and clarifying any doubts before the pull requests were merged. When reviewing pull requests, I followed a structured approach: first, I checked if the build had successfully completed, then reviewed the commits and commit comments to understand the changes, and lastly, I assessed the functional impact of the pull request on the codebase.

While daily commits were not heavily emphasized, I found that regular communication in our Discord channel kept the project on track. This informal, yet consistent, communication allowed the team to address any issues or updates without feeling the need for strict daily commits.

In terms of Test Driven Development (TDD), our team didn't follow the process to the letter. Initially, the concept of building code first and then writing tests didn't feel intuitive to me, so I started developing and testing in tandem on my own machine. Although I didn't commit or document these tests along the way, I did realize that this approach wasn't ideal. I now see the value of fully adopting TDD for maintaining clean, tested code, and I plan to continue practicing it to improve my skills.

Name: Paris Zhou

Term: Fall 2024

Overall, my experience with CI helped me grow as a developer, particularly with GitHub and TDD. I also learned the importance of being an effective team member, and in this project, I adopted a more follower role. This allowed me to learn from others, especially the repository owner, who had more experience with the CI workflow. Though my mentoring contributions were modest, I did offer minor suggestions for code improvements to my teammates.

Lessons for the Future

This project has provided valuable lessons that will undoubtedly influence my future work as a software developer. One of the key takeaways is the importance of Continuous Integration (CI) in facilitating better software development. CI helps resolve a common challenge in team-based projects: the goal alignment problem. When multiple developers contribute to the same project, it can be difficult to ensure that everyone is on the same page regarding changes and overall direction. With CI, changes are integrated regularly, and the build process ensures that everyone's contributions are validated and documented. This keeps everyone aligned, reduces conflicts, and makes it easy to understand what changes have been made at any given time.

The mandatory code review process was another essential learning experience. Code reviews are an effective safeguard against submitting sloppy, inefficient, or even non-functional code. By thoroughly reviewing each other's work, we were able to catch potential bugs, enhance code readability, and ensure consistency with project standards. This process fosters a collaborative environment where knowledge is shared, and developers learn from one another, thus improving individual coding practices and the overall quality of the codebase.

Incorporating Test Driven Development (TDD) into our workflow, though not fully embraced by the team, helped me understand how it leads to better software. TDD forces developers to consider edge cases and functionality early on, ensuring that the code not only works as intended but also handles unexpected scenarios. The process provides a layer of documentation and forces a deeper understanding of the problem at hand, which ultimately results in more robust and maintainable software.

Finally, solid test suites are vital when working on a shared code repository. With multiple team members contributing to the same codebase, tests serve as both a safety net and a measure of functionality. They ensure that new changes do not break existing functionality and contribute to good code hygiene by making it easier to identify and fix bugs. A well-maintained test suite makes the process of integrating code more efficient and provides confidence in the stability of the application as the project evolves.

Through this project, I have learned how these practices—CI, code reviews, TDD, and solid testing—are essential for high-quality software development. They ensure that development processes are smooth, code is maintainable, and the team is aligned toward common goals. These lessons have prepared me for future roles as a software developer, equipping me with the tools and mindset to work effectively in team environments while maintaining high standards of software quality.