CS4300 – Final WriteUp Lucas Estevez

CS3300 and CS4300

CS3300 was a great learning experience for working on a large project over the course of a long time and through multiple sprints, rather than the standard coding assignemnts which are usually only worked on for a week or so. It was also interesting to work on the same code with multiple other people while also working on completely different features, and compiling our work into one application. However, our group in CS3300 did not really use branching at all, rather all pushing to main on a regular basis (somehow, our stuff worked). We also did not have any real testing, very unlike CS4300.

CS4300 on the other hand was really neat because we got to see the backend of how projects come together. Learning about Pull Requests, AI Code Reviews, and separate branches for new features was very interesting for me and I really enjoyed getting to see how it all came together.

AI in Software Engineering

We've learned a lot about AI in this class and its benefits and abilities, as well as its limitations and deficiencies. I enjoyed getting to see the AI code reviews get implemented in everyone's projects, and thought it was really interesting to see what it thought about our project and individual commits. I also got to see what it could do with the grad students' project and how it was developed as well, but did get to hear plenty about its limitations and that group's frustrations with it. Developing with only AI, not even being able to make slight tweaks by hand, does not really sound enjoyable and I'm kind of glad I didn't have to do it, but it was cool to watch and hear about.

Key Software Engineering Concepts

I've gotten to learn about a whole bunch of Software Engineering concepts both throughout my degree and in this class. Examples of which include: version control, branching, CI/CD, API's,

proper testing, and more. The concepts that I especially got out of this class were branching/pull requests and CI/CD. CI/CD pipelines are a concept of which I had heard of (I got asked about them in an interview once, didn't go well), but didn't really have any knowledge or experience in. Getting to learn about automatic testing/reviews and deployments was really insightful and I'm glad I got to get my hands dirty with that. I had made a couple branches for features in CS3300, but that was less of a structured thing and more of just me wanting to try it out (I had seen it as an option on the GitHub Windows GUI and was curious). Seeing how branches and pull requests work in an actual dev environment was really cool (minus the merge conflicts).

Concepts and the SDLC

Version Control and Branching are absolutely critical when following the SDLC. Testing and deployment are more complex when not done automatically, and it also becomes easier to forget to test or to miss critical errors before merging code to a production environment.

Proper testing is really important to the SDLC, not the least important reason being that it's literally one of the steps. Without testing applications, it's possible for bad code to get pushed into production and for applications to completely break, allow bad actors to access critical information, and more. To prevent these events, it's very important to test regularly and ensure that tests are actually covering everything that happens in applications.

Tools of Software Engineering Concepts

Needless to say, GitHub and Git have been monumental in assisting with the concepts of version control and branching, without these tools I'm not exactly sure how one would go about following these principles.

The tools we got to learn about for CI/CD were really neat. Having Render auto-deploy after passing tests was pretty neat, all we had to do was push commits and make pull requests. Having Pylint, the AI Code Review, and Dependabot were really great for our goal continuous integration,

telling us about any real issues when we pushed to dev branches, long before anyone would see these issues show up in production.

Proper testing goes hand in hand CI/CD, but still needs its own explanation. Having tests which automatically run when commits are made is great to see how well features are working and if anything has been broken by code changes. Having pytest implemented with our project has been great to test everything both locally and through GitHub to see how the code is working beyond the frontend user experience and a simple error log.

Favorite Part of the Class

My favorite part of the class was by far learning about CI/CD pipelines, why they're important and how they're used. Getting to implement these concepts into a large project and see them in my classmates' projects was really interesting and I plan to implement similar ideas into personal projects in the future.

Class Comments

I actually really liked the way the class was structured, the only thing I would change is that I found the implementation of DevEdu as a requirement for the homework assignments to be a bit unnecessary. It was really nice to have the option to host it on Docker however, especially as someone who did not enroll in the TAAP program this semester.