COS333 Final Project: AACCNJ Job Board

Tanushree Banerjee*, Chris Pan*, Haoyuan He*, Yutong Shen*
* denotes equal contribution

Project Evaluation

1. How good was your original planning?

Positive aspects:

- In terms of splitting teamwork throughout the tech-stack, we found that it was very helpful to have an expert on each part of the stack (database, front-end, back-end) as well as a fullstack jack-of-all trades. When we needed to implement a new feature that only touched one part of the tech-stack, it was very helpful to delineate that work to the expert, who typically finished it much quicker and more thoroughly than any of the other team members could have. However, large changes, and features that touch every part of the tech-stack could be handled by the designated full-stack developer. During debugging time as well, the full-stack developer had a more holistic view of the codebase, and could spot bugs that were the result of interactions, whereas the experts could very quickly pinpoint bugs located deep within one aspect of the codebase.

Negative aspects:

- The planning of implementing an additional feature didn't work as expected that we originally planned to complete all features before the Beta version. We encountered the problem of finding an unpaid API for searching by location feature of the jobboard and the Google map API we planned to incorporate turned out to be a paid service. Therefore, we needed to have additional discussions after the Beta version to settle on the implementation of additional features.
- We also didn't do well on the planning of the documentation part of the project. After the presentation was done, we found that the timeline was relatively tight to work on the four main deliverables of documentation.

Learning:

We learned from the project experience that it will be better to have a thorough planning next time that includes what exact features we are going to have and what tools we are going to use in order to avoid the expected circumstances of spending time on finding alternatives. In addition, we need to pay more attention to the timeline to assign the task to teammates in a more reasonable way. We also learned how to plan out the entire project as a team and present the timeline to our major client.

2. How did your milestones go?

Positive aspects: We tried our best to meet the major milestones of the course, including the Prototype, Alpha and Beta version deadlines, and we were successful in meeting the deadlines. This helped us stay on track and make significant progress during the semester and avoid stress closer to the end of the semester.

We meet every milestone in our MVP and most of the stretch goals.

Negative aspects: Times around the milestone deadlines were a bit stressful to handle, especially as we had to manage them alongside regular assignments for this class as well as our other classes.

Learning: Start planning and actually doing the work to meet the milestone deadlines themselves, rather than doing all the work to meet the milestone deadlines a week before the milestone deadline.

3. What was your experience with design, interfaces, languages, systems, testing, and so forth?

Positive aspects:

- Our team did a really great job learning new APIs. It's often not easy to learn a new API system and integrate it into our app, but we did it very seamlessly and well. We feel our organizational structure of putting each new feature into a separate directory paid off long term, since our main controller file jobboardapp.py didn't become cluttered with methods that weren't flask endpoints.
- It was a smooth transition from the course projects to the app, which we all really appreciated. We quickly built a skeleton and populated it with a very minimal table, and we gradually added new features and interactive components, and built up our skeleton. This process went very smoothly.

Negative aspects:

- We should have been more thorough on testing towards the beginning. I think for many of the deadlines we made the mistake of trying to push a feature right before the checkpoint meeting. We didn't have enough time to debug this, and ended up having failures in many parts of our codebase right before our beta meeting.
- Collectively our weakest skill was the Javascript embedded scripts in the front-end. We relied heavily on tutorials that we cited in the Readme.md file in our codebase. During debugging time, we were thankful that not too many bugs were a result of the Javascript, however, when there was a bug in Javascript, it took much more time to fix.
- We also wanted to copy the structure of the Assignments, which blinded us to the fact that since we were abstracting data types into classes anyways, we could have used SQL alchemy.

Learning:

- One big lesson we learned was related to the first negative aspect. It is okay to push new features to github, but it's best to keep a "stable version" of the codebase that is heavily tested. During project meetings, instead of hastily demonstrating a new version, stick with the stable version instead. We definitely could have used git branches for this.
- We also learned that we should think about codebase structure very early on. We did a rough codebase skeleton, so we decided to create database and template directories, however the organization within those directory started off very haphazardly. This resulted in lots of large code refactors later on. It would be much easier to start off with a better structured codebase.

4. What surprises, pleasant or otherwise, did you encounter on the way?

Positive aspects:

- We realized the Google API was very powerful for many login features, and also very easy to import. We anticipated that OAuth2 would be a major pain point to integrate with our website, but it was quite smooth actually.
- We also figured out how to leverage multiple zip-code search API's very quickly. Typically using independent APIs in one pipeline can lead to

Negative aspects:

- We found out that our elephantsql free plan was a bottleneck to our stress test. The database frequently produce "too many connections" error and we discussed about whether we should upgrade it to a paid plan. We even have to create a new database instance because the old database seemed to be banned by elephantsql.

Learning: Perform the stress testing and other crucial testing well in advance of the deadline since it is likely that these tests will crucially break the system, and that we would require more time to debug our system after these tests, so we should budget time for debugging after stress testing and other types of testing.

5. What choices did you make that worked out well or badly?

Positive aspects:

- Our choice of using Elephant SQL for the database end worked out well that the PostGre language is similar to the one used in assignments so we didn't spend too much time on learning the new language. The online database platform is also relatively stable for querying and data management.
- Our choice of using Flasks-wtf and Jinja worked out well that the backend could successfully and securely send data to the frontend to display the table content. The use of flask table template also worked well in terms of aesthetics.

Negative aspects:

- The design of the database schema didn't work well since the final schema was changed several times from our original planning, and our client expressed her opinions of changing some of the columns in the database table. Therefore, we had to continuously update the function calls on the database side.

Learning:

We learned that planning is really important in the entire project. A good planning with what tools and framework we are going to use during the implementation will save more time and avoid repeatedly changing the structure. In addition, close communication with the major client at the early stage is helpful for the design of the project as the client could have more sense on what the database schema is going to be.

6. What would you like to do if there were more time?

If we have more time, we want to extend the filter by location features to pin the jobs nearby on an interactive map on our job board. This feature will make our app more user friendly. We also want to enable auto-completion functionality in the form when the user types something in the input. This will help to avoid typos/misspelling in the input when the user submit a new job posting. In addition, we want to work on improving the visual layout of the job board to make it more attractive.

7. How would you do things differently next time?

Positive aspects:

- I really appreciate that we sketched out each page of our app early on, which ensured everybody was on the same page about what the app would look like and what functionalities we hoped to add during the project, which made later stages smoother and allowed ou application to look for cohesive

Negative aspects:

- I wish we had planned out which exact API's we would be using later on in the project so that we could structure our code in a way that we are able to integrate the external API or framework into our own

Learning:

- I think I would really try to plan out minute details such as where I can find the data or whether the API we need to implement our functionality exists or not.
- This would ensure that we know for sure a feature can be implemented, and know what costs must be paid to external API providers

8. Overall project

Positive aspects:

I think this project was great for learning how coding with people works in a real-time setting, unlike the zoom semester last year. We all learned about a core computer science skill - web development. I think generally this project helped me become a better programmer, and I learned how to structure a bigger codebase well, as well as design a code base from scratch.

Negative aspects:

I think we mismanaged time quite a bit at the beginning of the semester, and thus couldn’t spend much time at the end of the semester to further work on making out features work as smoothly as they could have towards the end of the semester. I wish we had worked a bit more early on in the semester to finish the core coding of the main features early on. I also wish we had figured out how to use bootstrap CSS rather than writing the CSS styling manually ourselves, since that would have lent itself to a more consistent front-end design.

Learning:

Distribute workload evenly throughout the semester, and don't leave things until the very end of the semester/reading period.