GrabAGrad

Project Evaluation

Original Planning

In reflecting upon our original plan for GrabAGrad, it is safe to say that in some areas our planning process set us up well for success. However, in other areas, we exhibited a lack of clear understanding of our team's goals as well as the feasibility of those goals.

Our planning process excelled in Section 6: Design of our Project Overview document. In this section, we detailed a segmented approach to build our application using the MVC framework. This framework helped us break our code down into relevant modules as we developed our application. Using Model, View, and Controller directories, we were able to split our code between sections relevant for retrieving information from the database, displaying content to the user of the application, and relaying information between the two, respectively. Although Flask itself often has a design that doesn't perfectly correlate with the MVC framework, we found we were able to still utilize the MVC framework to keep our code organized and easy to maintain. Especially towards the end of the project, having these directories and focusing on modularity from the very beginning of our development process made it much easier to create new features and update old features in the code than if we had lacked this structure. Furthermore, maintaining and adhering to this modularity in the beginning of the project was sometimes a bit challenging and felt unnecessary (like having just a few lines of code or one function entirely separate in its own Python file), but in the end it was very worthwhile.

Where our planning process struggled was primarily in considering the finer details of our database schema and what GrabAGrad would require, as well as coming up with an appropriate timeline for the project and dividing work between ourselves. Our database schema changed dramatically throughout the course of the project, and oftentimes a change to our database schema would require changes to many of the files elsewhere within our app. While the MVC structure provided useful modularity between files and their specific purposes, whenever we wanted GrabAGrad users to be able to see information about a new table or a new column that we updated within our database, we would have to make sweeping changes to our codebase. Regarding the timeline for our project, when planning as a team, we did not feel confident in our ability to predict challenges, roadblocks, and future development needs that would arise down the line. This lack of confidence was reflected by assigning ourselves bulk tasks that didn't consider the nuance required to go about developing these tasks. For example, during the week of 3/7, spring break, we planned to download PostgreSQL for remote testing, but this didn't really consider the fact that because we were going to be launching our app using Heroku, we wouldn't actually need to download PostgreSQL, as we would instead be using a cloud hosted service. We also had other issues in our timeline where during our weekly meeting with our advisor, we would receive suggestions for future work that differed from what we planned in the timeline. Over the course of several weeks, this made our timeline no longer useful or reflective of the work we planned to do on a weekly basis.

Milestones

Over the course of the past few months, we worked hard to make sure we were consistently on track to meet our weekly goals and to stay on track for our alpha, beta, and final presentations.

Our first milestone involved meeting with the graduate student that proposed the GrabAGrad idea and creating an intuitive design for the application, while also making a plan for gathering test data from existing graduate students at Princeton.

Our second milestone was completion of the MVP, which included a simple landing page with two buttons. In the MVP, if the user selected that they were an undergraduate, they were brought to a much more limited graduate search page, which used a search bar with a very specific input mechanism to query our database. If the user selected that they were a graduate student, they were brought to a simple form where they could input their information. Note that within this large milestone, many smaller milestones also existed, including completing the database schema and designing a basic UI.

Our third major milestone involved adding to the user experience by making the search functionality more dynamic (as the user typed, the shown graduate students would update), updating the database schema to store more information related to each graduate, and modifying the UI to highlight the app's functionality more effectively (by giving each graduate a card and utilizing pop-ups for information).

Our fourth milestone involved implementing the Explore and Favorite pages on the application as well as debugging the application to prepare for final presentations. Also, large-scale modifications of the UI were made in order to make the application as responsive as possible to changing viewport sizes.

The last milestone involved completing all of the documentation associated with the application.

Note: Between each milestone many smaller milestones were present (only the large milestones are listed) and robust user testing was performed in order to ensure that our application was on track to be best poised to serve our intended users. Also, we framed these milestones around our weekly advisor meetings, especially the alpha, beta, and final meetings. We frequently completed all of our weekly work before our advisor meetings and were able to present our current product at all meetings with minimal bugs and to the satisfaction of our advisor's grading criteria.

Some Surprises

When we first sent out a form to collect graduate student information that would populate our database, we were surprised to see how many graduate students submitted responses. We only needed a few students to submit information to initially populate our database, so we were ecstatic to see that over twenty students were interested. Seeing how many graduate students were willing to spend time filling out our interest form also gave us a confidence boost that our app would be received well by the graduate students on campus and that it had a real chance at becoming pretty popular among students after we deployed it.

Another surprise was how tough it was for us to make the pages on our app responsive to different browsers, window sizes, and devices. For instance, Google Chrome and Safari have slightly different standards for certain HTML elements that make making pages consistent across both browsers more challenging than expected. Furthermore, it was also hard to format features such as the Filter By side bar in small vs large window sizes. For instance, in a small window size, it was challenging to move this side bar to instead be a drop-down menu on the top of the page. This issue carried over to formatting our site for mobile phones versus computers. While we originally only designed our site with computer browsers in mind, we had to adapt some of our formatting to also work with mobile browsers as we progressed.

Some other surprises involved the amount of time needed to implement seemingly simple features. For instance, utilizing Cloudinary for our image storage was much more complicated to implement than expected. Furthermore, as mentioned in the Programmer's Guide Design Challenges section, the fact that Heroku doesn't store global variables was a surprise that forced us to rework our code to only rely on parameters and local variables. In addition, trying to require specific formatting for user input took much longer than expected, as it seemed like each time we would solve an edge case, we would also discover a new one.

Good and Bad Choices

Good Choices:

- ❖ Splitting up the workload so that we had one team member primarily working on the user interface, one team member primarily working on the backend, and one team member shifting back and forth between the two sides of the application. Splitting up the workload in this way enabled us to maximize the functionality of our final product through delegating weekly tasks and keeping team members accountable for their weekly deliverables.
- ❖ Working with a graduate student mentor for our application was helpful in the sense that she was essential in our collection of graduate student data necessary for testing. At first, we were hesitant about working with an advisor as we felt it would stifle our ability to freely make decisions, but in the long run we felt that her hands-off approach enabled us to design freely but also check in for her feedback.

Bad Choices:

- ❖ Deciding to start coding too quickly without developing a robust plan of action / design beforehand was a poor decision. Neglecting to properly prepare our plans before starting to code caused future problems as we didn't consider how to address prominent issues beforehand, so at certain points we had to rewrite previous code to accommodate new changes.
  - ○ **What we should've done differently -** We could've created a thorough set of documentation and plans before starting

- Not utilizing bootstrap enough for functionality and instead largely relying on custom CSS for the majority of the project was helpful in the sense that it made it easier for us to customize our application, but detrimental in the sense that we didn't take advantage of the built in functionality Bootstrap already specializes in, especially in regards to responsive design.
    - **What we should've done differently -** Before starting development of the user interface, it would've been beneficial to map out what aspects of the frontend were going to be built using Bootstrap and what aspects were going to be built with raw CSS. These plans could be created after weighing the pros and cons of using CSS or Bootstrap for each aspect of the interface.

If There Was More Time

One of the biggest stretch goals that we had in our original plan was to save undergraduate profiles along with the graduate profiles. This way, we could bolster the Explore page to become a machine-learning based recommendation page for undergraduates to find graduates. In this sense, the Explore page could resemble a dating app where undergraduate users could "swipe" through graduate students, favoriting the ones that stood out to them the most. In order to implement this feature successfully, we would have to invest a good amount of thought on how we could develop a recommendation algorithm. But most importantly, this would require many more graduate students to create profiles so that we would have enough data for the algorithm to be optimized and practical.

Furthermore, we also would like to implement an in-app messaging platform for undergraduates to reach out to graduates. This way, graduate students who would prefer to not have undergraduates emailing them would still be inclined to use the app. On top of that, we could restrict the number of undergraduates reaching out to individual graduates so that no graduate student becomes overwhelmed with undergraduate students reaching out to them.

Besides coding, we would also like to spend more time marketing our app. By potentially receiving support from the Career Center or student government, our app could become much more popular. After all, the more users we have is directly correlated to how valuable our app is to those same users. One of the main ideas that we have for marketing the app is to offer a rewards system where connections between graduates and undergraduate students are incentivized with something like coffee vouchers, for instance.

Personal Experiences

Henry

Working on GrabAGrad was one of the most fun yet grueling programming experiences I've had. Coming into the project, I had very minimal experience with full-stack development. In fact, I don't remember ever writing an interactive HTML file, let alone touching any database programming. This project proved to me that the best (and perhaps only) way to learn a new coding skill is to learn on the fly. By being thrust into such a large-scale project with limited previous experience, I found that the learning curve was steep and I was able to learn new techniques, libraries, and conventions much more quickly than I expected. For instance, even though having to program a Postgres database may have seemed daunting at first, I now feel very comfortable working with SQL. I also now feel very confident with Flask, Jinja, and AJAX. I spent a majority of my time trying to optimize the workflow between certain Flask endpoints rendering HTML with Jinja and other endpoints responding to user interaction to update the UI with AJAX. Although this led to a frustrating amount of debugging and hours spent on trying to fix minute features, I feel much more confident in myself as a programmer who will be less intimidated by full-stack development (and any other new topic, for that matter) in the future.

The project was also rewarding in that I had a blast working on a team with Theo and Brett all semester. I realized just how much more I enjoy working on a team as compared to working individually. The three of us complemented each other's strengths and weaknesses well which resulted in a very productive team dynamic. It was especially gratifying to witness all of our features come together week-by-week and how each of us would seamlessly build off the progress made by the other two. We had a great line of communication and set deadlines for ourselves that made working on the team easy, enjoyable, and successful.

Lastly, I realized that it is much easier to spend substantial amounts of time debugging, conducting manual tests, and all the other tedious aspects of programming when the overall project is exciting and interesting. Finding myself looking forward to solving little bugs when they popped up was a pleasant surprise. Overall, the software tools I have learned throughout this semester along with the teamwork skills I've taken away will set me up well for future projects.

Brett

I really enjoyed working on the GrabAGrad project throughout this past semester. I feel like I learned a ton, especially in regards to front-end software engineering, simply through running into problems and then having to rely on my own research or asking my peers for solutions to bugs in the code. Also, I felt that this project really built my teamwork skills as they relate to software engineering as I had to constantly make sure that the work that I was producing was in line with the goals of the group. Also I had to ensure I was in constant communication with the other members of my team to make sure we were all on the same page regarding goals and weekly assignments. In terms of design, I felt that our robust user testing enlightened me to new ways of thinking about web page design, especially the ways in which design could be optimized based on the target users of an application. By the same token, I found it interesting to see the minor differences in the user experience that occurred based on different devices used or different browsers. This taught me the importance of considering the specifications of different devices and browsers and the implications those specifications have on the design of the site. Further working day in and day out with HTML, CSS, and JavaScript made me a lot more comfortable utilizing these languages to create intentionally designed user interfaces. Especially, the debugging portion of this project really helped me to learn the deeper concepts within these coding languages which I believe in the long term will make me a better software engineer. Lastly, the testing portion of this project has made me realize how important proper user testing is in making an application as successful as possible. Especially hearing feedback from peers of mine that are not at all familiar with software development really made me realize how valuable all types of users are in the testing process. Overall, I really enjoyed working with my team this past semester and am genuinely proud of all the work we created.

Theo

I entered COS 333 this past January excited to work on a project that I was passionate about. But without a team or a firm idea of a problem I wanted to solve, this excitement was largely undirected. I knew that many TigerApps came out of the course, and I liked the idea of contributing to a project that could have a real impact on the campus community, but I wasn't sure where to start. Fortunately, Oladoyin Phillips, a Resident Graduate Student in Whitman College, proposed the GrabAGrad idea and Henry, Brett and I jumped on the opportunity to work with this idea.

Right off the bat, I knew GrabAGrad was going to be a project unlike anything I had ever worked on before. Although I have experienced software engineering environments through two summer internships, the goals, tools and design decisions that were needed for GrabAGrad were very different from my prior experiences. I found that my internship experience prepared me well for learning how to use programming tools that I lacked experience with, and feeling comfortable searching up solutions to the problems I encountered in the process of creating GrabAGrad. I had never used JavaScript before, and had only ever composed very basic HTML files that lacked enormously useful front-end development tools like Jinja, AJAX, and jQuery. COS 333 lectures and assignments provided a nice introduction to these tools, but I felt like I was able to truly experiment with the tools and flex my understanding of them via GrabAGrad. When the GrabAGrad project needed an image management solution to enable graduates to upload profile photos, I was worried that this task would be beyond my programming abilities, and possibly a whole project's worth of work in and of itself. However, Anat, our project advisor, suggested we look into using Cloudinary to solve this problem, and after reading extensive documentation I discovered how to do this using a JavaScript widget. While this was, retrospectively, a small and rather simple task, working on this specific feature guided me through an important learning process about how to use other people's software within front-end development and provided me with knowledge and understanding that I will lean on as I go forward in my computer science studies and my future software engineering pursuits.

Working with new tools and developing additional skills throughout the GrabAGrad development process was extremely rewarding and makes me feel more confident in my ability to work through technical problems, and moreover, I am thrilled to have had the opportunity to work alongside a great team for this project.

Acknowledgements

We would like to extend a big thank you to Professor Dondero and Anat Kleiman for the help and guidance throughout the development of this project.

We would also like to thank Oladoyin Phillips for suggesting the GrabAGrad project to the class, for enabling us to bring her idea to life, and for spreading our request for graduate student data among her peers to help us populate GrabAGrad with the graduates.