TigerThrift 

Project Evaluation

Our Original Plan

Even though creating a semester schedule was a daunting task, overall, we did a good job of generally keeping consistent with what we had originally planned in the 'Anticipated Project Schedule' section of the Project Overview, which we wrote out at the beginning of the project. However, we did encounter problems that made us fall behind in some aspects of the planning. For example, we had planned to have the CAS authentication fully implemented for our first prototype demonstration (week of November 1st). Unfortunately, we had problems understanding the documentation behind how to use the CAS authentication, as we had yet to learn about it in lecture; thus, we had to push back the CAS implementation to be a goal we achieved by our Alpha version demonstration. Though we had to push this back, we made sure that we were still on track with all other parts of the project, and we were able to make sure that we did not fall behind with regards to our end goals.

In terms of the actual application we created, we believe that our original planning went fairly well. As we described in the last section of the Programmer's Guide, our original planning on the database went really well because we had spent so much time at the beginning to plan the main features we wanted and we designed the database around these features. By doing so, we had very few changes made to the database throughout the project.

We also believe that our original planning of determining the pages we need on the application went very well since we did not make major changes to the purposes of the pages. We also made a low-fidelity design of the pages using pencil and paper, and later on Figma, which we believe helped streamline the process of designing our application, as we did not make major design changes. However, we do believe that we could have done a better job with the planning of minor details of the pages. Implementing these minor details into our pages caused problems because the pages had not been structured to contain these minor details. Though we were able to work with these problems and implement the minor details, we believe that it would have been much more efficient if we had planned these details as well during our designing phases.

Milestones

We believe that we made a lot of progress at each of our milestones. As discussed above, we were able to make sure that we kept with the original project scheduling that we made at the very beginning of the project. We did our best to make sure that we implemented all the features we listed in our project overview for our prototype, alpha version, and beta version, and when we absolutely could not, we implemented other features to make sure that we were not falling behind. When comparing each of our milestones, we felt that we were able to add more features (outside of the ones that we had originally listed), as we moved forward with the project because we

gained more momentum and figured out what working methods worked best for us as a team (for example, creating a To Do doc to list out our next tasks – this will be discussed in a later section of this document).

Unfortunately, we were unable to reach the stretch goals that we listed in our document provided the bugs as well as new features that we realized we needed to implement on our application. However, we do not believe that this suggests that we did not make a lot of progress, as we implemented additional features that we had not originally considered (such as the tabs under 'My Account' that helps users keep track of the different statuses) and we also made sure that the features that we implemented were working fully without any problems.

Our Personal Experiences

Iroha:
TigerThrift is definitely one of the most exciting projects that I have worked on during my time at Princeton. Prior to this project, I had only ever made simple websites mostly structured around HTML and CSS, and websites where the majority of the information was set – it was not dynamic. Furthermore, I had limited experiences with systems and testing; my knowledge was mostly from what I learned in previous Princeton COS courses. For me, the main challenge in this project was getting more experience with PostgreSQL, Flask, Jinjia, and jQuery, as I had no prior experience with these applications. However, I appreciated this project in that I was able to use what I learned in class and actually implement it. Furthermore, it was a unique learning experience because we had to really think about how we used these applications, given that this was not like an assignment, where oftentimes the uses and the goals of the implementation are defined by the assignment specifications. Not only did we have to implement it so that it worked, but we also had to make sure that we used the correct application to achieve the goal that we had in mind. For the design aspect, I did have prior UI / UX design experience, but this project was unique in that I was on both the designing and developing side of the website. This aspect of being part of both sides was very valuable to me because it really reinforced the difficulties of trying to implement a design that we wanted. Throughout the project, we would want to make design changes with the way things were formatted on our page; sometimes these changes were simple to implement into the current code structure, other times, we had to completely rearrange the structure of our code to be able to make the small design choices. Thus, I definitely learned the difficulties in making major updates later on in the process, though, in a sense, this is inevitable given that as we make more progress, the more detailed user feedback we are able to receive. In addition, this taught me the importance of starting off with a very clean structure within our code. Overall, I really enjoyed this project; though I spent perpetual hours on different online websites trying to figure out how to use the applications that we used – with an exceptional number of hours on Bootstrap's

documentation trying to understand what it was actually doing in order to change Bootstrap's pre-set designs – it was definitely worth it. I also realized that I really enjoy working on front end aspects. This project was full of challenges and opportunities to learn from; one that I would not have been able to gain in a different Princeton COS course.

Katie:
I enjoyed working on this project from the beginning and grew to love it even more by the end of semester. I had never created a project of this scale before, so I didn't know exactly what to expect, and I'm very pleasantly surprised and proud of what we created. I had previous experience with Javascript and a bit of web programming, but the majority of the knowledge needed to complete this project was learned in lectures and assignments. I found myself taking over the majority of the Javascript/JQuery programming, as well as some database programming and other processing tier tasks. I grew familiar with PostgreSQL, and felt more and more comfortable with HTML and Javascript. I dabbled a bit in styling with bootstrap, but this was not a strength of mine.

Seeing the system break down with small bugs was a little bit frustrating, but it was very gratifying to squash bugs. I enjoyed user testing, and was impressed with the valuable feedback, good and bad, that I received. I was very grateful for the TODO list document we had that always kept us on task, allowed us to prioritize tasks and not forget anything. I found I relied on it more and more to figure out what to work on next, and it made us very productive. I learned a lot from this project, and had a great time and a great team.

Katelyn:
I thoroughly enjoyed working on TigerThrift. From the entrepreneurial idea that this technical project stemmed from to the meticulous detail we dedicated to designing the relational database, modules, querying functions, background processes, and user interface. After gaining exposure to front-end, back-end, and database programming, I found that my inclination is to work on back-end design. I enjoy problem-solving how to best modularize code and optimize the amount of operations we are executing, as well as minimizing the number of calls we make across modules to strive towards a weakly-coupled design. I also enjoyed learning about the apscheduler background processes since that was what I had never had exposure to before with regards to the languages and technologies I had worked with in the past. I discovered that I am a very detail-oriented person, so I prefer to test all scenarios fully before moving onto the next feature implementation. I also made sure to document all possible boundary/corner/error cases I could think of to make sure we addressed it when we formally tested our project towards the end of the semester. I surprisingly enjoyed database programming a lot. I've worked extensively with SQL databases, but did not

enjoy it as much as I did this time around! I think it was because I got to see the database queries actualized into a tangible product – a webapp.

What was frequently a challenge for me was front-end design. Despite having some exposure to user interface design, I found that since I tend to be a stickler for nitty-gritty details, I easily went down a rabbit hole in terms of design, and found myself googling css styling a lot. I think this would be an area for me to improve on going forward especially if I find myself in the web-dev environment again, but I did learn several valuable styling techniques with CSS and Bootstrap. On the contrary, I enjoyed learning some jQuery and JavaScript to manipulate the HTML logic. When I reflect on my experience in COS333 this semester, I'm very grateful for the abundant skills I acquired across a broad breadth of technical topics, and I am very optimistic about the value these skills will provide me in my future career!

Surprises

We were pleasantly surprised that the planning of our database schema proved to be very effective and flexible and to fit our needs. 3 relational tables worked perfectly. When we wanted to add additional fields for item information input, it was easy to simply add another column to the database. Figuring out early how to generate the self-incrementing itemid helped us easily identify items throughout the entire schema and project.

We were less pleasantly surprised with browser errors that only Katelyn was getting when trying to locally test CAS authentication. While it was working just fine for Katie and Iroha, there was an unresolved bug which prevented Katelyn from being able to test CAS authentication locally; this took up quite a bit of time.

We were also surprised by the need for a scheduled background process for reminder emails, which was not something we had considered in the planning process. This required considerable time spent researching the best way to create that.

We were also surprised by the speed at which the app came together, and some things took shorter than expected. Some other tasks, like styling, creating a footer, were much more challenging than expected and that we ended up not including in our final version because of the adverse effect these new aspects had on other functioning features of the app.

Choices that Worked Out and Didn't

There were some choices that we made that worked out well and some that did not work as well. Below are some of the choices that we made; the choices are not all about the actual application that we created, but also include choices we made to work as a team.

Well:

- **Scheduling weekly meetings**
  We decided to schedule team meetings twice a week to ensure that we were all on the same page and were all making progress. We found this to be very successful because this made sure that we were making progress between every meeting, and we were able to ask each other questions if we were unable to figure something out. As helpful as texting / emailing is, we definitely found in-person communication to be much more efficient.
- **To Do's Google document**
  We also created a '[TigerThrift] To Do's' Google document where we compiled the next steps we needed to complete. Rather than compiling big ideas and tasks that we needed to complete, we wrote down very specific tasks on this To Do document. We believe that writing down very specific tasks worked well because it helped break down the progress that we needed to make into very feasible tasks. Furthermore, by making it a Google doc, we were able to constantly update the tasks we needed to complete, as well as deleting / crossing out tasks that we completed. This was especially helpful when we were working on the project outside of our weekly meetings.
- **Simple, Minimalistic Design**
  Though we considered more colorful designs, we ultimately decided upon our very simple, minimalistic design. We believe that this design choice worked really well because we received a lot of feedback about this, with users very happy about the clean look of our website.

Bad:

- **Not Thinking about Lower Level Features Required on Each Page**
  Though we thought a lot about the main actions and features for a given page, we had not spent much time thinking about the lower level features, such as what types of buttons we would want. This later became a problem because the way we had structured the html files were not very flexible to new features, causing the code to be error prone. Though we ultimately re-structured the code to be able to make the features implemented, we feel that it would have been easier and less time consuming had we thought about the baseline code structure thinking about lower level features first.

- **Relying on Git Recursive Auto-Merge Strategy When Pushing / Pulling Changes**
  At the beginning of the semester our updates were a bit slower than at the end of the semester since we experienced a natural final sprint towards the end. When we were all making our individual edits at the same time and not pushing and pulling to/from Git frequently, this led to lots of sneaky bugs that often went undiscovered for a long time since we had just merged the commits using the recommended Git auto-merge strategy. This led to a bit of last

minute stress for our team fixing bugs. In retrospect, we could have avoided several of these bugs by making sure to push and pull our code often so that we were all working with each other's most up to date version of the code and not relying on the Git auto-merge strategy that was not always the most accurate.

If We Had More Time

There are various features that we would like to implement if we had more time.

Smaller scale next steps:

1. **Open to Princeton Graduate students**
   Not only did we want to make this platform a smaller subset of the Princeton community to ensure that the users felt safe with making transactions on the platform, we are also using Tigerbook API to get user information, which does not include graduate students. Therefore, if we had time, it would be great if we could navigate the other Princeton APIs available to see if we can get necessary information about graduate students; this would allow us to expand our platform to the graduate student community as well.
2. **Phone Notifications**
   Another next step would be to add automatic phone notifications. Our application currently sends email notifications using SendGrid, and given that undergraduate students check their emails every day (given how most communication related to school is done through email), we felt that this was a sufficient communication method to start off with. However, we believe that phone notifications may potentially be more effective than email notifications as they might be overwhelmed by the email spam, and thus, we would like to implement this.

Bigger next steps:

1. **Cart + Payment features**
   Given that our application focuses on the main actions of shopping and selling clothing items, it seems most fitting to have a cart and provide the option for users to pay on the platform. The car feature would allow the user to browse various items before making a final reservation. The payment feature would allow the user to choose different payment options. In our user testing, we received feedback that some people appreciated how the payment option was not limited to online transactions. Therefore, we hope to leave the non-online payment feature as well, but also integrate online payment for those who would prefer it, and we would consider using Stripe or Venmo API for this purpose.
2. **iOS/Android App**
   Given that our target is college students, we believe that it would be helpful to make our website into an app. Our website is mobile friendly, but making our

website into an app would allow us to integrate notification features directly through the app. In addition, it would better the user experience for a user who primarily uses our website on their phone.

3. **Social Media Aspect**
   If we make an app, we believe that it would be most fitting to add social media aspects to our platform. This would be features like being able to comment on products, add ratings to sellers, and adding a 'For You' page, or a suggestion page. These sorts of features would allow our users to engage and interact with other users of our application, making it more exciting to use.
4. **Admin interface (for moderating + reporting items)**
   We would also like to create an admin interface for our application. Though we did write up a programmer's guide to help the next generation of people who maintain this website, we felt that having an admin interface would make the maintenance process a lot simpler. Creating such an interface would allow people without coding / backend experience to help moderate and report our items.

What We Would Do Differently Next TIme

There are definitely things that we would want to do differently next time we work on a project similar to this one.

First, we would want to make sure we start off with a more detailed structure of the baseline elements that will be used in a given element. We found that as we added more id tags and styles to the elements in our code, the code became longer and more complicated, making it error prone to small changes. Therefore, it would be more time efficient and would reduce avoidable errors by making a clear outline of the necessary elements in a given page before implementing design features and adding smaller details to the layout. This would also mean solidifying exactly which technologies/languages/web-frameworks we are using and understanding their limitations and capabilities.

We would also want to gather more user feedback and do more user testing beforehand. Though we did receive user feedback throughout this semester, the majority of the user testing happened at the end after we had created our beta version. Though we needed the beta version to have a more detailed and presentable user interface, we feel that it may have been better to get user feedback throughout the process, starting with the prototype version, to get feedback about naming conventions and baseline features. This would be helpful so that we can receive feedback on the main features early on.