TigerSnatch: Project Evaluation

Original Planning

We unanimously agree that our initial planning process was critical in streamlining our workflow early on and important to our app's overall success. In the first weeks of our project, we spent many hours writing our Project Overview document together, making sure to thoroughly discuss our MVP, stretch goals, database design, and visual layout in particular. Indeed, we outlined in detail our MVP and stretch features and ended up accomplishing all the ones we wanted to pursue (and more that we did not even write down). We also sketched up mockups of key site pages (Dashboard, Course, Landing) in Figma and put these sketches in our Project Overview. Establishing our site design, such as the color scheme, content layout, and widgets, early on greatly simplified the coding of our interface later on. As per our adviser's suggestion, we also took quite some time planning our database schema together, considering factors like efficient data lookups and reducing data redundancy. Our final schema is almost the same as the original, with a few expansions like new collections and fields. Moreover, we did our best to follow the weekly timeline we outlined in the Project Overview and, thanks to our early planning, actually went ahead of our timeline by at least one week.

One component that we wish we had planned earlier on was our previous version of the Trades feature, called "Mutual Section Swap (MSS)". Although we had written down the general idea of MSS in our Project Overview, since it was a stretch goal, we didn't think it was necessary to discuss its implementation until we had fully completed the MVP. When we finished our MVP mid-late semester and finally came around to brainstorming the implementation of MSS, we found that MSS would be much more challenging to implement than we had anticipated. In particular, we did not design our original database schema to accommodate MSS, so integrating MSS would require adding an entirely new collection to store graph objects as well as many new fields to our existing collections. Although this oversight in planning ahead for MSS was frustrating at the time, we ended up adapting MSS into the more technically-feasible Trades feature, for which we have received positive feedback from our pilot users.

Finally, an essential part of our planning process was discussing how to best collaborate as a team. Unfortunately, we were unable to meet in person this semester, so working on the project concurrently was challenging. Still, from the first week, we agreed to call twice per week (and more near the end of the project). For each meeting, we would write out a quick agenda -- whether it was talking about the next milestones, divvying up tasks for the week, or preparing a demo and questions for tomorrow's adviser meeting, we were conscientious of each other's busy schedules and the limited time we could call each time, so we planned ahead to ensure that we made valuable use of our collaborative calls.

Milestones

We are proud of the progress that we achieved for each milestone. For the prototype, alpha, and beta versions, we finished our milestone features one week ahead of the deadline, so that we would have time to patch any bugs that went unnoticed and improve the user interface before our demo. After each milestone, we also made sure to get user feedback. For each version, we interviewed a different group of users, so every user came in with fresh eyes to the product. For our prototype, we simply showed them our mockups and explained the idea of TigerSnatch; for alpha and beta, we gave them a set of typical tasks to complete while we took notes on how they interacted with our page. In each iteration, we received both compliments and useful suggestions for improvement -- but it was overwhelmingly clear that for the beta session, the users had much less to say about refining our features. One embarrassing moment was during our alpha session: this is the first time we had real users test out our site, but due to a flawed database change we made right before the session, the TigerSnatch course search resulted in an error page for certain courses -- from this small mishap, we learned that it is important to run quick tests on your site before piloting it with users and never deploy large-scale backend changes without thoroughly testing it!

Our prototype, alpha, and beta meetings with our adviser were also quite successful. We meticulously tested our app before meetings to ensure that the demo would run smoothly during each meeting. One concern that was brought up following our alpha demo was that our ending product would be too “lightweight”. At that point, TigerSnatch only consisted of our main “Subscriptions” feature along with other minor UI add-ons. Implementing the Admin panel was on our schedule as the next heavy-weight feature, but we were planning to drop another heavy-weight stretch feature, Mutual Section Swap, due to its infeasibility (as discussed previously). We took this feedback very seriously and spent all weekend brainstorming a new heavy-weight stretch goal and more UI extensions to increase our app’s value. As a result, between the alpha and beta builds, we incorporated many new features that are now central to our app’s usage, including the entire Trades feature, Tutorial page, Activity page, Admin page, Quick Links, better mobile friendliness, more tooltips and status indicators, etc. The feedback we received for our milestones were crucial in advancing TigerSnatch to its current state.

What was your experience with design, interfaces, languages, systems, testing, etc.?

Shannon: Developing TigerSnatch with Byron and Nick was certainly one of the most valuable learning experiences for me as a COS student. In terms of languages, I had written short programs in Python and created small websites in HTML, CSS, and JavaScript, but never had I applied these languages to a large-scale project. A main challenge was teaching myself how to use various libraries and frameworks with these languages, especially jQuery with JavaScript. For the subscription and course search features, I had to work extensively in JavaScript to control the switch on and off functionality and render the course pages. When I first started working on these features, no one in the team knew what JavaScript code to write to do this dynamic functionality -- at this point, we hadn’t even created our app.js file yet. Although it was exhausting having to spend hours reading jQuery documentation and StackOverflow posts to

figure out the implementation, it was definitely worthwhile: I learned how to write advanced listener methods, make jQuery post requests to interact with Flask endpoints, control props and attributes of HTML elements, etc -- all of which I realized are essential web-based programming practices. In terms of database design, I had used MongoDB a little bit previously, but I had never worked with a database with hundreds of data entries that would frequently be read from and written to. The large load and frequent operations on our database pushed me to think critically about optimizing database design. For UI design, given my previous minimal UI/UX experience, I was accustomed to glossing over, what I thought were, minor front-end details. Thanks to user feedback sessions, I learned that even the smallest changes in the interface -- from adding an info tooltip to explain a feature, switching the order of columns in a table, to underlining a hyperlink -- can significantly boost user experience, especially for first-time users of our app. Finally, feature testing was definitely challenging. I realized early on that catching errors and logging messages, which I was previously not used to doing, are critical for not only me, but also my teammates, to detect issues. This project forced me to think outside of the box about stressful or corner inputs that a user could enter and helped me become more cognizant of developing safeguards against such unusual inputs. Overall, I appreciate the many challenges, experiences, and learnings I have gained through developing TigerSnatch.

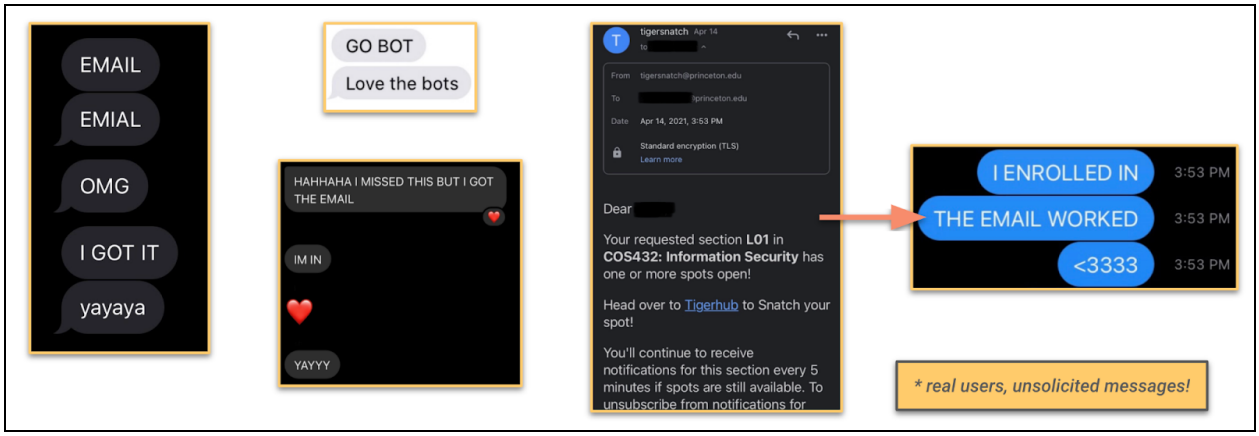
Nick: TigerSnatch is by far the most rewarding project I've contributed to during my time at Princeton. Prior to working on TigerSnatch - and more generally, taking COS 333 - I didn't have solid experience with technologies like Flask, Jinja, jQuery, Bootstrap, and Heroku; and all my previous work with Python during past summer internships had nothing to do with building backends, and wasn't using good coding practices. Building TigerSnatch with Shannon and Byron significantly improved my personal coding skills, both in terms of raw ability to apply technologies and frameworks in terms of applying code paradigms. I learned to write efficient modular methods, especially on the backend, that interact with a database and process data in ways that are robust against runtime errors and protect the database against corruption. I also learned to take advantage of Python's Exception handling functionality to raise descriptive error messages and catch them at a certain level in the code; Shannon and Byron were also good about throwing describe Exceptions whenever possible, which made future code that relied on previous methods *much* easier to debug and develop. I learned so much more while working on the backend, but for the sake of space (and the grader), I'll move on to the processing tier. The first thing that comes to mind here is Jinja templates. Prior to starting the TigerSnatch processing tier, I was puzzled as to how we could directly pass data from the backend to the frontend HTML pages - I thought we'd have to use JavaScript for that task, and I wasn't too excited given that JavaScript is a notoriously annoying language to work with. But learning to use Jinja came to be *extremely useful*, mainly because it integrates perfectly with Flask and allows us to pass entire objects (rather than just plain strings or numbers) to its templates. More generally, though, I learned how flexible Flask is in providing all the needed functionality - both for page generation and for transferring data to and from the backend - for a web app's functionality. Given how popular Flask is, I think that learning to better apply its features is one of the most useful takeaways from the project. Finally, in terms of the frontend, I especially

learned to focus on the fine details that improve user experience beyond default Bootstrap; e.g. adding auto scrolling for mobile devices, hiding certain elements to optimize small-screen real-estate, adding subtle loading indicators like using "wait" cursors and blurring parts of the window, utilizing tooltips, and overall designing a minimal UI that emphasizes features without clutter. I learned to take user feedback *seriously* and not react negatively to criticism; without feedback, our site would look and function quite differently than it does now (for example, Subscriptions would still use ordered waitlists!). Developing the website also taught me a ton about jQuery - a technology with which I'd had zero previous experience - and how it can be used to implement advanced listeners and dynamically control every element on the page. Like with Flask, jQuery's popularity makes it extremely worthwhile to learn! TigerSnatch's functionality and intuitiveness benefitted massively from our use of jQuery. Overall, I genuinely enjoyed the entire process of contributing to the development of TigerSnatch, even during difficult times like when we had to ditch Mutual Section Swap and build Trades from the ground up. The project challenged me intellectually and technically, encouraging me to think in creative ways and making me a better programmer, both technically and non-technically. If given the opportunity, I'd definitely do it again with this team!

Byron: TigerSnatch is definitely a highlight amid this special semester. I am very glad to have the opportunity to work with Shannon and Nick to create a product that might benefit the Princeton community by applying knowledge learned from the COS 333 class. Like Shannon, I also have had some experiences building small projects in Flask and HTML, but never tackled something as large as TigerSnatch, so witnessing how the product slowly came together is a very rewarding experience. There are three challenges that I think made my experience worthwhile: MongoDB, Jquery, and Bootstrap. Before starting the project, I also did not know how to use MongoDB. MongoDB was slightly difficult to learn at first, because its syntax is very different from the SQL we learned in class, and it also requires more steps to set up. However, after Nick and Shannon gave me a quick tutorial, I was very happy to after mastering the basics of MongoDB due to its speed and flexibility, and its pythonic syntax. it definitely provided substantial benefits to the development process of TigerSnatch. With a few lines of Python code, we were able to perform all kinds of database queries and updates that suited our needs. Using JQuery was also something new for me. Before COS 333, although I had some knowledge of the Javascript language, I mostly built applications using React, which is not ordinary Javascript language. The TigerSnatch team actually planned on using React at first, but I am glad now that we did not pursue that route, because Jquery was a lot more light-weight, and serves the purpose of our application well. Finally, I think learning Bootstrap was probably one of the most challenging parts of the TigerSnatch experience. Bootstrap is a giant library, and some of its classes have a “hierarchy.” In other words, if you wrap a class inside an element with another class, the application sometimes produces undesirable effects. However, Bootstrap was a very important part of our application, and it is one of the reasons why our UI received positive feedback from most of our users. Therefore, it was worth spending time browsing the documentation. Although there were many more challenges along the way, I am very satisfied with the experience of developing TigerSnatch with Shannon and Nick. Through this project, I

became a better software developer, problem solver, and critical thinker. I look forward to opportunities like this in the future!

Surprises Encountered (pleasant and otherwise)
During the development of TigerSnatch, there were a couple notable surprises we encountered. Perhaps the most pleasant surprise occurred on April 14 during the junior course enrollment period, when we invited a few Princeton students to use TigerSnatch Subscriptions to nab spots that might open in COS432 (Information Security). Keep in mind that this was our first real-world test with live and constantly-changing course data, so we were expecting various bugs (we warned our pilot users not to fully rely on TigerSnatch but also check enrollments themselves regularly that day) and for the email notification algorithm to fail. In reality, the test was 100% successful, totally not what we expected! Our pilot users received an email from TigerSnatch right after seats opened in COS432, 20 minutes before the official announcement from the department head, and grabbed a slot in the highly coveted course before non-TigerSnatch users did. See below for screenshots of happy reactions from those users (we displayed this in our project presentation):



Transitioning to the other end of the spectrum, we encountered an unpleasant surprise when we began planning the Trades system (formerly named Mutual Section Swap) after demonstrating the alpha build. We hadn't realized the number of challenges - both technical and non-technical - as well as design issues that stood in the way of implementing the system. For instance, we had to re-evaluate the paradigm of *automatically* matching users and notifying both of them using the TigerSnatch official email address; we ultimately ended up stashing this design in favor of a model that requires users to find Trades and reach our themselves. This threw us off a bit, since we didn't foresee having to re-envision a significant feature from the ground up. See Design Problem 3 in the Programmer's Guide for more details about the shift from Mutual Section Swap to Trades.

We wrap up with one more pleasant surprise - during our beta feedback session, we found that users were able to (very) quickly figure out how to use TigerSnatch, especially Trades, a feature

in which we'd spent a lot of time making the UI intuitive but were still doubtful as to its usability by real (non-developer) people. Beta users were able to fully and clearly explain how Trades worked and to confidently execute a correct series of actions when prompted to perform a series of Trades and Subscriptions tasks (see Tasks 7-9 in the Product Evaluation). Generally, users praised the TigerSnatch UI, much to our surprise given how many UI changes we'd made since the last feedback session during the process of implementing the Trades system and improving the Subscriptions feature.

What choices worked well

1. **Using a switch for Subscriptions**, which is intuitive for users to interact with and easily visualizes the “on” and “off” nature of receiving email notifications.
2. **Including Tutorial & tooltips** to guide inexperienced users. Tooltips were not difficult to implement but significantly improved user experience as they were navigating new features on our app, according to our beta users. Automatically showing the Tutorial page to first-time users sped up user onboarding and having it accessible in the navigation bar allowed users to reference it again.
3. **Including an Admin panel**, which enables admins to maintain the website from the front end. Before creating the Admin panel, we needed to perform tasks such as clearing Subscriptions and updating our database to reflect the current term from the terminal. Since we are planning to hand off TigerSnatch to USG, an admin panel would be very helpful for site maintainers from USG TigerApps team.
4. **Starting the UI early**, and building upon it incrementally throughout the semester. As mentioned in our planning process, we sketched up mockups in the first week of our project, and closely followed these mockups when coding our interface. We wanted to design our UI together early on so we wouldn’t have design conflicts when we started coding it up. Especially with responsiveness, we made sure that all added features were usable on a mobile device as we implemented them, as opposed to optimizing them all at the end and potentially having to make substantial changes.
5. **Assigning very specific tasks to each team member**, which allowed for efficient and ahead-of-time completion of all features. Ahead of each week’s adviser meeting, we would have a call to discuss the next week’s objectives and assign specific tasks to each member. Although our tasks were often related to a single feature or process, we tried to divide up the tasks so that each user could work independently. Our efficient delegation of tasks provided us ample time to review each other’s work, find bugs, improve UI, and perform testing.

What choices worked badly

1. **Initial choice to make waitlists ordered**. Before our alpha, we implemented subscriptions as an ordered waitlist, so only one user will get emailed at a time, in the order that they subscribe to a section. We initially thought ordered waitlists would be more “fair” since it rewards students who subscribe earlier and thought emailing all subscribers at once would be too competitive. However, during our initial feedback sessions, users

expressed skepticism about the fairness of ordered waitlists and that we'd be artificially imposing an enrollment ordering. Moreover, by spacing out when emails would be sent, a student who is continually checking the Registrar's page would probably get the enrollment spot before users at the end of TigerSnatch waitlists. In response to this feedback, we decided to make waitlists unordered, and changed the name "waitlist" to Subscription to avoid ambiguity about ordering and distance ourselves from official Registrar waitlists. The new unordered Subscriptions feature was well received by users.

2. **Not initially establishing standards for code organization and method naming**, with `database.py` and `app.js` in particular. Because of this, we sometimes wasted effort to reimplement methods we didn't know another teammate had already written or spent time trying to find certain methods. At the end of the project, we had to spend a lot of time re-organizing functions into categories and factoring out helper functions into files.

If We Had More Time

If we had more time to develop TigerSnatch, we would definitely try to expand the Subscriptions service by allowing integration with official course/departmental waitlists. This would alleviate the need for course instructors and department heads to maintain Google Form and/or Sheets waitlists. As part of this feature expansion, we would likely create a special type of account for instructors that allows them to manage the waitlist for a specific course with features such as "Admit ___ Students from Waitlist" or "Send Announcement to All Students on Waitlist." As part of this, we would re-implement TigerSnatch's original ordered waitlist system, which seems to be the preferred system for official course waitlists (this differs from Subscriptions in that users who enroll in a waitlist earlier are notified earlier, i.e. waitlists are queues rather than sets). In general, this feature would essentially standardize the paradigm of official course waitlists, making it easier on students wanting to enroll in courses with waitlists, and allowing TigerSnatch to become a central platform for getting into full courses.

Additionally, if we got funding, we would definitely implement text notifications, since users are more likely to be active on SMS/instant messaging rather than email. This would certainly increase user satisfaction. Finally, we might want to implement in-app communication to replace the current need for Trades users to use Gmail to contact their matches. Although this would likely be a complicated feature, since we'd have to design a push notifications system and a messaging platform, it would probably be useful because it would enable users to perform Trades - including meeting their match and coordinating a time - all in one place.

What We'd Do Next Time

Next time we work on a project, we would change several things. First, we would adamantly seek user feedback at regular intervals - before, during, *and* after implementation. Although we did host feedback sessions after our prototype, alpha, and beta milestones, looking back now, it would have been ideal to host a couple more sessions in between those checkpoints. In our opinion, user satisfaction is the most important metric by which to measure the success of an application such as TigerSnatch, so getting more feedback from real users is an extremely

important element of the app building process. Additionally, especially for more-advanced features like Trades, we would want to create technical and UI outlines earlier on, mainly to better understand the feasibility and constraints of those features. As talked about earlier, we didn't realize the immense difficulty of building a fully-automated Trades system until fairly late; we'd definitely want to not make the same mistake. Furthermore, as a team, we would have liked to establish standards for our source code - especially in terms of method naming and organization - at the beginning of our project, in order to streamline our code collaboration. Finally, we would definitely seek to coordinate more-frequent discussions with our adviser to get feedback about future plans; Sata's advice was extremely helpful in improving the usability and clarity of the TigerSnatch UI, and we would certainly do everything possible to speak with him more.

Acknowledgements
We would like to extend our sincerest gratitude to our adviser Sata Sengupta and instructor Professor Robert Dondero for their guidance throughout this project. This class has been an incredibly valuable learning experience for all of us, and we look forward to carrying on TigerSnatch throughout the remainder of our time at Princeton.