A New Approach to Picking Our Courses

Princeton Course Planner

Vinay Ramesh
under the guidance of Dr. Robert Dondero

Motivation

The main reason why Princeton Course Planner was created is to consolidate the functionalities of the TigerApps into a mobile application, a one-stop-shop of sorts for Princeton students who are planning out their course schedules each semester.

The status quo of Princeton University sponsored applications that deal with student course planning are listed on the website www.tigerapps.org. Namely, students can choose between Princeton Courses, ReCal, and Princeton Pounce.

- Princeton Courses
    - Browse through courses in the Princeton Registrar, by placing filters onto a query string. Additionally, one can see class reviews and ratings through the platform, allowing a comprehensive course-browsing experience.
- ReCal
    - Search through classes and position classes and precepts onto a visual schedule. Students can experiment with different schedules and put multiple classes into a schedule to scout multiple possible combinations of classes they may be interested in taking.
- Princeton Pounce
    - Students can search for a course that may be at full occupancy. If the student decides, they can opt to received an email notification if at any time the class has an open vacancy. This way, as soon as the student is notified, they can enroll into the class they want to occupy the open seat.

These three applications (or TigerApps) are all currently used by Princeton University students to plan out their course schedules each year. They are all online, web applications that students browse to.

We gathered information about these applications from Princeton students through a Google Survey sent out on the residential college listservs. Participants expressed a strong preference for a singular application that combines the functionalities of the TigerApps. That is, one singular source from which they can browse for courses they may be interested in, put that course onto a hypothetical schedule (and experiment with multiple schedules within a given term), and "pounce" on this course in order to get updates on class openings. In addition to the aforementioned features, participants reported that the mobile responsiveness of the TigerApps is lacking, with blocky interfaces that clutter the elements on the page.

It was apparent that a highly-functional mobile application with an intuitive design would fix this User Feedback. A mobile interface would look more appealing to the eye, and would provide quick access on ones phone. Moreover, mobile apps tend to be faster as several pieces of data can be stored locally on the user's device rather than on the web servers, like on a website. Additionally, features like push notifications (useful for the “pounce” feature, Apple ID security verifications (important for general security) and geolocation are only supported on mobile applications, which makes room for additional features that would not be possible on a website.

Functionality

This section is structured into separate use cases.

Downloading and Installing the App

For the purposes of downloading and installing the product for the application, we will be using TestFlight, as for now keeping the application in the Beta testing phase is the most desirable for the project.

You must first download the TestFlight app on the following link: https://apps.apple.com/us/app/testflight/id899247664.

After downloading TestFlight, use the following link to download a build of the Princeton Course Planner into your TestFlight application: https://testflight.apple.com/join/eOjxJSOD.

This above link will also give a link to download TestFlight in case you have not already done so. Once clicking the link, click the button that says “Start Testing” under Step 2 (assuming TestFlight has been downloaded). After this, the TestFlight app should walk you through the steps until the build is installed onto your phone. Click “Install” and then click “Open”. The application should then immediately download onto your phone, and after some setup/instruction screens from TestFlight, you should be able to start using the Princeton Course Planner application.

Creating an account with the application

As soon as you are at the home screen of the application, click the second orange button that says “Sign up”. In the field “Princeton NetID” put in your Princeton University netid, and hit “Sign Up”. If successful, an Alert saying “You’re request has been received!” should come up. Follow the message and go to your Princeton email, where an email with the title “Verify your account” should appear. Please allow a few minutes for the email to send, or check the junk/spam folder in order to locate the email. Once you open the email, click on the word “here” which has a hyperlink to the page in which you will set your new password for the application. If the email didn’t send, feel free to click the “Sign Up” button again with your netid in the text field. Otherwise, if all else fails, route directly to http://princeton-mobile.herokuapp.com/set_password (This is the same hyperlink as in the email).

Routing to this link should open a page that says "Welcome to Princeton Course Planner". In the text fields, input a new password (preferably separate to the one you normally use to login to Princeton CAS authentication), confirm the password, and input a phone number you can receive text messages on WITH the country code. So, if your regular USA phone number is 609-258-9999, you should input 16092589999. Click "Submit", and a message should prompt you to return to the mobile application to log in. Once opening the application, input your netid and the password you just indicated, and you should be able to log in.

Creating and editing a schedule within the application

Creating a schedule from the calendar

There are two ways to create a schedule; the second way will be discussed at a later time. Once logged into the app, click on the top left portion of the app with the "ios-menu" icon. It is an icon with three horizontal lines. A menu should come from the right side of the screen, and then click on the orange button that says "Add Schedule", and then click on "New Schedule". Name the new schedule "First".

Searching a course, and using filters with the search

A new schedule has been created for the Spring 2020 term (or whatever term is the most recent at the time of reading this), so let's add a course to the schedule. Let's try and add the COS 217 course to our schedule named "First". Reopen the menu and click the button that says "Add a course", and you should be directed to the Course Search Screen. On this screen type in "cos" as the query (case does not matter). As you type, a prompt should come up reminding you to select a term to search. Before selecting the term, however, click on the top right "ios-options" icon, and under "Course Levels", choose the filter "2xx". Then, close the menu and click the orange button saying "Please select the Term...", and select "Spring 2020" from the dropdown options. The search should commence immediately, and a rotating activity indicator will let you know that the request is still in progress.

Browsing Course Details, Adding/Editing a course to a schedule

As soon as the search has finished, you should see some instructors and courses listed on the screen. Scroll through the results and click on the result that says "COS 217". This should redirect you to a Course Details screen that lists the instructor of the course, a description, as well as a list of the sections (precepts, lectures, seminars, etc.) in the course along with their current enrollment and capacities. On the top left, click the icon saying “Add to Schedule” and click “First” to add the course to your schedule name “First”. Now, on the bottom tab navigator, click Calendar to see the course added to your calendar. The lectures are set classes which cannot be changed, so they are displayed with a grey bar at the top. The precepts are not set, and so they don’t appear with a grey bar. Let’s choose precept number P01, which meets Mondays and Wednesdays at 1:30 pm. To do this, click on either of those precept times that say P01, and the schedule should update by removing the rest of the precept times while keeping the P01 precept on the calendar with the grey bars. We have now successfully scheduled our first course to our schedule.

Creating a schedule from the Course Details Page, changing between schedules on the Calendar Page

Now, let’s go back to the Courses tab on the bottom tab navigator. The COS 217 Course Details page should still be open. If not, simply re-click that course on the search page. Click “Add to Schedule” again, but this time, instead of adding to the schedule named “First”, click the “Create a new schedule!” button, and enter “Second” as the name of the new schedule. Now when you route back to the Calendar, the new schedule with COS 217 classes should appear. Pull up the menu by clicking the “ios-menu” icon on the top left, and click on the button that says “Change Schedule”. You should see that both “First” and “Second” are now listed. Simply click on either schedule name in order to pull up that particular schedule.

Creating a schedule in a different term

Now, let’s create a schedule in a different term. Press the “Courses” button on the bottom tab navigator to return to the search screen (you may have to press the back button to get there). Now, change the selected term to “Fall 2019” and wait for the results to update. This time, select “COS 226”, and click “Add to Schedule” on the top left. You’ll notice that the schedules “First” and “Second” don’t show up on this screen, since those schedules are part of the Spring 2020 term. This is a class from the Fall 2019 term. Select “Create a new schedule!” and name it “Third”. Now, when you route back to the “Calendar” screen on the bottom tab navigator, you should see the class on the schedule. In the pop-out menu, you can “Change Term” back to Spring 2020 to see the “First” and “Second” schedules from that term, and likewise do the same for any other term as you build schedules for those terms.

Browsing through an Instructor’s Prior Courses

Now, let’s go back to the Courses screen by clicking “Courses” on the bottom tab navigator. The Course Details for the COS 226 class should still be up. Note that this is the Fall 2019 edition of the class. Under the instructors list, “Kevin Wayne” should show up in orange-colored text. Click on his name, and you should be directed to an Instructor Details screen, on which all of the courses that Kevin Wayne has taught during the past few semesters are listed. The first item in the list should be the Spring 2020 edition of COS 226. Click anywhere on that item, and you should be redirected to the Course Details page of the Spring 2020 edition of COS 226. Let’s now add this course to our schedule “First” by clicking the “Add to Schedule” icon on the top left, and clicking the “First” button. Now, click the “Calendar” on the bottom tab navigator to see this course now added to the schedule “First”. Click the P02 precept at 3pm on Thursday, and now we have two classes scheduled on “First”.

Additional Filtering and Adding a Second Course to a Schedule

It is apparent that there are open spaces at 9AM and also 12:30PM everyday for the schedule “First”. So, let’s look for a class that may fit at this time. Now, let’s go back to the “Courses” screen by clicking the bottom tab navigator (You may have to click the back button to go back to the search screen). Open up the menu by clicking the “ios-options” icon on the top right, and select the 12:30PM and 9AM under the “Start Times” category. Additionally, select the “3xx” filter under the “Course Levels” category. Close the menu, and you should see that there are four chosen filters. Let’s undo the “2xx” filter by clicking the “2xx” text under “Chosen Filters”. Finally, let’s change the Term back to “Spring 2020” as our “First” schedule belongs to the Spring 2020 term. You should see two classes: COS 302 and COS 340. Let’s click on COS 302, and add it to our schedule “First” by clicking on the “Add to Schedule” icon on the top left. After doing so, we can navigate back to our “Calendar” on the bottom tab navigator and see the COS 302 lecture at 12:30PM which meets on Mondays and Wednesdays. Feel free to choose or not choose any precept in the class that you desire.

We now have multiple schedules to experiment with for our course selection. At this point, open the menu on the calendar screen by clicking the “ios-menu” icon on the top left. At the very top of the menu, there should be a button to “Log out”. Go ahead and click that button to be directed back to the Login Screen. Quit the app now as well.

Using Princeton Pounce

Let’s say we are a student who is interested in taking a Chinese class during the Spring 2020 term. Let’s log in to the application, and route to the “Courses” screen. Open up the filter menu by clicking the “ios-options” icon and click the “CHI” filter under the category “Departments”. Then, in the text box, input the string “ch”. Again, this is case-insensitive. You’ll need to select a term to search, so click the button and choose “Spring 2020”. Click on the class “CHI 102”. At the time this paper will be reviewed, there should be a section with open spots. On the course details page for this Chinese class, click the top right “ios-paw” icon that says “Pounce”. This will launch a background thread on the server side, which will periodically (currently every 5 minutes, but this can be configured on the server side) track the Registrar’s enrollment records. As soon as a spot opens in the class, you should be notified by text and email notifications. Wait for about 5 minutes, and you should receive an email and text message on the phone number you provided at authentication describing that the class has an open spot. Please observe the contents of the email and the text message.

Now, do the same thing with another course. Press the back button if you are not already on the “Courses” screen of the bottom tab navigator. Pick the class entitled “CHI 406”, and press “Pounce” on this course as well (It is unlikely that this class will be filled up at the tie this paper will be reviewed). Wait for about 5 minutes or so, and you should receive an email and text message indicating that a spot has opened in this class as well. Please observe the contents of the email and the text message.

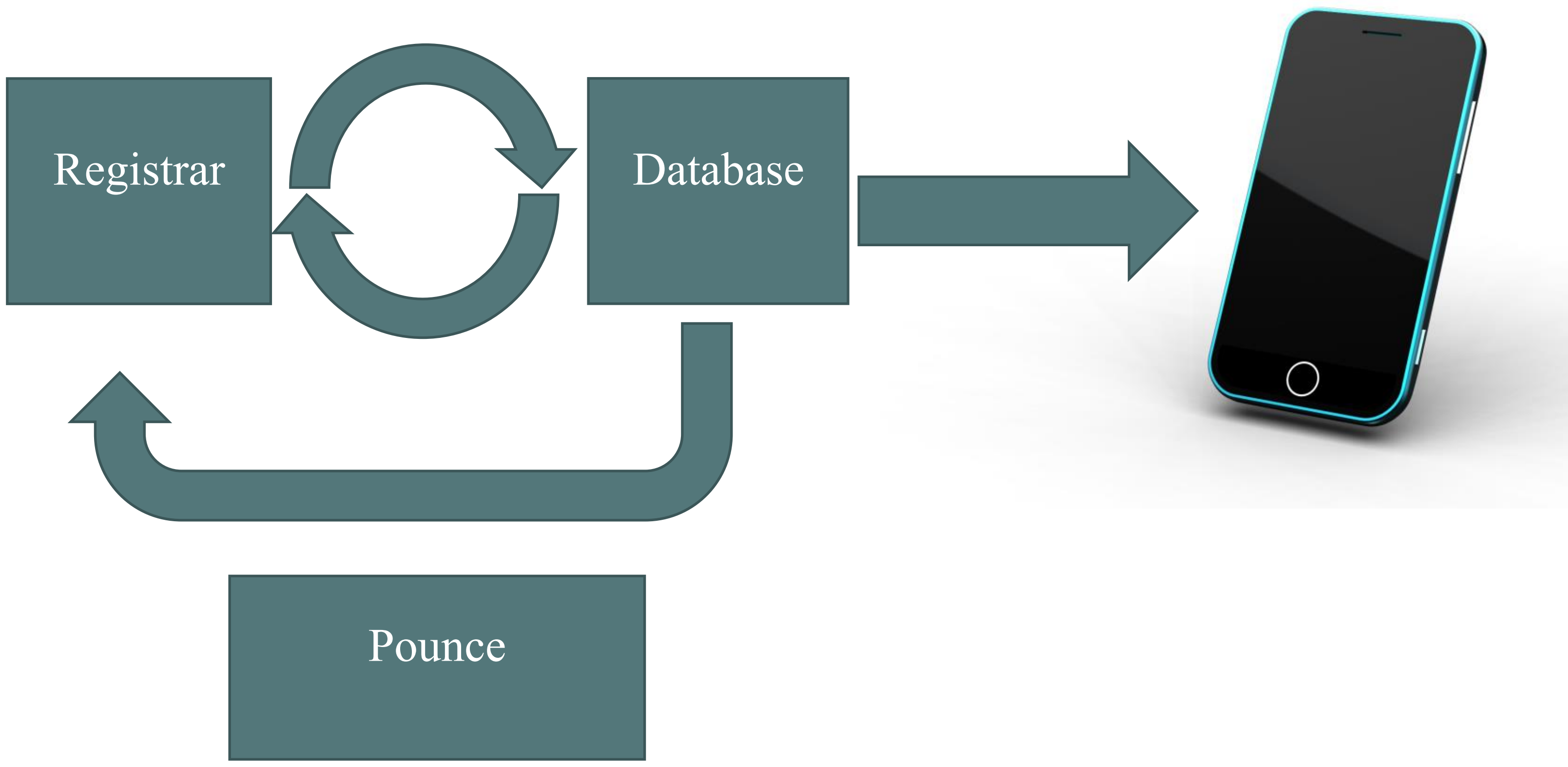
Typically, if a course is currently closed, the “Pounce” feature will send you an email as soon as it detects that the course has an opening. In these cases, the class is already open, but this is okay for demonstration purposes. If, after 48 hours, the “Pounce” feature is unable to find any openings in the course that was requested, a courtesy email is sent to the User letting them know that the Pounce has stopped tailing their chosen course.

Summary

The application has three main components/features: Calendar to visualize a schedule, Search Engine to browse through courses, and Pounce to inform a User when a course has an opening in its enrollment. The above “Functionality” section walks a User through all of these Use Cases.

How It Works (Programmers' Guide)

The below diagram represents the overall structure of the Princeton Course Planer system:



Basic Structure

The application is three-tiered. The usage of React Native encompasses the first layer, the graphical user interface, in the form of a mobile application. This tier is further subdivided into the “Calendar” Screen and the “Course Browsing” Screen, not sharing any common screens among each other. The second tier is the server processing tier. Its client-facing interface is a RESTful API that accepts GET or POST requests when the client would like to view data currently stored in the database or would like to update data in the database. The final tier is the database which stores relational information about users, courses, schedules, classes, and instructors.

The Registrar and the Database are kept in sync within one hour of each other. That is, a script is launched every hour on the server side and it monitors the Registrar for changes in the course data. The script updates our database during this process, and so our database is ultimately in sync (within one hour) with the Registrar.

The reason this design decision was made is that querying straight from the Registrar’s website would result in queries of almost 5 minutes (observed through empirical testing). By having our own custom database with its own HTTP endpoint, we can reduce this query time to just a few seconds in the worst case. Therefore, the mobile application never directly communicates with the Registrar website. Rather, it indirectly communicates to the Registrar’s website through our database.

Another reason for the choice to use our own database is to accommodate user schedules. That is, when users add a course to a hypothetical schedule, the information must be stored in some location; it certainly cannot be stored in the Registrar, as we don’t have write access to their database. This information is stored in our database.

The Backend

The backend is designed and created using the micro-framework Flask. Whenever a network request is called from the mobile application, it will reach the endpoints defined by this Flask application, and not those of the Registrar. The root endpoint for this application is http://princeton-mobile.herokuapp.com.

The client interacts with this backend through a RESTful API. Since the application is structured around the relational nature of the database, GET and POST requests to the server are sufficient to access or modify data without having a stateful connection to the server.

There are also several extensions that are used such as Flask-Mail (Princeton Pounce notifications, sign up notifications), SQLAlchemy (communication with the database), Flask-Admin (managing the database as a programmer), and Nexmo (notification through SMS messaging).

The Database

The database is created with a Flask-SQLAlchemy extension. On Heroku, this database uses Postgres. Postgres is really good for this type of an application for the following reason. Already mentioned was the script that periodically updates the database, keeping it in sync with the Registrar. This script takes around 12 minutes to execute. If the database were SQLite, the database would be locked during the execution of this script, which means that any GET/POST request that wishes to retrieve/update information would not be allowed to do so.

On the other hand, PostgreSQL does not lock the database during execution of the script, and so GET/POST requests can retrieve/update during execution of the script. This is obviously much more desirable behaviour, and so motivated our choice to use Heroku Postgres as our database version.

Below are the fields of the databases, along with the tables they belong to:

A. User Table

id: primary key for this table

name: Name of the User

schedules: relation to the schedules that a User may have

phone: phone number of the User

email: email of the User

netid: netid of the User

loggedIn: Boolean whether the User is logged in or not

password: the password of the User

B. Term Table

id: primary key for this table

cal_name: name of the term (e.g. “Spring 2020”)

code: Registrar’s code for this term

start_date: When this term starts

end_date: when this term ends

schedules: the schedules that are made by users for this particular term

courses: the courses that are offered during this term

C. Schedule Table

id: primary key for this table

name: name of the schedule

classes: the classes that have been added to this particular schedule

user_id: which user_id is the owner of this schedule

term_id: which term_id this schedule belongs to

D. Course Table

id: primary key for this table

catalog_number: the catalog number of this course in the Registrar

classes: the classes (section) that belong to this course

dept_code: department code of this class

guid: unique GUID for the class in the Registrar

reg_id: Registrar's reg_id

title: Title of the course

description: description of the course

start_date: When this course starts

end_date: when this course ends

track: The track this course is a part of

crosslistings: course's crosslistings

term_id: term_id this course belongs to

dept_id: dept_id this course belongs to

E. Class Table

id: primary key

class_number: Registrar’s class id for this course

section: Which section number this class is

status: whether or not the class is open

type_name: what type of class this is (Lecture, Precept, and/or Seminar)

capacity: Total capacity of this class

enrollment: How many people are currently in the class

course_id: course_id that his class is a part of

meetings: The meeting times for this class

F. Instructor Table

id: primary key

name: Name of the instructor

emplid: Registrar’s employee id number

courses: The courses that this instructor teaches

G. Department

id: primary key

name: name of the department

dept_code: department code of this department

courses: The courses offered by the department

H. Meeting

id: primary key

start_time: when this class starts

end_time: when this class ends

building: where this class meets

room: room this class meets

day: days this class meets

class_id: class_id that this meeting is a part of

The Frontend

The frontend was implemented using the JavaScript framework React Native. We chose React Native because of the speed with which apps can be created on the platform, and additionally to make it easier to release this application on both iOS and Android (as React Native translates the application to both). Additionally, React Native provides an easy to use library to make fetch GET/POST requests to RESTful API's, so this made technical sense in terms of integration with the server.

The interface is built around a "Bottom Tab Navigator" which routes to both the "Calendar" screen and "Courses" screen. Internal storage is managed through a mechanism known as Redux. Redux is a mechanism that allows us to keep internal "global" state throughout the app, or certain pieces of information that several screens share. In React Native, state is normally restricted to a specific screen but Redux allows us to keep state that is persistent across all screens.

Evaluation

The Princeton Course Planner application succeeds in its primary goal of integrating the functionalities of the TigerApps (Princeton Courses, ReCal, and Princeton Pounce). From the beginning, the app was created with one primary goal: simplistic yet highly functional. After conducting several evaluations both by experts (in this case, the programmers) and users (other Princeton students), we are confident that Princeton Course Planner meets the needs of its users.

The core functionality of the backend was tested thoroughly throughout the semester with Postman, a client that issues networking requests to servers, to access the database contents and make changes, and retrieves a response. We formatted these requests as JSON objects, and sent them to our database with every possible input that we could think of, especially those that may cause injection attacks. Every time we created a new endpoint that affected the database in some way, we would do this type of testing before adding the change to the application. Another layer of protection provided by our database is the Flask_SQLAlchemy developer community. Our database is not created by a bunch of SQL statements, but rather with the Flask-SQLAlchemy framework. Therefore, the database is robust in its design and its implementation.

The main goals for the frontend of the application are intuitiveness and high-functionality. On the calendar screen, this means having a calendar that fits within the screen and the ability to edit and delete courses from the schedule. Users should be able to monitor and view their schedules from all terms, and they should be able to see ALL schedules they've made within a specific term. On the courses screen, users should be able to type in a search query, choose certain filters, select a term, and then browse the search results. These search results should be individual courses. Users can click on these courses to get to a Course Details Screen that gives further information about the course (instructors, meeting times, and course description). On this screen, there should also be a way to add the course to some schedule, and a way to pounce on the schedule. Pouncing on a schedule will allow a User to receive a notification (email and/or text) whenever there is an opening in the class. These features also allow the user more flexibility and customizability while they are browsing for courses. Our application encompasses all of these necessary features, and so the functionality that is necessary is present.

We also received feedback from our User Group. Since this is a mobile application and we only have an Apple Developer Account, our users were made up of iOS users. We distributed the application through TestFlight, a beta-testing platform for iOS applications. After a brief approval period from Apple, we were able to do "External Testing" which allows any user with an iOS device to download the beta version of the app. We received feedback by sending out user surveys and having peers conduct user tasks. With surveys, we released one Google form that asked users what they feel is missing from the current TigerApps, and another that asked questions about their user experience. Testers were instructed to use the application without much direction, so as to find out whether the application was intuitive. After about 30 minutes of use, they would answer for us the following questions:

1. Do you feel the application is intuitive?
2. Did you encounter any bugs during your testing?
3. What additional features would you like to be added to elevate your experience with the app?

Although we didn't receive a whole lot of written feedback (only two responses to the first Google form and 6 for the second), what we did receive was wholly positive. Users praised the intuitive nature of the application, and no bugs were reported. A few complaints about the speed of the application were brought up, but at the time, we were using a free Heroku dyno that goes to sleep after 5 minutes of inactivity. The speed of the application greatly improved when we switched to a Heroku performance dyno, so we aren't very concerned about the speed of the server at this point. Few others commented on stylistic elements to the application. Notably, that the application has a monotonous color-scheme. This was also more towards the beginning of development, where the focus was more on functionality rather than aesthetic appeal, but this feedback was taken seriously, and the colors have been enriched since then.

Many users responded to the application with comments along the lines of “This application is extremely useful!” exclaiming that “[they] would definitely use this over other TigerApps when it comes out.”

Based on our own walkthroughs of the app, coupled with user evaluation from peers, we concluded that the Princeton Course Planner is highly intuitive and provides its users with valuable information in their Princeton lives. The application in its current state is good enough to be used by the mass Princeton community, which means that once the spring semester rolls around, we will be able to do a mass release across campus. This is the next step in our testing going forward.

Conclusion

All in all, the Princeton Course Planner application provides many thought out features that are fully integrated with the University Registrar, our own Heroku database, as well as mobile frontend which ultimately gives the user an intuitive and overall pleasant experience. The product was created using frameworks that are easy to scale up and improve upon for the future. This is important, as there are currently plans to continue work on this application for the spring semester. Some possible additions to the application can be made during this time, new features that could take the application to a whole new level.

One major thing that could be done to improve the application is fixing how it deals with CAS. That is, currently, a user receives an email prompting them to log in to CAS authentication, thereby creating an account with our application. A far more desirable flow would be for the user to authenticate through CAS within our application itself, without having to open their emails. Early beginning of this were attempted, namely using the WebView component of React Native to open up a browser. However, the problem with this is that each subsequent network request has no access to the CAS cookie that was sent back to the WebView component. If this feature could be fixed, it would greatly enhance the user experience and intuitive nature of the app.

Another thing that could be done, and that is being planned for the second semester is to incorporate a social media component into the application. Nowadays, social media is often times what can make or break an app. Namely, if this application were a place in which students comment on, rate courses, post evaluations, recommend courses to friends, and coordinate schedules with friends/other students such that they are in the same class, it would greatly enhance the user experience. This would mean that students can create a schedule on the current version of the app, then instantly share a schedule with a friend that also uses the application. Then, the app could provide suggestions on classes that both friends may like to take depending on prior courses they have taken.

These features would immensely increase the appeal of the application, and would be excited to see in place. Overall, this application does its job to help students in the course planning stage, and should make an impact immediately in the Princeton community. Once it launches, it is only a matter of time before its usefulness becomes apparent.