TigerSnatch: Project Overview

Shannon Heh (lead), Nicholas Padmanabhan, Byron Zhang
{shannon.heh, nicholaspad, zishuoz}@princeton.edu
Code: github.com/shannon-heh/TigerSnatch

We confirm that the lead instructor and TA adviser have read-only access to our source code repository.

Elevator Speech

Have you ever wanted to take a course so badly, but it filled up and now you're stuck refreshing the Course Registrar page every minute waiting for someone to drop? Or perhaps you've needed to switch into a full precept but can't find someone to swap sections with you? Worry no longer! TigerSnatch notifies you if a spot in your desired course opens up and can help you coordinate section swaps – eradicating your enrollment problems once and for all!

Overview

Inspired by the intent of Princeton Pounce, TigerSnatch will not only restore the basic functionality that notifies Princeton students when a course opens up, but also provide new features, including a system to facilitate mutual section swaps and a personalized dashboard of awaited courses with a real-time waitlist count. TigerSnatch is a responsive web application that queries data from the OIT MobileApp API and scrapes information from the Course Registrar page.

Requirements

Problem

Many courses, especially small seminars and popular introductory courses, reach capacity quickly after enrollment opens. Sometimes, students want to enroll in full courses or sections, but there's no way to know when a spot opens up without continually monitoring the Registrar's course page. Further, it is currently difficult to find and coordinate section swaps with students in full sections or precepts in which you want to enroll.

Intended Users

TigerSnatch's primary audience will be the community of Princeton students. Administrators and course staff may find the live waitlist data useful for gauging interest in a course and potentially adjusting enrollment limits.

TigerSnatch will likely be most popular during the course enrollment periods at the end of each semester and add/drop periods at the start of each semester.

Existing Solutions

The Course Registrar's page (registrar.princeton.edu/course-offerings) shows the current enrollment for each course, but does not provide any way to track when spots free up. Using the Registrar's page, the only way to know when a slot opens is to continually refresh the web page.

One of the older TigerApps, Princeton Pounce (pounce.tigerapps.org) was created in 2013 to notify students via text or email when a spot opens up in their desired course. However, the app has been inactive since Fall '19, and its UI and features are unsatisfactory.

TigerSnatch is superior to both solutions because our platform:

- Displays a personalized user dashboard that lists awaited courses and offers "quick" controls
- Finds and coordinates mutual section swaps between students in full sections
- Maintains ordered waitlists, rather than notifying all waiting students at once
- Allows users to remove themselves from waitlists

Functionality

Features

TigerSnatch will implement the following features (from the user's perspective):

- User authentication through Princeton's Central Authentication Service
- Personalized dashboard with list of current courses and sections student is on the waitlist for
    - Displays number of students currently on waitlist (in addition to other information)
    - Provides switches to remove yourself from a particular waitlist
- Searching for courses
    - Fuzzy text-based searching based on course name, department, course number
    - Advanced search may allow filtering courses based on course popularity
    - When user submits query, list of matched courses is displayed and user selects desired course to display all lecture/section options
- Individual course pages displaying information about a single course
    - Lists lectures, precepts, labs, etc. for the course in current term along with basic details (e.g. class times)
    - For full lectures and sections, flip switch on if you want to be notified, and turn off switch to remove yourself from waitlist
    - Displays warning messages if you may not be able to enroll in course for any reason (e.g. departmental waitlist, grade-based caps)
- Notification system that sends emails and/or texts to students on waitlists when spot(s) open up in a particular course
    - Starting from the top of the waitlist, students are notified one-at-a-time and are given a 10-minute window to enroll before the next student is notified.
    - After their 10-minute window finishes, the student is removed from the waitlist, but can re-enter the waitlist (at the bottom) if their desired enrollment was unsuccessful
    - Notifications halt once the course fills or the waitlist empties - whichever comes first
- Search and notification system for mutual section swaps
    - *Without* mutual section swap notification enabled, the only notification a student would get would be if a spot opens up (same mechanism as with course waitlists)
    - *Optionally*, students may specify their current precept/lab/etc. enrollment for the course
        - If the student's current section field is filled in, the student will either receive a notification if a section slot opens up, OR receive a notification if a mutual section swap becomes available with a student in one of their desired sections
        - If the mutual swap doesn't work out, students must re-enter waitlist for mutual swaps by re-entering their precept/lab/etc. and flipping the switch to green

Sample Use Cases

Scenario 1: “I’m interested in enrolling in a full course”.

A user wishes to be notified when a spot opens up in a full course, let’s say COS126.

1. The user navigates to the TigerSnatch website and logs in with CAS authentication.
2. The user is directed to the dashboard. If the user is a first-time user of TigerSnatch, under the Contact Preferences section, they should enter their email and phone number, and click Save. This is how the user will be notified of a spot. Note that email defaults to netid@princeton.edu and is a required notification method. Phone number is optional.
3. The user enters “COS” into the search bar on the left. A list of course entries offered by the COS department are displayed. User selects the “COS126” course.
4. The right pane displays course details for COS126, including a table with the times for lectures, precepts, and labs. Since the course is full, the lecture row is highlighted to reflect this.
5. Under the “Notify” column of the table, the user flips the switch to green for the lecture entry. The user has now entered the waitlist for COS126 L01.
    a. Similarly, to enter the waitlist for a COS126 Precept, the user will flip the switch for their desired precept (let’s say P02). If the switch is greyed out, this means the precept has spots available.
6. After entering the waitlist, the user clicks back to the dashboard. The dashboard should list COS126 L01 and COS126 P02 as awaited classes. The user can see their spot in line in the waitlist for both the lecture and precept.

Scenario 2: “I’m in a full precept, but want to switch into another full precept”

A user wishes to be notified when a spot opens up in a full precept, and optionally when a mutual section swap becomes available.

1. The user undergoes steps 1-2 in Scenario 1.
2. The user searches for a course, typically one they're already enrolled in (which is why they're interested in a particular precept/section time). Suppose the user selects the "COS126" course.
3. The right pane displays course precepts/sections (in addition to lectures) for COS126, and the user scrolls to their desired precept/section. Since that precept/section is full, it is highlighted.
4. Under the "Notify" column, the user flips the notify switch to green for the precept/lecture entry. They have now entered the waitlist for their desired precept.
    a. Note that this can be done with as many full precept/section times as desired.
5. In the "Mutual Section Swaps" pane at the bottom right, the user may opt to receive a notification by providing the current precept/section in which they're enrolled and flipping the "Notify" switch to green.
    a. Situation when a mutual section swap notification would trigger: User A desires P02, and has indicated that they are currently enrolled in P01. User B desires P01, and has indicated that they are currently enrolled in P02. Both P01 and P02 are full. A mutual section swap notification will be sent to User A and B to share their contact info and let them coordinate a section swap.
6. After entering the waitlist for a precept/section and optionally opting into the mutual section swap system, the user navigates back to the dashboard, which should reflect that the user has entered the waitlist (and show their current spot in line) for their desired precept/section.

Scenario 3: “I’m first the waitlist for a full course, and a spot in the course opened up”

A user receives an email (and text message, if they choose to provide their phone number on the dashboard) notification.

1. One course/precept/section for which the user is in the waitlist now has a slot available.
    a. Note that if this a mutual section swap notification, both users will receive an email containing the email address of the other user so that they can facilitate contact and arrange a swap time.
2. The user has 10 minutes (24 hours for mutual section swaps) from the time the notification is sent to enroll in the course on TigerHub, before the next user in the waitlist is informed.
3. After 10 minutes (24 hours for mutual section swaps), the user will be automatically removed from the waitlist for this course or section (and the corresponding entry is removed from the user's dashboard).
4. If the course or section is still not full, the next user on the waitlist will receive a notification and steps 2-3 are repeated.

Design

Database

TigerSnatch will use a MongoDB database, hosted on the same location as the server (using Heroku). The database will contain collections of:

- `users`: their contact information, enrolled waitlists, and desired swaps
    - Data in users is set by the user
- `mappings`: maps a department + course number (which matches a user's search query for a course) to an internal courseid
    - Data in courses is gathered via calls to the OIT courses API
- `courses`: course information (i.e. title, department, number, times, days)
    - Data in courses is gathered via calls to the OIT courses API, and if necessary, scraping from Courses Registrar page
- `enrollments`: enrollment and capacity for each course
    - Data in enrollment is gathered via OIT courses API
- `waitlists`: list of students awaiting each lecture, section, precept
    - Data in waitlists is set by user

Server

The TigerSnatch server will be hosted on Heroku and run the Python runtime environment. The server will have the following key modules (subject to change):

- `app.py`: starts the server to listen for browser requests, loads all the other modules
- `config.py`: handles configurations settings including database credentials
- `update_enrollment.py`: calls OIT courses API periodically to check for changes in enrollment numbers in courses or sections
- `notify_users.py`: handles logic for notifying relevant users via text or call
- `api.py`: defines and implements API endpoints, handles client requests, and returns response to the client after extracting necessary data from the database

`api.py` defines the following endpoints (subject to change):

- **POST** /api/user/contact: Store user's contact information
    - The request contains the user netID + user's email and phone number.
- **POST** /api/user/waitlist: Adds user to the specified course or section's waitlist.
    - The request contains the user netID + the ID of course or section whose waitlist the user should be added to.

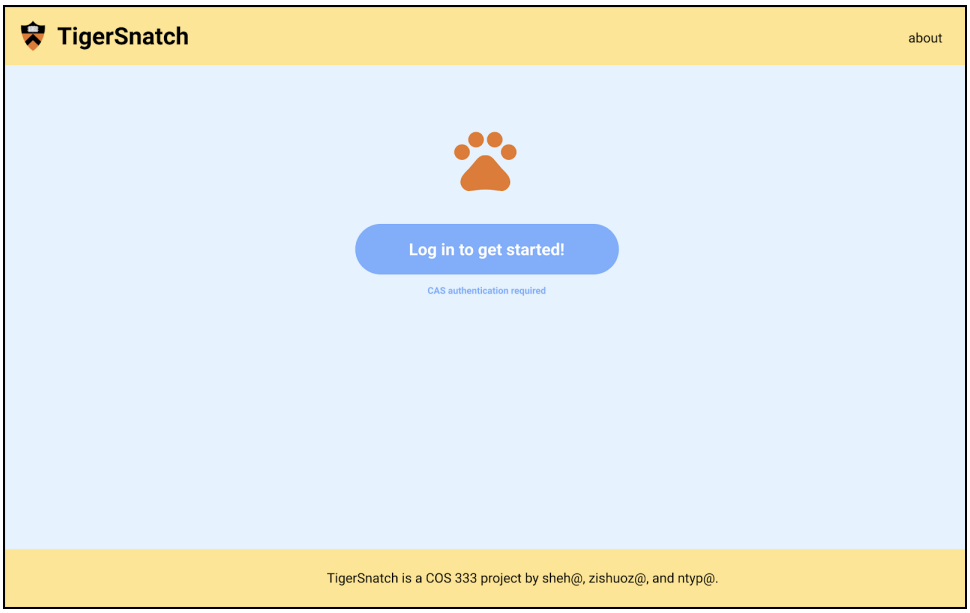
- **POST** /api/user/mutual: Add course to user's list of desired mutual swaps.
    - The request contains the user netID + the current section of the course that student is enrolled in.
- **DELETE** /api/user/waitlist: Remove the user from the specified course or section's waitlist.
    - The request contains the user netID + the ID of the course or section from which to remove the user.
- **GET** /api/user/waitlists: Return all courses that a student is waiting for.
    - The request contains the user netID.
- **GET** /api/courses: Return course data for a specific course
    - The request contains a query parameter (e.g. a substring) with the course department and course number (for now -- we will later advance the search to allow search by course title, professor, etc.)

Client

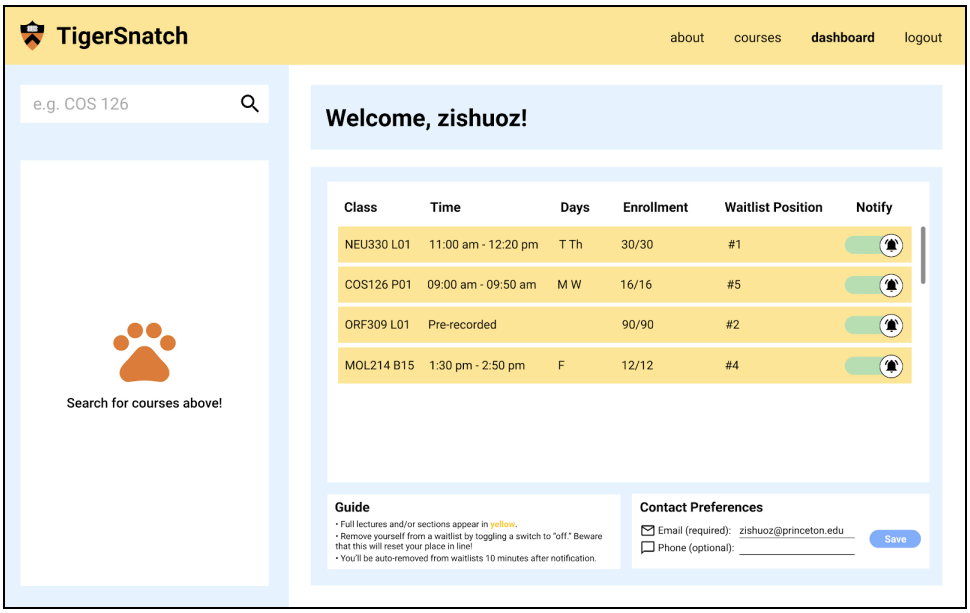
The front-end (a web application) will be built using HTML, CSS (with Bootstrap), and JavaScript (with jQuery and AJAX). The application will consist of four pages - Princeton CAS login, "dashboard" (displaying current waitlist enrollments and user notification preferences), "courses" (consisting of a course search feature to enroll in lecture, precept, and/or section waitlists and optionally mutual section swap notifications), and "about" (containing information about us, the developers).

As shown below (subject to change), the dashboard and courses pages will be similar in layout, with two main panels. The left panel enables course search - if the user searches for courses while on the dashboard page, the application will dynamically redirect (without interrupting the user) to the courses page to allow for search results to be shown. The right panel contains a header with an overview of the current displayed data (e.g. welcome message, course title, course department and number, applicable course warnings).

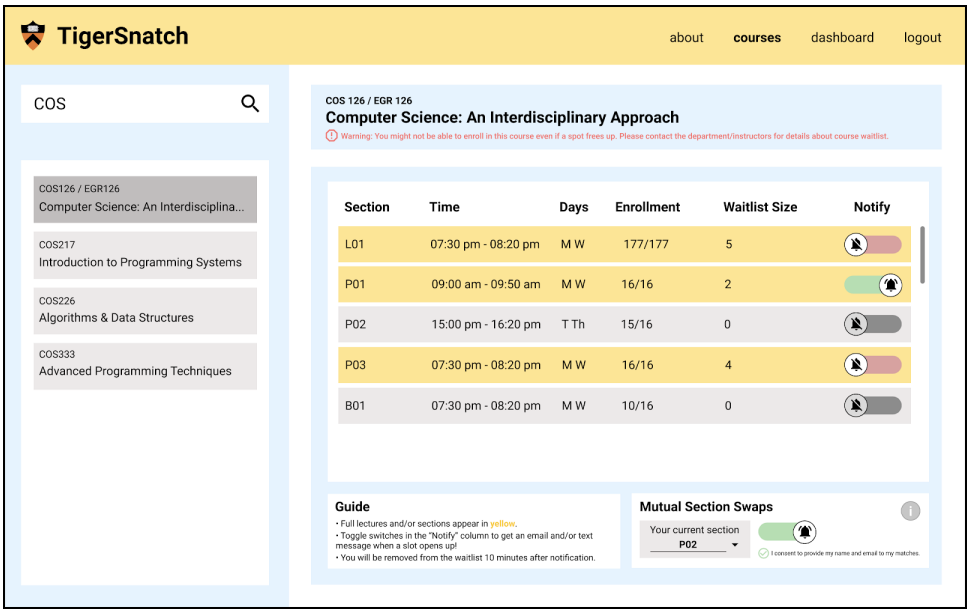
The main body of the right panel contains either the user's current waitlist enrollments (if on the dashboard) or course search results (if on the courses page). The bottom of the right panel contains a short "guide" with tidbits on how to use the current page. It also contains fields to change contact preferences (if on the dashboard), or a field and switch to enable mutual section swap notifications (if on the courses page).



Login



Dashboard (landing page after login)



Course search with waitlist notification switches

Milestones

Minimum Viable Product

- The user can search for a course based on department + course number.
- The user can add themselves to the waitlist for a course, precept, or section.
- The user will receive a notification (email/phone) when a spot opens in an awaited course, section, or precept.
- Waitlists are ordered and when a spot opens, the front-most user has a 10-minute window to enroll until the next student is notified. Then the user is removed from the waitlist.
- The user can view awaited courses, sections, and/or precepts and edit contact information on a dashboard.
- The user can remove themselves from a waitlist via a direct search.
- User login is CAS-authenticated.

Stretch Goals

- The user can remove themselves from a waitlist on the dashboard (i.e. "quick" controls for current notifications).
- The dashboard displays current user's spot in line in a waitlist.
- The user can specify preference for a mutual section swap. If two students are both in full sections of the same course and want to switch into each other's section, the system will notify both students of this potential mutual swap. Students themselves must arrange the swap - each will be given the other's email address.
- The user can search for courses based on additional terms, like professor name, course title, etc. The search will be more advanced to allow fuzzy text-based searching.
- Each course page, in addition to class times, will display useful information about a course, including how many students are on the waitlist for a course and disclaimers that may affect a student's ability to enroll in the course (e.g. departmental waitlists, application-based course).
- (if time allows) An administrative interface that allows future maintainers of TigerSnatch to perform "root" level actions, e.g. clear waitlists, pause waitlist enrollment, reset profiles, etc.

Project Schedule

Week	Goals	Deadline
3/8	● Create modules ● Set up database (MongoDB) ○ Populate database with course info from MobileApp API ● Set up key endpoints (Flask) ● Set up skeleton HTML files (Jinja2) ● Set up CAS authentication	
3/15	● Basic course search feature ● Implement key endpoints ● Basic “add/remove waitlist” feature ○ Set up ordered waitlist ● Basic “user notification” feature ○ Set up email and phone notifications ● Basic “update enrollment” feature ○ Read from courses API to update enrollment numbers (will need to create dummy data) ● Secondary goal: CSS	
3/22	● Basic “waitlist countdown” feature ○ Script (cron job): check enrollments for waited classes (at most) every 2 minutes, send notifications for available classes and removes user(s) from applicable waitlist(s) ○ Updates course page on click if it’s been more than 2 minutes since the last click ○ *remove enrollments modifications in Monitor scripts* ○ I.e. bridge Monitor, Notification, and Waitlist ● Hosting site and server (Heroku), database (Atlas) ● Set up endpoint to edit email & phone number field ● Basic dashboard and course page UI ● Other - strip spaces from course search queries (enables e.g. "COS 126")	3/26 - prototype due
3/29	● Test/debug basic features setup in previous week ● Get feedback	
4/5	● Basic dashboard and course page UI ● Get and implement feedback on MVP ● Swap feature ● Scrape and show warnings for each course ● Test/debug basic features setup in previous week	4/9 - alpha due
4/12	● Get and implement feedback on alpha version ● 4/13-4/15: test app with real-time data from course enrollment period ● Advanced dashboard and course page UI ● Show waitlist statistics within course listings (opt.) ● Fuzzy course search feature (opt.)	

4/19	● Get and implement feedback on stretch features ● Finalize dashboard and course page UI ● Work on presentation and slides	4/23 - beta due
4/26	● Get and implement feedback on beta version ● 4/26-4/29: test app with real-time data during add/drop ● Begin User's Guide Document, Programmer's Guide Document, Product Evaluation Document, Project Evaluation Document	4/30 - presentation and slides due
5/5	● Finalize and submit User's Guide Document, Programmer's Guide Document, Product Evaluation Document, Project Evaluation Document, Source Code	5/5 - entire project due

Risks

Data Dependencies

The course data we require depends on the Registrar's maintenance of accurate course information on the Registrar Course Offerings website and the Office of Information Technology (OIT's) real-time maintenance of their Courses API. We will try to retrieve most of the course data from the Courses API and resort to screen scraping for the rest. Since we will have to screen scrape the Course Offerings site for course-specific disclaimers/warnings, there is the risk that the Registrar will change the structure of their website and we will need to adjust our code accordingly.

Personal Challenges

None of us have worked with jQuery or AJAX before. We are not too concerned because we all have JavaScript experience and understand the basic concepts of asynchronous web development, but we anticipate spending extra time on searching for existing code samples and reading documentation.

Sustainability

If we find that Princeton students are commonly using TigerSnatch, we will have to consider how to sustain this project and ensure it does not succumb to the same fate as Princeton pounce. In particular, we must think about who will take over our codebase and the potential of choosing more robust data sources.