Princeton Courses Design Document

Team

- Bensu Sicim: bsicim@princeton.edu
- Caterina Golner: cgolner@princeton.edu
- Kara Bressler: karab@princeton.edu **(Team Leader)**
- Mel Shu: mshu@princeton.edu
- Sebastian Hallum Clarke: sebastian.hallum.clarke@princeton.edu

Overview

Princeton Courses is a website that provides the Princeton community with an easy-to-use and high-powered way of learning about courses and discovering new ones. The system allows students to find the courses that are most relevant to the next step on their individual academic journeys. Princeton Courses lets student search for and view course information and evaluations on a user-friendly interface.

Princeton courses receives its data from the Registrar.

Requirements and Target Audiences

Problem

Princeton Courses solves the problem that students currently do not have an easy way to learn about courses. The Registrar's Course Offerings website is slow, visually unappealing, and hard to use. Our project will address these problems and offer additional features.

Intended Users

Our primary target audience is Princeton students, however other community stakeholders (such as faculty and administrative staff) may also find the system useful.

Most of the traffic to Princeton Courses will occur during the course selection period in the second half of each semester and during the add/drop period in the first two weeks of each semester. These are the times when students most desire information about courses.

When students are searching for information about courses, these searches are typically either the student learning about courses they have already heard about (such as courses in their

department or core courses for certificates) or discovering courses they haven't already heard about (such as courses fulfilling distribution requirements).

When considering whether to take a course, students typically evaluate these criteria:

- Whether the course is scheduled at times that are compatible with other courses the student wants to take
- How the course contributes towards the student's academic journey (as a prerequisite, towards a major/certificate, satisfying a distribution requirement, etc…)
- Whether the content of the course will interest the student
- The amount and type of work the course demands
- The experiences of past students of the class/professor

Princeton Courses will allow students to search for courses and easily learn the information they require to decide whether to take the course.

Existing Solutions

Princeton Courses is a replacement for the Registrar's Course Offerings website. Princeton Courses is superior to the Registrar's system because our platform:

- Presents a user interface that is faster, more attractive and easy to use
- More prominently features course evaluation information
- Is optimised for devices of all modern screen sizes
- Allows for favoriting courses for easy access in the future
- Allows for fast and intelligent searching with results delivered while the user is typing

Functionality

Features

From the perspective of the user, Princeton Courses will implement the following features:

- User authentication through Princeton's Central Authentication Service
- Powerful searching for courses and instructors
    - Fuzzy text-based searching across all relevant course attributes
    - Filtering of search results based on whether the course conflicts with courses on the user's favorites list
- Sorting search results intelligently by relevance or specific course attributes
- Displaying information about a single course or instructor
    - Most of the information the registrar currently lists for a course
    - Numerical course evaluation information
    - Insights about the course based on common phrases in students' comments
    - Numerical evaluation of the instructor based on all the courses taught
- Saving courses to a list of favorites

Sample Use Cases

Scenario 1: Finding a Course in User's Department

A user wishes to find departmental courses in their major.

1. The user accesses the Princeton Courses website and authenticates through CAS.
2. The user enters "COS" (for the computer science department's code) into the search bar. A list of all courses affiliated with the computer science department appears in the course listings pane.
3. The user clicks/taps on courses that he finds interesting. The details of the course appears in the course details pane.
4. The user reads the course description and evaluations.
5. The user clicks a button to add the course to his favorites list. The course appears under the "Favorites" list in the course listings pane.
6. The user repeats steps three through five for a variety of courses.

Scenario 2: Finding a Fifth Course

A user wishes to find a fifth course for their semester. The user doesn't have many specifics but rather is window-shopping for something they might find interesting and has great reviews.

1. The user accesses the Princeton Courses website and authenticates through CAS.
2. The user searches for the "HA" distribution area and sorts the courses based on their rating.
3. The user clicks on each one in succession, and based on the evaluations snapshots and common responses, marks the ones which don't conflict with the four other classes the user has for the 'watch list.' Later, the user reviews workloads and ultimately adds one to the 'favorite' list.

Design

Server

A server hosted with Heroku running the Node.js runtime environment. The server will have the following modules:

- App.js, which is the entry-point into the app. This module loads all the other modules and starts the server listening for web requests.
- Config.js, which loads configuration settings such as database credentials.
- Database.js, which connects the app to the database using Mongoose.
- Api.js, which handles requests from the client, interfaces with the models, and returns the requested response.
- The following models (each corresponding to a collection in the database):
    - User, which contains a user's netID and list of favorite courses

    - Course, which contains all the information about a course and a method for saving new courses to the database.
    - Semester, which contains information about semesters (name, start/end date, registrar code, etc…).
    - Instructor, which contains an instructor's name, courses taught, and aggregate data on evaluations for these courses.
- Modules for importing into the database course offerings from OIT's webfeed and for scraping additional information and course evaluations from the Registrar's website.

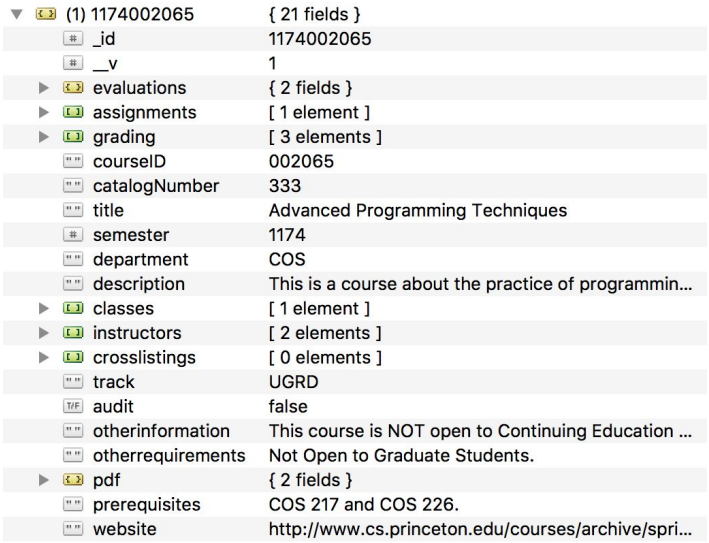
The client API will expose the following endpoints to the client:

- POST /api/courses: List courses in the database
    - The request must contain a query parameter, which is either a string for running a Full Text Search of courses or an object for more fine-tuned searches.
    - The request may contain a sort parameter, which is a string. Valid parameters are "relevance", "title", "department", or "code". The course results will be sorted by this parameter.
- PUT /api/user/favorite: Add the course supplied in the request's course parameter to the user's favorites list.
- DELETE /api/user/favorite: Remove the course supplied in the request's course parameter from the user's favorites list.
- GET /api/user/favorites: List all of this user's favorite courses.

All of these endpoints require the user to be authenticated.

Database

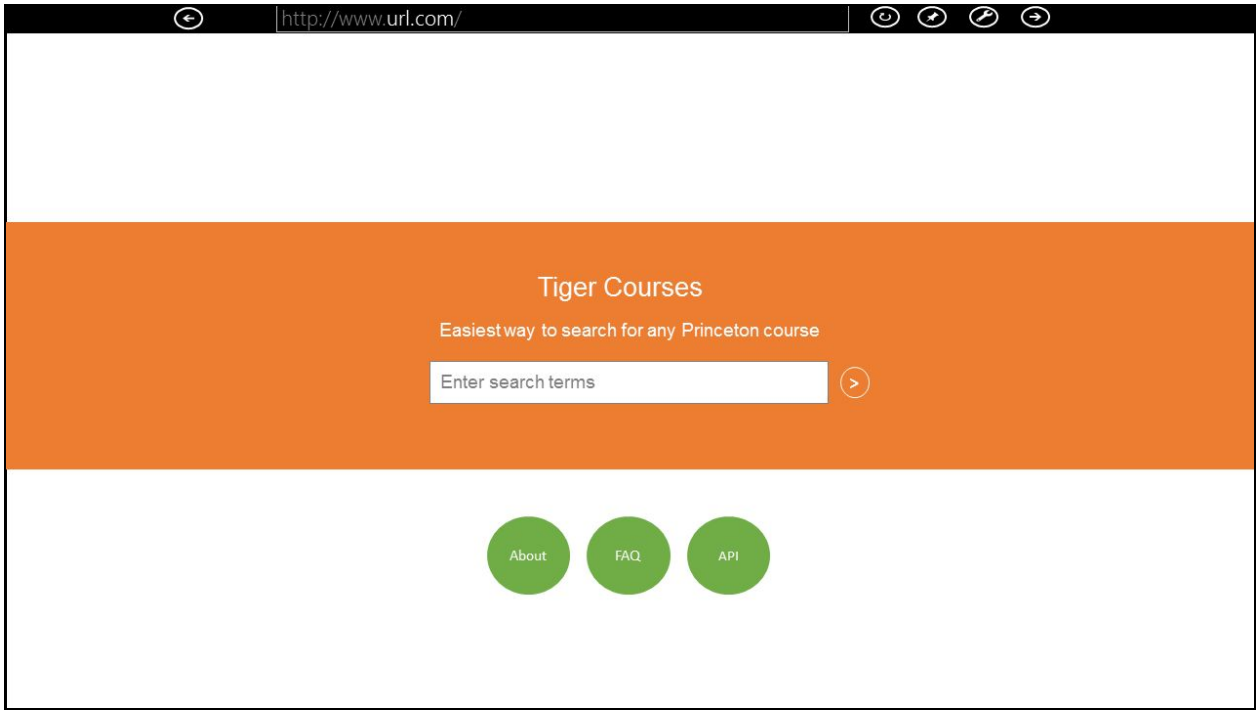
A MongoDB database hosted with mLab. The database will contain collections of users, courses, instructors, and semesters. MongoDB uses schema-less documents. This is an example of a course document in our database:

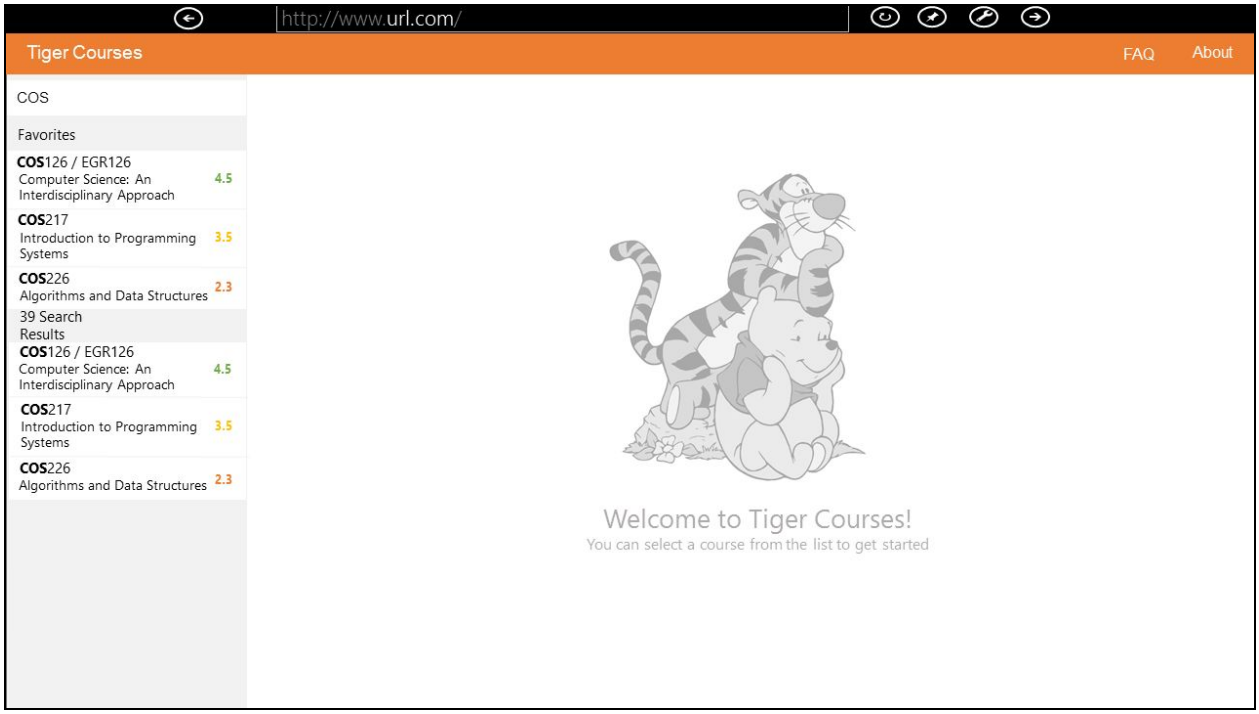


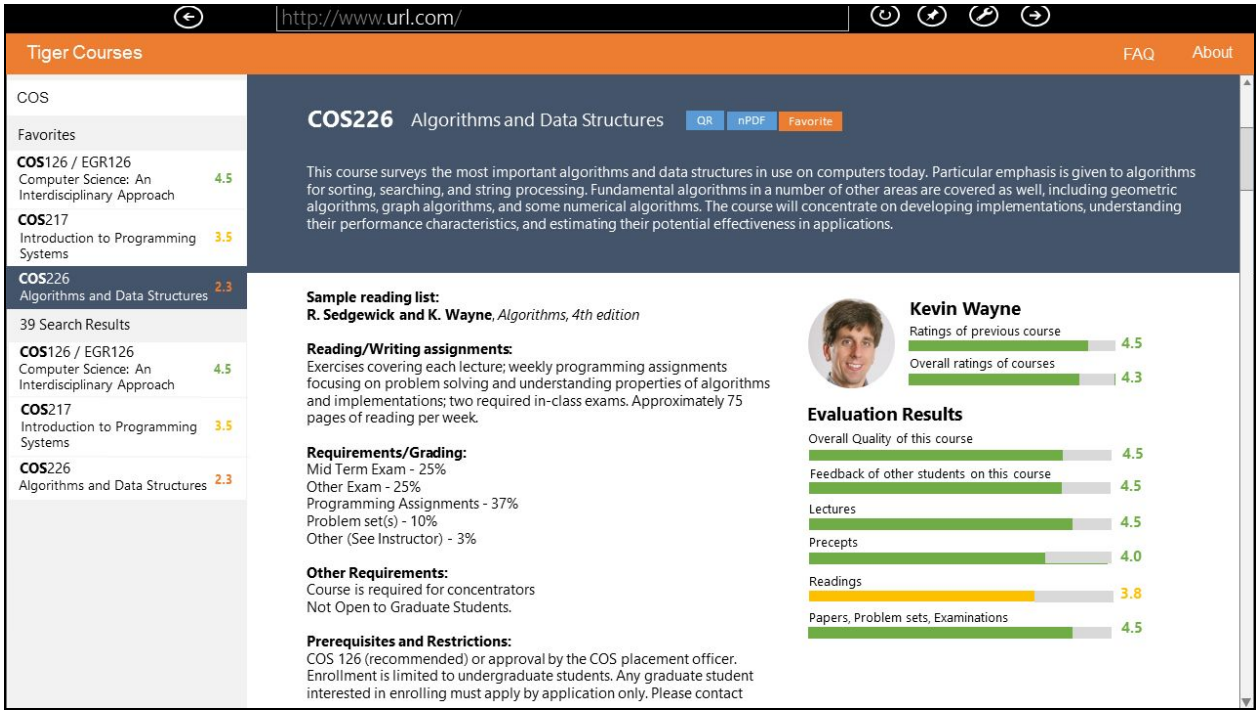
The data in our database has been scraped from the Registrar by our scripts.

Client

A front-end will feature a system built using HTML (generated by the EJS templating library), CSS (enhanced by Bootstrap), and JavaScript (jQuery). We will be featuring two pages with our app – (1) our splash page and (2) our main page. The splash page will be a general greeting page, allowing for the user to login to his/her Princeton netid and password via CASS. As well the splash page will allow the user to first start querying for course information.
The main app page will feature most of the content in a three pane display as basically outlined below. Students comments and various class rankings will be displayed prominently, as will overall course information (consisting of what the Princeton Registrar already includes).







Timeline

We already have a basic implementation of our website running live. This makes us optimistic about our ability to further develop Princeton Courses.

Pre-Design Document

- Server, git, and database all running properly
- Scripts for importing data via web scraping working
- Core server functionality (models, CAS authentication, request routing, etc…) working
- Front-end design started, search & display courses

Sunday 19 March

- Design document submitted
- Project name and domain decided, bought, migration complete
- Separate development and production systems set up
- Front-end design complete
- Front-end architecture decided (How will the Javascript work?)

Sunday 26 March

- Front-end design implemented for core features
- Course clash detection algorithm designed
- Course evaluation text analysis planned
- Google Analytics installed on production
- Project status website running on a subdomain

Have core features stable and in production anticipating course offerings for Fall 2017 being released by the Registrar on or after 6 April 2017. Share Princeton Courses with friends for alpha testing.

Sunday 2 April

- Solicit feedback, bug reports, and advice from alpha testers
- Course clash detection algorithm implemented
- Course evaluation text analysis in progress
- Front-end connected up with course clash detection algorithm and course evaluation text analysis features

If alpha testing is going well, broadcast Princeton Courses more widely for beta testing.

Sunday 9 April – Onwards

- Fix bugs reported by users
- Implement user feedback
- Course evaluation text analysis implemented
- Based on user feedback, explore implementing “stretch goals” such as integrations with ReCal and Princeton Pounce.

Demo Days: 8, 9 and 10 May

Submission: 14 May

Risks and Outcomes

To reduce the risk of not having an actual product, we've designed our project such that it is essentially a beautified version of the existing course evaluations initially, and gradually build on small features that each provide an independent benefit. This is meant to avoid having a "big-bang" style project.

One of our most exciting features that we wish to element is sentiment analysis of reviews to provide users with a numerical snapshot of what a course's written reviews said. However, this will inevitably involve using a pre-existing package, and if no good ones can be found or modified, writing up a rough algorithm ourselves, which will involve learning a fair amount of tone in natural language, specifically as it applies to reviews.

Performance: Scraping the course evaluations take over an hour for the current algorithm that we have. Although this process is done only when we want to include new evaluation data, the process time should be considered.

Sustainability: We should also consider the sustainability and upkeep of this project, in the event that Princeton students find our tool to be useful.