Design Document for COS 333 Project

Noah Apthorpe, Garrett Disco, Luke Paulsen, Jocelyn Tang, and Natalie Weires

March 10, 2013

1 The Group

Project name (tentative): "Course Aggregator Tool" (CAT)

Project manager: Garrett Disco

Noah Apthorpe apthorpe@princeton.edu
Garret Disco gdisco@princeton.edu
Luke Paulsen lpaulsen@princeton.edu
Jocelyn Tang jmtang@princeton.edu
Natalie Weires nweires@princeton.edu

2 Overview

The Course Aggregator Tool is an online platform for viewing and querying Princeton course information available at various OIT and Registrar sites. This will help Princeton undergraduates make course selection and scheduling decisions, including long-term course planning. The site is different from other tools such as ICE and the Registrar site in that it will allow easy visualization of all available data on a course, make it easier to compare and rank courses, and provide historical data on older semesters of a course, but it is not designed to provide calendar-style scheduling the way ICE is.

CAT will provide a graphical interface that will make it easy for users to get a sense of what a course is like and select what information they want to see. It will also provide a way to compare two (or possibly more) courses side-by-side. The course data being displayed will include: professors, class size, class times, ratings in each category, written course reviews, P/D/F status, and textbook costs. It will be possible to search for, rank, and filter courses by any of these categories, either over the whole university or by department or distribution

requirement. We will also include a system for rating course reviews "up" or "down" for its helpfulness.

CAT will be built with Python/Django and hosted on Amazon AWS. We will scrape course data periodically (e.g. once per semester) from the Registrar website and store it in a Mongo database. We are still deciding on what to use for the frontend, but it is likely to involve heavy use of Javascript/CSS and/or tools built on top of them.

3 Functionality

CAT has several different possible use cases, all of which involve a comparison of multiple courses.

Use case 1: A user wants to find a class fitting a particular kind of description (e.g. "find a large LA class that is well-reviewed and offered after 11 AM"). The user goes to the CAT homepage, which has a prominent search bar in which all of these parameters can be put in (either a selection of fields, or a single text bar with smart interpretation, or both). Once the user has entered the search parameters, a results page is displayed with the best matching courses, from which the user can select one or more courses to view the full data for.

Use case 2: A user is only considering a few courses (e.g. COS 333 and COS 402) and wants to compare their ratings in detail. The user opens both courses in the full-data view mode (by typing their codes into a search bar) and compares them side-by-side in terms of lecture rating, assignment rating, and overall course recommendation rating (all of these can be selected to display in the full-data view mode).

Use case 3: A user is browsing courses and is considering what to take in future semesters. The user puts in very general search parameters (e.g. all COS classes) and looks at classes from the result list one at a time. When viewing courses, the user uses the "timeline" option, which brings up the current (or most recent) semester of the course as well as past semesters. Then user can then view how course ratings, class size, readings, and so on have changed over time.

4 Design

5 Milestones

1. Set up site on AWS (i.e. have "Hello World" working)

2. Set up backend on AWS, send data back and forth
3. Get in contact with OIT / Registrar
4. Define database structure (all categories we will need)
5. Scrape one semester worth of course data
6. Display single semester data in bare-bones format
7. Scrape multiple semesters and display all at once
8. Search, filter, and rank by any category
9. Decide on frontend design specs for course view (move earlier?)
10. Decide on frontend design specs for search view

    [No particular order from here on out]
11. Implement frontend design specs for search view
12. Implement frontend design specs for course view
13. Implement ranking system for course reviews
14. Implement textual analysis of course reviews
15. Implement future semester predictions
16. Implement CAS login wall
17. Deploy for alpha test
18. Deploy for beta test
19. Deploy for demo

6 Risks

7 Timeline