Design Document for COS 333 Project

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

March 12, 2013

1 The Group

Project name: "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. 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 more courses side-by-side. The course data displayed will include: professors, class size, class times, ratings (in various categories), written course reviews, P/D/F status, distribution area, and textbooks. 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 a course review "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 how to set up the frontend, but it is likely to involve heavy use of Javascript-based tools and Ajax.

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 logs in to CAT, which displays a prominent search box where all of these parameters can be entered (either a selection of fields, or a single text bar with smart interpretation, or both). Once the user has entered the search parameters, the best-matching courses are displayed, 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 (by typing the course code into the search box, then opening another search box and repeating the process) and compares them side-by-side with respect to those attributes that they find relevant (e.g. lecture ratings, overall ratings, enrollment numbers, etc.)

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 activates the "timeline" option, which brings up past semesters of the course as well as the most recent semester. Then user can then view how course ratings, class size, readings, and so on have changed over time.

Core functionality that will be required:

- Search for and rank courses by one or more categories
- Display full data for one or more courses from search results
- Add additional boxes to search for and view courses in
- Extend a single semester to a timeline for that course
- Choose which data to display for courses

Additional features we are considering include:

- Display textbook costs
- Predict what future semesters of a class will be like
- Allow student course reviews to be rated up or down
- Include prerequisites and certificate requirements
- Analyze text of course reviews for key terms, positive versus negative, etc.
- Maintain favorite courses for each student

4 Design

We have decided to use MongoDB for the database because a lot of the data is irregular - for example, classes have different numbers of sections, multiple professors, etc. Each class is a document with fields for each feature: professor, sections, ratings and reviews, pdf status, distribution category, sample readings, grading breakdown, class number, pre-requisites, course description.

Our primary source of data is the registrar. We plan to scrape the course offerings website as well as the ratings and reviews, which come from applyweb.com. Much of the data is also available in XML format from an OIT web feed.

We plan to use the Django framework to structure the website. It provides a powerful and well-organized system for accessing the database and sending information to the client.The Django view functions will be used as the interface for interacting with the client side, and we will use the pymongo package (or possibly a more specific Django-Mongo engine) to retrieve information from the database.

We plan to use some kind of Javascript (most likely jQuery) for the frontend. This will let us provide an intuitive and interactive user interface, while accessing server-side data efficiently using Ajax requests.

5 Milestones

Already complete:

- Set up minimal Django site on localhost
- Get in contact with OIT and Registrar

- Produce basic frontend design

For 3/22:

- Set up site on AWS or OIT, be able to send data
- Set up MongoDB, be able to send data
- Produce a minimal scraper (e.g. get 1 piece of data for a given course)
- Define frontend API (what does Django need to send and what does Javascript take care of?)

For 3/29:

- Have a minimal site (Django + Mongo) up on AWS
- Define database structure (all categories we will need)
- Scrape all relevant data for any given course
- Get a minimal version of the frontend working (Javascript or similar communicating with Django)

For 4/5:

- Display single-course data on frontend
- Scrape data for many courses automatically
- Search, filter, and rank courses by any category
- Decide on final frontend design specs

For 4/12:

- Implement CAS login wall
- Display search bar and results on frontend
- Scrape data for all courses, including corner cases
- Project prototype ready

For 4/19:

- Implement interactive stuff (tabbing and choosing which data to view) on frontend
- Open multiple course boxes on frontend
- Test system's protection against basic attacks
- Store user data on backend

For 4/26:

- Finish frontend navigation
- Implement additional features / categories
- Deploy for alpha test

For 5/3:

- Evaluate alpha test feedback
- Finish frontend styles / graphics
- Finish adding features
- Deploy for beta test

For 5/10:

- Finalize software
- Produce demo presentation

6 Risks

- Getting access to data: We may have to meet with OIT or the registrar to scrape the ratings and reviews for courses, since this data requires login information.
- Changing course numbers: In the case that departments change course numbers, we need to be able to track this for each semester. We can do this by comparing numbers of classes that have the same names over the course of the semester.
- Integration: Interaction between Django and Mongo is not as throughly structured or documented as it is for relational databases like MySQL. Some experiementation will probably be needed to get this to work.

7 Timeline

Our timeline is posted at
`http://www.princeton.edu/~nweires/timeline.html`.

The timeline (and the rest of our project) can also be viewed on Github at
`https://github.com/lpaulsen333/cat`.