Princeton Courses Product Guide

User Guide

Overview

Twice a year, every Princeton student spends hours planning their courses for the following semester. Course selection choices are incredibly important: choosing well can make a semester fun, intellectually stimulating, and exciting, while choosing poorly can result in boredom, academic struggles, and disengagement.

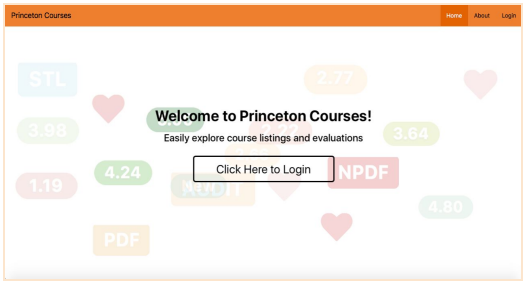
Princeton Courses is the best way to discover and learn about courses offered at Princeton. The website offers users effortless, precise searching and, for every Princeton course, the most comprehensive collection of information about the course's details, schedule, evaluations, student comments, and history.

Access Princeton Courses at https://www.princetoncourses.com.

Using Princeton Courses

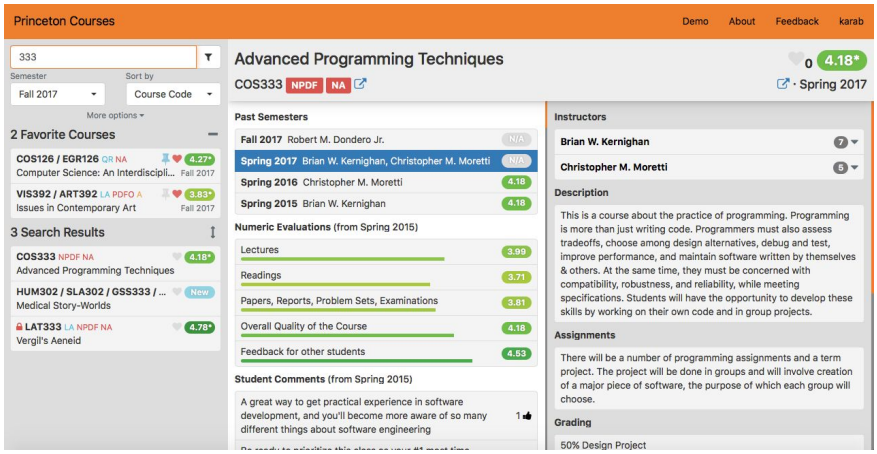
Splash Page

The splash page provides description of how Princeton Courses works and invites users to login through the Princeton University Central Authentication Service.



Main Application Page

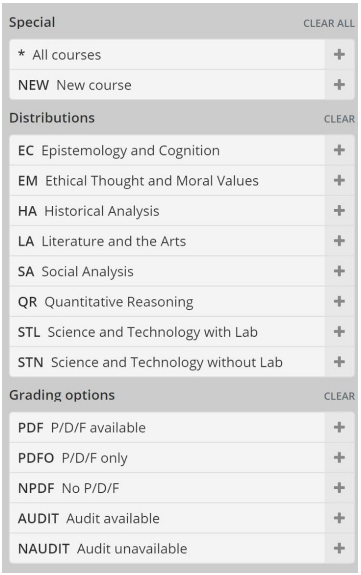
Once logged in to Princeton Courses, users can search for courses or browse any courses they have previously saved as favorites. Further help can be found through the Demo and About links.



Search

Courses and instructors can be found using the search bar or the search suggestions pane. Search results appear and update immediately. The following are valid search terms:

- Distribution areas (i.e. *EC*, *EM*, *HA*, *LA*, *SA*, *QR*, *STL*, *STN*)
- Grading options (i.e. *PDF*, *PDFO*, *NPDF, AUDIT, NAUDIT)*
- Course level (i.e. *1XX*, *2XX*, *3XX, 4XX, 5XX, UGRD*, *GRAD*)
- Departments (i.e. *COS*, *ELE*, *ECO*, *MAT*, *SOC,* etc…)
- Course number (i.e. *333*, *126*, *340*)
- Course title (i.e. *western way of war*, *hustlers, info sec,* etc…)
- Instructor name (i.e. *moretti*, *Brian Kernighan*, *katz*, etc…)



If multiple distribution areas, departments, or course levels are specified, the union of these results will be returned.

The filter icon in the search box toggles the Search Suggestions pane (right), which illustrates the variety of search parameters permitted. Clicking a suggestion searches for this term, while clicking the plus icon appends the term to the current search.

Search results can be filtered by semester and sorter based on different parameters using the drop-down menus below the search bar.

Viewing a Course

Once a course has been opened, a plethora of information about the course is presented. Princeton Courses displays information about cross-listings, grading options, instructors, assignments, grading, prerequisites, readings, the course's schedule, evaluation scores, student comments, and the course's history.

For courses that have not yet been evaluated, Princeton Courses will display the evaluations from the most recent semester for which the current instructor taught this course. If this instructor has never taught the course, the evaluations from the most recent semester for which evaluations exist will be displayed. Each student comment can be up-voted by clicking its thumbs-up icon. Student comments are sorted by the number of upvotes, ensuring that the most helpful comments are seen first.

Clicking on an instructor's name displays a clickable list of all of the courses the instructor has taught.

Favorite and Pinned Courses

Clicking a course's heart icon adds it to the user's Favorite Courses list, which is located above the Search Results, for easy access in the future. Clicking the pin icon on a favorite course causes search results to display a warning if a course in the search results has a time conflict with the pinned courses. Clashing courses can be excluded from search results through "More Options".

Developer Guide

Installation

Follow these instructions to deploy your own installation of Princeton Courses:

1. You must have Node.js and the Heroku CLI (command line interface) installed. If running both `node -v` and `heroku -v` return reasonable-looking version numbers, then these are correctly installed. Otherwise, go install these.
2. Clone the Princeton Courses repository to your computer by running `git clone https://github.com/sebthedev/PrincetonCourses.git`.
3. `cd` into your clone of the repository and run `npm install` to install the app's dependencies.
4. In this folder, create a file named `.env` which will be used to store environment variables. Add a new line to this file with the text `MONGODB_URI='DATABASE_URL'` where `DATABASE_URL` is replaced with the `mongodb://` address of your MongoDB server and database. If you don't have a MongoDB database already, you can host a database locally on your computer (instructions) or use a service like mLab. Save this file.
5. Populate your database by running the following commands:

```
node importers/importBasicCourseDetails.js; node importers/scrapeEvaluations.js
node importers/scrapeExtendedCourseDetails.js; node importers/importDepartments.js
node importers/insertMostRecentScoreIntoUnevaluatedSemesters.js
node importers/setNewCourseFlag.js
```

Backend

Princeton Courses is a Node.js-based web app currently run on the Heroku platform using a MongoDB database hosted with mLab. The app uses the Model-View-Controller architecture.

Models

Princeton Courses uses the object modeling system Mongoose to interact with our data. These models are built from schemas which ensure data consistency and facilitate attaching static and instance methods. The `models` folder contains models for courses, departments, evaluations, instructors, semesters, and users.

Model Population

Some models use population (similar to SQL joins) to embed documents of one type inside a document of another type and reduce data duplication. For example each course contains only the ID number of its instructors. When a course is fetched from the database, Mongoose replaces this field with the complete details of the instructor with the given employee ID.

Controllers

Control flow within the application is managed by the top-level `app.js` script and the scripts within the `controllers` folder. The script `app.js` is executed when the application is launched. This initialises Mongoose, Express (which manages HTTP requests), and connects to the database. On every web

request to the server, the app checks the user's authentication status, and loads the requested resource.

Page Requests

Requests to load web pages are passed directly to EJS for rendering.

Authentication

User authentication, via the Princeton University Authentication Service, is handled by `controllers/authentication.js`. Users login by visiting `/auth/login` and logout through `/auth/logout`.

API Requests

Authenticated HTTP requests to `/api/*` are handled by `controllers/api.js` and then passed to the relevant endpoint-specific script inside `controllers/endpoints`.

/api/semesters, /api/departments

Return all of the semesters or departments for which the database contains course information.

/api/course/:id, /api/instructor/:id

Return the data for the course or instructor with the specified `:id`.

/api/search/:query

Return an array of the courses and instructors matching the URL-encoded query string `:query`. The following optional URL parameters can be supplied to further modify the result:

- `semester` – the semester ID of the semester to which to restrict the search
- `detectClashes` – the strings "true" (default), "false", or "filter", indicating whether the results should contain information about whether each course clashes with the user's list of courses for detecting clashes (the clashDetectionCourses list). If the value is "filter", then any clashing courses will be excluded from search results.
- `sort` – a string representing the key on which search results should be sorted, such as "relevance" (default), "rating", "commonName"

/api/user/favorites

Return the list of the user's favorite courses.

/api/user/favorites/:id, /api/user/clashDetectionCourses/:id

Append to (for requests with HTTP verb `PUT`) or remove from (for requests with HTTP verb `DELETE`) the user's favorite courses list or clashDetectionCourses list the course with the ID `:id`.

/api/evaluations/:id/vote

Increment (for requests with HTTP verb `PUT`) or decrement (for requests with HTTP verb `DELETE`) the number of votes of the the evaluation with the ID `:id`.

Course Clash Detector

`courseClashDetector.js` determines whether a course can possibly be included in a student's schedule, given that the student definitely wants to take a supplied set of courses. This is used in search results and when adding a course to the favorites list.

Importing Data

Princeton courses has a collection of scripts in the `importers` folder that fetch course information from the Registrar's servers. Details of what each of these scripts do and when each script should be run can be found at `docs/importingData.md`. These scripts may require updating as the Registrar modifies their systems.

Front-End

The client-side components of Princeton Courses consists of HTML (generated by the EJS templating library), CSS (enhanced by Bootstrap), and JavaScript (jQuery) pages. Relevant HTML files (i.e. `app.ejs`) can be found in `/views/partials` and `/views/pages`, CSS files (i.e. `app.css`) in `/public/stylesheets`, and JavaScript files (i.e. `app.js`) in `/public/scripts`.

Independent modules

To promote modularity, many features common across different views are stored as partials in `/views/partials`. In particular, `header.ejs` holds the website's public-facing metadata and the implementation of Google Analytics, all elements common inside the head element of the pages.

Navigation bar `navbar.ejs, navbar.css, navbar.js`

Feedback `feedback.ejs, feedback.css, feedback.js`

Logout `logout.js`

These files handle the display and functionality of the navigation bar (conditionally with feedback and login/logout button) at the top of all pages (except the navbar in `app.ejs`). The feedback is sent to a Google Form, currently located here.

Splash page `splash.ejs, splash.css, splash.js`

The Splash page welcomes users and invites them to login through Princeton's Central Authentication Service. The introductory graphic was inspired by work on CodePen.

About page `about.ejs, about.css, about.js`

The About page explains Princeton Course's history and lists the project's team members. Original templating for About page comes from CodePen.

Main App page `app.ejs, app.css, app.js`

The main app page provides the core functionality of the website. It contains the search interface and displays the course information and evaluations. The page is divided into panes for search suggestions, searching, and displaying courses (including evaluations and course information). The code that initializes the functionality of the main app page can be found in the file `app.js`. The first screen that users see following authentication is the Welcome Page, at `welcomepage.ejs.`

Icons `icon.js`

HTML strings generated for favorite icons, rating badges, distribution/pdf/audit tags, course listings and locked icons are located here.

Search `search.js`

This script implements the search feature. It also contains the functions which return DOM elements for the search result corresponding to an instructor or a course, which are also used elsewhere (e.g. favoriting / displaying instructors in the info pane).

Search Suggestions `suggest.js`

This script handles the initial displaying of the suggest pane and the functionality of the suggest terms. It contains a list of departments and distributions that is used globally.

Favoriting and Pinning `fav.js, pin.js`

These scripts handle clicks of a favorite and pin icons (with the corresponding AJAX request) and the updating of the favorites list in the search pane. The page maintains a local list of all favorited courses in `document.favorites`, and all pinned courses in `document.pins`.

Display of Course Information `display.js, eval.js`

These two scripts handle the displaying of course information in the info pane and the eval pane. There is a helper function to fill each section of the display pane, which extracts the relevant information from a course object and inserts it into the corresponding section of the view pane.

Demo `app.ejs, demo.css, demo.js`

The walkthrough of our app is implemented using the open source Intro.js library. To change walkthrough content, edit `conductInitialDemo()`'s `steps` variable in `demo.js`.

History `history.js`

This script handles browser history using the standard History API. This allows users to move through Princeton Courses using their browser's forwards/back buttons.

Scrollbar `scrollbar.css`

Size and shape of the scrollbar can be edited within `scrollbar.css`. Our code structure for this functionality was implemented with the help of Michael Schwartz's tutorial on Webkit scrollbars.

Mobile/desktop compatibility `layout.js`

The script `layout.js` handles the layout for mobile and desktop, as well as transitions between the two. It also handles the default view of the main app page (displaying of the welcome page).

Desktop `resizable.js`

The desktop version allows for resizable panes using Rick Strahl's plugin. It was edited to resize with measurements in % and vh rather than px, and also to save the resized widths of the panes into `localStorage`.

Mobile

The mobile version allows for horizontal swiping between panes. This was implemented using nested Slick carousels. Intricacies involved with handling swipes are dealt with within the `layout_mobile` function in `layout.js`.