TigerPath

Product Guide

Richard Chu (PM)
Barak Nehoran
Daniel Leung
Adeniji Ogunlana

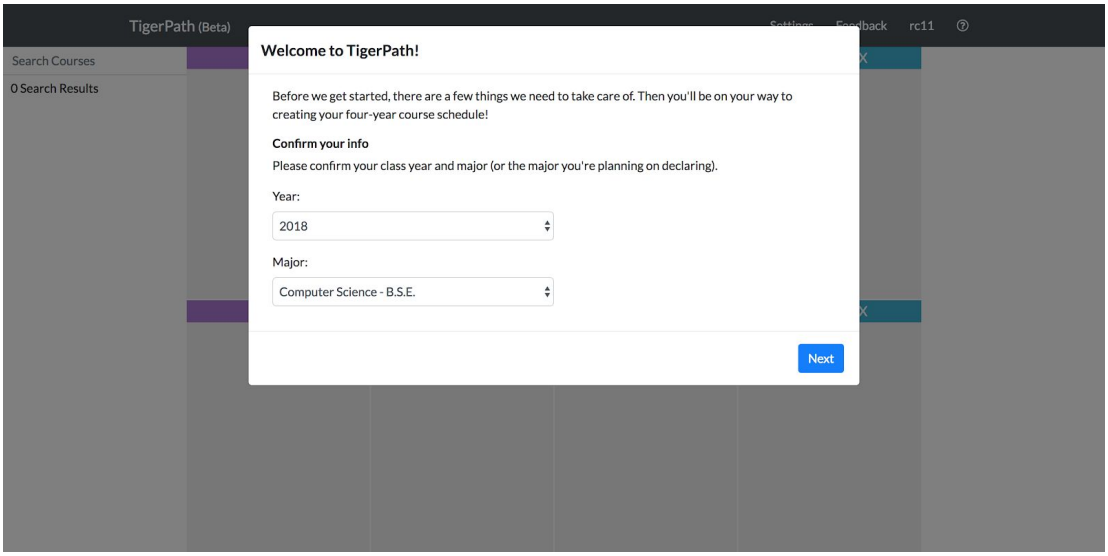
TigerPath User Guide

A. Getting Started

TigerPath is a web application that can be accessed at http://www.tigerpath.io. If this is your first time using the application, then you will be taken to the landing page where you can find out more information about the app. You can then login to the app using CAS authentication by entering your NetID and password.

B. Onboarding

On your first login, you will be greeted with a two-page onboarding screen. The first page asks you to confirm your class year and major, which are automatically populated with the information on your TigerBook profile.

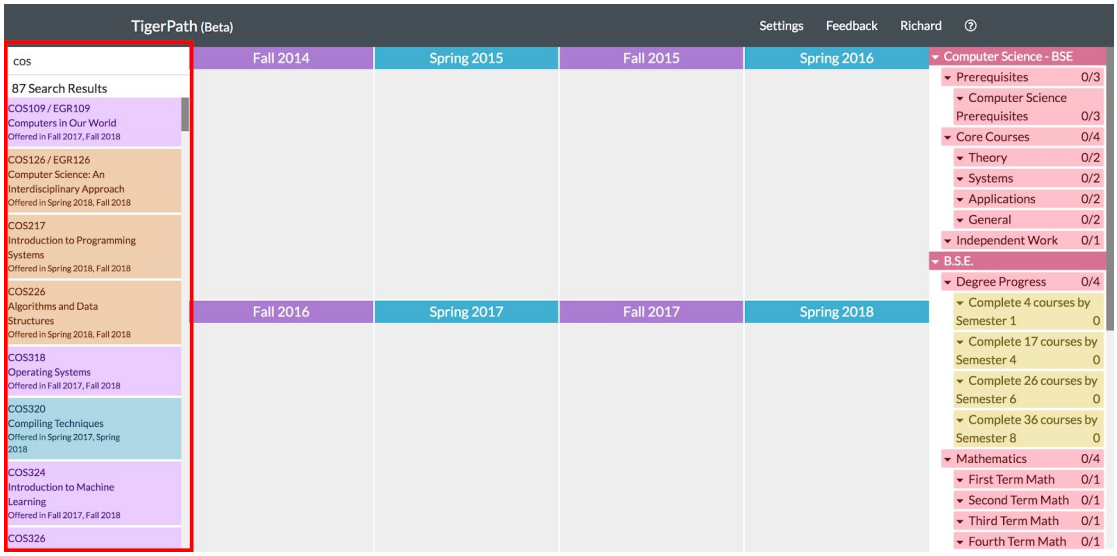


On the second page, you can optionally use the Transcript Service to prepopulate your schedule with the courses on your transcript. Regardless of whether you use the Transcript Service or not, after you finish the onboarding, you will be presented with a tutorial guiding you through the different sections of the app. During the tutorial, each section of the app is highlighted and paired with a brief description describing what the section is and how to use it. Once the tutorial is completed (or skipped), you can begin using our app.

C. Searching for Courses

You can use the search pane on the left to search for courses from a combined course list compiled from the last five semesters (with more semesters to be added soon). You can search by any combination of the three letter department code, the course number, and the course title. As you type in the search box, the app automatically updates the search results with a list of courses that match your query. The displayed information for each course includes the course code, the course title, and the semesters that the course was offered in. The courses are color coded based on the semester that they were previously offered in. Fall courses appear in purple,

Spring courses appear in blue, and courses that are offered in both semesters appear in orange. If you hover over a course in the search pane, links to the course evaluations and the course details on the Registrar's Course Offerings page are shown.

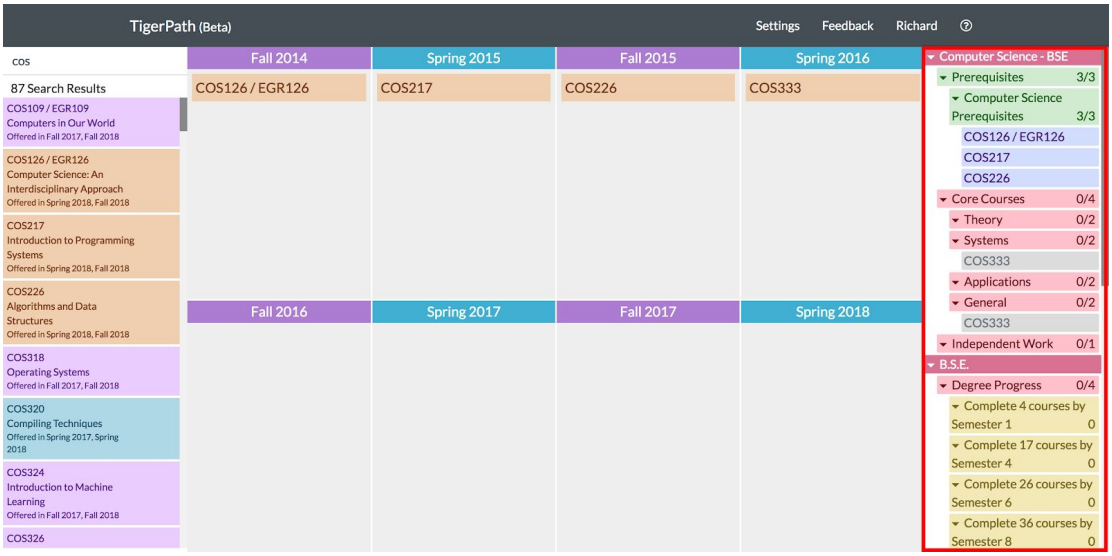


D. Adding Courses to Your Schedule

You can drag and drop any of the courses in the search results to one of the eight semesters in your schedule in order to add it. By hovering your cursor over a course in your schedule, you can see relevant information about it, such as the full course title and whether or not it already exists in the schedule. The courses can be rearranged within a semester and between semesters by dragging and dropping. You can delete a course from your schedule by hovering over it and pressing the "x" button.

E. Requirements Pane

TigerPath shows the requirements for your major and your degree (AB or BSE) in the requirements pane on the right side. In this pane, you can view the specific sub-requirements you've satisfied and have yet to satisfy for your major and your degree. The headers are shown in three different colors to indicate the status of the particular requirement in relation to your proposed four-year course schedule. Red headers indicate incomplete requirements, yellow headers indicate optional requirements, and green headers indicate completed requirements. The information in the requirements pane continually updates as you change the courses in your schedule, and it automatically puts a course underneath the requirement that it satisfies. However, when a course can satisfy multiple requirements, it will show up in gray underneath each of these requirements. You should then click on the course underneath the particular requirement that you want it to satisfy.



F. Settings Page

You can change your nickname, your class year, or your major in the settings page, which can be accessed by clicking on “Settings” in the navigation bar. Changing your major in the settings page allows you to see what requirements in another major that the current courses in your schedule would satisfy, and could help inform you on whether you can switch your major or not. In terms of showing requirements, we currently fully support Computer Science (AB and BSE), Electrical Engineering, and the Woodrow Wilson School.

On the Settings page, you can also use the Transcript Service to upload your transcript and populate your course schedule with courses you’ve already taken. This is helpful in case you initially skipped this step during the onboarding page. Note that using the Transcript Service will clear out your current four-year course schedule; this action cannot be undone.

TigerPath Developer Guide

A. Getting Started

Learning how to set up your environment to develop, run, and deploy TigerPath is detailed very thoroughly in the README file of our GitHub repository. In this developer guide, we will describe the relevant parts of the code that are used to implement the main functionality of our app.

B. Data Collection

1. Requirements `(tigerpath/majors_and_certificates/)`

We've defined a data structure to encode each major in a JSON file which our app parses through to display the requirements tree. You can find more information about how the JSON requirement files are structured in the README of the `majors_and_certificates` directory in our Github repository. We had to manually make these JSON files for each major that our app supports, but to help streamline the JSON creating process, we also created an HTML form that automatically generates a major JSON; this can be found in `majors_and_certificates/scripts/JSON_creator.html`.

2. Course Scraping `(tigerpath/scraper/)`

We adapted code from ReCal's scraper and COS 333's Assignment 4 scraper. We first scrape for course information using ReCal's scraper, which gets its information from the OIT Web Feed Registry. However, the Web Feed doesn't provide some data, such as the distribution areas that each course satisfies; to solve this, we also ran the Assignment 4 scraper, which scrapes directly from the course offerings page and adds the distribution areas to the existing courses scraped by ReCal.

In addition, we added a list of cross listings to each course so that it would be more efficient to search for courses, as getting cross-listed courses through foreign key relationships turned out to be too inefficient and required too many database queries. We also modified the scraper to maintain a single model for each course instead of making new models for each course for every semester. This makes course searching more efficient because it doesn't have to search through copies of the same course in different semesters.

When a new semester of courses is released, running the scraper for the new semester updates the relevant information for each course (e.g. the course description and the cross listings). It also adds a new semester to the semester field, which holds all of the semesters a course was offered in. This semester field is used on the frontend to inform the user of the most recent two semesters a course was offered in.

C. Backend Systems

1. Database Table Structure `(tigerpath/models.py)`

Using ReCal's scraper, our courses table had Course models with these foreign key relationships: Semester ← Course ← (Professor, Sections ← Meetings, Course Listings). A lot of this information is not necessary for our application (like Professors and Sections), so we mostly worked with the Course model. We moved some information from other models to the Course model as well for convenience and efficiency. The relevant information for each Course model include: title, description, registrar_id, dist_area, all_semesters, is_master, cross_listings.

We also created a UserProfile model to store user data relating to our application. We decided to store our application data in a separate model than the User model because Users are handled by Django and django-cas-ng. The UserProfile table stores data such as the user's nickname, their major, their class year, some user_state (such as whether or not they have completed onboarding), and the user_schedule (which holds their four-year course schedule).

2. Course Search `(tigerpath/views.py)`

Our course search algorithm takes a search query as a parameter, splits it by spaces, and checks to see if each subquery matches a department code, course number, or course title. After we match a subquery on one of these three categories, we then narrow down the set of courses based on the subquery. After going through all the subqueries, we finally sort the resulting course list by the department and course code before returning the list.

3. User Schedule/Requirements

`(tigerpath/majors_and_certificates/scripts/verifier.py)`

We have a field in the UserProfile model to hold the user's schedule. Whenever a course is added on the frontend, a POST request is sent to the backend updating the user's schedule in the UserProfile model with the new course. The verifier script is also run which recalculates what requirements are satisfied and sends an updated requirements JSON to the frontend, which displays the JSON as a tree.

The verifier works by running a simplified version of the Ford-Fulkerson algorithm to find the largest flow from the leaves of the tree (the courses) to the root (the full major requirement). If a course only satisfies one requirement, then the course is automatically added to a "settled" list in the corresponding requirement. Courses in this list will be displayed in the tree in light blue. If the verifier sees that a course could satisfy more than one requirement, it adds the course to an "unsettled" list in all of the requirements that the course could satisfy. The frontend then displays courses in "unsettled" lists in grey. The grey course signifies to the user that he/she should click on it to remove the course from all other "unsettled" lists and "settle" that course underneath the chosen sub-requirement.

D. Frontend Systems

1. ReactJS `(frontend/src/Search.js, frontend/src/Requirements.js)`

Our interface had certain components that had to be constantly updated, like the course searching pane and the requirements pane, so we used ReactJS to help us efficiently update and

render these components. Whenever the search query is changed, React automatically renders the new search results, and whenever a course is added to the user's schedule, React renders the new updated requirements tree.

2. Dragula `(frontend/src/Search.js)`

The drag and drop interface is implemented with the Dragula library, which made things much simpler than trying to implement dragging and dropping manually. We selected the containers that we wanted to use with dragging and dropping, and its children elements automatically became draggable. It took some additional code to implement more specific functions like one-way dragging for courses from the search pane to the schedule, however.

3. Tree View `(frontend/src/Requirements.js)`

We used the react-treeview library to display the requirements pane. Using a recursive function, we parsed through the requirements JSON object returned by the backend and created a tree with nodes at each level. The react-treeview library allowed us to easily customize each label of the tree to include any additional information such as the requirement count.

E. Frontend-Backend Integration

1. AJAX

We used AJAX to send and receive data between our frontend and backend. For search queries, the frontend sends a GET request, the backend filters courses based on the query, and the filtered list is returned to the frontend. Whenever the user drags a course onto their schedule, we send a POST request to update their schedule in the backend. We then follow up with a GET request to run the new schedule through the requirements verifier, and then we display the verifier output on the frontend.

2. API `(tigerpath/urls.py)`

We implemented a RESTful API for our application. We have API urls for getting courses for a search query, updating a schedule, retrieving a schedule, and retrieving requirements.