– Independent Work Report Spring 2024 –

hoagieplan | Dashboard
Princeton's Comprehensive Academic Planner

George Chiriac
Adviser: Robert M. Dondero

Abstract

The hoagieplan project[1][2] represents a comprehensive approach to academic planning at Princeton University, aimed at consolidating multiple functionalities into a single, user-friendly application. This report details the development and implementation of the Dashboard tab within hoagieplan, which allows users to plan their academic curriculums over four years. Developed in response to identified limitations in existing academic planning tools, hoagieplan integrates and enhances the features of its predecessors—TigerPath, Princeton Courses, and ReCal—providing comprehensive support for degree, major, and minor planning. The Dashboard caters to semester planning by providing course and requirements information and checking which requirements a student fulfills. This report outlines Dashboard's architecture, functionalities, and user feedback, highlighting its successful deployment and potential for future enhancements. Through hoagieplan, Princeton students receive streamlined and effective support, reducing the complexity and time associated with academic planning.

[1] hoagieplan link.
[2] GitHub repository link.

Contents

1. Introduction

1.1. Motivation

Major and Degree Planning

Academic planning at Princeton University is hard. Students must parallel plan their coursework in order to satisfy their major and degree requirements. Bachelor of Arts (AB) students must take a writing seminar, complete one to four terms of foreign language, and take courses that fall into each of the following distribution areas: Culture and Difference, Epistemology and Cognition, Ethical Thought and Moral Values, Historical Analysis, Literature and the Arts, Quantitative and Computational Reasoning, Science and Engineering, and Social Analysis[3].

Bachelor of Science and Engineering (BSE) students must take four terms of mathematics, two terms of physics, and one term each of chemistry and computer science. They must also take a writing seminar and a minimum of seven courses distributed among the humanities and social sciences distribution areas.

In addition to degree requirements, students must fulfill their major requirements, which typically present themselves as lists of required courses within their department and lists of elective courses. These can be very difficult to keep track of since there often are multiple pathways through a major. Additionally, restrictions exist such as deadlines for fulfilling requirements and constraints on the grading bases of the courses taken.

Minor and Certificates Planning

Princeton undergraduates can supplement their diploma with one or more certificate programs. The University also introduced minor programs for the Class of 2025 and beyond. Minor and certificates requirements are similar to major requirements but fewer in number. Currently, there is no application to help students plan their minor and certificate programs.

[3]Adapted from Princeton's Undergraduate Announcement

Resolving Conflicts

Students must also pay attention to the classes' time schedule in order to avoid conflicts between the classes they pick. Thus, students must survey at least five different websites to decide which courses to pick each semester: the Undergraduate Announcement for the degree requirements; the major department's website or handbook for major requirements; the minor department's website for minor requirements; the registrar's website to see available courses; and TigerHub's course planner to manage time conflicts. These websites are old and hard to navigate. Princeton students have created applications that circumvent this issue.

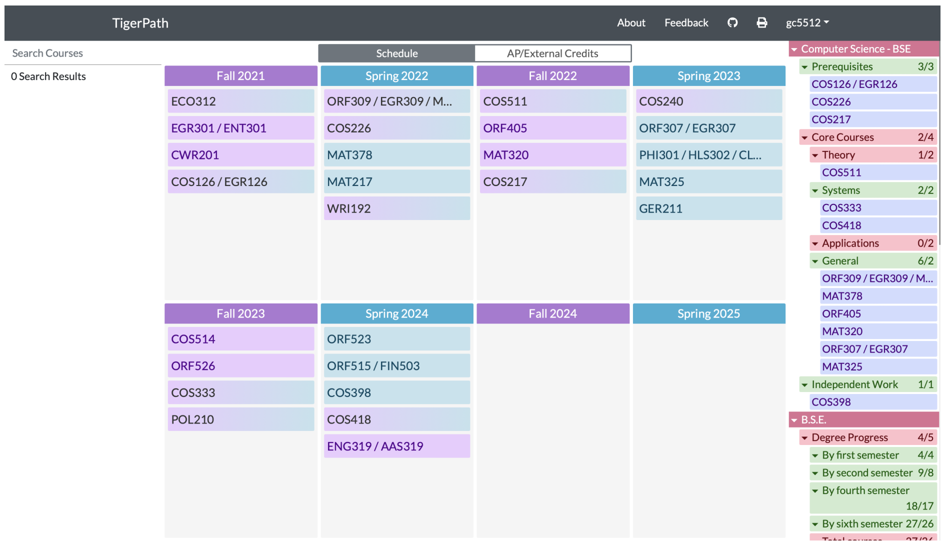
1.2. Related Applications

TigerPath

Released in 2018, TigerPath[4] is a four-year course planner designed to help students navigate their major and degree requirements. The layout of the application is shown in 1. Students can search for courses, drag and drop them into their semester schedule, then see which major and degree requirements they do and do not satisfy.

TigerPath's last major update was in 2021 so as of the time of writing this report, its requirements data is outdated. Furthermore, it does not have minor and certificates support. Lastly, the UI / UX of the application does not look modern.

[4]TigerPath link.



Fall 2021	Spring 2022	Fall 2022	Spring 2023
ECO312	ORF309 / EGR309 / M...	COS511	COS240
EGR301 / ENT301	COS226	ORF405	ORF307 / EGR307
CWR201	MAT378	MAT320	PHI301 / HLS302 / CL...
COS126 / EGR126	MAT217	COS217	MAT325
	WRI192		GER211

Fall 2023	Spring 2024	Fall 2024	Spring 2025
COS514	ORF523		
ORF526	ORF515 / FIN503		
COS333	COS398		
POL210	COS418		
	ENG319 / AAS319		

Computer Science - BSE	
Prerequisites	3/3
COS126 / EGR126	
COS226	
COS217	
Core Courses	2/4
Theory	1/2
COS511	
Systems	2/2
COS333	
COS418	
Applications	0/2
General	6/2
ORF309 / EGR309 / M...	
MAT378	
ORF405	
MAT320	
ORF307 / EGR307	
MAT325	
Independent Work	1/1
COS398	
B.S.E.	
Degree Progress	4/5
By first semester	4/4
By second semester	9/8
By fourth semester	18/17
By sixth semester	27/26

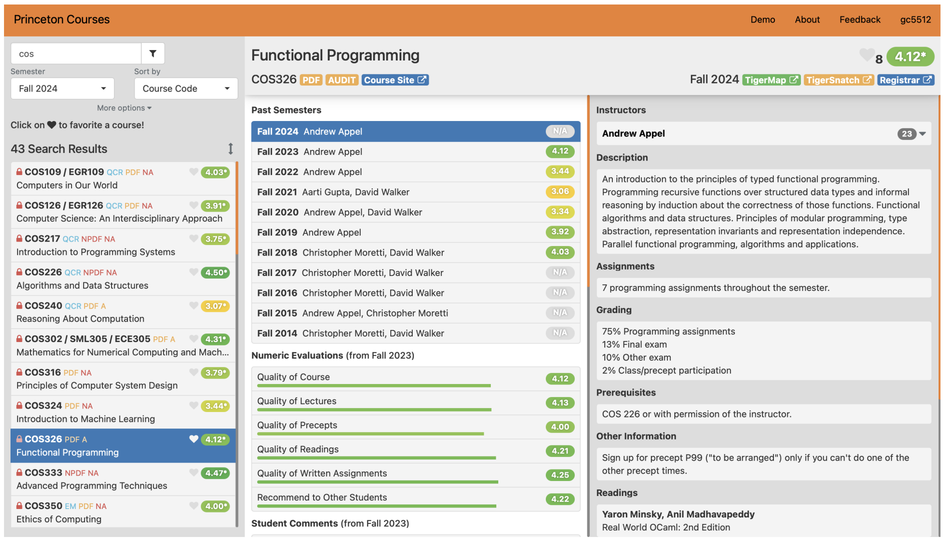
Figure 1: TigerPath layout. Left: course search. Middle: semester schedule. Right: requirements menu.

Princeton Courses

Released in 2017, Princeton Courses[5] displays the information offered by the registrar's website in a more intuitive and compressed manner. The layout of the application is shown in 2. Users can search for courses by title, course code, or department, and see the description, distribution area, instructors, and reviews of the selected courses on one page.

Princeton Courses is used by 45% of undergraduates, making it the most popular academic planning application at Princeton. Similarly to TigerPath, however, the UI / UX of the application is dated.

[5] Princeton Courses link.


Princeton Courses

Demo About Feedback gc5512

cos

Semester: Fall 2024

Sort by: Course Code

More options

Click on ❤ to favorite a course!

43 Search Results

COS109 / EGR109 QCR PDF NA — Computers in Our World	4.03*
COS126 / EGR126 QCR PDF NA — Computer Science: An Interdisciplinary Approach	3.91*
COS217 QCR NPDF NA — Introduction to Programming Systems	3.75*
COS226 QCR NPDF NA — Algorithms and Data Structures	4.50*
COS240 QCR PDF A — Reasoning About Computation	3.07*
COS302 / SML305 / ECE305 PDF A — Mathematics for Numerical Computing and Mach...	4.31*
COS316 PDF NA — Principles of Computer System Design	3.79*
COS324 PDF NA — Introduction to Machine Learning	3.44*
COS326 PDF A — Functional Programming	4.12*
COS333 NPDF NA — Advanced Programming Techniques	4.47*
COS350 EM PDF NA — Ethics of Computing	4.00*

Functional Programming

COS326 PDF AUDIT Course Site

8 4.12*

Fall 2024 TigerMap TigerSnatch Registrar

Past Semesters

Fall 2024	Andrew Appel	N/A
Fall 2023	Andrew Appel	4.12
Fall 2022	Andrew Appel	3.44
Fall 2021	Aarti Gupta, David Walker	3.06
Fall 2020	Andrew Appel, David Walker	3.34
Fall 2019	Andrew Appel	3.92
Fall 2018	Christopher Moretti, David Walker	4.03
Fall 2017	Christopher Moretti, David Walker	N/A
Fall 2016	Christopher Moretti, David Walker	N/A
Fall 2015	Andrew Appel, Christopher Moretti	N/A
Fall 2014	Christopher Moretti, David Walker	N/A

Numeric Evaluations (from Fall 2023)

Quality of Course	4.12
Quality of Lectures	4.13
Quality of Precepts	4.00
Quality of Readings	4.21
Quality of Written Assignments	4.25
Recommend to Other Students	4.22

Student Comments (from Fall 2023)

Instructors

Andrew Appel 23

Description

An introduction to the principles of typed functional programming. Programming recursive functions over structured data types and informal reasoning by induction about the correctness of those functions. Functional algorithms and data structures. Principles of modular programming, type abstraction, representation invariants and representation independence. Parallel functional programming, algorithms and applications.

Assignments

7 programming assignments throughout the semester.

Grading

75% Programming assignments
13% Final exam
10% Other exam
2% Class/precept participation

Prerequisites

COS 226 or with permission of the instructor.

Other Information

Sign up for precept P99 ("to be arranged") only if you can't do one of the other precept times.

Readings

Yaron Minsky, Anil Madhavapeddy
Real World OCaml: 2nd Edition

Figure 2: Princeton Courses layout. Left: course search. Middle: ratings and reviews. Right: course information.

ReCal

ReCal[6], released in 2014, is a semester planning app that allows students to visualize the times at which their courses take place and resolve potential conflicts. The layout of the application is shown in 3.

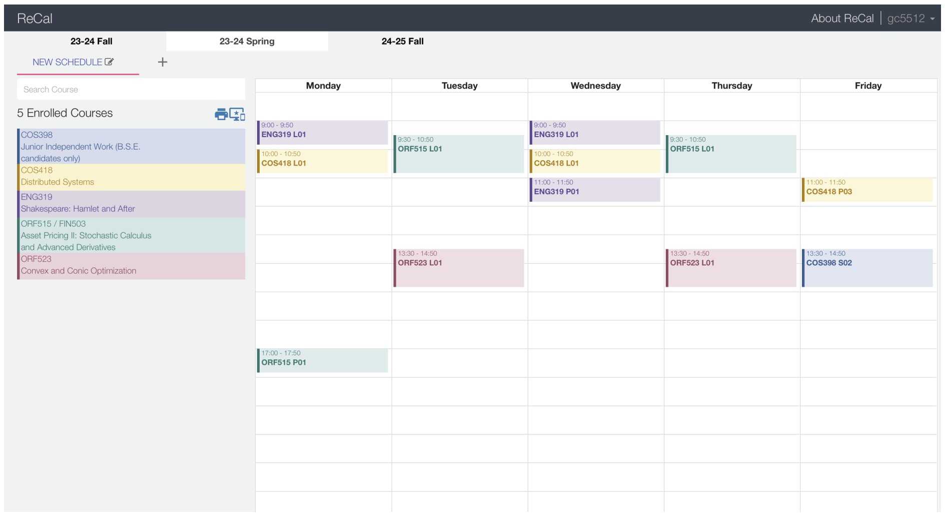


Figure 3: ReCal layout. Left: course search. Middle: calendar view.

ReCal+[7], released in 2023 as part of the TigerJunction project, is an updated version of ReCal

[6]ReCal link.
[7]ReCal+ link.

that introduces different color themes and advanced filters such as “Days” (that allows users to search for courses that have classes during the selected days) and “No Conflicts” (that allows users to search for classes that do not conflict with their current schedule). The layout of the application is very similar to ReCal and is shown in 4.

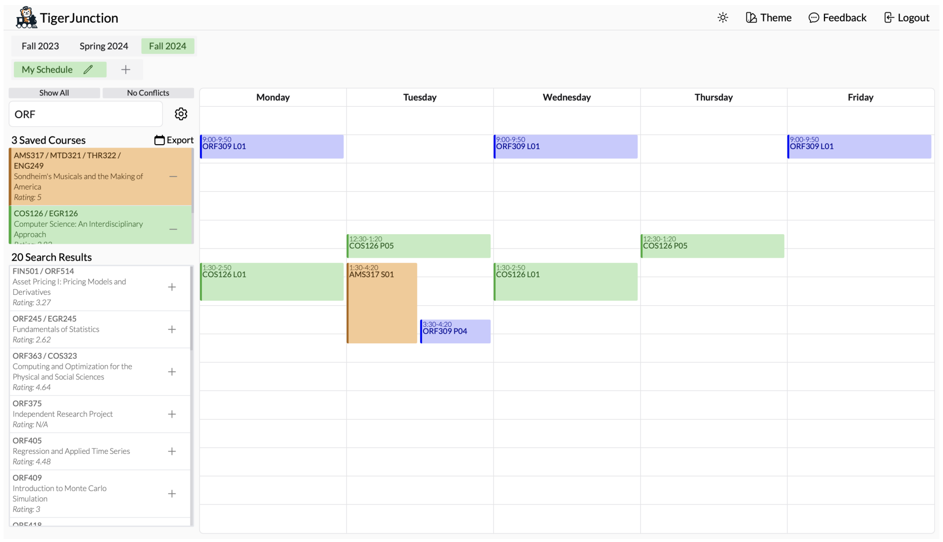


Figure 4: ReCal+ layout. Left: course search. Middle: calendar view.

Both of these applications look modern and are utilized by a large fraction of undergraduates.

Compass

In Fall 2023, I sent Princeton undergraduates an interest form for an application that helps users plan their minor requirements. It received 257 affirmative responses, confirming that Princeton students struggle planning out their minor programs. Furthermore, respondents expressed frustration regarding the need for multiple academic planning applications.

As a solution, I, Windsor Nguyen, Kaan Odabas, Ijay Narang, and Julia Kashimura set out to create an all-in-one academic planning app as our project for COS 333 – Advanced Programming Techniques in Fall 2023. We named it Compass[8]. Given the time constraints, we managed to implement most of TigerPath’s functionality, as well as Princeton Courses’ functionality through course information modals. The layout of the application is shown in 5. Compass had many limitations:

[8] Compass link.

- No support for cross-listed courses.
- Only had support for 6 majors, 30 minors, and no degrees or certificates.
- Did not check double-counting restrictions and requirement deadlines.
- No search filters, no delineation of course information and reviews by semester.
- Drag and drop behavior was slow and unstable.

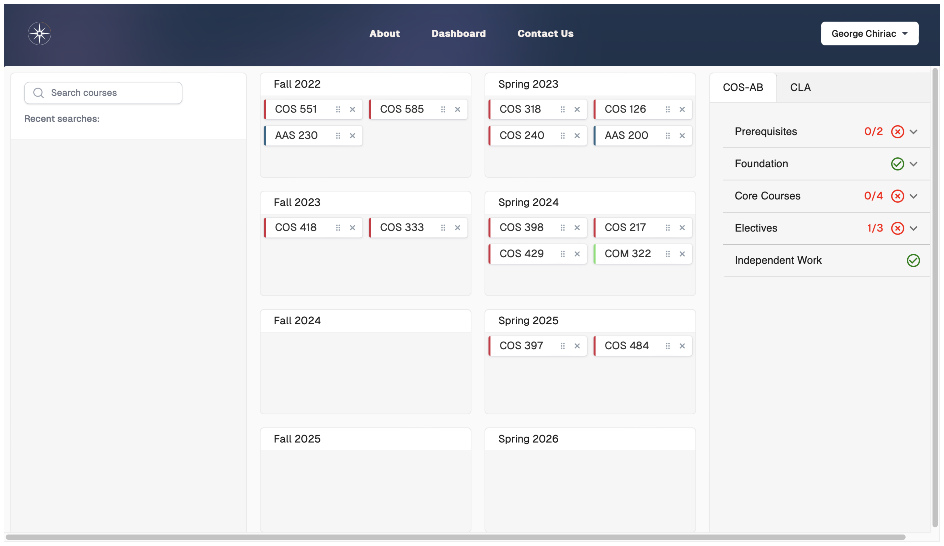


Figure 5: Compass layout. Left: course search. Middle: semester schedule. Right: requirements menu. Top right: account settings

1.3. Goal

hoagieplan is a continuation of Compass. Its name comes from hoagie.io, the application platform that hosts it. hoagieplan aims to modernize and integrate the functionalities of TigerPath, Princeton Courses, and ReCal. It is divided in two tabs, Calendar and Dashboard. Calendar, developed by Windsor Nguyen, is the tab where users visualize class times and manage conflicts. Dashboard, developed by me, is the object of this report, and it is the tab where users can plan their four-year schedule. The layout of Dashboard is shown in 6. The course information modals are the same as for Compass.

The goal of Dashboard is to resolve the limitations of Compass listed above. Specifically, users should be able to plan out any of the 2 degrees, 38 majors, and 38 minors. Furthermore, users should be able to refine their search using filters, see course crosslistings, and be informed about the courses that satisfy each requirement. Certificate programs will be withdrawn starting with Class of 2027 so, in the interest of time, I decided against certificate planning support.

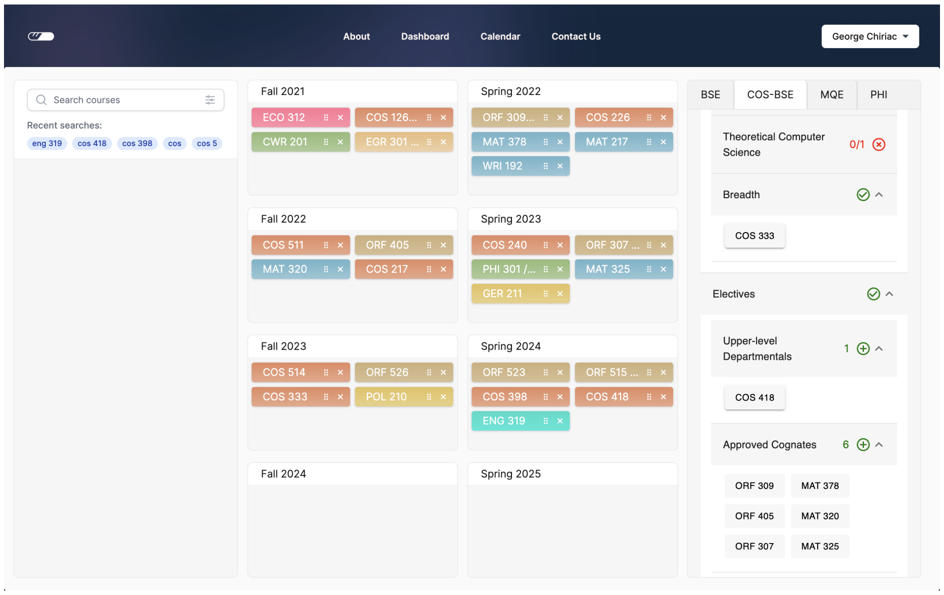


Figure 6: hoagieplan Dashboard layout. Left: course search. Middle: semester schedule. Right: requirements menu. Top right: account settings.

2. Functionality

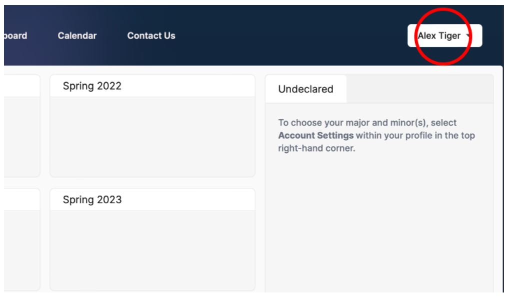
2.1. User Scenario

I will demonstrate hoagieplan's functionality by walking through a scenario users may find themselves in. Two more scenarios are listed in Appendix 6.

User Scenario – Alex

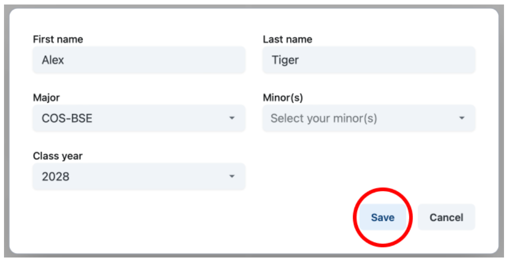
Alex is a freshman interested in machine learning and robotics. He is deciding between majoring in Computer Science (COS-BSE) or Electrical and Computer Engineering (ECE). He needs to make a decision and pick his courses for freshman spring.

- He clicks on his name on the top right section of the screen to open the Account Settings menu (7). He selects 2028 as his class year and first chooses COS-BSE as his major (8).



To choose your major and minor(s), select **Account Settings** within your profile in the top right-hand corner.

Figure 7



First name	Alex
Last name	Tiger
Major	COS-BSE
Minor(s)	Select your minor(s)
Class year	2028

Figure 8

- He scrolls through the COS-BSE requirements on the right of the screen. He clicks on the Artificial Intelligence and Machine Learning requirement (9) and presses the "Search Courses" button (10) to populate the search results with the courses that satisfy this requirement. He then clicks on the course cards on the left of the screen (11) to show more course information (12). He decides he wants to take COS 424 / SML 302, COS 429, and COS 484.

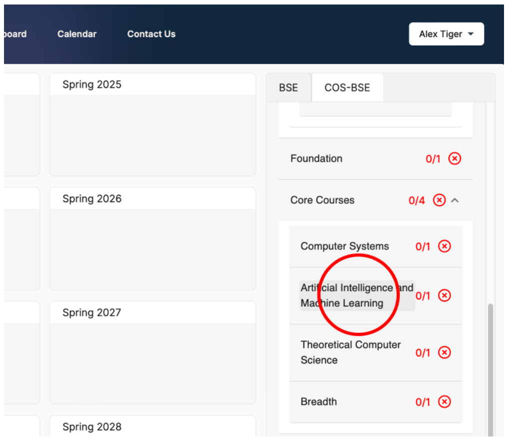
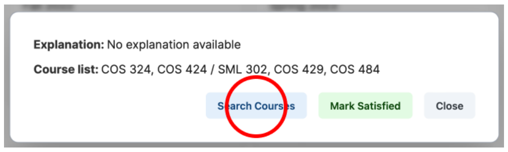


Figure 9



Explanation: No explanation available

Course list: COS 324, COS 424 / SML 302, COS 429, COS 484

Figure 10

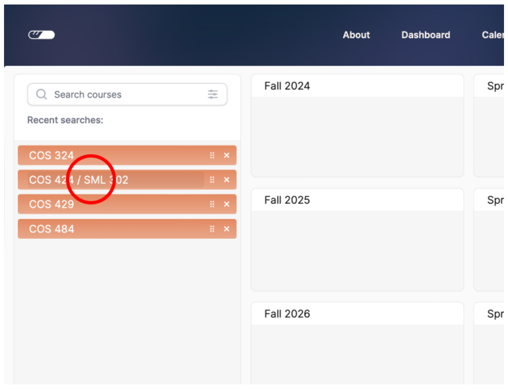
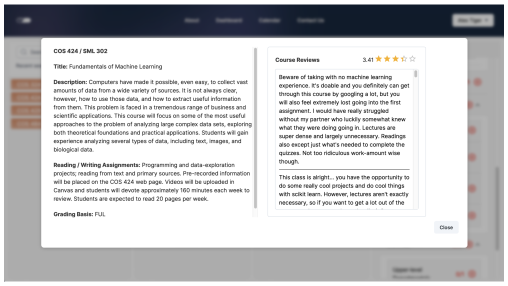


Figure 11



COS 424 / SML 302

Title: Fundamentals of Machine Learning

Description: Computers have made it possible, even easy, to collect vast amounts of data from a wide variety of sources. It is not always clear, however, how to use those data, and how to extract useful information from them. This problem is faced in a tremendous range of business and scientific applications. This course will focus on some of the most useful approaches to the problem of analyzing large complex data sets, exploring both theoretical foundations and practical applications. Students will gain experience analyzing several types of data, including text, images, and biological data.

Reading / Writing Assignments: Programming and data-exploration projects; reading from text and primary sources. Pre-recorded information will be placed on the COS 424 web page. Videos will be uploaded in Canvas and students will devote approximately 160 minutes each week to review. Students are expected to read 20 pages per week.

Grading Basis: FUL

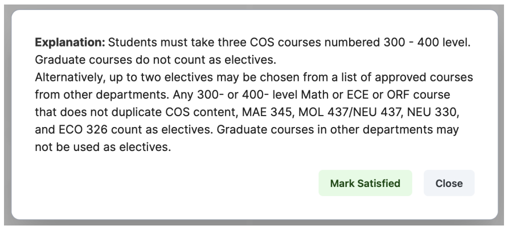
Course Reviews 3.41 ★★★⯪☆

Beware of taking with no machine learning experience. It's doable and you definitely can get through this course by googling a lot, but you will also feel extremely lost going into the first assignment. I would have really struggled without my partner who luckily somewhat knew what they were doing going in. Lectures are super dense and largely unnecessary. Readings also except just what's needed to complete the quizzes. Not too ridiculous work-amount wise though.

This class is alright... you have the opportunity to do some really cool projects and do cool things with scikit learn. However, lectures aren't exactly necessary, so if you want to get a lot out of the

Figure 12

- He clicks on the Electives requirement and sees that at most 2 ECE courses can count as COS electives (13).



Explanation: Students must take three COS courses numbered 300 - 400 level. Graduate courses do not count as electives.
Alternatively, up to two electives may be chosen from a list of approved courses from other departments. Any 300- or 400- level Math or ECE or ORF course that does not duplicate COS content, MAE 345, MOL 437/NEU 437, NEU 330, and ECO 326 count as electives. Graduate courses in other departments may not be used as electives.

Figure 13

- Then, Alex returns to Account Settings and chooses ECE as his major (14).

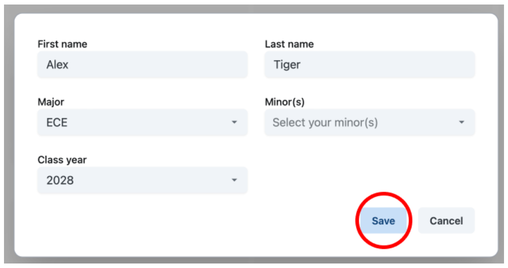
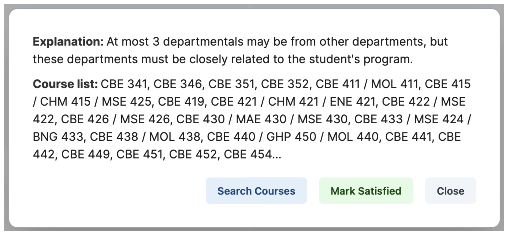

First name: Alex
Last name: Tiger
Major: ECE
Minor(s): Select your minor(s)
Class year: 2028

Figure 14

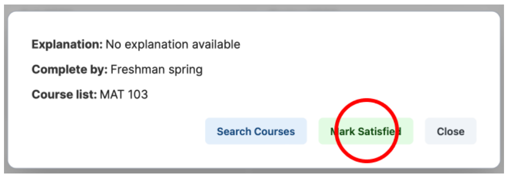
- He inspects the ECE requirements. The "Robotics and Cyberphysical Systems" is the track he finds most fit. He realizes he can take robotics classes for this requirement and can count COS 424 / SML 302, COS 429, and COS 484 toward the Balance and Completeness requirement (15).


Explanation: At most 3 departmentals may be from other departments, but these departments must be closely related to the student's program.

Course list: CBE 341, CBE 346, CBE 351, CBE 352, CBE 411 / MOL 411, CBE 415 / CHM 415 / MSE 425, CBE 419, CBE 421 / CHM 421 / ENE 421, CBE 422 / MSE 422, CBE 426 / MSE 426, CBE 430 / MAE 430 / MSE 430, CBE 433 / MSE 424 / BNG 433, CBE 438 / MOL 438, CBE 440 / GHP 450 / MOL 440, CBE 441, CBE 442, CBE 449, CBE 451, CBE 452, CBE 454...

Figure 15

- Alex decides to major in ECE, as it is more flexible to his interests.
- He scrolls through the BSE requirements. He has AP Credits in Mathematics, so he clicks on the First- and Second Term Math requirements and marks them as satisfied (16, 17). This leaves room for ECE classes in his spring semester.


Explanation: No explanation available

Complete by: Freshman spring

Course list: MAT 103

Figure 16

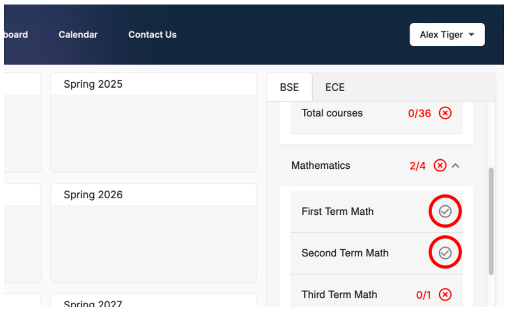


Figure 17

2.2. Selected Features

In this section, I list some features of hoagieplan Dashboard that I implemented this spring.

Comprehensive Requirements Checking

Degree, major, and minor requirements may present themselves as course lists ("Students must take one of these courses"), department lists ("Students must take a course from one of these departments"), distribution areas or course counts ("Students must take at least this number of courses"). Some requirements have deadlines:

"Computer proficiency is a requirement for the B.S.E. degree fulfilled by taking COS 126 General Computer Science or ECE 115 Introduction to Computing: Programming Autonomous Vehicles. This requirement must be satisfied before the beginning of the junior year."[9]

Others have double-counting restrictions. In other words, if a requirement has multiple subrequirements, a course may only count toward one of them:

"Courses must come from 3 out of 4 subject areas. No double counting. You can only use NEU courses and electives one time (in one category)"[10]

To complicate things, requirements may be in interdependence relationships with one another:

"A student majoring in a department other than East Asian studies must take six language courses, two or more of which must be beyond the second-year level."[11]

[9]Computer Science requirement for the BSE degree.
[10]Electives requirement for Neuroscience.
[11]Language requirement for the Chinese minor.

hoagieplan navigates these complexities and tracks which requirements are satisfied by a student's courses. Some courses may satisfy multiple requirements. In this case, the user has to pick which requirement a course is settled into, so the unsettled course shows up as a grayed out button under the potential requirements (18, 19).

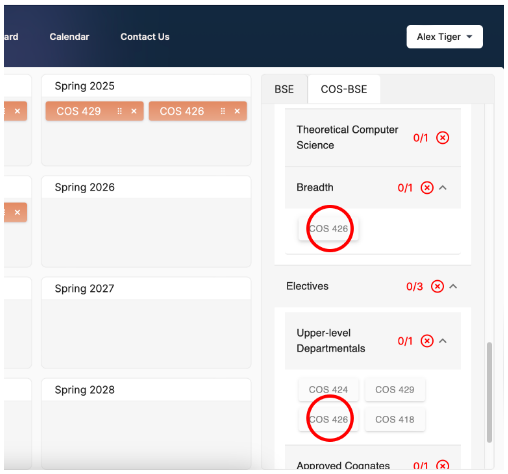


Figure 18: COS 426 can be settled either into Breadth pr into Upper Level Departmentals.

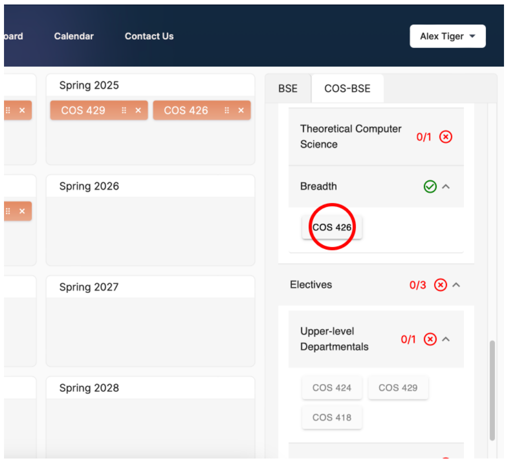


Figure 19: The user settled COS 426 into Breadth, so it no longer appears under Upper Level Departmentals.

Course Crosslistings

Princeton courses may be listed under multiple departments. For example, THR 203 / AAS 204 / DAN 203 / GSS 378 is listed under the Theatre, African American Studies, Dance, and Gender and Sexuality Studies departments. We say that this course is "crosslisted." Theatre (THR), the first crosslisting in the course code, is the primary department. hoagieplan displays all the crosslistings in the course cards. In the case of courses whose crosslistings changed over time (e.g. AAS 230 / ENG 231 to AAS 230 / AMS 231), only the most recent crosslisting is shown on search. However, if the user filters the search results to a previous semester, the crosslisting pertinent to that semester is shown. Search results are sorted by the catalog number of the crosslisting that is matched by the search query. 20 illustrates this.

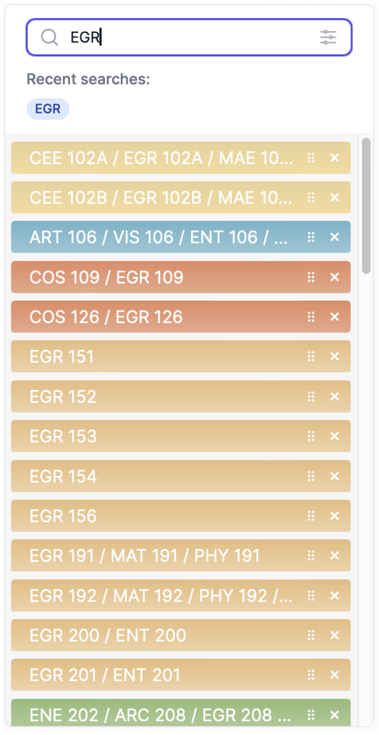

EGR
Recent searches:
EGR
CEE 102A / EGR 102A / MAE 10...
CEE 102B / EGR 102B / MAE 10...
ART 106 / VIS 106 / ENT 106 / ...
COS 109 / EGR 109
COS 126 / EGR 126
EGR 151
EGR 152
EGR 153
EGR 154
EGR 156
EGR 191 / MAT 191 / PHY 191
EGR 192 / MAT 192 / PHY 192 /...
EGR 200 / ENT 200
EGR 201 / ENT 201
ENE 202 / ARC 208 / EGR 208 ...

Figure 20: Search results are in ascending order of the catalog number corresponding to the "EGR" crosslisting.

Search Filters

Users can filter their search by semester, distribution area, course level, and allowed grading bases, as shown in 21.

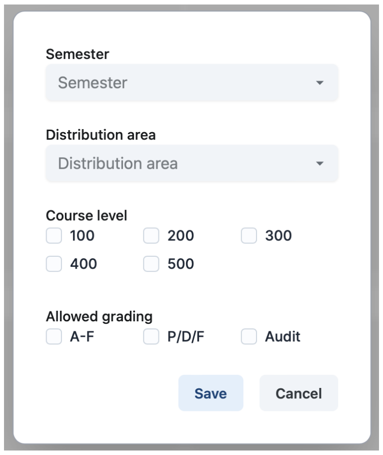


Figure 21: Search filters modal.

Search for Requirement's Course List

After opening a requirement modal, users can click on the "Search Courses" button to close the modal and populate the search results with the courses that satisfy the requirement. If the requirement is satisfied by any course from one or multiple departments, then the search results are populated with samples of 4-5 courses from each of these departments.

Mark Requirement as Satisfied

A student may have satisfied a requirement through AP credit, placement tests, or course substitutes that are not listed on the department website. In this case, users can open that requirement's modal and click the "Mark Satisfied" button to manually mark the requirement as satisfied. The satisfaction mark that appears next to the requirement's name will be gray instead of the usual green to indicate that it was manually marked as satisfied.

3. Implementation

3.1. Overview

By the numbers, hoagieplan contains:

- 14,248 lines of code across 179 files.
- 47.8% Python code, 46.7% TypeScript code, 5.3% in SCSS, and 0.2% other languages.
- 18,017 lines of manually collected requirements data across 108 YAML files.

- 419,216 database entries across 33 tables.
- 147 early testing users.

3.2. Architecture

hoagieplan has a frontend and a backend server. The frontend server routes user commands to the backend for processing. The backend server handles database operations such as course search, requirement checking, and updating account states.

Frontend

hoagieplan uses React-based Next.js as its UI library. This has important performance benefits. First, Next.js allows React components to be rendered on the server and sent as HTML to the browser, which significantly improves the load times of pages. Second, Next.js splits the code into smaller bundles at build time, which can be loaded on demand. Only the necessary code is loaded initially, again improving performance. Finally, Next.js' Fast Refresh feature speeds up the debugging process; the UI is updated after code changes without me having to restart the servers or refresh the page. It is an industry-grade framework used by TikTok, Twitch, Nike, and others.

The prevalent programming language in the frontend is TypeScript with embedded JSX (.tsx), which is standard for Next.js applications. It has the powerful web development features of JavaScript with significantly improved type safety, as it checks type correctness at compile time.

hoagieplan mainly uses Tailwind CSS for styling, as it provides beautiful inline style presets. For style sheets, it uses SCSS, which is a superset of CSS with support for variables and mathematical functions. This proved to be very useful for course card resizing and drag and drop animations.

The frontend is deployed on Vercel. It is the owner and maintainer of Next.js, so it is optimized for the framework hoagieplan uses. It also provides GitHub integration and analytics. Customers of Vercel include The Washington Post, Stripe, and TripAdvisor, which speaks to its quality.

Backend

The backend code is written in Python with Django as the web development framework. It is a high-level framework that is less flexible but better suited for large scale applications than Flask.

It handles database operations through models, which are easy-to-use Python objects that mirror database tables. Django's migrations functionality allows for updating the structure of a table in a few lines of code by updating its respective model. Django allows for easy manipulation of foreign key and many-to-many table relations, as well as serialization of table data into JSON format, which is how backend data is sent to the frontend.

For the database, hoagieplan uses PostgreSQL. The relational structure provides the speed required for fast requirement checking. The schema of the database is attached in 23 and 24 in Appendix 6.2. I used JetBrains' DataGrip software to manage, monitor, and debug the database.

Authentication is done using Princeton's Central Authentication System (CAS). This allows hoagieplan to prepopulate students' accounts with their names and class years.

3.3. Code Structure

A simplified directory structure is given in 22. The two main directories, frontend and backend, contain the code for their respective servers.

```
backend
├── compass
│   ├── models.py
│   ├── serializers.py
│   └── views.py
├── config
│   ├── settings.py
│   └── urls.py
├── data
│   ├── check_reqs.py
│   ├── fetch_courses.py
│   ├── fetch_evals.py
│   ├── insert_data.py
│   ├── insert_evals.py
│   └── insert_yamls.py
├── degrees
│   ├── AB.yaml
│   └── BSE.yaml
├── majors
│   ├── AAS.yaml
│   ├── ANT.yaml
│   ├── ARC.yaml
│   └── ...
├── minors
│   ├── AFS.yaml
│   ├── ASA.yaml
│   ├── CHI.yaml
│   └── ...
└── certificates (old data, unused)
    ├── AAS.yaml
    ├── ACE.yaml
    ├── ACM.yaml
    └── ...
```

```
frontend
├── app
│   ├── dashboard
│   │   ├── Canvas.tsx
│   │   └── page.tsx
│   ├── globals.scss
│   ├── layout.tsx
│   └── page.tsx
├── components
│   ├── Container
│   ├── InfoComponent
│   ├── Item
│   ├── RecursiveDropDown
│   ├── Search.tsx
│   └── UserSettings.tsx
├── store
│   ├── filterSlice.ts
│   ├── searchSlice.ts
│   └── userSlice.ts
├── tailwind.config.ts
└── types.d.ts
```

Figure 22: Simplified directory structure of hoagieplan.

Frontend

The core of the functionality resides in `frontend/app/dashboard/Canvas.tsx`. This is the file that renders the Dashboard page. It generates the search results container and the semester containers. It handles the rendering of the search results and the drag-and-drop functionality. It makes API calls to the backend for fetching a user's courses from the database, checking requirements, and updating the user's courses in the database as they are moved from one container to another.

The `frontend/components` directory contains the TypeScript–JSX code (.tsx) and style sheets (.scss) for the various components of the application. The following files and directories are important:

- `frontend/components/Container` contains the implementation of the semester containers and the search results container.
- `frontend/components/Item` contains the implementation of the draggable course cards.
- `frontend/components/InfoComponent` contains the implementation of the course information modal and the content on the course cards.
- `frontend/components/RecursiveDropdown` contains the implementation of the recursive requirement menu.
- `frontend/components/Search.tsx` contains the code for the search functionality.
- `frontend/components/UserSettings.tsx` contains the code for the Account Settings, including the functionality to update an account's state in the database.

hoagieplan uses Zustand, a React-based library for state management. It allows for passing the state of variables accross different files in the code, which is essential for persisting the state of search results and user settings within a session. The TypeScript code (.ts) for Zustand is in the `frontend/store/` directory:

- `frontend/store/userSlice.ts` defines the state of the user and the functions that update it.

- `frontend/store/searchSlice.ts` defines the state of the search results and the functions that update it.
- `frontend/store/filterSlice.ts` defines the state of the search filters and the functions that update it.

Backend

The URLs of the requests from the frontend are picked up by `backend/config/urls.py`, which routes them to be handled by functions in `backend/compass/views.py`. This is a standard Django setup. The directories `backend/degrees`, `backend/majors`, `backend/minors` contain data for degree, major, and minor requirements in YAML format. `backend/data` contains database-facing code:

- `backend/data/fetch_data.py` makes API calls to Princeton's StudentApp and processes the responses into CSV files, one per semester.
- `backend/data/insert_data.py` inserts the course data CSVs into the database.
- `backend/data/fetch_evals.py` scrapes course evaluations from the registrar's website and writes them to a CSV.
- `backend/data/insert_evals.py` inserts the course evaluations CSV into the database.
- `backend/data/insert_yamls.py` inserts the data from the degree, major, and minor requirement YAMLs into the database.
- `backend/data/check_reqs.py` checks a user's requirements against their courses.

3.4. Challenges

Data Collection

Manually collecting the data for 2 degrees, 38 majors, and 38 minors took over 200 hours of work. A lot of time was spent finding the best format of the requirement YAML files to reduce the work. The format I chose is described in the next section. It was also difficult to decide which courses should satisfy some underspecified requirements:

"Focal points include, but are not limited to

- *Greek Language, Literature, and Culture*
- *Latin Language, Literature and Roman Culture Classics and Reception*
- *Medicine, Science, and the Body"*[12]

For this requirement, I read the description of all the courses in the Classics department to reason whether they fall under one of the focal points or not.

Princeton's StudentApp API, which I used to get course data, has a few broken endpoints and often times out. This led to incomplete course data. The solution was to manually look for and exclude the broken endpoints, as well as implement an "at-least-once" algorithm for API calls: on failure, each call is retried up to 3 times.

Checking Requirements

As detailed in the Selected Features section, verifying a Princeton student's degree, major, and minor requirements is hard due to deadlines, double-counting restrictions, and interdependence relationships. Inspired by TigerApps' Departmental Data repository, hoagieplan's approach is encoding requirements in YAML format.

Requirements follow a tree structure. The root node is either a degree, a major, or a minor, and its children are the respective requirements. Most requirements have their own subrequirements, so they are internal nodes in the tree. Leaf nodes may be represented by a course list, a department list, a distribution area, or a course count. Nodes have the following properties[13]:

- `name`: requirement's name.
- `max_counted`: maximum units passed to the parent requirement.
- `min_needed`: minimum units demanded of children.
- `explanation`: human-readable explanation of the requirement.
- `double_counting_allowed`: whether courses may count for multiple subrequirements of this requirement.
- `completed_by_semester`: semester by the end of which the requirement must be complete.
- `course_list`: (leaf nodes) list of courses that satisfy this requirement.

[12]Classical Studies with a Focal Point requirement for the Classics minor.
[13]Adapted from the Princeton Departmental Data Repository.

- `dept_list`: (leaf nodes) list of departments from which any course satisfies this requirement.
- `distribution_area`: (leaf nodes) distribution area of the courses that satisfy this requirement.
- `num_courses`: (leaf nodes) number of courses required to satisfy this requirement.

25 and 26 in Appendix 6.3 illustrate how I encoded the information for two different requirements into the YAML format.

The requirement checking code performs an inorder traversal of a user's requirement tree and compares their courses against each node.

A complication is that requirements may be satisfied with AP credits, and others are underspecified or otherwise impossible to encode into the above format:

"One course, which may or may not be one of the nine courses taken for the major, must be dedicated in its entirety to historical periods, literature or cultures before 1800 C.E."[14]

To circumvent this, I implemented the "Mark Satisfied" button that allows users to manually mark a requirement as satisfied. I also appended editor's notes to the requirement explanations:

Editor's Note: This requirement is too broad. Courses not in the list may satisfy the requirement and hoagieplan does not check this.

This makes the application useful for bookkeeping purposes even for requirements that cannot be automatically checked.

Another complication was that each user can have over 50 requirements to check, which is slow due to the large number of database transactions involved. The solution was to cache each user's requirement tree in the database, so that it is not reconstructed each time a course is inserted into their schedule. This significantly reduces the slow database transactions.

[14]Departmental Distribution Requirement for Comparative Literature

4. Evaluations

4.1. Feedback Form

A beta release of the application was conducted with 147 users. I sent out a feedback form which received 60 responses. The responses indicate overall satisfaction with the application. With a few exceptions, the users found the functionality intuitive.

A recurring piece of feedback is that users want to plan certificate programs as well. Although certificates will be withdrawn in 2026, the responses indicate that a large number of undergraduates still pursue them:

- *“All the certificates are missing. I’m urban studies and I need to plan for that”*
- *“Missing legacy certificates like OQDS”*
- *“You can’t add certificates”*
- *“Have you guys added every minor/certificate or is that still to come? I don’t see Near Eastern Studies in the selection dropdown.”*
- *“I’m wondering if you guys are planning to update the app with certificate requirements as well? It would be super helpful for me as I’m planning to navigate 2 certificates.”*

Furthermore, users found many requirement data mistakes. This was expected, as the data was collected manually and was sometimes interpretable:

- *“Correct me if I’m wrong, but it seems that you do not account for double counting with regard to minors. For example, the finance minor allows you to double count a maximum of 2 courses, although it double counts every course I enter (my major is econ).”*
- *“I input all of my classes, but the field "Seven Total Courses" under the ’BSE’ ’Distribution Requirements’ tab reads 0/7 when I have actually completed almost all of them.”*
- *“In addition to my previous bug report about the distributions requirements being checked off, the foreign language isn’t satisfied with my ASL track (four classes total)”*
- *“I put it my courses, and it missed counted the number electives I have completed from ’Other Departments.’ My actual number of completed electives is 1, but it said 2/10”*

- *"the prereqs for math is wrong, doesnt have to be 215 and 217 or 217 and 218. its (215 or 216) and (217 or 218)"*

4.2. Live Testing

Three live tests were conducted. The testers were Robert M. Dondero, Jeanie Chang, Kyung Lee, and a Princeton student chosen at random in the Firestone library, who I will call Alice. The task list given to the testers and my notes during the tests are in Appendix 6.4. The points of improvement identified during the tests are:

- Implement a short user guide that is displayed on first login.
- If settling one or more courses into a requirement would satisfy it, use a different color for the satisfaction mark.
- Highlight the filters button when filters are active.
- Make course cards draggable from their entire surface, not just the drag handle.

5. Summary

5.1. Conclusions

This application was a much bigger undertaking than I initially expected, but it is close to being done and hopefully it will have a large and stable user base for the next few years. I learned that it is better to complement other applications before implementing features that are already available. Specifically, the application would have been more useful to beta testers if certificates support was implemented instead of, e.g, course information modals. Time is limited and constant communication with the users is essential to determine which features to sacrifice. I also learned that an application is much less intuitive to a user compared to a developer, which is also an argument for constant communication with users. Finally, I learned that requirement tracking at Princeton is intrinsically hard. Gathering all the requirements data should not be a one-person job, but should be a collaboration between different departments to list their requirements in a common format that students can use. A partial solution is having a dedicated team of people updating the requirements each year and communicating with departments to ensure correctness.

5.2. Limitations

The application has a few technical limitations. First, and most importantly, there is no support for certificates. Second, users cannot see the course titles in the search results and cannot search courses by title. Third, the course information modals do not display as much information as Princeton Courses does. Specifically, hoagieplan does not partition course reviews by semester and does not display course instructors. These three limitations will be addressed in future versions of hoagieplan.

A higher level limitation is that there is no way to automatically update requirements data when departments make changes. This means that, in the absence of a team that maintains it, hoagieplan will no longer be useful in a few years. I plan to consult with Princeton departments in the hope of finding a long-term solution. In the meantime, requirements data will be maintained by the hoagie.io team.

5.3. Future Work

I will first implement the points of improvement highlighted in the evaluations. Then, I will address the rest of the limitations mentioned in the previous section. Finally, I will integrate the UI of hoagieplan with that of other hoagie.io applications such as hoagiemail. I intend to release the application to the public in August 2024.

6. Appendix

6.1. User Scenarios

User Scenario – Vanessa

Vanessa is a Comparative Literature (COM) major double-planning Creative Writing (CWR) and

Visual Arts (VIS) minors. She wants to optimize her junior fall course selection in order to satisfy as many requirements as possible.

- She clicks on her name on the top right section of the screen to open the Account Settings menu. She selects 2026 as her class year, COM as her major, and CWR and VIS as her minors.
- She clicks on the COM requirements one by one, clicks the "Search Courses" button for each of them, and drags the CWR and VIS courses that she finds interesting into her schedule.
- Then, she looks at the VIS and CWR requirements to see which VIS and CWR courses in her schedule satisfy them. She finds that CWR 305 / COM 355 / TRA 305 can count both toward the Creative Arts requirement in COM and the Advanced Coursework requirement in CWR, so she keeps it in her schedule. Similarly, she finds that VIS 301 can count both toward the Creative Arts requirement in COM and the Studio Courses requirement in VIS, so she keeps it in her schedule.

User Scenario – Cole

Cole is a School of Politics and International Affairs (SPI) junior trying to decide on minors to declare.

- First, he fills up his semesters with his past coursework. Based on his past courses, he guesses that the History (HIS), History of Science, Technology, and Medicine (HSTM), and Quantitative Economics (MQE) minors are a good fit for him.
- Cole selects 2026 as his class year, SPI as his major, and HIS, HSTM, and MQE as his minors.
- Browsing through the HIS, HSTM, and MQE requirements, he sees that he is 1 course away from satisfying the HIS requirements, 5 courses away from satisfying the HSTM requirements, and 4 courses away from satisfying the MQE requirements. Based on this, Cole decides to minor in History and Quantitative Economics.
- Looking at the AB requirements, he realizes he needs one more Social Analysis course. Cross-checking with MQE requirements, he finds that ECO 327 is a Social Analysis course that also counts toward the Elective Courses requirement in MQE, so he picks it for next semester.

6.2. Database Schema

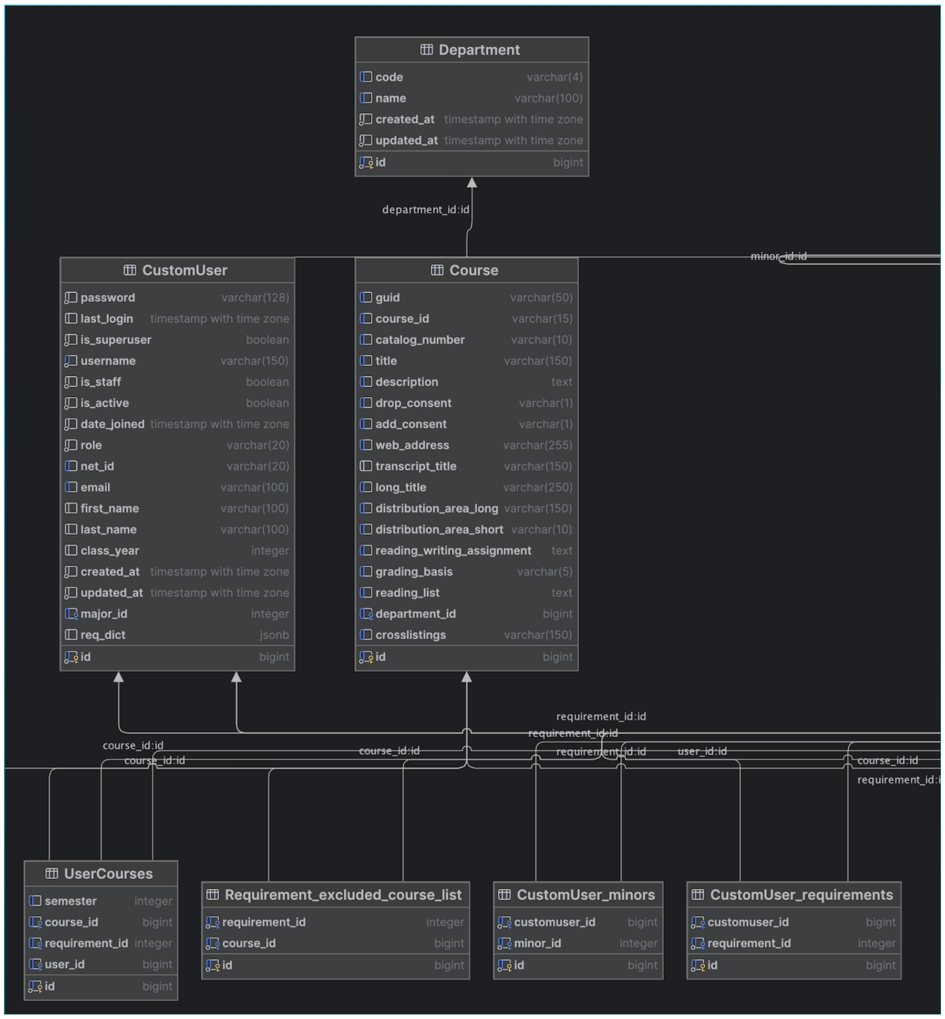


Figure 23: Database schema section pertaining to courses.

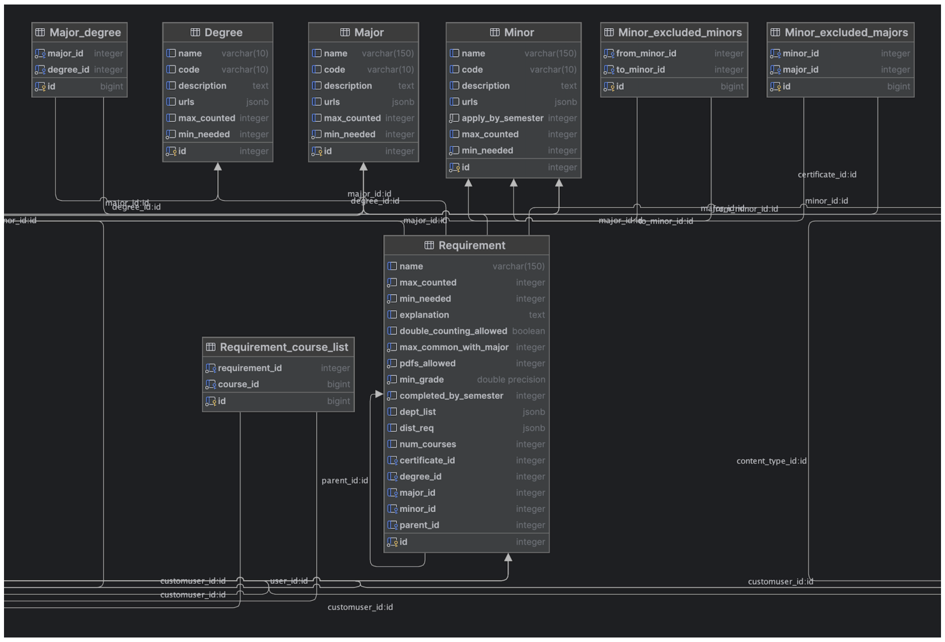
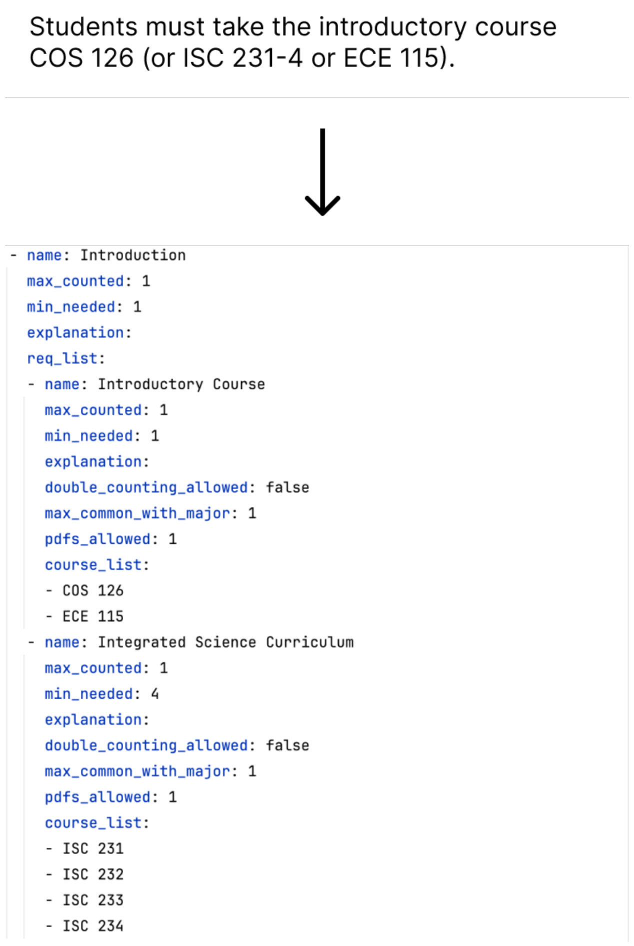


Figure 24: Database schema section pertaining to requirements.

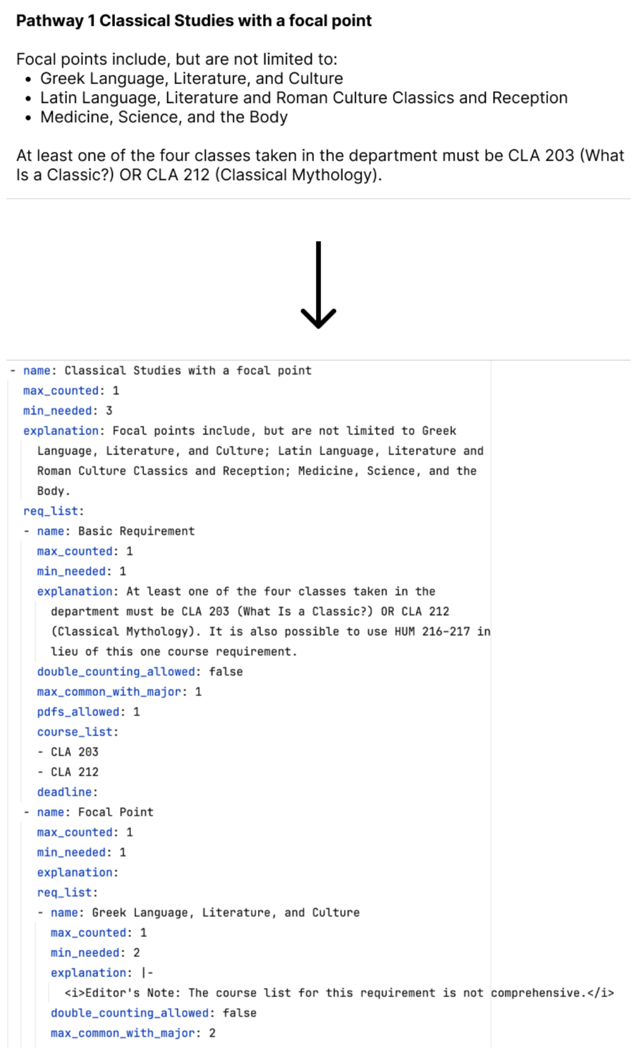
6.3. YAML Encodings



Students must take the introductory course COS 126 (or ISC 231-4 or ECE 115).

```yaml
- name: Introduction
  max_counted: 1
  min_needed: 1
  explanation:
  req_list:
  - name: Introductory Course
    max_counted: 1
    min_needed: 1
    explanation:
    double_counting_allowed: false
    max_common_with_major: 1
    pdfs_allowed: 1
    course_list:
    - COS 126
    - ECE 115
  - name: Integrated Science Curriculum
    max_counted: 1
    min_needed: 4
    explanation:
    double_counting_allowed: false
    max_common_with_major: 1
    pdfs_allowed: 1
    course_list:
    - ISC 231
    - ISC 232
    - ISC 233
    - ISC 234
```

Figure 25: YAML encoding of the Introduction requirement for the Computer Science minor.



Pathway 1 Classical Studies with a focal point

Focal points include, but are not limited to:
- Greek Language, Literature, and Culture
- Latin Language, Literature and Roman Culture Classics and Reception
- Medicine, Science, and the Body

At least one of the four classes taken in the department must be CLA 203 (What Is a Classic?) OR CLA 212 (Classical Mythology).

```yaml
- name: Classical Studies with a focal point
  max_counted: 1
  min_needed: 3
  explanation: Focal points include, but are not limited to Greek
    Language, Literature, and Culture; Latin Language, Literature and
    Roman Culture Classics and Reception; Medicine, Science, and the
    Body.
  req_list:
  - name: Basic Requirement
    max_counted: 1
    min_needed: 1
    explanation: At least one of the four classes taken in the
      department must be CLA 203 (What Is a Classic?) OR CLA 212
      (Classical Mythology). It is also possible to use HUM 216-217 in
      lieu of this one course requirement.
    double_counting_allowed: false
    max_common_with_major: 1
    pdfs_allowed: 1
    course_list:
    - CLA 203
    - CLA 212
    deadline:
  - name: Focal Point
    max_counted: 1
    min_needed: 1
    explanation:
    req_list:
    - name: Greek Language, Literature, and Culture
      max_counted: 1
      min_needed: 2
      explanation: |-
        <i>Editor's Note: The course list for this requirement is not comprehensive.</i>
      double_counting_allowed: false
      max_common_with_major: 2
```

Figure 26: YAML encoding of the Classical Studies with a focal point requirement for the Classics minor.

6.4. Live Testing Notes

Each tester was given the same task list, which is outlined below.

1. Log in, find the Account Settings.
2. Suppose you are a sophomore in the MAE department. Select 2026 as your class year, MAE as your major, and MQE as your minor.
3. Inspect your major requirements by scrolling up and down the requirements menu and by clicking on requirement titles.
4. Search for courses to satisfy the Engineering Design Courses requirement under the Mechanical

Track. Drag them into your semester schedule. If there are multiple courses that satisfy the same subrequirement, click on them to see their ratings and pick the highest rated one.

5. Settle the courses into requirements so that they show up as satisfied.
6. Switch to the BSE tab of the requirements menu. Inspect your degree requirements.
7. Suppose you want to take a Drawing class next semester to satisfy the Literature and the Arts distribution requirement. Go to search filters, select the Fall 2024 semester and the LA distribution area. Then, search for the VIS department and look for a Drawing class. Drag it into the Fall 2024 container and see the Literature and the Arts requirement satisfied.
8. Suppose you passed the Chemistry placement test. Mark the Chemistry requirement as satisfied.

The rough notes I took during each test are pasted below.

Robert M. Dondero

- Not sure whether to click on get started or log in, expected to click on get started.
- Correctly clicked on upper left hoagie to go back to home screen.
- Found the account settings. That was intuitive.
- Liked that you can type up minor name.
- Correctly saved.
- Figured out how the tabs work for the requirements menu.
- “What does x mean?” Thought the “x” mark for the requirements was a delete button.
- Liked that internal nodes of the requirement menu give explanations, child nodes give course lists.
- Figured out how to settle courses into requirements, but it is probably not obvious.
- Took a while to find the filters button.
- Would like a “Clear filters” button.
- Did not understand the difference between drop shadow versus no drop shadow in course settling.

Jeanie Chang

- The account settings were easy to find.
- The settings menu looks clean.
- The settings menu is very intuitive.

- It is easy to find majors and minors, likes how they are alphabetically listed.
- Likes that requirements have explanations. Likes the gray highlight on hover of requirement title.
- Likes that requirement menu is expanded by default.
- Missed “Search Courses” button.
- Likes the course information modal, stars for ratings at the top and distribution area are very convenient.
- Figured out how to settle courses.
- Overall found the application very intuitive.

Kyung Lee

- Not enough distinction between different subrequirements, hard to figure out the parent-child structure of requirements.
- Confused with the drag handle of course cards, would like to be able to drag from any point of the course card.
- Confused how settling courses into requirements works.
- Confused about the way children requirements satisfy parents.
- Wants to see course titles displayed, functionality to search by title.
- Likes how the satisfaction mark is grey for manually satisfied requirements.
- Would like the dropdown button for requirements to be at the indent, not right side.

Alice

- Initial reaction was to drag from any point of the course card but she figured out the drag handle quickly. She would like course cards to be draggable from any point.
- Did not understand settling courses into requirements. Suggested that the requirement marker is different if the requirement could be satisfied by settling courses into it.
- Took her a while to find how to set her major.
- Wants to see the titles of courses on the course cards or on hover.
- Likes the design.
- Likes that the data is current.

- Did not notice the “Mark Satisfied” button.
- Did not notice that you can click on course cards, and requirement titles to see information.
- Likes TigerPath, still uses it.