



# CS CAPSTONE DESIGN DOCUMENT

FEBRUARY 16, 2020

## OSU CS APPLIED PLAN PORTAL

PREPARED FOR

OREGON STATE UNIVERSITY

DR. ROB HESS

\_\_\_\_\_  
*Signature*

\_\_\_\_\_  
*Date*

PREPARED BY

GROUP CS72

THE PORTAL TEAM

CLAIRE CAHILL

\_\_\_\_\_  
*Signature*

\_\_\_\_\_  
*Date*

JACKSON GOLLETZ

\_\_\_\_\_  
*Signature*

\_\_\_\_\_  
*Date*

PHI LUU

\_\_\_\_\_  
*Signature*

\_\_\_\_\_  
*Date*

ZACHARY THOMAS

\_\_\_\_\_  
*Signature*

\_\_\_\_\_  
*Date*

### Abstract

This document provides the architectural and visual design of the CS Applied plan Portal. The document introduces the scope, purpose, and intended audience of the web application while giving a much more detailed look at the application programming interface and the development tools and processes of the system. Throughout this document, the team gives assessments of the infrastructures provided by OSU APIs and discusses the development process as well as the usability of the application.

## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Scope . . . . .	3
1.2	Purpose . . . . .	3
1.3	Intended Audience . . . . .	3
1.4	Conformance . . . . .	3
1.5	Glossary . . . . .	3
1.5.1	Acronyms and abbreviations . . . . .	3
<b>2</b>	<b>User Interface Design</b>	<b>4</b>
2.1	OSU Brand Guide . . . . .	4
2.2	Web-page Layout . . . . .	5
2.2.1	Student Home . . . . .	5
2.2.2	Advisor Home . . . . .	6
2.2.3	Create/Edit Plan . . . . .	6
2.2.4	View Plan . . . . .	7
2.2.5	Set Roles . . . . .	8
<b>3</b>	<b>Usability</b>	<b>9</b>
<b>4</b>	<b>Data Storage and Handling</b>	<b>11</b>
4.1	Database Model . . . . .	11
4.2	Relational Database Management System . . . . .	11
4.3	Database Schema . . . . .	11
4.3.1	Plan . . . . .	12
4.3.2	Course . . . . .	12
4.3.3	User . . . . .	12
4.3.4	Comment . . . . .	12
4.3.5	PlanReview . . . . .	13
4.3.6	SelectedCourse . . . . .	13
4.4	Concerns About Course Data . . . . .	13
<b>5</b>	<b>Application Programming Interfaces</b>	<b>14</b>
5.1	OSU APIs . . . . .	14
5.1.1	OAuth2 . . . . .	14
5.1.2	Directory . . . . .	14
5.1.3	Textbooks . . . . .	14
5.1.4	Courses . . . . .	15
<b>6</b>	<b>Central Authentication Service</b>	<b>15</b>
6.1	Using a Central Authentication Service . . . . .	15
6.2	Central Authentication Service Attributes . . . . .	15

		2
<b>7</b>	<b>Development</b>	<b>16</b>
7.1	Component-Level Development . . . . .	16
7.2	Containerization . . . . .	16
7.3	Hosting Platform . . . . .	16
<b>8</b>	<b>Mobile Support</b>	<b>17</b>
8.1	Mobile App Type . . . . .	17
8.2	Mobile Web App Constraints . . . . .	17
8.2.1	Mobile Browser Limitations . . . . .	17
8.2.2	Screen Resolutions . . . . .	17
8.2.3	Optimization . . . . .	17
<b>9</b>	<b>Conclusion</b>	<b>17</b>

## LIST OF FIGURES

1	Student Home Page. . . . .	6
2	Advisor Home Page. . . . .	6
3	Create/Edit Plan Page. . . . .	7
4	View Plan Page. . . . .	8
5	Set Roles Page. . . . .	9
6	Sequence diagram. . . . .	9
7	System use case diagram. . . . .	10
8	Database entity relationship diagram. . . . .	11
9	Data flow diagram. . . . .	13

## LIST OF TABLES

1	OAuth2[9, Tab. 1]. . . . .	14
2	Directory[11, Tab. 1]. . . . .	14
3	Textbooks[12, Tab. 1]. . . . .	14

# 1 INTRODUCTION

## 1.1 Scope

This document covers the front-end and the back-end designs of the OSU CS Applied plan Portal. For the front-end, this document shows the team's mockups of web pages for students and advisors. The document defines the color system, iconography, and graphic patterns of the application such that they adhere to OSU's conventions. For the back-end, this document shows the entities involved in the data storage and handling processes as well as the relationship between the entities. The scope of this document also covers how we utilize OSU APIs to manipulate the data and how we decide on various technologies used for this web application.

## 1.2 Purpose

This document provides an insight to the team's designs for the application. The purpose of these designs is to establish a concrete plan to develop the system. We will be directly using the design specifications when implementing the front-end and the back-end. This document acts as a proposal to our client of how we as a team moves towards the development process.

## 1.3 Intended Audience

The intended audience of this design document is our client, who will assess our approach to developing the application. This document guides the client in the selection, organization, and presentation of the design information. It helps to ensure that the client is aware of the development process and, vice versa, helps the team establish complete and concise design descriptions to communicate more effectively with the client.

## 1.4 Conformance

The design specifications listed throughout this document, especially the user interface design, adhere to the theme and convention of Oregon State University websites. Since this application implements most, if not all, of its API endpoints based on the established OSU APIs, the back-end processes that handle these API endpoints should already conform to OSU's standards. This document also discusses our design of the security and authentication features to ensure that the application is compliant with FERPA.

## 1.5 Glossary

### 1.5.1 *Acronyms and abbreviations*




- **API:** Application Programming Interface.
- **ARIA:** Accessible Rich Internet Applications.
- **AWS:** Amazon Web Services.
- **CAS:** Central Authentication Service.
- **CS:** Computer Science.
- **CSS:** Cascading Style Sheets.
- **ECE:** Electrical and Computer Engineering.
- **GPA:** Grade Point Average.
- **HTTP:** HyperText Transfer Protocol.

- **JS:** JavaScript
- **JSON:** JavaScript Object Notation.
- **LDAP:** Lightweight Directory Access Protocol.
- **OSU:** Oregon State University.
- **ONID:** OSU Network ID.
- **PAAS:** Platform as a Service.
- **RDBMS:** Relational Database Management System.
- **SQL:** Structured Query Language.
- **SSO:** Single Sign-On.
- **UI:** User Interface.
- **UML:** Unified Modeling Language.
- **URL:** Uniform Resource Locator.
- **WAI:** Web Accessibility Initiative.

## 2 USER INTERFACE DESIGN

### 2.1 OSU Brand Guide


As this application will eventually be transferred to Oregon State University for use as an official EECS tool, we have been instructed to adhere to the university's branding guidelines wherever possible. Our primary source of information on how to accomplish this will be Oregon State University's Brand Guide[1], created by University Relations and Marketing. While there are many themes and templates that one could use to construct a web app, our choice of React (discussed later in the document) reflects the fact that we will be creating many of our components ourselves, and so will not be using pre-made templates or themes. OSU's brand guidelines provide us with all of the information we could need in order to begin designing our web application. They include:

- a list of typefaces;
  - [Stratum 2](#) for header display text
  - [Georgia](#) for serified text and subtitles
  - [Helvetica](#) for sans-serifed text and subtitles
  - [Kievit](#) for body copy and smaller text
  - [Open Sans](#) for a free body copy alternative
- iconography guidelines;
  - Rounded, monoweight lines
  - Monochrome palette
  - Incomplete shapes with gaps at intersections
- a list of primary colors;
  -  Beaver Orange
  -  Paddletail Black
  -  Bucktooth White

- a list of secondary colors;

 Pine Stand

 High Tide

 Luminance

 Stratosphere

- imagery;
- graphics patterns; and
- guidelines for message, audience, and personality, among other things.

In addition to the purely visual design elements included in Oregon State's branding guidelines, we also found guiding principles for the university's media as a whole. For instance, the messaging guidelines mentioned in the list above describe characteristics and qualities of OSU students and faculty that URM believes should be represented in the things they produce. For our project, we believe the following items are most relevant:

- "The ability to navigate situations nimbly"
- "Utilize available resources for maximum impact"
- "Demonstrate steadfast resilience to get things done"

These principles and qualities are at the core of our project, and so should serve as significant factors in our decision-making. With the information provided by Oregon State University Relations and Marketing, we can confidently proceed with scaffolding out our pages once development starts, even if we don't apply Oregon State University's visual identity to it at the outset.

## 2.2 Web-page Layout

### 2.2.1 Student Home

The student home page is the first screen a student sees when they login to the portal. Users can create a new plan, they can search through already accepted plans, or they can view a plan that they have already created. Clicking the plus icon makes the new plan field appear at the top of the screen, giving the user the option of creating a plan from scratch or by using a template.

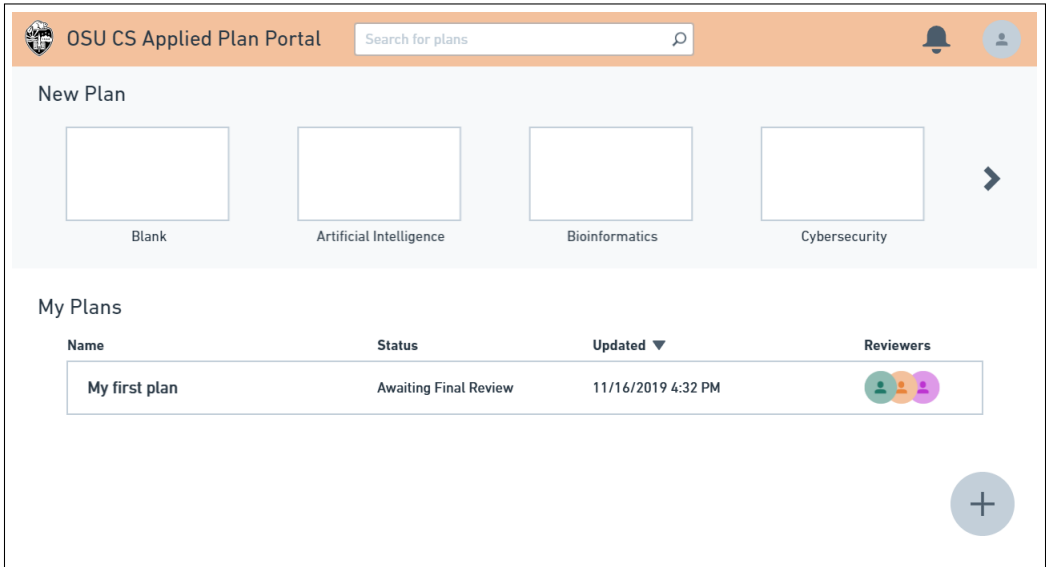


Fig. 1: Student Home Page.

2.2.2 Advisor Home

This is the default page that advisors will see upon log in. It lists student plans and can be sorted by "Awaiting Review", "Awaiting Final Review", "Awaiting Student Changes", "Accepted", or "Rejected". Advisors can search for students by their name or ONID number. Each plan shows the advisor(s) who have already reviewed it. Only the head advisor has an additional button on this page that brings them to the "Set Roles" page.

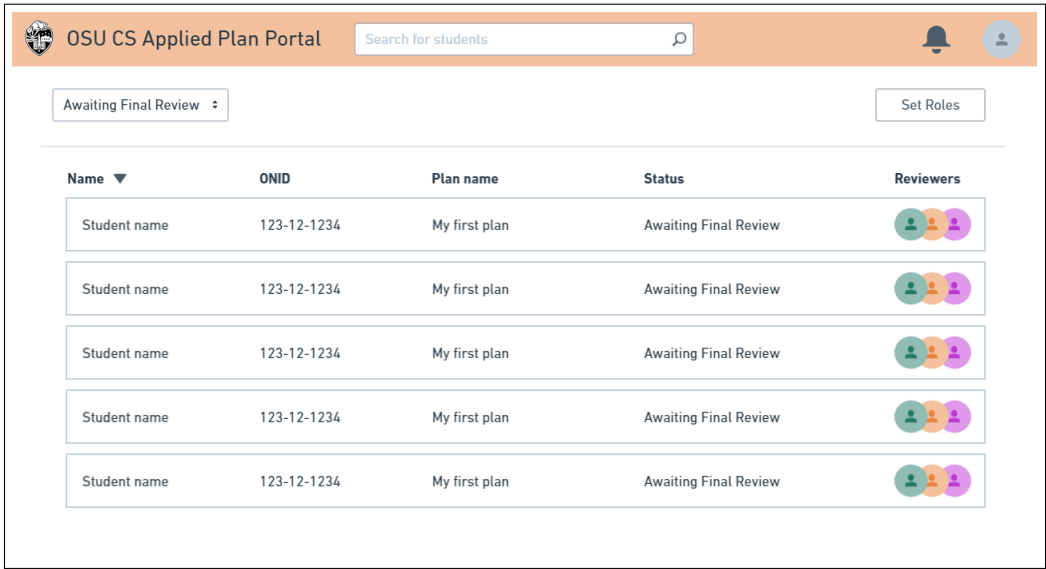


Fig. 2: Advisor Home Page.

2.2.3 Create/Edit Plan

A user is brought to this page when creating a new plan or when editing a plan. This page is split into two different sections.

**Right section:** Users can use this section to search for and find information about courses offered at OSU. Course information includes the course code, name, description, and acceptance rate. The course acceptance rate is a value indicating what percentage of all applied plans containing the course were approved.

**Left section:** Users can use this section to enter a name for the plan and select courses. When a user types out a course name the field attempts to auto-complete. Once a user has filled out their plan they can press the submit button at the top of the screen to save their changes and complete the plan. Users can only submit/update a plan if it does not violate a hard constraint. Users who attempt to submit/update a plan that violates a soft constraint will get a warning message, the user can ignore this warning. The overall acceptance rate in this section is the average acceptance rate of all of the courses in the plan.

**Hard constraints:**

- A plan must have at least 32 credits.
- A plan must not have any courses that are required as part of the base CS degree.
- A plan may not select the same course multiple times.
- A plan may not have a course that does not exist or is no longer offered.

**Soft constraints:**

- If a plan has the "Final Review" status, users will be warned that editing the plan will revert it to "Review" status.
- If a plan has less than 20 credits of upper-division CS courses the user will get a warning.
- If a plan has an unusually low overall acceptance rate the user will get a warning.
- If a duplicate plan has been rejected before and has never been accepted, the user will get a warning.

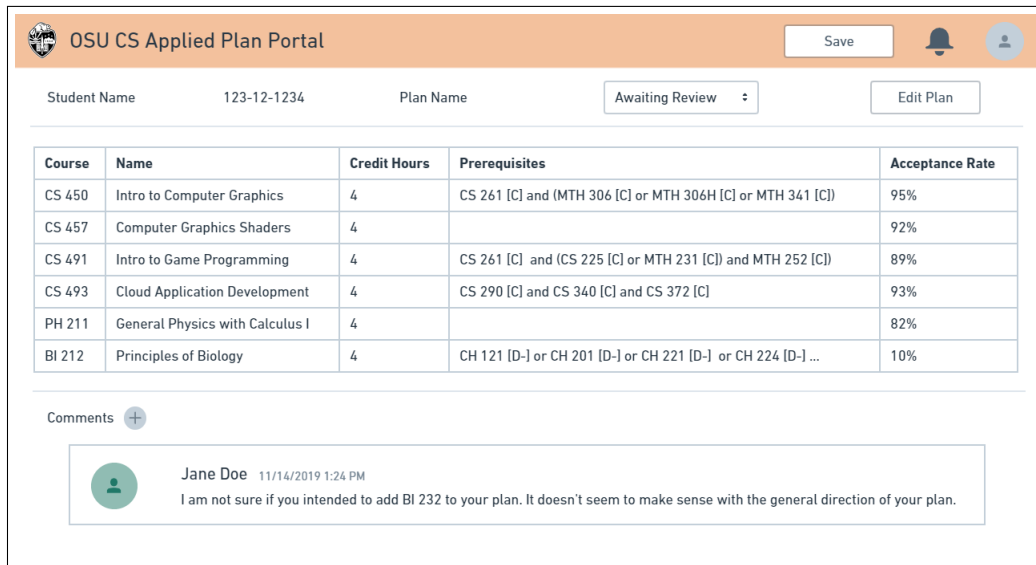
Fig. 3: Create/Edit Plan Page.

## 2.2.4 View Plan

This is the page users are brought to after selecting a plan to view. In order to approve or reject plans that have been submitted by students, advisors must have a way to view their contents. This is especially true for custom plans, but



even for template applied plans, there is a significant amount of flexibility in exactly which courses a student can choose. To address this, we will be creating a page for advisors to view the details of an individual plan, comment on it, and update its status as defined in the previous section. When a user opens the page for a specific plan, they will see the student's name, the student's OSU ID number, the plan's name, and the plan's status. Only advisors can change the plans status, and only the head advisor can select the "accept" and "reject" statuses. Below this header section, a table of the selected courses will be displayed, along with data about each course. The rightmost column contains a value indicating what percentage of all applied plans containing the course were approved. Below the table is a section where each user can add comments about the plan. Users may also select the edit plan button to be brought to the edit plan page.



The screenshot shows the 'OSU CS Applied Plan Portal' interface. At the top, there's a header with the OSU logo, the title 'OSU CS Applied Plan Portal', a 'Save' button, a notification bell, and a user profile icon. Below the header, the student's name '123-12-1234' and the plan name 'Awaiting Review' are displayed, along with an 'Edit Plan' button. A table lists the courses in the plan:

Course	Name	Credit Hours	Prerequisites	Acceptance Rate
CS 450	Intro to Computer Graphics	4	CS 261 [C] and (MTH 306 [C] or MTH 306H [C] or MTH 341 [C])	95%
CS 457	Computer Graphics Shaders	4		92%
CS 491	Intro to Game Programming	4	CS 261 [C] and (CS 225 [C] or MTH 231 [C] and MTH 252 [C])	89%
CS 493	Cloud Application Development	4	CS 290 [C] and CS 340 [C] and CS 372 [C]	93%
PH 211	General Physics with Calculus I	4		82%
BI 212	Principles of Biology	4	CH 121 [D-] or CH 201 [D-] or CH 221 [D-] or CH 224 [D-] ...	10%

Below the table is a 'Comments' section with a plus icon. A comment from 'Jane Doe' dated '11/14/2019 1:24 PM' is shown, stating: 'I am not sure if you intended to add BI 232 to your plan. It doesn't seem to make sense with the general direction of your plan.'

Fig. 4: View Plan Page.

### 2.2.5 Set Roles

This page is only accessible by the head advisor. Here the head advisor can set the role of any user. Each user can be assigned to at most one role, these roles are "Student", "Advisor", and "Head Advisor". If a user is set to "Head Advisor" a warning message appears requiring explicit approval of this action. Users can be filtered by their current role, sorted by name, email, or ONID. The head advisor can also use the search bar to search for a specific user.

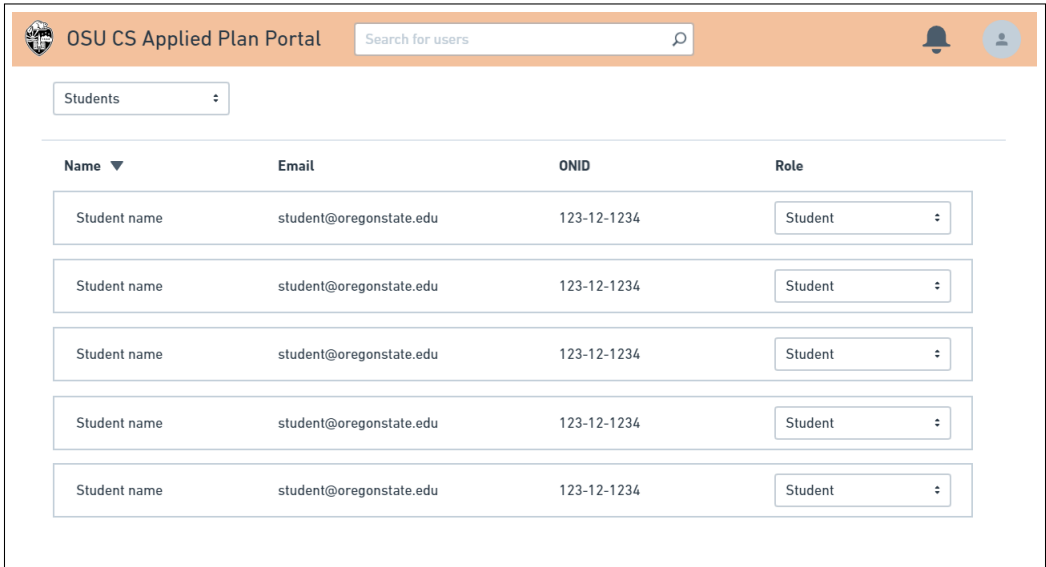


Fig. 5: Set Roles Page.

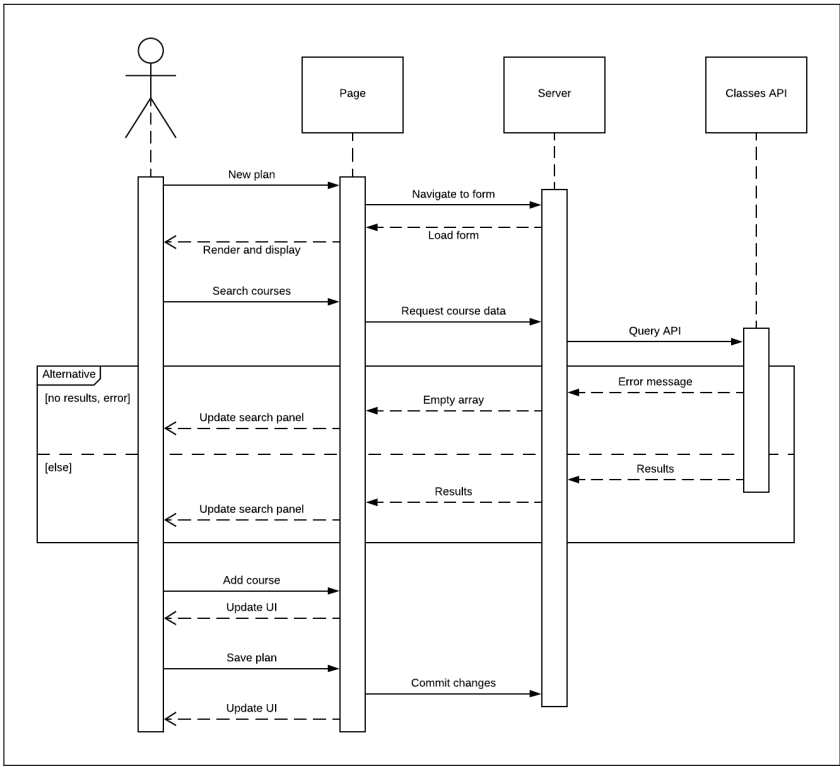


Fig. 6: Sequence diagram.

3 USABILITY

Usability is, in many respects, a highly subjective measure. Moreover, usability and UX testing are necessarily integrated into the development process, with testing of individual components and pages coming after those items are developed. To outline comprehensive usability standards and testing plans at this stage of the project would essentially require us

to mock up the precise functionality of each of our pages. As previously stated, scaffolding and development will not begin until the start of next year. By listing usability as a responsibility, we are not seeking to define steadfast standards for our own project's functionality and behavior.

Rather, our definitions of usability here seek to identify specific needs of our user base (namely students and advisors), and adopt industry standard practices to ensure that our final product meets those needs. Oregon State University's brand personality traits state that OSU must be "Welcoming"[1]. More specifically, the guidelines state that OSU is "open to all and enriched by difference;" it follows that the products created by and for Oregon State should reflect this value. Since OSU's students come from a wide array of backgrounds and have a wide array of abilities, we believe it is important that our final product work as seamlessly as possible to all individuals.

One major step towards accessible web apps is the WAI-ARIA standard, a comprehensive manual of standard practices for developing accessible web applications[2]. The goal of the WAI-ARIA standard is to allow all users, especially those with impairments, to interact with web apps in a way that closely resembles the developer-intended experience, and does not cause additional burdens to those with accessibility issues[3]. The ARIA standard has been carefully drafted and refined over more than a decade to guide developers in creating web apps with accessible experiences. Crucially, the WAI-ARIA standard has been established for so long that the vast majority of screen readers and other accessibility devices natively support applications that comply with it. Using the WAI-ARIA standards ensures that we are in compliance with industry norms and can create an app that is usable by all students at the university, all without adding a large amount of unnecessary work to our agendas. When we begin developing our application at the start of next year, we will be adhering to WAI-ARIA standards wherever possible to ensure that the product we create is usable by everyone in the Oregon State community, something which the current Qualtrics solution does not do.

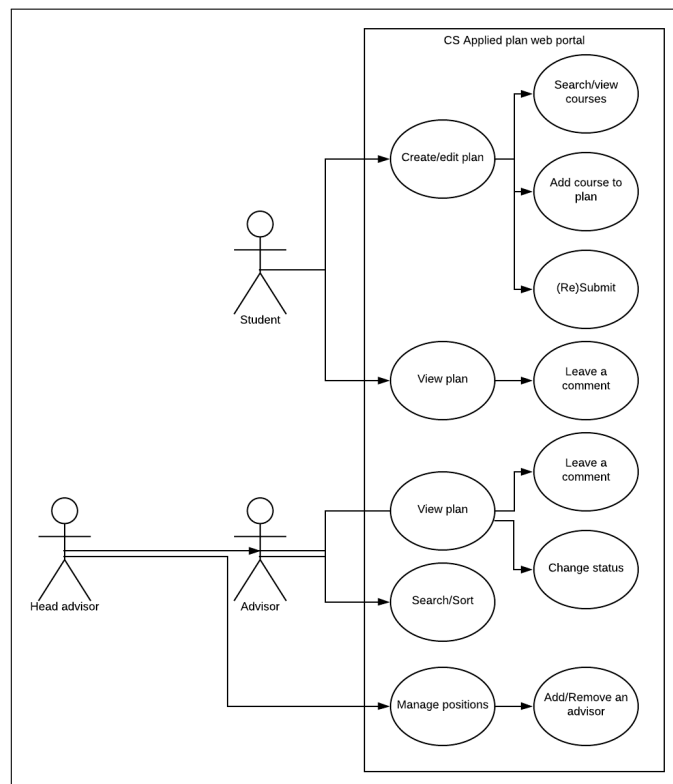


Fig. 7: System use case diagram.

## 4 DATA STORAGE AND HANDLING

Working with data is a critical part of our application. Students' Applied plans shall be securely stored and advisors shall be able to quickly look at plans and query similar plans. This means that this application shall use a database to help manage and secure data.

### 4.1 Database Model

Relational databases are made up of tables, each table is broken up into columns, which represent different types of data, and rows that represent separate entries. Each table also has a primary key, and these keys can be used to form a relationship between tables by referencing it in a query [4].

Relational databases are the most popular model, as they are reliable, predictable, and avoid storing redundant data [4]. The trade-off of using a relational database is its structured nature, which makes it less useful for storing random unstructured data [4].

The most important data that our database needs to store is very structured data about custom Applied plans. We might want to know which user created a specific Applied plan, and we also might want to know which advisors have reviewed the plan. Since we are storing very structured data and we care about some relationship between entities, it makes sense that we would use a relational database for our project.

### 4.2 Relational Database Management System

There are a number of possible Relational Database Management Systems (RDBMS) to choose from. Some of the more popular ones are MySQL, PostgreSQL, MariaDB, SQL Server, and SQLite [5]. For our purposes we will use MySQL as it is reliable, secure, and offers a comprehensive set of features [5].

### 4.3 Database Schema

The database will be divided up into six tables: Plans, Courses, Users, Comments, PlanReviews, and SelectedCourses.

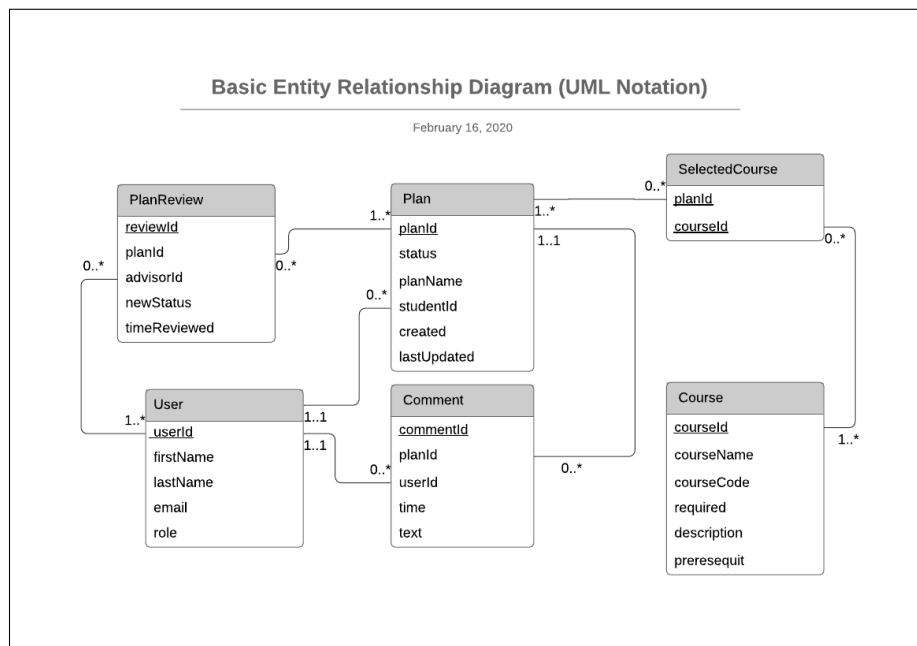


Fig. 8: Database entity relationship diagram.

#### 4.3.1 Plan

Each row in the Plans table represents a custom Applied plan created by a single student and that may have been reviewed by any number of advisors.

- **planId**: ID to uniquely identify the plan.
- **status**: This is an integer value that represents a specific status for a plan. 0 is "Rejected", 1 is "Awaiting Student Changes", 2 is "Awaiting Review", 3 is "Awaiting Final Review", 4 is "Accepted".
- **planName**: This is the name of the plan, decided by the student.
- **studentId**: ID to uniquely identify the student that created the plan.
- **created**: This is a timestamp of when the plan was created.
- **lastUpdated**: This is a timestamp of the last time the plan was changed.

#### 4.3.2 Course

Each row in the Courses table represents a different course that is offered at OSU.

- **courseId**: ID to uniquely identify the course.
- **credits**: The number of credits that this course is worth.
- **courseName**: The name of the course.
- **courseCode**: The course code (CS101, MTH250, etc.).
- **restriction**: Set to 0 if there is nothing stopping a student from taking this course. Set to 1 if the course is a required course for the CS applied option, and set to 2 if it is a graduate course (level 500+).
- **description**: Describes the course.
- **prerequisites**: Describes prerequisites for the course if there are any.

#### 4.3.3 User

Each row in the Users table represents a student or advisor.

- **userId**: ID to uniquely identify the user.
- **firstName**: The first name of the user.
- **lastName**: The last name of the user.
- **email**: The user's OSU email.
- **role**: The role of the user. 0 is "Student", 1 is "Advisor", and 2 is "Head Advisor".

#### 4.3.4 Comment

Each row in the Comments table represents a comment that has been left on a plan.

- **commentId**: ID to uniquely identify the comment.
- **planId**: The ID of the plan that the comment is attached to.
- **userId**: The ID of the user who created the comment.
- **time**: The time that the comment was created.
- **text**: The text content of the comment.

#### 4.3.5 PlanReview

Each row in the PlanReviews table represents a many-to-many relationship of users/plans. A plan may have many advisors review it and an advisor may review many plans.

- **reviewId:** ID to uniquely identify the review.
- **advisorId:** ID to uniquely identify the advisor who reviewed the plan.
- **planId:** ID to uniquely identify the plan.
- **newStatus:** The status value that this review changed the plan to.
- **timeReviewed:** The time that the review was created.

#### 4.3.6 SelectedCourse

Each row in the SelectedCourses table represents a many-to-many relationship of courses/plans. A plan may have many courses and a course may be used in many plans.

- **courseId:** ID to uniquely identify the course that is used in the plan.
- **planId:** ID to uniquely identify the plan.

### 4.4 Concerns About Course Data

To obtain course data for the database we will have to periodically make requests to the Office of the Registrar, these requests take a minimum of two weeks to complete, so they should be done well in advanced. This method is somewhat tedious, but it will ensure that course data does not become outdated. A new courses API is also being developed at OSU, but it may not be complete by the time that this application is finished. We will have to ensure that our application will accommodate such an API, so it can easily be used with this application when it is ready.

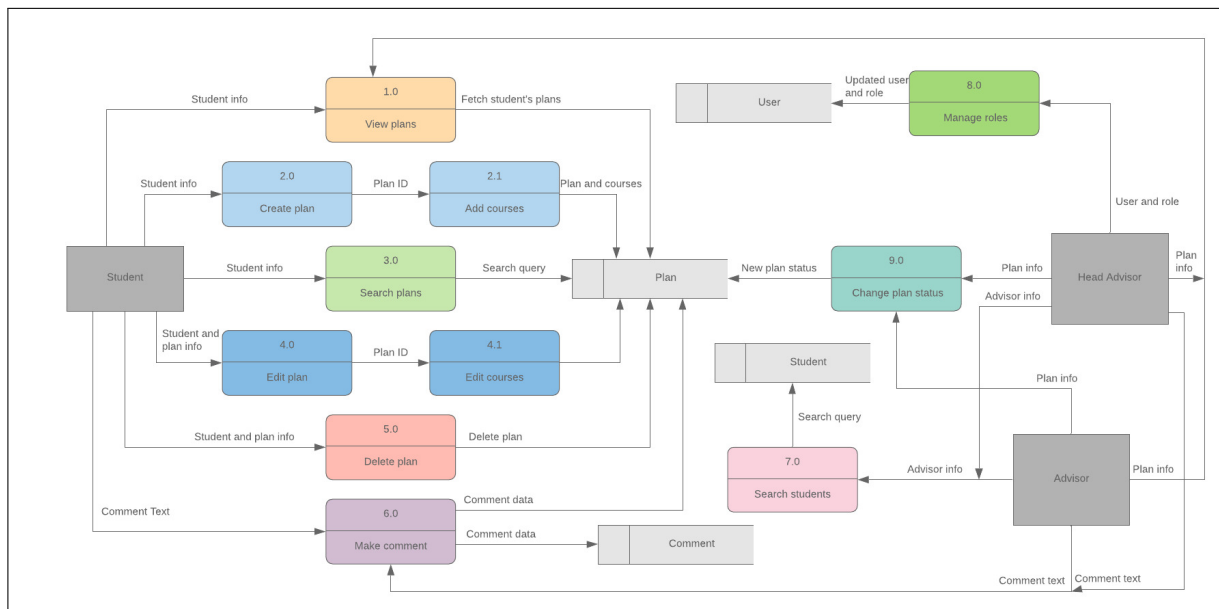


Fig. 9: Data flow diagram.

## 5 APPLICATION PROGRAMMING INTERFACES

An Application Programming Interface (API) allows two applications to communicate with each other, in most cases this will be a client communicating with a server [6]. Being able to simply request data from an API without having to implement a function for finding and processing the data will help to free up time and resources that can be spent on implementing other features of our application.

### 5.1 OSU APIs

For our application we will be using the APIs provided by OSU. The APIs we are interested in supply information about people, textbooks, and courses. All of the data handled by the APIs are also classified into one of three categories: "Unrestricted", "Sensitive", and "Confidential" [7]. Applications that have access to data must meet specific standards before they are allowed to access data from the "Sensitive" or "Confidential" categories [7]. For our application we will only use "Unrestricted" APIs.

#### 5.1.1 OAuth2

We will use the OAuth2 API to get access tokens that will allow our application to use the other APIs supplied by OSU [8].

TABLE 1: OAuth2[9, Tab. 1].

METHOD	TITLE	DESCRIPTION
POST	getAccessToken	Get an access token to make API calls

#### 5.1.2 Directory

We will use the Directory API to search LDAP directory entries using first / last name, email, or username [10]. Much of the information in this API will already be supplied by OSU's CAS, however this API will allow us to look up users without first requiring them to login to OSU's CAS. For example, if an advisor searches for a student that hasn't created any plans yet, we could still supply the advisor with that student's email and other information.

TABLE 2: Directory[11, Tab. 1].

METHOD	TITLE	DESCRIPTION
GET	getBySearchQuery	Get entries in directory matching all terms in search query
GET	getByOSUUUID	Get entries in directory filtered by query parameters

#### 5.1.3 Textbooks

We will use the Textbooks API to allow students to see which textbooks they will need for any of the courses that they may be interested in adding to their plan.

TABLE 3: Textbooks[12, Tab. 1].

METHOD	TITLE	DESCRIPTION
GET	getTextbooks	Retrieve textbooks by parameters

#### 5.1.4 Courses

At the time of writing this document OSU does not have a functioning Courses API. However communications with the OSU Development team has revealed that they are currently developing a Courses API. To accommodate this future API we will setup our application to first attempt to communicate with the API and if it does not get a valid response it will default to using the static course data in its database.

## 6 CENTRAL AUTHENTICATION SERVICE

We will be using a Central Authentication Service, which is a single sign-on protocol [13]. This protocol will allow our application to authenticate a user without requiring access to a users password. This is ideal for users as it prevents them from having to remember any additional passwords and it reduces the risk of their passwords being compromised [13].

### 6.1 Using a Central Authentication Service

This is how OSU suggests using CAS for a web application:

“Basic Steps:

- 1) A user comes to your web site.
- 2) Your web application redirects the client web browser to the OSU Login web page.
- 3) The user enters their username and password to login.
- 4) If they are successfully authenticated, OSU Login redirects the browser back to your web application and includes a Service ticket.
- 5) Your web application validates the Service ticket against the CAS endpoint.
- 6) Your web application creates a local session for the user and logs the user in.” [13]

### 6.2 Central Authentication Service Attributes

Once a user has been authenticated by OSU’s CAS a variety of attributes can be accessed using the “serviceValidate” endpoint [13]. These attributes will let us find out information about the user that can be automatically applied to custom Applied plans. Additionally, we can use the email attribute to make it easier for students and advisors to contact each other.

- “uid - username
- UDC\_IDENTIFIER - Unified Digital Campus ID, a Banner unique ID
- lastname - last name
- firstname - first name
- fullname - full name
- email - primary email address
- osuuuid - LDAP attribute ‘osuuuid’, used as a unique key in LDAP
- osupidm - OSU Banner PIDM
- eduPersonPrincipalName - a scoped user identifier value “username@oregonstate.edu”
- eduPersonPrincipalNamePrior - prior values of eduPersonPrincipalName, if it has changed
- eduPersonPrimaryAffiliation - the user’s primary affiliation with OSU



- eduPersonAffiliation - all of the user's affiliations with OSU
- osuprimarymail - primary email address
- commonName - full name
- surname - last name
- givenName - first name" [13]

## 7 DEVELOPMENT

### 7.1 Component-Level Development

In light of the resources found in the User Interface Design section, we believe it is most prudent to develop pages modularly, creating individual components first, then assembling them into pages. Component-level development helps to ensure that identical functionality in different pages does not need to be created twice; each instance can simply be added in from a central repository of modules. There are a number of solutions in use throughout the industry, some nascent, others established. Our tentative choice is to use React, a virtual-DOM component library created by Facebook, Inc. First released in 2013, React is a fast and lightweight[14] library for building single page apps. React has become increasingly popular in recent years, and currently holds a 78.1% market share[15]. In addition to having a significantly larger market share than competitors like Angular, Vue.js, and LitElement, React is the front-end component framework with which the members of our team are most familiar. Additionally, React's large market share means it has more community-based support, which should make it easier to overcome technical roadblocks when using it. Given this, we believe that React is the best tool out of the ones summarized above, and have tentative plans to use it for the project. Because development is not planned to start until the start of next year, our choice may change in the interim, but we believe this is highly unlikely.

### 7.2 Containerization

To make the development process easier, we plan to use Docker to containerize the project and all of its dependencies [16]. However, until we develop a clear design of what the web application should be and what technologies should be involved, Docker is an optional technology. We still would like to record it in this document, though, so the team can refer to this section when we decide to use containerization in this project.

### 7.3 Hosting Platform

The Applied plan Portal will be hosted as a cloud application. With that being said, it is important to consider the different platforms with which it can be hosted on. One of the most popular PaaS companies is Heroku. It allows for quick and effective building, deploying, and scaling of web applications[17]. It includes a fully flexible runtime environment with dynos, which are virtual computers that can be powered up or down. It also allows for manual horizontal and vertical scaling, rollbacks to the database code, and a monitoring system to keep track of metrics like throughput, response time, and memory[17]. Although there are viable alternatives, Heroku is the best option for hosting the Applied plan Portal because it has a usable command line interface, it allows for development in any language, it helps focus on innovation, not operations, and it offers plenty of supportive tools[17].

## 8 MOBILE SUPPORT

While our application is intended to work on desktop and laptop computers, both the team and the client would like to see this application fully supported on mobile devices. A mobile app will allow both students and advisors to see updates to plans while away from home or work, expediting the approval process.

### 8.1 Mobile App Type

One of the most important decisions we had to make for mobile development is if we want to create a mobile web app, a native app, or a hybrid of the two. This choice greatly impacts the time we spend developing the app, the features that are available, and the number of devices that will be supported.

We ultimately settled on creating a mobile web app as it is the quickest of the three options and it will also give us the most supported devices [18]. The trade-off is that we will not have access to many of the device specific features that would be available when using a native app [18].

### 8.2 Mobile Web App Constraints

#### 8.2.1 Mobile Browser Limitations

Some features that work on standard web browsers will not work on mobile web browsers [18]. We will need to avoid relying on the HTML5 <video> tag auto-play feature, the opacity property, and the transition property as they are unsupported and may exhibit undefined behavior [18].

#### 8.2.2 Screen Resolutions

There are a wide variety of screen resolutions for mobile devices, accommodating all of these resolutions is not a trivial task [18]. We will support both portrait and landscape views for devices [18], we will also need to support Apple's Retina displays. Retina displays generally have about double the resolution of similarly sized screens, and as a result, lower resolution images often appear blurry or pixelated [18]. To combat this, we will serve higher resolution images to Retina devices when they request images, in addition to using vector graphics wherever possible.

#### 8.2.3 Optimization

Users of mobile devices are very concerned with response time and will actively avoid apps with long load times [18]. This is why it is critical to consider areas where performance can suffer. Image load times can greatly hinder performance, we will need to find a balance between image size and quality to ensure the best user experience possible [18]. Waiting on database queries can also lead to a poor user experience, when appropriate we will use server-side caching to expedite the process of retrieving data [18].

## 9 CONCLUSION

This document gives an in-depth view of the architecture and visual design of the front and back ends of the OSU Applied plan Portal web application. It will utilize React, a JavaScript library to create reusable components and follow the standards outlined by the OSU branding guidelines to create a usable, WAI-ARIA standard application. Our team will use MySQL to create a relational database, including tables for plans, courses, users, comments, plan reviews, and selected courses. It will connect to Oregon State APIs to supply information about people, textbooks, and eventually courses. The entire app will be hosted on Heroku and will utilize OSU's central authentication service to grant students and advisors access to the portal.

## REFERENCES

- [1] Oregon State University Relations and Marketing. (2019). The Oregon State Brand, [Online]. Available: <https://communications.oregonstate.edu/brand-guide> (visited on 11/03/2019).
- [2] World Wide Web Consortium. (2016). WAI-ARIA Overview, [Online]. Available: <https://www.w3.org/WAI/standards-guidelines/aria/> (visited on 11/03/2019).
- [3] World Wide Web Consortium, J. Diggs, S. McCarron, M. Cooper, R. Schwerdtfeger, and J. Craig. (2017). Accessible Rich Internet Applications (WAI-ARIA) 1.1, [Online]. Available: <https://www.w3.org/TR/wai-aria/> (visited on 11/03/2019).
- [4] W. Hempel. (2018). How to choose a database in 2018, [Online]. Available: <https://arcentry.com/blog/choosing-a-database-in-2018/>.
- [5] J. Germain. (2019). Choose your rdbms flavor: Mysql or a better clone, [Online]. Available: <https://medium.com/linode-cube/choose-your-rdbms-flavor-mysql-or-a-better-clone-48dacb52a081> (visited on 11/14/2019).
- [6] MuleSoft. (2019). What is an api? (application programming interface), [Online]. Available: <https://www.mulesoft.com/resources/api/what-is-an-api> (visited on 10/31/2019).
- [7] Oregon State University. (2019). University data management, classification, and incident response, [Online]. Available: <https://is.oregonstate.edu/policies/university-data-management-classification-and-incident-response> (visited on 11/01/2019).
- [8] —, (2019). Frequently asked questions, [Online]. Available: <https://developer.oregonstate.edu/faq-page> (visited on 11/01/2019).
- [9] —, (2019). OAuth2, [Online]. Available: <https://developer.oregonstate.edu/oauth2/apis> (visited on 11/01/2019).
- [10] wutso. (2018). Andrew's student developer experience, [Online]. Available: <http://blogs.oregonstate.edu/developer/> (visited on 11/01/2019).
- [11] Oregon State University. (2019). Directory, [Online]. Available: <https://developer.oregonstate.edu/directory/apis> (visited on 11/01/2019).
- [12] —, (2019). Textbooks, [Online]. Available: <https://developer.oregonstate.edu/textbooks/apis> (visited on 11/01/2019).
- [13] —, (2019). Cas (central authentication service) information, [Online]. Available: <http://onid.oregonstate.edu/docs/technical/cas.shtml> (visited on 11/15/2019).
- [14] S. Martin. (2019). Angular vs React vs Vue: Which is the Best Choice for 2019? [Online]. Available: <https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847> (visited on 11/03/2019).
- [15] TechMagic. (2019). React vs Angular vs Vue.js — What to choose in 2019? [Online]. Available: <https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d> (visited on 11/03/2019).
- [16] Docker. (2019). Why docker? [Online]. Available: <https://www.docker.com/why-docker>.
- [17] Guru99. (2019). Heroku vs aws: 10 most important differences you must know! [Online]. Available: <https://www.guru99.com/heroku-vs-aws.html#2>.
- [18] T. Agrimbau. (2019). Developing mobile web applications: When, why, and how, [Online]. Available: <https://www.toptal.com/android/developing-mobile-web-apps-when-why-and-how> (visited on 11/01/2019).