OSU
Oregon State
UNIVERSITY
College of Engineering

CS Capstone Requirements Document

April 20, 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 outlines the client requirements for the OSU CS Applied Plan Portal project. This web application will allow students to create custom applied plans, EECS advisors to provide input on proposed custom applied plans, and the EECS Head Advisor to approve plans. This document ensures that both the client and the Portal Team agree on what the final product should achieve and provides methods to verify that the product adheres to these requirements.

CONTENTS

1 Introduction

The OSU CS Applied Plan Portal aims to be a full-featured, easy-to-use, and aesthetically ~~appealing~~ pleasing web application. This application will allow students to create custom applied plans, EECS advisors to provide input on proposed custom applied plans, and the EECS Head Advisor to approve plans.

1.1 Purpose

This document outlines the agreed upon requirements for the OSU CS Applied Plan Portal project.

1.2 Scope

This document describes the complete scope of the OSU CS Applied Plan Portal project. This includes both the ~~front-end and back-end~~ front end and back end of this web application.

- **~~Front-end~~ Front end (Web Client):** This is the part of the application that the user interacts with. It is primarily composed of HTML, CSS, and JavaScript. The front-end is further divided into a student interface and an advisor interface.
- **~~Back-end~~Back end:** This includes the server, the database, and the parts of the application that help to support the front-end.

1.3 Product overview

1.3.1 Product perspective

The user interface should be easy to navigate on ~~both standard monitorsand standard mobile devices~~most standard monitors. Users should also be able to clearly read text and see all relevant elements on any modern screen. Interactive elements must be easy to select either with a mouse or by touch screen. The most common display ~~ratios~~ sizes that this application must work with are ~~16:9, 9:16, 8:5, 5:4, 4:3, 3:5, 3:4, and 2:3.~~ 1024x600, 1024x768, 1280x768, 1280x950, 1366x768, 1440x900, 1600x900, 1680x1050, 1920x1080, and 1920x1200.

1.3.2 Product functions

The OSU CS Applied Plan Portal has two different user interfaces. The user interfaces that is used depends on if the user is a student or an EECS advisor. Additionally all users will have access to a navigation bar throughout the application and will be required to login using their ONID credentials.

- **Student ~~View~~view:** When a student logs into their account, they are recognized as a student ~~are~~ and brought to the student homepage.
  - **Homepage:** From the homepage, students are able to create new applied plans, view their ~~custom~~ created applied plans, and ~~see~~ check if they have received any new feedback on their submissions.
  - **Create ~~Plan~~plan:** When creating a new plan, students select which courses they would like to include~~in their plan~~. Invalid courses ~~are filtered out, attempting to select~~ will not be able to be added to a plan; attempting to add an invalid course will ~~show~~ display an error explaining why the course is ~~not applicable~~incompatible with the plan. Users may also view detailed information about specific courses before making a selection.

- **View ~~Plan~~plan:** Once a student has created at least one plan they can can view that plan and its status. When viewing a plan, students can see any comments that an advisor has left on their plan~~, they can~~. They can also leave their own comments, modify the courses in their plan, ~~or~~ and see the current status of their plan. Each plan will have one of the following statuses: "Awaiting ~~Initial~~ Review," "Awaiting Final Review," "Awaiting Student Changes," "Rejected," and "Accepted". A student will be unable to edit a plan if it has the statuses "Awaiting Final Review", "Accepted", or "Rejected". A student will be unable to comment on a plan if it has the status "Accepted" ~~.~~ or "Rejected".

* **Advisor ~~View~~view:** When an advisor logs into their account, they are recognized as an advisor and ~~they interact with~~ will be brought to the advisor homepage (search plans page).
  - **Homepage / Search plans:** From the homepage advisors are able ~~to see all of the student plansthat they are currently working on, as well as notifications for any new responses that they have received. There is also an option to~~ search for plans~~.~~
  - ~~**Search:** Advisors can search for plans by any of the statuses: "Awaiting Initial Review", "Awaiting Final Review", "Awaiting Student Changes"~~, ~~"Rejected", or "Accepted". Advisors can also search for a plan by a students name or student ID number.~~ they are also able to filter out plans based on the plan status. The search field will accept partial or complete plan names, user names, and user IDs.
  - **View ~~Plan~~plan:** When an advisor views a plan~~they are able to~~, they can leave comments, ~~mark specific courses as requiring a change,~~ edit the plan, or change the ~~plans status(only~~ plan's status. Only the head advisor can set the status of a plan to "Accepted" or change the status of a plan ~~to/from Acceptedor Rejected~~that has already been set to "Accepted" or "Rejected"). When an advisor views ~~the plan~~a plan, they can also see if a similar plan (~~same collection of~~ one with the same courses) has been accepted or rejected ~~previously.~~ in the past.
  - **Manage user roles:** Only the head advisor should have access to the "Manage User Roles" page. Here, the head advisor can search for users by username or user ID. Once the head advisor has found a user, they can see and change the user's role to one of "Student", "Advisor", and "Head Advisor".

* **Login system:** All users will have to log in to the application using their ONID account. Our application must never have access to any user's ONID password. To prevent the storage and checking of passwords our application must interface with OSU's Central Authentication Service.
* **Navigation bar:** All users will be able to see the navigation bar while navigating the application. Depending on the user's role, different buttons will be shown on the navigation bar.
  - **Log out:** All users will be able to log out of their current session.
  - **Notifications:** All users will be able to see a list of recent notifications. For students, this will include notifications about comments and status changes made to plans that they have created. For advisors, this will include notifications about comments or status changes to plans that they have begun to review (by commenting or changing the status of a plan).
  - **History:** Advisors will be able to see a list of the last five plans that they have opened using the history tooltip in the navigation bar.
  - **Manage roles:** The head advisor will have a button in the navigation bar that allows them to navigate to the manage roles page that will allow them to change users roles.

- **Update courses:** The head advisor will have a button to check and update courses in the database. These will be the same courses that students will see when creating a plan.

1.3.3 User characteristics

There are two main user groups.

- **CS Students:** These are undergraduate students that are comfortable with software and technology. While these students are unlikely to have issues with navigating a web application, they may have limited understanding of the courses and plans that offered to CS students.
- **EECS Advisors:** These are employees of OSU that advise CS students. These advisors are likely to have a firm understanding of software, technology, and the courses and plans that are offered to CS students at OSU.

1.3.4 ~~Limitations~~

~~Since the OSU CS Applied Plan Portal will accept and store student data it is critical that FERPA laws are adhered to and that student privacy is a primary focus of this application. Also, we will have to perform informal user studies as formal studies are beyond the scope of this course.~~

2 Specific Requirements

This section will provide in-depth descriptions of what the application shall do and how it works~~to achieve~~, in order to provide a better user experience ~~for~~ to EECS advisors and students.

2.1 External interfaces

External interfaces of the application ~~are~~ constitute its inputs and outputs. The web application shall support the following external interfaces:

1) **Inputs**

   - **Mouse events:** This type of input allows the users to navigate to various sections on the web page. The source of this type of inputs can be either a hardware pointing device (primarily on desktops) or the touching and dragging actions by the users (primarily on mobiles). The website shall have pixel-level input accuracy, though ~~it should have a comfortable layout to tolerate a~~ the layout should enable comfortable navigation with a reasonable margin of error in user input.
   - **Keyboard events:** This type of input allows the users to navigate throughout the web page ~~as long as to~~ and submit written responses or text search requests. The source of this type of ~~inputs~~ input can be either a hardware keyboard (primarily on desktops) or a ~~software~~ software-driven on-screen keyboard (mainly on ~~mobiles~~mobile devices).

2) **Outputs**

   - **Visual outputs:** This type of output is the primary source of output from the application. Therefore, the application shall have fast response time and shall display the mouse cursor and/or text cursor so the users can know where they are on the web page.

2.2 Functions

The application shall perform the following fundamental actions to process the inputs and produce the output:

1) **Login/Logout~~/Registration~~**

   - The application shall ~~perform an input validation, including, but not limited to, null-checks on required fields, duplicate checks, and unmatched passwords checks, and shall reject any bad or malicious request.~~
   - ~~The application shall use OSUAPIs to compare the provided credentials with existing credentials in the database and determine the validity of the login /logout /registration process~~ use OSU's Central Authentication Service to manage the login and logout process so that our application never has access to user's ONID passwords.
   - If the login/registration process ~~was~~ is successful, the application shall redirect the user to the ~~dashboard~~homepage. Otherwise, the application shall inform the user of the issues ~~and redirect back to the login /registration~~ with their input and remain on the login page.
   - If the logout process ~~was~~ is successful, the application shall log the user out and redirect to the ~~login page.~~
   - ~~Edge cases are handled in the input validation step, and since this shall be a secured operation, any abnormal request shall be rejected.~~ successful logout page.

2) **Dealing with permissions based on roles**

   - ~~The application shall have a role for each type of users. Only administrators~~ Each user must have only a single role (Student, Advisor, or Head Advisor).
   - Only head advisors shall have permissions to manage ~~the~~ user roles.
   - ~~The application shall configure the roles such that a student can have access only to his/her Applied plan, and an advisor can have access only to the Applied plansof students they are advising~~A student will only have access to their own applied plans.
   - ~~An edge case would be a user having multiple permissions. To prevent this, each user shall only have one role.~~
   - ~~The application shall output a "Forbidden Error" (code 403) response whenever users try to access something they should not~~Only advisors and head advisors are able to search for applied plans.

3) **Creating Applied plans**

   - The application shall ~~ensure that the current user is a student or an administrator, otherwise shall output a "Forbidden Error" (code 403) response and stop the operation~~only allow students to create applied plans, otherwise the operation will be aborted.
   - A student will create a plan by searching for and selecting courses for their plan and providing a name for the plan.
   - When a student sends a request to create an ~~Applied~~ applied plan, the application shall receive it and validate the request. As per the usual rule, the application shall ~~stop~~ abort the operation and inform the user of the issue if the validator rejects the request.
   - ~~The application shall perform a sequence of steps to collect the student's response to create a draft for his/her Applied plan~~ A plan will not be created if any constraints are violated. The minimum required

constraints are as follows: A plan must have at least 32 credits. A plan must not have graduate or required courses. All courses in the submitted plan must be offered at OSU. ~~This process is quite complicated depending on the type of the student's Applied plan . What needs to be included in this sequence of steps will be continually discussed and optimized by the EECS advisor that the team works with.~~

- After a successful creation, the system shall update the student's profile and ~~show the new plan on his/her dashboard~~redirect them to the plans page, where they will be able to leave comments or view the status of the plan. If the student returns to their homepage they will see their new plan listed on the homepage.
- Any failure in the process shall make no changes to the system ~~but redirect the user back to the dashboard and include~~ and send a feedback message explaining the error, instead.

4) **Viewing Applied plans**

- ~~The application shall ensure that the current user is a student or an advisor or an administrator, otherwise shall output a "Forbidden Error" (code 403) response and stop the operation~~Advisors are allowed to view any plan.
- The application shall allow ~~a student to view his/her—and only his/her—Applied plan~~students to view their—and only their—applied plans.
- The application shall allow advisors to view ~~their students' Applied~~ students' applied plans and give feedback on ~~it~~them.
- When the user sends a request to view an ~~Applied~~ applied plan, the application shall perform validation ~~. If~~ on the request. If the validation is successful, the system shall fetch the ~~requested Applied~~ specified applied plan from the database, return it to the client, and display it on the screen.
- ~~An edge case in this operation is a student having no Applied plan. In this case, the application shall display a "No Applied plan" message on the dashboard.~~
- Any failure in ~~the process~~ attempting to view a plans page shall make no changes to the system~~but redirect the user back to the dashboard and include a feedback message instead~~, but show the "Page Not Found" (HTTP 404) error.

5) **Searching Applied plans**

- The application shall ensure that the current user is an advisor~~or an administrator, otherwise shall output a "Forbidden Error" (code 403) response and stop the operation~~, otherwise the user will be shown an error message stating that they lack permission to search for plans.
- The application shall allow advisors to search for ~~their students' Applied plans and perform a viewingaction on it~~students' applied plans and open them for viewing.
- When the user sends a request to ~~view an Applied plan~~search for applied plans, the application shall perform validation. If validation is successful, the system shall search ~~for Applied plans from the database~~ the database for applied plans based on the search query and display the ~~result~~ results on the screen.
- An edge case in this operation is an empty search result. In this case, the application shall display a "No ~~Applied plan~~plans found" message on the dashboard.
- Any failure in the process shall make no changes to the system ~~but redirect the user back to the dashboard and~~ and will include a feedback message instead.

6) **~~Retrieves student information~~Reviewing applied plans**

- The application shall ensure that the current user is an advisor or ~~an administrator, otherwise shall output a "Forbidden Error" (code 403) response and stop the operation~~head advisor, otherwise the user will be show an error message outlining that they lack the permission to change a plans status.
- The application shall allow ~~an advisor to view students' Applied plans and give feedback on them.~~
- ~~When the user sends a request to view~~ advisors to make a decision to change the status of a student's ~~Applied plan~~ applied plan draft.
- When the advisor changes an applied plan's status, the system shall notify the student who created the plan about the change in status, and will update the plan's status.
- Advisors will have limited control over the statuses to which they can set a plan. Advisors cannot change the status of a plan that is already "Rejected" or "Accepted." They also are unable to set a plan to the "Accepted" status; this is solely the responsibility of the head advisor. Instead, advisors can set a plan's status to any of: "Awaiting Student Changes", ~~the application shall perform permission checks on the user and then try to retrieve the student's information from the database~~"Awaiting Review", and "Awaiting Final Review".
- ~~Any failure in the process shall make no changes to the system but redirect the user back to the dashboard and include a feedback message instead.~~
- ~~**Accepting and rejecting Applied plans**~~
- ~~The application shall ensure that the current user is an advisor or an administrator, otherwise shall output a "Forbidden Error" (code 403) response and stop the operation.~~
- ~~The application shall allow advisors to make a decision to accept or reject a student's Applied plan draft.~~
- ~~When the user accepts or rejects an Applied plan, the system shall make changes to the student's record to update the plan result~~Head advisors can set any plan to any status.
- Any failure in the process shall make no changes to the system ~~but redirect the user back to the dashboard and include~~ and return a feedback message~~instead~~.

2.3 Usability requirements

One of the most important attributes this application must have is usability. To ensure that the application is easy to use, it must satisfy the following qualities:

1) **Learnability:** Students and advisors should take at most 3 hours to learn how to use the external interfaces of the website. All components shall have ~~articulate~~ descriptive names and possibly ~~descriptions~~ details of what they do, if necessary.
2) **Friendliness:** The user interface's design shall follow popular design ~~concepts that have been known for creating~~ standards that the industry at large believes create a friendly environment ~~to the~~ for users. The application shall be as interactive as possible and shall give feedback on ~~every operation~~ operations that the user performs, when relevant.
3) **Productivity:** The application shall surpass the current web-form planner in terms of productivity—the average time spent to build an ~~Applied~~ applied plan shall be considerably shorter, and ~~the averagefrustration shall be~~

~~much less (or even better, none)~~users, on average, should self-assess their frustration to be lower when using our system compared to the current solution.

4) **Manageability:** The application shall keep track of every student's ~~Applied plan. It shall save all old drafts of a plan and allow the students to continue building the latest version of their plans~~applied plans. Only if a plan is manually deleted should it no longer be kept in the database.

2.4 Performance requirements

- The application shall be scalable and be able to simultaneously serve at least 300 users.
- 95% of ~~the~~ responses shall be ~~less than~~ returned within 3 seconds ~~after the requests are~~ of the request being sent.
- 95% of the responses from the back-end shall be ~~asynchronous—meaning the responses will be~~ displayed on the front-end without the user having to reload the web page.
- The average time it takes for a student to complete his/her ~~Applied plan shall be approximately twice faster than that of the current web-form~~ applied plan (including time spent searching for courses) using our solution shall be faster than when using the current form-based planner.

2.5 Logical database requirements

To retrieve and update the information of different entities in the system, the application must store information in ~~relational databases, as shown with the following entities~~a relational database, with at minimum the following tables and fields:

1) **~~Student~~User:** Includes ~~student~~ user ID number, first name, last name, ~~date of birth, contact information, major(s), their preferred advisor(s), their Applied plan draft(s), grades, course history, etc. The information about a student is used frequently to determine who are on track on their Applied plan. This data shall be accessible only to the students themselves and to the advisors. Students shall have a many-to-many relationship with advisors, meaning a student can have many advisors, and an advisor can advise multiple students.~~ email, and the users role.

2) **~~Advisor~~Plan:** Includes ~~advisor ONID, first name, last name, contact information, specialization(s), list of students they are advising, etc. The information about an advisor is used quite often to help students build their Applied plans. This data shall be accessible publicly depending on the property of this entity (e.g. students they are advising should only be visible to the system and the administrators). Advisors shall have a many-to-many relationship with students, as explained in the paragraph above.~~

3) ~~**Plan:** Includes the corresponding student's ID , plan progress, related drafts, reviewing advisor(s), etc. The information about an Applied plan is used quite often to help the student and the advisor(s) succeed in building the~~ the plan name, status, time created, time last updated, and the user ID of the student that created the plan. This data shall be accessible only by the student ~~and his/her advisor(s)~~who created the plan and advisors. Plans shall have a many-to-one relationship with students, as a student can build multiple ~~Applied plan~~applied plans, but an ~~Applied~~ applied plan is associated to only a single student.

4) **Course:** Includes courses ID number, name, course code, description, and number of credits. Users will have access to course data when creating or viewing plans.

~~All data shall be FERPA-compliant and shall be stored in a stable database management system (DBMS) and shall have backups in case of emergency. Finally, note that the entities and their properties as well as the relationships between the entities are subject to change as the team sees fit as the project moves forward.~~

2.6 Design constraints

~~We are restricted to create an interface for~~ This application must be usable on desktop PCs and laptops. ~~We are restricted by FERPA to protect the privacy of student records when we are developing this portal. We have to monitor the way we handle data as to not interfere with any of the privacy laws~~This application must be designed to easily be intuitive and easy to use with all of the following screen resolutions: 1024x600, 1024x768, 1280x768, 1280x950, 1366x768, 1440x900, 1600x900, 1680x1050, 1920x1080, and 1920x1200.

2.7 Software system attributes

The portal will include the following attributes in order to ensure the system is user-friendly and maintainable.

1) Reliability - Once it is launched, the system should be available through the EECS website and should be maintained in good quality.
2) Availability - This system should be made available on the OSU EECS website so both students and advisors can access the portal.
3) Security - The portal shall use OSU ONID authentication with the two step Duo login to manage student accounts.
4) Maintainability - The system will be maintained by this team until it is passed off to the EECS advisors and Rob Hess. ~~The portal~~ Our team will conduct sufficient testing of the final system before passing it off to the advising team.
5) Portability - This portal will be made available to desktop PCs and laptops. ~~If time allows, it will also be accessible through mobile devices.~~

2.8 Supporting information

This portal ~~solution solves the problem of creating a custom applied plan~~ solves the current problems with creating custom applied plans at Oregon State, namely that the existing solution is complicated and time consuming to use. The current system is clunky, inefficient, and non-interactive. ~~This~~ Our project will make it easier for students to create their plan and track their progress toward graduation, while also reducing the amount of work required by advisors to review applied plans.

3 Verification

3.1 External Interfaces

As a web application, our product will likely be used with mouse and keyboard in the majority of cases, but we should also take care to opportunistically consider the site's accessibility to impaired users who ~~, for instance, might navigate entirely withthe keyboard or while using a screen reader. With this in mind, verification of our external interfaces will overlap considerably with our usability verification.~~ may have trouble seeing or interacting with smaller visual elements. We will demo the application for our client and get confirmation that our application is easy to navigate and interact with. Buttons and links should not require advanced fine motor skills to click.

3.2 Functions

~~Our testing of functionality will be largely automated, using a system like Jest. js. For a larger, multi-platform application like ours, we believe it is much better to use a testing library instead of unreliable and laborious human testing.~~ We will demo the application's functionality for our client. The following functionality must be present and fully functional during the demo.

- Users must be able to sign into the application using their ONID credentials.
- Users with the student role must be able to create a new plan and see it listed on their homepage.
- Users with the student role must be able to view any of their plans.
- Users with the advisor role must be able to search for a plan.
- Users with the advisor role must be able to change the status of a plan (keeping in mind that normal advisors have restrictions on the statuses they can change/set).
- Users with the head advisor role must be able to change a plans status to "Accepted".
- Users with the head advisor role must be able to change any users role.

3.3 Usability Requirements

We will ~~perform two phases of informal user testing to ensure that our applicationmeets our usability requirements.~~ hold informal user tests. The users will only know the specific tasks that we will give them to perform using the application; they will not know the layout or look of the application before testing it. We will also have the users attempt to create a plan using OSU's current form for submitting applied plans. We will time both attempts, including the time it takes to look up courses to add to the plan. We will also have users fill out a user experience rating for each method which will let us know which method users prefer to use.

- On average tested users should take less time to create an applied plan using our application.
- On average tested users should prefer to create a plan using our application.
- On average tested users should be able to re-perform a task in in less time than the first attempt when using our application.

3.4 Performance requirements

The performance requirements will be verified by having the development team test the system with the client present. Response times will be recorded ~~, the scalablility~~ and the scalability requirement will be ~~tested by simulating high traffic, and the average completion times for the web application vs web-form will be tested by attempting one method and then the next and then comparing the times.~~

3.5 ~~Logical database requirements~~

~~We will review the database with our client to ensure that the structure and contents of the databasematches the logical databaserequirements.~~ assessed using the following formula:

- (number of CPU cores / Average Page Response Time in seconds) * 60 * User Click Frequency in seconds = Maximum simultaneous users

We will be using a Heroku server for deployment with access to at least 8 CPU cores and we will want to be able to handle no less than 300 simultaneous users. We also estimate that users will perform a operation that will make a request to our API server on average every 5 seconds. The updated formula:

- (8 / Average Page Response Time in seconds) * 300 >= 300

So our application, on average, must be able to respond to user requests in 8 seconds or less to be able to support 300 simultaneous users.

3.5 Logical database requirements

Simply viewing the database using PHPMyAdmin or viewing the db-init.sql file in our GitHub repository will allow us to prove that the minimum required tables and fields are being used in our database. Additionally, our database must be accessible and function properly while the application is running. To prove that the user, plan, and course tables are being accessed correctly, we can create a plan and then view it. The user information on the plan is taken from the active user that is stored in the database. The plan page itself is generated from the plan in the plan table, and the listed courses in the plan are taken from the course table.

3.6 Design constraints

We will test the application on a variety of ~~desktop PCs and laptops~~ common desktop resolutions to ensure that the application does not have any unforeseen responsiveness or usability issues. We will ~~also review FERPA laws in regards to any sensitive data we are working withto ensure no violations occur.~~ seek to support the following resolutions:

- 1024x600
- 1024x768
- 1280x768
- 1280x950
- 1366x768
- 1440x900
- 1600x900
- 1680x1050
- 1920x1080
- 1920x1200

Since we have limited access to monitors to test with, we will use our browsers' developer tools to simulate these resolutions.

3.7 Software system attributes

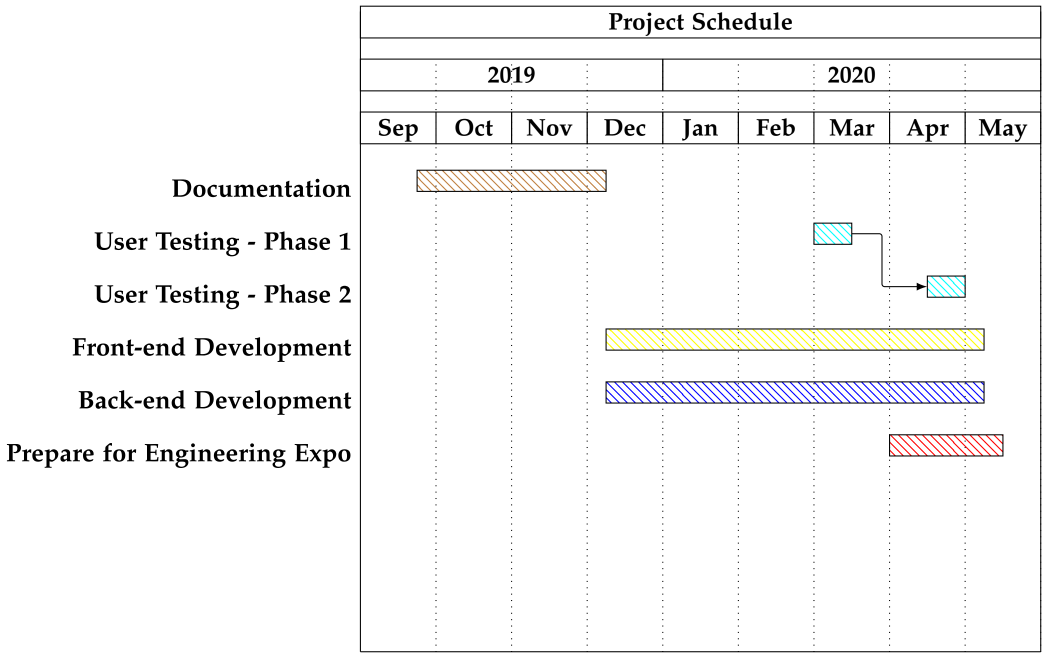
~~The development team will confirm that the reliability, availability, and security attributes are met by confirming that the system is available through the EECS website, and that~~ We will test that users are able ~~to login using~~ login using their ONID credentials. We will also ensure that once a user has logged out, a new login must take place before users can once again navigate to the application. Users who login to the application must also be able to see their name and email on plans they create without every having to enter them into the application (this user data is provided to the application by OSU's Central Authentication Service).

4 Appendices

4.1 Acronyms and abbreviations

- **API:** Application Programming Interface.
- **CFR:** Code of Federal Regulations.
- **CS:** Computer Science.
- **CAS:** Central Authentication Service.
- **CSS:** Cascading Style Sheets.
- **DBMS:** Database Management System
- **EECS:** Electrical Engineering and Computer Science.
- **FERPA:** Family Educational Rights and Privacy Act.
- **HTML:** Hypertext Markup Language.
- **URL:** Uniform Resource Locator.
- **OSU:** Oregon State University.
- **ONID:** Oregon State Network ID.

5 Project Schedule


Project Schedule
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Documentation
User Testing - Phase 1
User Testing - Phase 2
Front-end Development
Back-end Development
Prepare for Engineering Expo