ISU POS Design System

DESIGN DOCUMENT

Team 4

Tina Prouty (Adviser) Maruf Ahamed (Academic Advisor)

Elisabeth Bair (Meeting Facilitator), Andrew Goluch (Frontend Lead), Lucas Bell-Steckel (Backend Lead), William Peng (Full-Stack Coordinator), Cavin Leeds (Test Engineer), and Thomas Hotard (Report Manager)

sddec21-04@iastate.edu

https://sddec21-04.sd.ece.iastate.edu/

Revised: April 21st, 2021 Version 3.0

Executive Summary

Development Standards & Practices Used

Life cycle management, software testing standards, and development environment considerations were used.

Summary of Requirements

- Support the creation of programs of study for Electrical, Computer, Software, and Cyber Security Engineering majors
- Require students to accurately complete a program of study with the goal of graduation
- Provide warnings for missing classes or courses being added without prerequisites
- Enable students to edit their program of study
- Allows advisors to add new classes to the program
- Classes are able to be labeled as required, based on a given catalog year
- Individual courses are able to have prerequisites that must be met within the POS
- Advisors must be able to approve exceptions to general requirements
- Advisors are able to give feedback to a student's POS

Applicable Courses from Iowa State University Curriculum

Courses most applicable to this project are COMS 309, SE 329, and COMS 363. Other courses have added to general problem-solving skills and are indirectly applicable.

New Skills/Knowledge acquired that was not taught in courses

We project to acquire new knowledge and in JavaScript, MySQL, and HTML/CSS. As well as acquire a better knowledge of scripting to pull data from websites using Python.

Table of Contents

1 Introduction **6**

1.1 Acknowledgment 6

1.2 Problem and Project Statement 6

1.3 Operational Environment 6

1.4 Requirements 7

1.5 Intended Users and Uses 7

1.6 Assumptions and Limitations 7

1.7 Expected End Product and Deliverables 8

Project Plan **9**

2.1 Task Decomposition 9

2.2 Risks And Risk Management/Mitigation 9

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 10

2.4 Project Timeline/Schedule 11

2.5 Project Tracking Procedures 11

2.6 Personnel Effort Requirements 12

2.7 Other Resource Requirements 13

2.8 Financial Requirements 13

3 Design **13**

3.1 Previous Work And Literature 13

3.2 Design Thinking 14

3.3 Proposed Design 15

3.4 Technology Considerations 16

3.5 Design Analysis 16

3.6 Development Process 17

3.7 Design Plan 17

4 Testing **18**

4.1 Unit Testing 18

4.2 Interface Testing 19

4.3 Acceptance Testing 19

4.4 Results 19

5 Implementation 19

6 Closing Material **20**

6.1 Conclusion 20

6.2 References 20

6.3 Appendices 21

Figures

Figure 1: Project Timeline Gantt Chart Pg. 11

Figure 2: Front-end Component Diagram Pg. 17

Figure 3: Back-end Component Diagram Pg. 18

Figure 4: Sketch of potential website design Pg. 21

Tables

Table 1: Risks and Mitigation Pg. 9

Table 2: Milestones and Metric Pg. 10

Table 3: Tasks Efforts Requirements Pg. 12

Definitions

Term	Definition
Program of Study (POS)	The Program of Study, or POS, as referred to here in this document is the plan of courses for a student majoring in Computer, Software, Electrical, or Cyber Security Engineering. This list of courses includes personal choices to fulfill all major requirements and is the basis for the project.
Advisor	An advisor is a staff member of the Computer, Software, Electrical,

	or Cyber Security Engineering departments responsible for helping students choose courses and fulfill the requirements for graduation. Advisors also teach courses where students are required to create and submit an accurate POS for a grade.
Student	A student is any student majoring in Computer, Software, Electrical, or Cyber Security Engineering using the electronic POS tool to design their POS. A student also may be a participant in the 166 course being evaluated on their creation of a POS by an advisor.

1 Introduction

1.1 Acknowledgment

Special thanks to Patrick Determan, a colleague of Tina Prouty, who contributed valuable insight to the design and functionality of this project.

1.2 Problem and Project Statement

Every student who attends Iowa State has certain coursework requirements they need to fulfill before they graduate. Meeting all requirements can be confusing and overwhelming, especially to freshman students who have a lot to learn and little experience with the courses available. Because of this, every student in the Electrical and Computer Engineering Department (including Software and Cyber Security Engineering majors) needs to create a Program of Study (POS) in their Introduction to Engineering course. These programs are checked by hand, taking enormous amounts of time and leaving room for mistakes to be made.

The solution to this problem is an Electronic Program of Study. This program will be electronic and create a simple and effective tool for the creation of a student's POS. This tool will include features like prerequisite checking, graduation requirement assurance, different course requirements based on student catalog, and more. All of these features will aid advisors and students to maximize their time and provide powerful knowledge to ensure success in the selection of classes.

By the end of this project, there will be a web application with the capability to store, update and validate the Programs of Study for students in Electrical, Computer, Software, and Cyber Security Engineering. Students will be able to create this POS, and advisors will be able to view students' POS, update available courses, and approve exceptions to general prerequisites and requirements. This program will be accessible via an Iowa State account and will require minimal software updates going forward, making this a tool that can be used well into the future.

1.3 Operational Environment

The Electronic POS application will be a web application capable of being used in any popular and well-supported web browser. The use of a web browser can present some hazards during production of the product. One such hazard may be integrating existing user authentication with our application. One design decision was not to incorporate a new database of users, but to instead have the students use their existing login credentials provided by Iowa State. That means we have to incorporate existing google login verification into our application, which means the application will need to accommodate the settings that google requires to verify account information. This may or may not include cookies or other settings that can affect how google reads login settings or data.

Another possible hazard we are taking into account is the possibility of dealing with different browsers or versions of browsers that students could possibly use. Different browsers/versions can cause differences in page layouts and user interactions which we will need to take into account. This hazard can completely change how our web application functions for different users and we will need to be prepared to account for variables in student environments.

A future hazard that we will need to address is the longevity of our application through changing APIs. The project will require us to use existing APIs to build our application effectively, and as time passes, APIs will change/or become deprecated. The application must be able to evolve with the changing of APIs and still function, as one of the design requirements is that the application must be able to be modified and used over time.

1.4 Requirements

- Support the creation of programs of study for Electrical, Computer, Software, and Cyber Security Engineering majors
- Require students to accurately complete a program of study with the goal of graduation
- Provide warnings for missing classes or courses being added without prerequisites
- Enable students to edit their program of study
- Allows advisors to add new classes to the program
- Classes are able to be labeled as required, based on a given catalog year
- Individual courses are able to have prerequisites that must be met within the POS
- Advisors must be able to approve exceptions to general requirements
- Advisors are able to give feedback to a student's POS
- Application can be used by future students and is maintainable

There are some functionalities not included in these requirements because they are not necessary to a successful end product.

1.5 Intended Users and Uses

In this project, there are two primary users- students and advisors. These users have different needs and uses for this program.

First, students need a way to create a POS that can be a living document with immediate feedback about accuracy and potential errors. Most students make some changes when they select their first year at Iowa State, so they need to be able to access and edit their POS for at least four years after creation. Students also need to have access to requirements and a list of courses that satisfy the requirements of their catalog. They also need to receive notifications of advisor feedback and identify errors and have some idea how to fix them.

Second and finally, advisors need to be able to make updates to degree requirements, access student POS's, and give feedback to students. One critical feature is the ability for advisors to approve exceptions to the standard degree and course requirements. They also need to have access to all student programs in an organized way. This will allow for time to be saved when grading the POS assignment for the Intro to Engineering course.

1.6 Assumptions and Limitations

Assumptions-

- This Electronic Program of Study will only be compatible and maintained for the ECPE department and majors discussed in this document

- Minors and double majors, though nice to include, are not necessary to the success of the project. Courses associated with programs outside of the majors mentioned may be added to a POS but may not be checked for accuracy.
- The end product will be available only as a web app
- The end product of a POS will be available both in HTML on the student's profile (and advisor view) and as a downloadable pdf

Limitations-

- Canvas will not be used as the development environment due to extreme limitations and hassle. The web application can be linked in a Canvas module.
- As a web application, the application will have limited functionality while not connected to internet
- Due to the constantly changing University-wide course catalog, not every course will be included in the program. Students will be able to enter their own course numbers/credit amounts for gene eds and required courses and tech electives will be kept up to date
- The cost of server operation will be managed and funded by the ECPE department
- The end product needs to find a balance between usability and accessibility to ensure all students can utilize it
- Students will need to find another colleges course equivalents if they are a transfer student, as the application will not know any student information besides what they input
- Students will need to add in existing college credits from AP exams/high school, as per the explanation for the previously listed limitation

1.7 Expected End Product and Deliverables

The end product will be a web application that will meet all requirements laid out above. There will be user experience research and design based on that. There will be a few major deliverables to meet the goals outlined in this introduction. Expected date: November 1, 2021

The first deliverable will be the student side of the application. The student side will include text boxes for each of the semesters or some other way to clearly and intuitively input courses. It will also include a "check the POS" button that generates specific warnings of inaccuracies or green light that all coursework meets requirements. It will also include links to course lists, a spot for credits being brought in from other universities and colleges, and have login features enabled. This will be done at the latest halfway through the fall semester. Expected date: September 20, 2021

The second deliverable will be the advisor side of the web app. This will include the ability to view the POS, modify courses and requirements, and approve exceptions to the norms. Advisors will likely have the option to "bulk add" courses and requirements via a spreadsheet, or manually enter. At this point, the minors and double majors will not be supported but everything else should be functional. Most importantly, advisors will be able to have advisees assigned to them and see if their POS has a green light or not. Expected date: October 18, 2021

2 Project Plan

2.1 Task Decomposition

Phase 1. Planning

Task 1. Talk to client

Phase 2. Feasibility and Requirements Analysis

Task 1. Develop project plan

Phase 3. Design

Task 1. Design prototype

Phase 4. Software Development

Task 1. Setup database

Task 2. Create user interface

Task 3. Connect front and back end

Phase 5. Testing

Task 1. Test back end

Task 2. Test front end

Phase 6. Implementation and Integration

Task 1. Let some students try the software

2.2 Risks And Risk Management/Mitigation

Risks	Probability	Impact	Mitigation
Late delivery of software	30%	1	Set an earlier deadline than the final deadline to have some flexible time for accidents.
Technology will not meet expectations	20%	1	Decide 1 or 2 more alternative technology at the beginning to use if the original decision does not meet expectations.

Changes in requirements	10%	1	Implement a modular design so it is easier to add or remove functions.
Less reuse than planned	40%	2	When finished planning, ask the advisor for advice.
Deviation from software engineering standards	10%	3	Check if our code is still valid to the standards every week.
Poor commenting of source code	20%	3	Make sure to comment the code before pushing the code to Git.

Table 1:. Risks and Mitigation

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone	Metric	Finish Date
Get all requirements from client	List of the requirements requested by the client	April 7th, last client meeting of first semester before starting production of product
Develop a project plan	No comments on what to fix in the feedback	April 21st
Setup the database for one major	All the course for one major stored in a database	August 29th
Create a simple user interface	An interface that can search, drag, and drop course	September 5th
Setup the database for two more major	Add all the courses for two more major into database	September 5th

Create a more user friendly interface	Design an interface that has instructions for all functions	September 19th
Have a prototype of the project	A sample app that a student can use to perform all functions	October 18th, the day 2nd half semester courses begin and the pilot group of students can test
Complete project	Satisfied all the requirements the client requested	December 5th

Table 2: Milestones and Metric

2.4 Project Timeline/Schedule

The following Gantt chart is our project's estimated timeline. Our team has completed talking to the clients and finished the designing the project plan. We are currently designing the prototype of our project. Everything is on schedule by the submission of the project plan.

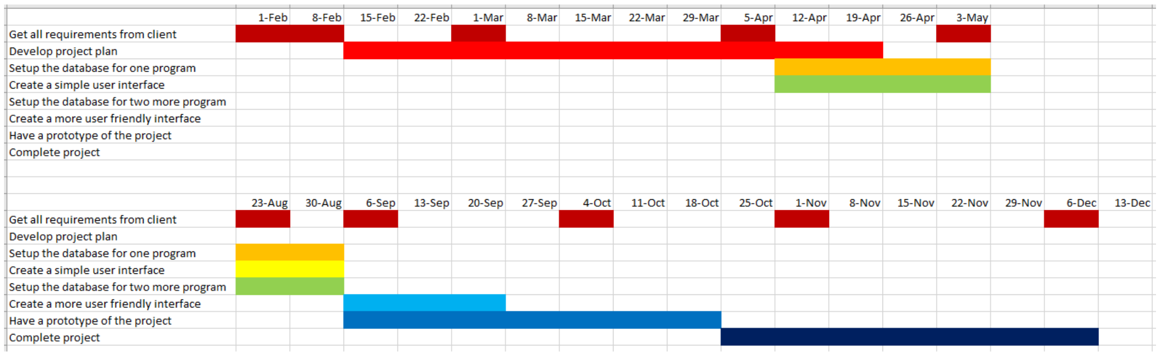

Figure 1: Project Timeline Gantt Chart

2.5 Project Tracking Procedures

Our group will be using our provided GitLab repository to manage our code and track our progress. Each group member will have their own goals to work towards and soft deadlines by which to upload their code to Git. Certain aspects of the project may require git branching to implement, which will be another way in which our group can track progress. Our use of the agile development process will allow us to set our goals for ourselves individually and collectively before getting to work and pushing our work to Git.

2.6 Personnel Effort Requirements

Requirement	Explanation	Estimate hours to complete
Get all requirements from client	Holding multiple conversations with clients to determine the needs and possible solutions. Includes presenting ideas to receive feedback and improve our understanding of what we want the project to become.	6 hours
Develop project plan	Taking information from our meetings with clients to develop the best product for the client. Figure out the tools and techniques we will use to create the product.	15 hours
Design prototype	Plan the structure of our final design. Decide how we want each component to interact with each other component and the information we will be sending to between them.	20 hours
Set Up Database	Create the database schemas and how the tables will interact. Enter in some basic information to test the functionality and the usability of the database design.	15 hours
Create User Interface	Create our basic user interface to test the entering of information and the usability of our interface design.	20 hours
Connect front and back end	Connect our basic front end and back end designs to send information to each other and interact fully. Discover and fix any problems we encounter with the information being sent between our front and back end designs.	10 hours
Improve User Interface	Create a more user friendly interface which utilizes the same techniques of sending and receiving information, but presents the information with usability in mind.	12 hours
Create Prototype of Product	Finalize a prototype to move into our testing phase and improve upon for future user testing	8 hours
Testing	Test front end and back end with as many use cases as possible to find and fix code failures and errors. Fix said failures and errors to move the prototype into the user testing phase.	15 hours
User testing	Let current students try the software and give feedback on the shortcomings or problems they encounter. Team will then use feedback to improve the product and finalize our end	10 hours

	product.	
Finalize Product	Finish polishing the product until it meets our criteria and reflects the feedback we received from students. Present final product as finished.	10 hours

Table 3: Tasks Effort Requirements

The entire project will take roughly 136 hours to complete. The number of hours it will take are also split among multiple people working towards common goals in the project. For instance, connecting the front end and back end will require 10 hours of combined work among all 6 team members.

2.7 Other Resource Requirements

The only requirements our group will need are access to our git repository, working code editors, database software (such as mySQL), and access to any subscriptions we may need to edit or test code. The school will need to provide a server to support the application in the long term, though not for most of the development process.

2.8 Financial Requirements

The only financial requirements required to conduct the product would be server costs to support the project for the classes in the long term. We do not believe at this time that we will need any additional financial resources to fund the project, besides maybe some kind of subscriptions we may discover we need to develop the final product.

3 Design

3.1 Previous Work And Literature

There are no similar products that exist, though the advisors of the EE/SE/CprE/CyE departments have implemented certain procedures for the creation of a program of study for their students. Some of the previous methods that advisors have had their students use to create their program of study(POS) include filling out an Excel spreadsheet and filling out a pdf form. Both of these previous methods come with serious drawbacks however, being clunky and tedious to fill in as no convenient method to check for prerequisites or corequisites. They do have the benefit of being commonly used formats and the software to edit them are readily available.

As with most software projects, this one will be divided into the front end and back end. Each portion will be worked on in different languages, and this project will utilize at least two primary languages. For the front end, the logic will be developed in JavaScript, and we are choosing this language for a variety of reasons. Firstly, this is something the team has a good understanding of as well as prior project experience. The second reason is the extensive amount of existing packages designed specifically for web application design. We hope to utilize a wide range of these libraries to deliver a solid, secure, and well-developed professional application. We have narrowed it down to two initial frameworks. First, we will utilize jQuery for our event handling logic. The second framework is Ajax. This framework allows us to have a highly dynamic and interactable application. The primary use for this will be to develop an interactable flowchart based on the user's schedule. As our group runs into obstacles, we will continue to look for additional frameworks to ensure the application meets or exceeds all requirements.

The web app needs to be intuitive, as this is something students will have to use, so it must be accessible to a wide range of persons while acting comfortably. For the appearance of the web application, the backbone needs to be Hyper-Text Markup Language or HTML. This is a requirement by the internet for an interactive web application to existing. To add style to the backbone, we will implement Cascading Style Sheets or CSS. This language gives us the ability to add additional design options for the website. CSS will provide the traditionally bland HTML a professional look to create an application that appears identical to any other Iowa State University website.

The backend will be developed using python as we do not expect heavy load most of the time. Therefore, we decided to use a language that is very common and easy to understand while also being readily available to implement new functionalities without requiring much effort. While having an interpreted language has some expansive drawbacks, we found that for our particular use case, almost no heavy load, relatively low peak load, and lightweight architectural design, an interpreted language such as python would be a perfect fit. While we could have chosen to code the backend in javascript using NodeJS, more of our members had experience with python and are more familiar with certain python libraries. We will be using two main libraries for our backend implementation, those being Flask and MySQL. Flask will be used to create REST endpoints to serve our frontend, helping to ensure safety and security through Flask extensions such as Flask-Authorize and Flask-Authenication which have ways of working with Google's login APIs.

3.2 Design Thinking

The first core concept that this project was designed with was that it needs to be modifiable. New classes should be able to be created by advisors or by automation.

- Application that utilizes Google sign in.
- Internal Database that holds the entire list of classes for major.
- Classes can be classified into different categories.
- Select classes from a drop down list.
- Automatically checks for prerequisites and corequisites.
- Academic advisors can view students' POS.
- Academic advisors can comment on students' POS.

There were other potential design choices that would not be necessary to the success of the project, but that would potentially be nice in the final product. These design choices are ideas that we came up with in the ideation phase of designing. Some of these ideas were the ability for advisors to view the student live editing the POS much like a Google Document. The method of linking with Iowa State's Okta authentication system in the place of a stand-alone Google sign-in. Also the thought of arranging courses in alphabetical order vs courses most used in the program was brought up. For design choice of course arrangement, we decided to go with the alphabetical order due to ease of coding and user friendliness. This will be a point to incorporate in the testing phase starting November 1st.

3.3 Proposed Design

There are two main methods to approach solving this problem, either having a web application or a standalone application that students download whether on PC or mobile.

Web application

This design allows for the advisors to easily edit the catalog of classes and lets advisors view student's POS without any additional work for students to supply to their advisors. Team members have plenty of experience with web applications. As a web application, a database is easy to implement and can integrate with multiple sign-in apis. The database in question will be MySQL and will sit on the same server that backend business logic will run on. Since web applications are highly modifiable, almost all requirements can be implemented, whether new or original. There are very few standards that apply to this project, most of which are simply IEEE cyber security standards. As web applications already exist in a browser, it becomes simple to have multiple users that can view separate POS, i.e. having advisors that can see multiple students but students can only view their own POS.

This application would meet all of our requirements which we have worked with the client to set. With the course catalog system database we will categorize the courses by college and program similar to how Iowa State's website works. Along with this courses will have parameters to be marked as essential to the selected students program and year. Another parameter for the courses will be their prerequisites courses.

There will be two types of accounts one for Students and one for Advisors. The logins for both will be through the Google Sign-In API since the university provides students and faculty with a google account already. The Student account is designed to use rules set for a four year plan set by the advisors. They will be able to make a schedule tailored to the Electrical, Computer, Software, or Cyber Security Engineering major. They will also be able to add in minors, double majors, graduate programs (concurrent), and a MBA if they desire. If areas of a program of study have multiple options there will be a drop down menu with available courses to pick. There will also be a general search by program similar to the Access Plus course registration.

The Application will review the students POS decisions and provide warnings if certain courses don't have the necessary pre-reqs met, or if they are missing general requirements for graduation. They will still be able to complete a program of study with warnings, but their advisor would get to see straight away there is an issue.

The Advisor accounts would be the maintainers for the platform and have the unique ability to add and remove courses to the database as well as new catalog rules, with the necessary rule changes added. This will be made easy with the ability to duplicate a previous year's catalog rules and make small changes. Advisors will also have access to their students' POS in order to provide feedback within the application and approve any exceptions to a warning an individual student may have. The application will also need to be easily maintained so that it can be used by multiple future years of students.

Alternative approaches to implement this project would be to only have a database on a remote server and have all business logic located in the client. This accomplishes the exact same thing that having a server contain the business logic with the database. Another alternative would be to handle signing into accounts with the application; this however, is a less viable alternative as

ISU already uses Okta and Google sign in for their accounts. Integrating into this system is simply the only choice as creating a new system to integrate with the already existing system would be more difficult than it is worth.

3.4 Technology Considerations

Having our application be a web based application, imposes many restrictions for the application as a whole. First of all, without the internet, the application becomes unusable and the entire functionality of the application becomes mute. Even with this drawback, a web application allows advisors to be able to easily view the POS of their assigned students. Security however, is a major concern with a web application that can be slightly mitigated with a downloadable application. There are solutions to these problems, the first one being that the application will be able to download the POS if the internet connection fails. On the concern of security, there are plenty of tools available to find security vulnerabilities and there are plenty of tools available to fix security vulnerabilities.

We considered several alternatives to a web application, and these included: mobile Applications, Desktop Applications, and a Canvas Plugin. We found drawbacks in all of these technologies that pushed web applications high above the rest for usability, convenience, and redundancies reduction. We considered a mobile application since almost every student has some mobile device. This would be convenient for their use, but since we cannot require a student to have a mobile device, this technology does not meet our requirements. Another drawback would be that we would need to develop at least two versions of the application for Apple and Android. While we have a larger group, it would make more logistical sense to develop twice on one application than half as much on two almost identical applications. The second technology consideration was to create a desktop application. This technology is better suited for the project because all Iowa State Students have access to computers through labs and laptop loaner programs. However, it still faces the issue of needing to develop multiple systems. In this case, we would need an application for at least a modern version of Windows, macOS, and Linux. Because of these duplicate applications, we chose desktop applications. The last consideration was a canvas plugin. This was almost an ideal solution because of the user's access to and familiarity with the Canvas program. The drawback we faced was the unfamiliarity we had with developing tools for canvas. The team also predicted that university and canvas would have issues with students messing with essential parts of the application. After all these considerations, our group decided to move forward with a web application.

3.5 Design Analysis

There have yet to be any major design modifications for our project. This is mainly due to having well defined requirements and discussing a multitude of scenarios with our clients. However, in the future, our project is designed to be modular in both its functionality and its expansion. One major change that has happened in the creation of our application design was whether to handle logins and accounts in the application or utilize already existing API's for ISU accounts and logins. Another change that we made early on was switching our end goal of a Canvas integration to a general web application.

3.6 Development Process

For this project, the team is following a hybrid approach between Agile and Waterfall. The reason for this approach comes down to the fact that communication with our client is vital to the success of the project, but face-to-face communication is limited in scope and availability. It would be fair to say that we are using Waterfall methodology while implementing aspects of the Agile methodology. Along with the limitations in communication between the client and the team, the scope of the project lends itself to the streamlined nature of the Waterfall approach.

3.7 Design Plan

- Application that utilizes Google sign in. (Front end Account)
- Internal Database that holds the entire list of classes for major. (Back end Class Database)
- Classes can be classified into different categories. (Back end Class Database)
- Select classes from a drop down list. (Front end Courses)
- Automatically checks for prerequisites and corequisites. (Front end Courses and Back end Class Database)
- Academic advisors can view students' POS. (Front end Advisors)
- Academic advisors can comment on students' POS. (Front end Advisors)
- Application can be used by future students and is maintainable. (Modular Design)

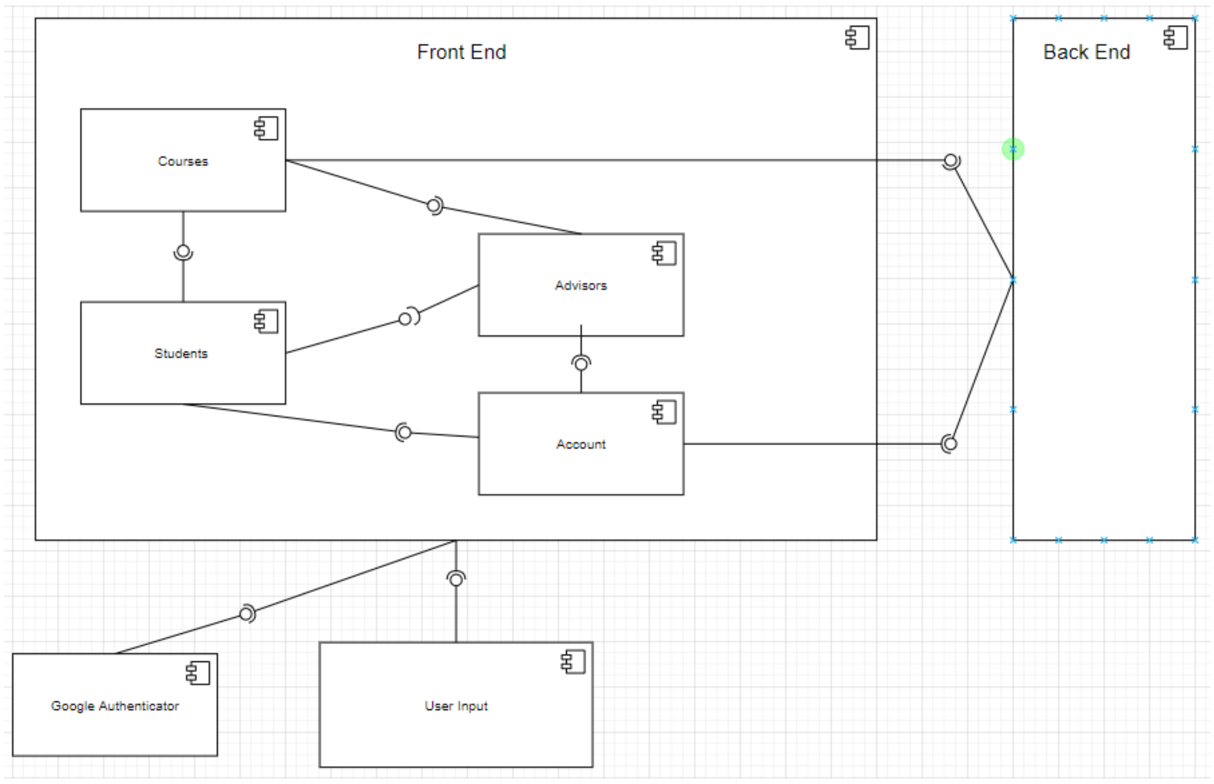

Figure 2: Front-end Component Diagram

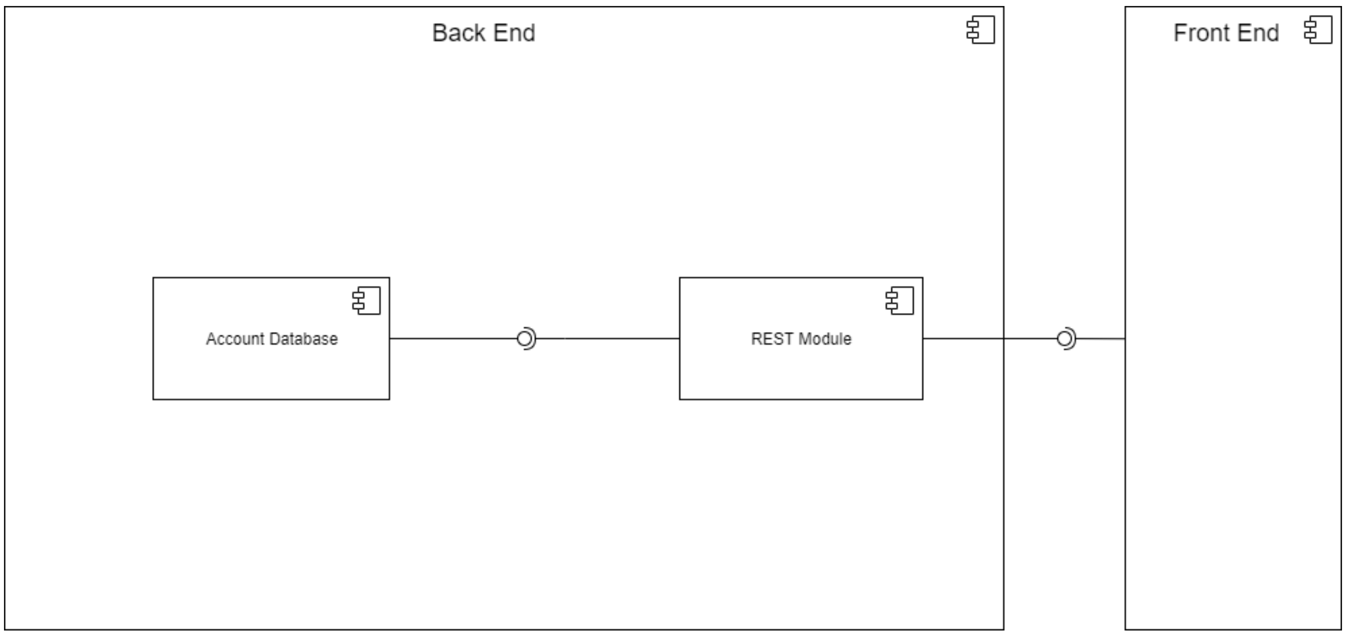

Figure 3: Back-end Component Diagram

4 Testing

Selenium will be the main tool for testing this project as it allows for unit and functional testing. Selenium is an industry standard for testing. Its cross platform and browser capabilities will make it a useful tool for testing our code. We will be testing the followings:

- If a student can search the courses they need.
- If the frontend can pass the user's schedule to the backend after saving.
- If a prerequisite does not meet, an alert will show which course is it.
- If the comments from the advisors can be shown correctly on the students side.
- If the advisor can approve exceptions to the rules.
- If the schedule created is missing courses to complete the degree, an alert will show which courses are missing.
- If advisors can edit, create, and delete courses, catalog rules and degree programs

4.1 Unit Testing

Tests will be written for each component of the software to ensure all components are functioning as intended. Examples of possible units to be tested include: inputting valid and invalid combinations of classes to see if the software correctly distinguishes between the two.

4.2 Interface Testing

Along with testing individual components, combinations of components will be tested as errors can appear anywhere in the code. It is important to test combinations of components to ensure overall functionality is maintained. Inputting full valid and invalid inputs to verify that our program is working as intended. Another important component of interface testing is to ensure our program interacts with the different API's in the code properly. Lastly, the GUI will be tested to make sure the user's experience is satisfactory. This will be tested by a group of students taking the course in the second half of the Fall semester.

4.3 Acceptance Testing

We will write robust tests to show each requirement is met along with having some students from S E or CPR E 166 use our product. Both of these classes will have the assignment to develop their four year plan. We will have one class utilize our working application in order to complete this assignment. From their experiences utilizing our application we will be able to test our application in its real-world environment. Edge cases from our client to produce better testing to ensure the best experience for the students and advisors using our software. These edge cases include the logic behind prerequisites and corequisites being met, advisors being able to approve exceptions to course and program requirements, and student saving degree programs while having another browser session open. Another piece of acceptance testing will be the simultaneous use of the program by the student and advisor editing and commenting.

4.4 Results

Unit and Interface testing will be developed alongside the working project development in Fall 2021. We will conduct the Acceptance testing with one section of either SE 166 or CPR E 166 around midterms of the Fall 2021 Semester.

5 Implementation

We would be designing a web application to house the Program of Study. This application will be supported by a database developed by the backend team. This database will store all the courses, rules, and account privileges. The website will be developed by the frontend team, who will design and create an easily interactable PoS which can be applied to the students 4 year plan. Our goal is to have a working and tested product by shortly after the semester midpoint.

The product itself would be a web application written in JavaScript. This application will have two account types one for students and one for advisors/faculty. The web app will store some information from users, like their current 4 year plan as well as additional plans they chose to save.

6 Closing Material

6.1 Conclusion

In developing this plan we have laid out initial work outside of development, we will spend the next period of time developing web scraping tools to create our course database. Our goal is to have a working version of this application near midterms of the Fall 2021 semester. The web app will allow for an easily editable, verifiable, viable, and accessible Program of Study for ECpE students. The software will allow for the addition and modification of majors,minors, degrees, courses, and courses. We believe a well developed web application will satisfy these needs and deliverables.

6.2 References

N/A

6.3 Appendices

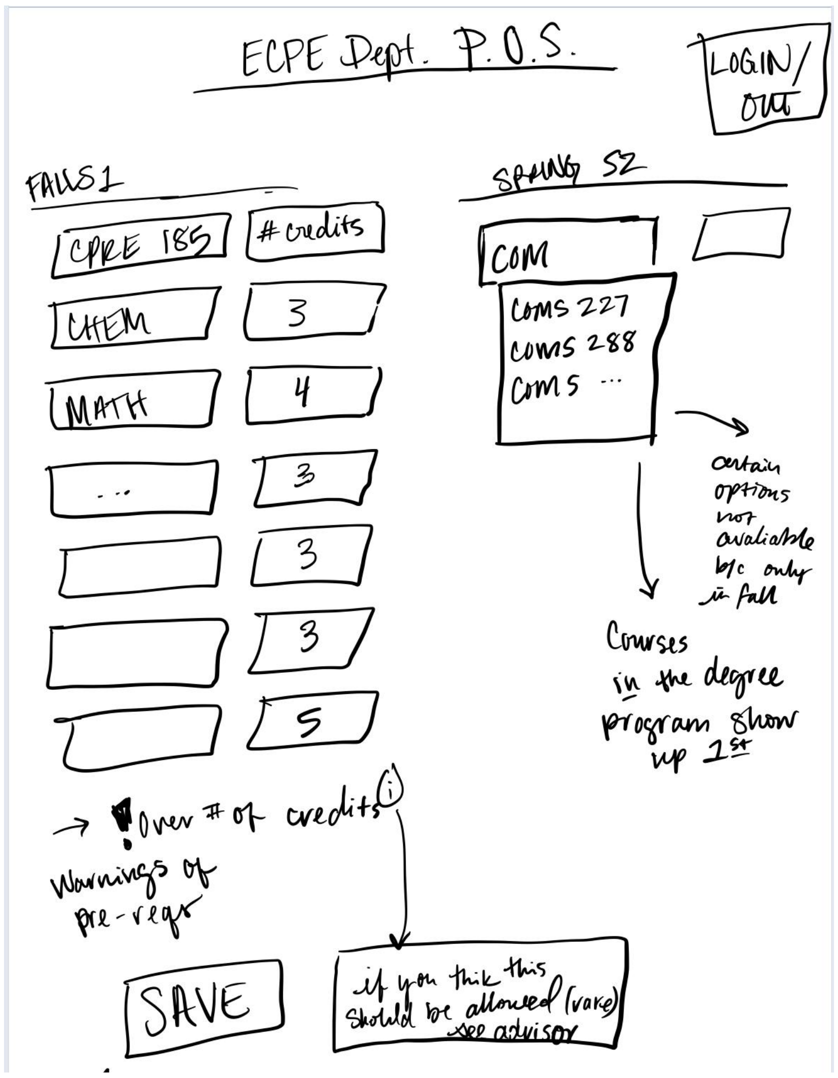

Figure 4: Sketch of potential website design