Critical Design Review Document

Boardman Computer Science Lab Web Portal

Client: Mr. Christopher Dufour

Development Company: In-House Operations

Development Team:
Klei Bendo
Jack Brisson
Alex Landry
Samuel Morse
Aaron Schanck
Forrest Swift

Date: 11/30/2021
Version 1.0

Executive Summary

In 2017, 50% of incoming computer science freshmen failed COS125, one of the introductory classes to the major. Just three years later, in 2020, that same class' fail rate dropped to just 27%. One of the largest contributors to this phenomenon has been the growing prevalence of the Boardman computer science lab, a help resource for UMaine computer science students. As successful as the lab has been, there is still a need to more effectively inform students of the current schedule of lab helpers, set up meeting times in the lab and remote during the pandemic, as well as record student's comments on their help sessions and involvement at the Boardman lab to improve it further. This first semester, the In-House Operations team completed much of the preliminary work in setting up the project, including identifying requirements and use cases, designing the user interface, and creating a database to store student comments and participation using PostgreSQL. In the coming semester we will use a Django framework to build a web application that has scheduler functionalities, feedback for help sessions and data analytics for instructors. We will use the Google OAuth API and access token so that Umaine students can log in to the application using their student emails. With this application, we hope to drop the fail rate even lower.

Foreword

The Boardman Computing Lab in BD 138 provides support for students taking introductory and intermediate computing courses, as well as a supportive community for students in computing majors. This lab is staffed by TAs and MLAs for COS courses, as well as a supplementary group of lab monitors. While some staff may be primarily associated with a specific course, all lab staff are available to help students in any computing course. Traditionally, some students would come to the lab specifically to get in-person help with programming assignments. Other students would work in the lab or associated lounge in order to be part of a community and to be close to help resources that might be needed. In response to the pandemic, lab operations have expanded to include remote support using Discord and/or zoom, as well as email to specific course staff. This more complex support ecosystem would benefit from a better understanding of what support students need and how they can best access it.

We wish to have a software system that would help coordinate and deploy help resources in the Boardman Computing Lab and its online extensions. This system would log help requests, track responses to requests, and support analysis of resource needs and effectiveness. The system should be web-based, easy to use, easy to maintain and adaptable to integrate data from multiple sources, able to support different levels of access, protective of student information, and capable of flexible analysis of data.

-Mr. Christopher Dufour

Preface

The Boardman Computer Science Lab Web Portal was designed and curated as a computer science senior capstone project at the University of Maine. This document encompasses the work and research done by our student team during the first of two capstone semesters. The In-House Operations team hopes that our application will be a valuable resource to undergraduate computer science students for years to come, and that this documentation will be an all-encompassing companion to the application itself to better understand our purposes, processes, and plans for the next semester's development.

Boardman Computer Science Lab Web Portal

Critical Design Review Document

Table of Contents

1. Introduction

This section directs reader traffic to specific locations for figures and tables and summarizes the overall content of the document. Note: sections 1.1 and 1.2 only cover the main segment of the documentation excluding Appendix B.

1.1 Figures

1.2 Tables

1.3 Summary

In 2017, 50% of incoming computer science freshmen failed COS125, one of the introductory classes to the major. Just three years later, in 2020, that same class' fail rate dropped to just 27%. One of the largest contributors to this phenomenon has been the growing prevalence of the Boardman computer science lab, a help resource for UMaine computer science students. As successful as the lab has been, there is still a need to more effectively inform students of the current schedule of lab helpers, set up meeting times in the lab and remote during the pandemic, as well as record student's comments on their help sessions to improve the lab further.

In-House Operations in conjunction with Mr. Christopher Dufour, the director of the Boardman lab, is producing the Boardman Computer Science Lab Web Portal to mitigate these shortcomings and continue to drop the failure rate for computer science underclassmen. This application will include features for students to view available helpers availability and subjects helpers can help with, in day, week, and month views, allow them to schedule one-time and recurring help sessions with specific helpers and leave feedback on help sessions. Data will be collected on help requests to better understand where students are struggling with certain classes, subjects, or at certain times, as well as saving user feedback to gauge the effectiveness of specific helpers.

During the first of two semesters, In-House Operations completed much of the preliminary groundwork for the application. This included gathering user opinions and data through a survey, we generated requirements for the

application from both the client and user base, designed the architecture and user interface, and created the database in which data will be stored and used to generate reports.

In the coming second semester, In-House Operations will more extensively integrate the Django framework to build the front and backend integration and functionalities of the application including a scheduler library for the calendar requirements. Additionally, we will connect to OAuth services in order to allow users to use their UMaine login information when signing in. Since the database is mostly completed, it will only be a matter of connecting it to the application on that front.

Every member of In-House Operations is an undergraduate computer science student who has used the Boardman lab extensively during our time at the University of Maine. This project is very important to each of us, and we hope to improve the functionality of the lab so that students for years to come will be able to have access to every tool needed to help them succeed.

2. Project Background & Methodology

This section covers the background information needed to understand the circumstances and motives behind the creation of this application. Provided are the purposes and situation this application was designed for, as well as an overview of the methods used to address the application's purposes.

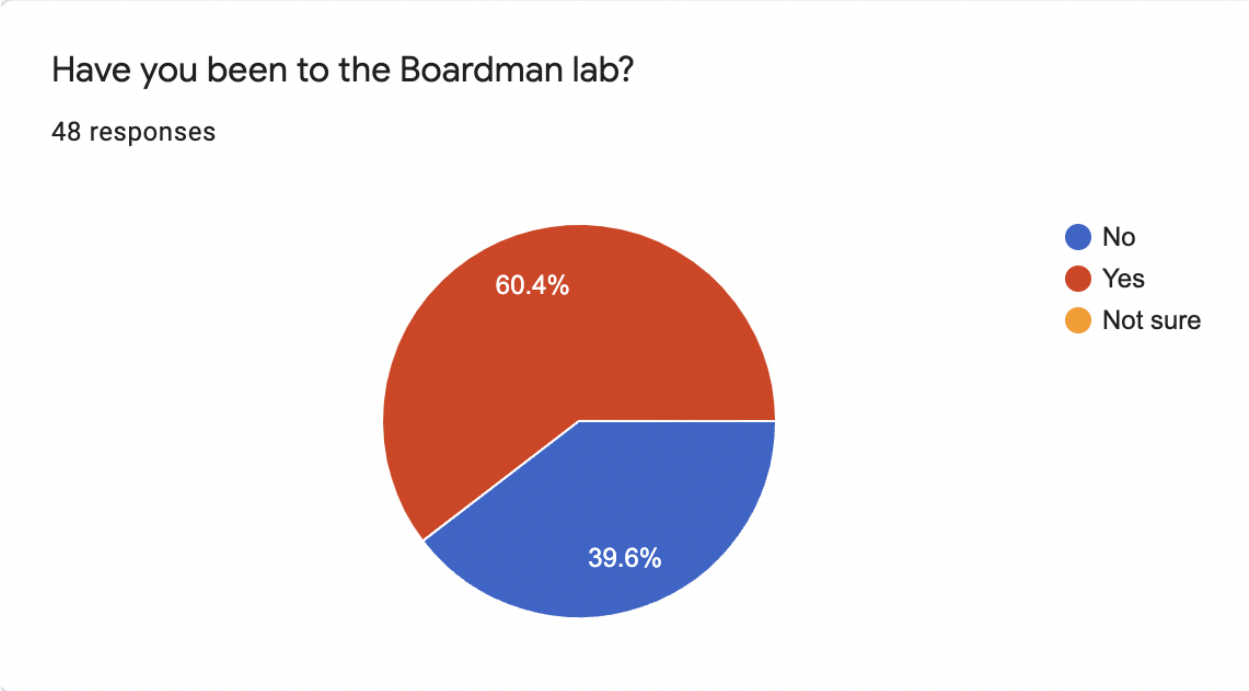
2.1 Introduction

The Boardman Computer Science Lab is a long-standing resource for UMaine Computer Science students to receive help with computer science and other related issues and questions and as a meeting place for students working in groups. Since 2017, the failure rate of COS125, one of the staple freshman courses, has dropped every year from 50% in 2017, 43% in 2018, 33% in 2019, to 27% in 2020. This is largely due to the emergence of this lab. However, some common complaints and issues are facing the lab that currently prevents it from attracting many undergraduate students to help them reach their full potential.

Firstly, there is a lack of clarity in regards to when lab helpers are available and the areas in which these helpers are able to lend assistance. There is, at present, only a schedule printed on the lab's door to inform students of the helper's availability. This can be subject to change and confusing to students who are not already familiar with the helpers themselves or how the lab works.

Additionally, there is an issue with reaching undergraduate students who are not already familiar with the lab and its location. In-House Operations conducted a survey of freshman students on various topics to gather expectations and suggestions from the primary users of our application. From this, it was discovered that almost 40% of the surveyed students had never once been to the lab, seen below in ***figure 1.1***.

Figure 1.1: Current Student Lab Usage



Description: In a freshman student survey of 48 participants, 60.4% indicated that they had been to the Boardman lab at least once while 39.6% indicated that they had never been to the lab.

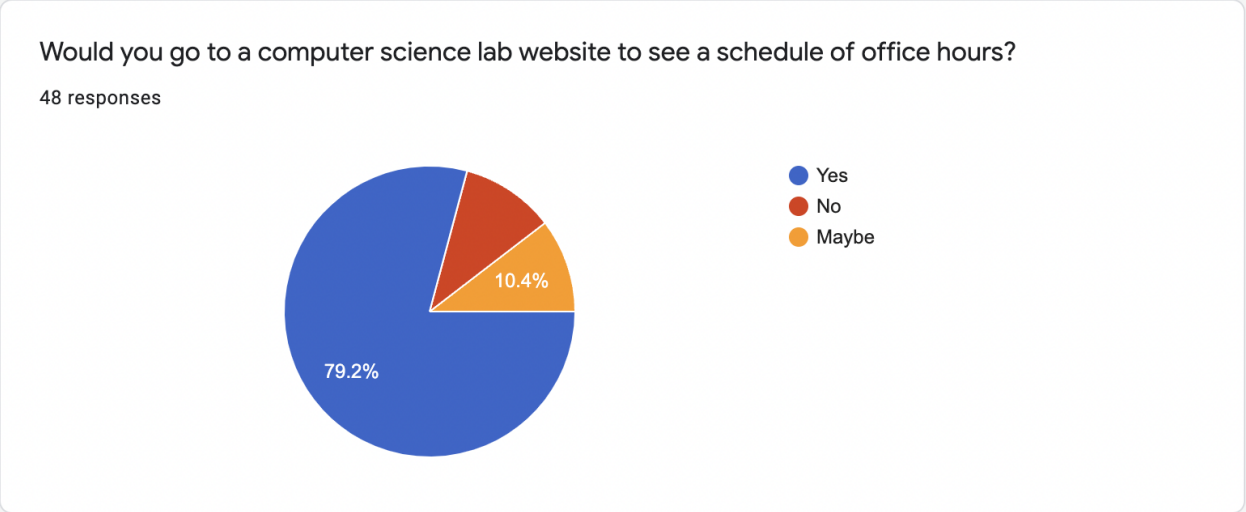
There could be a variety of reasons for students not having visited the lab, but it became abundantly clear that this invaluable resource needed to be more accessible to some students, especially given the current state of the COVID-19 pandemic making being physically in the lab difficult for many.

A need has also arisen for the lab to track certain data on the help sessions that occur both in-person and remote. In order to provide the best experience possible, student-provided feedback on help sessions and the lab itself needed to become more commonplace.

2.2 Purpose

The Boardman Computer Science Lab Web Portal is a program aimed to both alleviate the aforementioned issues (see section 2.1) attracting the primary users and collecting data to help the lab ever improve the student experience. In our survey, there was overwhelming support for an online companion to the Boardman lab. Almost 80% of students stated that they would use a website dedicated to updating them on when helpers are available in the lab, with another 10% indicating that they might.

Figure 1.2: Indicated Student Usage of the Proposed Application



Description: In a freshman student survey of 48 participants, 79.2% indicated that they would use a Boardman lab web companion to see a helper schedule, 10.4% indicated that they would possibly use the application, and only 10.4% indicated that they would not.

With our application, we aim to make getting help with computer science work even more accessible to undergraduate students. The Boardman lab has already improved the undergraduate experience, as seen in the shrinking failure rate of introductory classes. The Boardman Computer Science Lab Web Portal will aid in furthering this pursuit by making help resources more accessible and intuitive for the student body.

2.3 Method

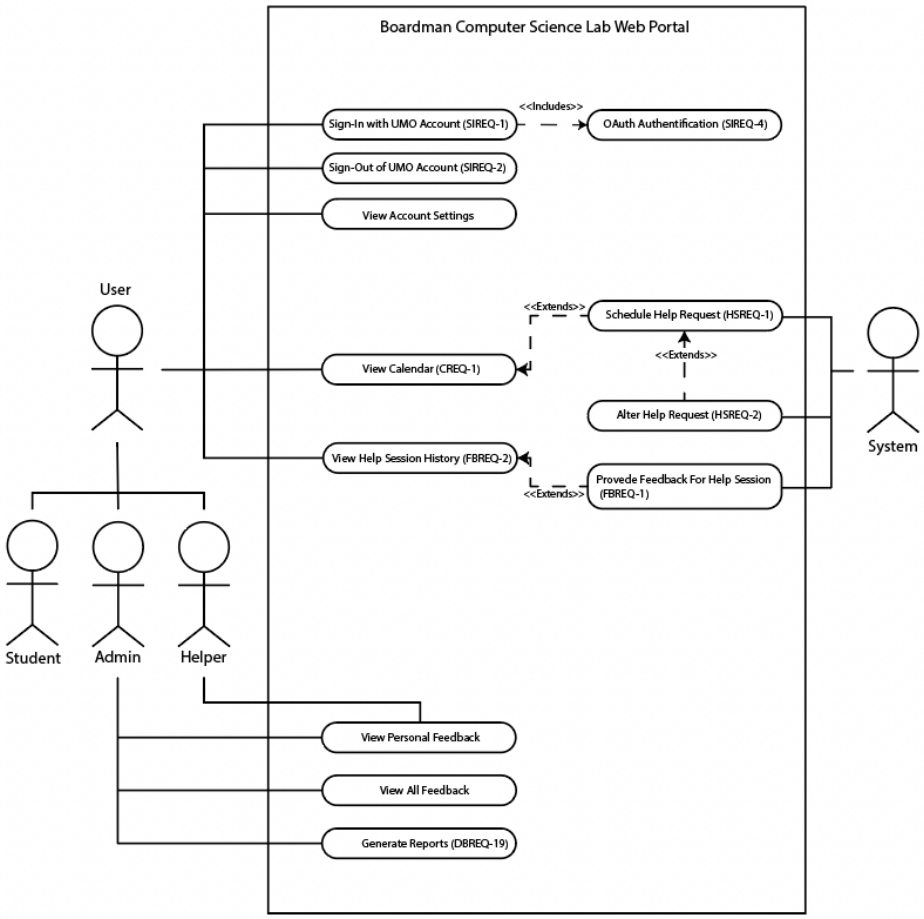
2.3.1 Application Goals

It was immediately clear that there needed to be a distinction between types of users to allocate permissions for selective functionalities such as viewing user feedback on helpers. The main focus, however, is the student users of the application. In-House Operation's main goals in the production of this application are simplicity of use and attractive design in order to appeal to student users as much as possible, as well as to provide a simplistic, yet effective way to record user feedback. To this purpose, these goals have been at the forefront of our development process.

2.3.2 Approach to Application Goals

To aid in simplicity, our planned student UI only has three major functionalities that students have access to. Students will only have access to an interactive calendar, their help sessions, and their account information (for a more in-depth explanation and UI mockup examples, see *section 3*). Further aiding in the simplicity and ease of use of our application will be the use of UMaine issued email addresses for login, to save in the creation of another account. Helpers and administrators will have access to a slightly more extensive set of functions. Helpers will be able to view feedback they have received from students in help sessions, and admins will be able to view feedback on all helpers and generate feedback reports, and number on help requests from specific users, courses, and timeframes to help record where improvements that need to made to lab functions. In *figure 2.1* below, a full scope of major user functionalities is shown as an application scope.

Figure 2.1: Application Scope



Description: The application scope shows the major functionalities available to all users and also functionalities available to select types of users only. Note: this is an updated application scope to that found in Appendix B, there were some stylistic and functional changes needed since the SRS was generated.

Having an attractive appearance is such a high priority to ensure the continued use of students. One response to our freshmen survey was: “don't make it look like it was made in 2001.” To accomplish this goal, the development team decided less was more and created our design as sleek and uncluttered as possible. We stuck to a small color palette and few graphics. For more information on the design of the application, see *section 3*.1.

In an effort to ever improve the Boardman lab’s functions, generating reports is also crucial. Whenever a help session is requested, the specific data from the request will be recorded for report generation including the student, course, subject matter, helper requested, time requested, type of meeting requested, and whether or not it is a recurring meeting time. Additionally, helper feedback will be recorded for posterity and improvement.

2.3.3 Frameworks, Libraries, and Outside Resources

Inhouse Operations will be using some existing frameworks and libraries in the creation of the application. For our front and back end integration we will be using the Django framework, a free and open-source “high-level Python web framework that encourages rapid development and clean, pragmatic design (Django).” A library for Django

that we will be using extensively is the Django Scheduler, a calendar library for Django that will cut down on development time significantly.
We are using ElephantSQL for our database integration. This framework installs and integrates PostgreSQL databases which we are using for data recording and processing.

Lastly, OAuth 2.0 will be used to generate user tokens for login access using UMaine login credentials. Being that the University of Maine already uses this framework for other university services, it will conserve work in creating a new user database for sign-in access.

2.3.4 Development Methodologies

Our team used a hybrid system of both agile and waterfall development during the course of this semester. In terms of agile development, we implemented a modified scrum approach for physical application development in which we met at least once a week, oftentimes more, to discuss progress on goals set in place during the previous session, and to also set goals for the coming development period. We used waterfall methodology for project artifacts throughout the semester which include our SRS, SDD, and UIDD (see *Appendix B*).

This served dual purposes for both structured goal setting to moderate expectations, and also to give the course's professor material to grade. This first semester was more waterfall-driven, as much of it was spent designing the project itself and doing much of the foundation for later development. The second semester will be more development-focused, so agile strategies will be more integral to our process. Put together, the workflow of the course as a whole begins to resemble a v-structure where the left side is the progress made during the first semester and the right being the second. It begins to look quite jumbled when the agile methodologies are also present, so it is separated into two graphics.

3. Project Design

This section details the design of the application in its entirety at the current time. It explains the guidelines set upon us at the beginning of the project in terms of its design, the team's approach to these parameters, any problems we encountered in our approach, and specific details about our design.

3.1 Design Parameters

As a team, we were given a large amount of freedom from our client for the design of the application. Our only explicit direction was to design the application to appeal to underclassman students and to retain their usage. This was a broad expectation, however, every member of the team are both undergraduate students and avid members of the Boardman lab, allowing us to have insight into this request. What we did have, however, was a list of functional and non-functional requirements to base our design around. These requirements can be found in *Appendix B.* They told us what needed to be included in the design functionally including a way to track user help sessions, see helper availability times, and create lab reports. The design came from how best to integrate these functions.

3.2 Design Approach

We decided to create a general layout for the application that is as simple and easy to navigate as possible. There is a navigation bar on the left-hand side of the display and content in the center (justified right). We worked on creating a layout that would be usable for mobile devices, with the option to hide the navigation bar for optimal viewing. Each page has tabs that are specific to user privilege level, and maintain a simple design standard.

A dark color palette was used to both reduce eye strain and save energy. Colors included are brought directly from the University of Maine 2018 Branding Standards to ensure continuity with the user's workflow.

There are no soft edges in the design (except for special buttons) to portray a utilitarian application - the design focuses on use while maintaining a clean organized appearance. We were influenced by the design of other applications that students commonly engaged with such as Slack and Discord.
University of Maine Design Standards:
https://umaine.edu/marketingandcommunications/wp-content/uploads/sites/209/2020/06/Brand-Standards-FINAL-web.pdf

3.3 Design Problems

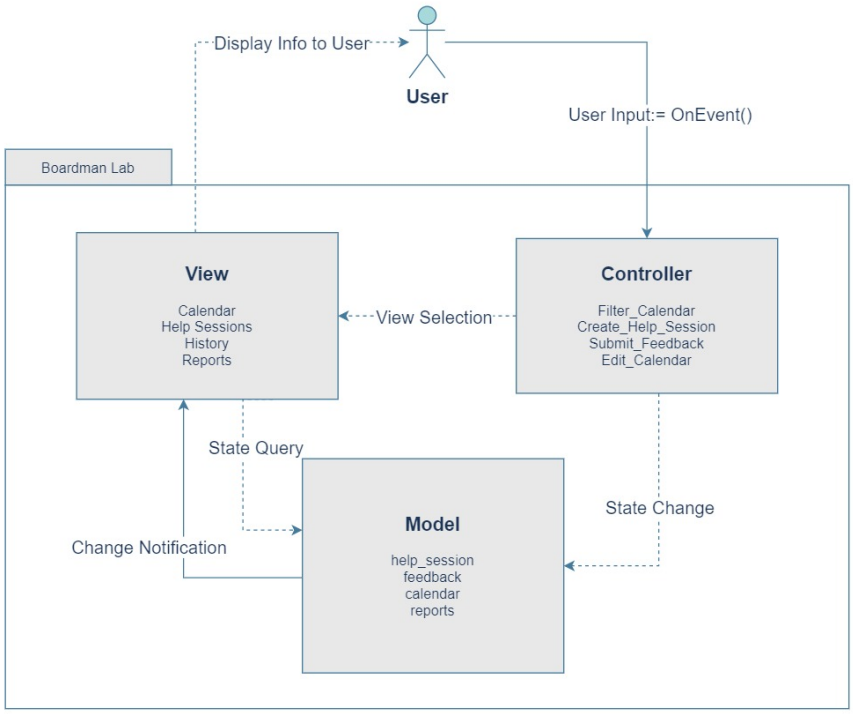
During the design process of our application, the only area we had extensive trouble with was finding a method to include a calendar feature in the application that suited all the requirements we were trying to accomplish. After extensive research, we settled on a Django library, the Django-scheduler which allows for scheduling of events, changing events, and day, week, and month views.
https://github.com/llazzaro/django-scheduler/blob/develop/README.md

3.4 Design Details

3.4.1 System Architecture Design

For the system architecture, we are using the model view controller design pattern. The frontend and backend will be using the Django framework with PostgreSQL. Django will handle the queries to the database and will handle display and user interaction (view, controller, and models). We are going to be implementing a single sign-in using Google Oauth and Umaine login information to streamline user interactions and allow us to not store sensitive user information on our database. This means that we will not be storing identifiable user data - only an encrypted email address that will keep track of help sessions, attendance metrics, calendars, and feedback.

Figure 3.1: Logical Architecture Overview



Description: Figure 3.1 describes the architectural view of the application. The user is able to interact with the system in a few different ways seen in the controller sections. The system records changes made by the user and displays appropriate visuals to the user.

The Model View Controller design pattern is very useful for web applications because it allows for a highly dynamic user interface. Since we will have different user types, it is important that we manage permissions across user types. Typically, user permissions will be dictated by the authority prescribed via the administrator control panel, but depending on the information we can get from Google Oauth, permissions also might be handled directly by the UMaine account system (for example, instructors and student aids have specific designations within their account).

Models are essentially the classes that are described below and shown in their respective class diagrams. The relationships between these classes will be represented in the database and their methods will be written in Django. The various views of the application are going to be coded in a combination of HTML, CSS, and javascript while utilizing the Node.JS framework that will allow us to

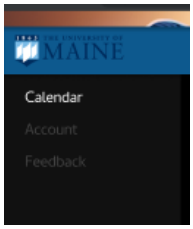
3.4.1 UI Standards

There are some UI elements that will remain constant throughout the application. These will not differ from page to page. Each UI standard to be used is included below with descriptions of appearance and how they will be used.

Figure 3.2: Button Standards

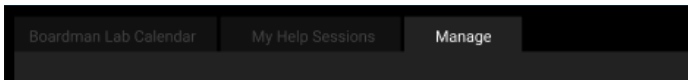
Description: Green buttons are for Creating and Saving actions, such as adding a new help session. Red buttons are for removing and deleting actions, such as removing a help session. Light blue buttons are for editing and altering actions, while dark blue buttons are for navigation.

Figure 3.3: Left Side Navigation Standards



Description: The left side navigation pane is reserved for larger scope navigation such as account view, calendar view, and feedback. This navigation should be a hideable entity for the web version of the application.

Figure 3.4: Tabs In Content Standards

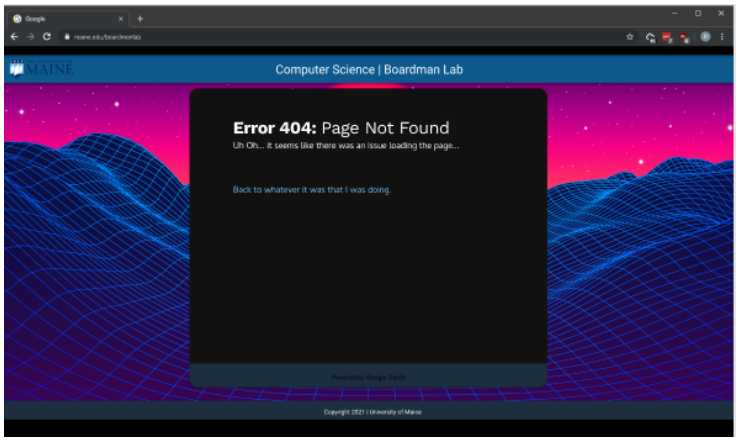


Description: Each content page may have multiple navigation tabs to enable other features of the application to be more easily accessible to the user. Currently selected tabs are highlighted while the other tabs are darkened.

Figure 3.5: Button Link Standards

Description: There are a series of buttons that will direct the user to other content that is contained within the current section, such as switching between day, week, and year views on the calendar page.

Figure 3.6: Error Handling Standards

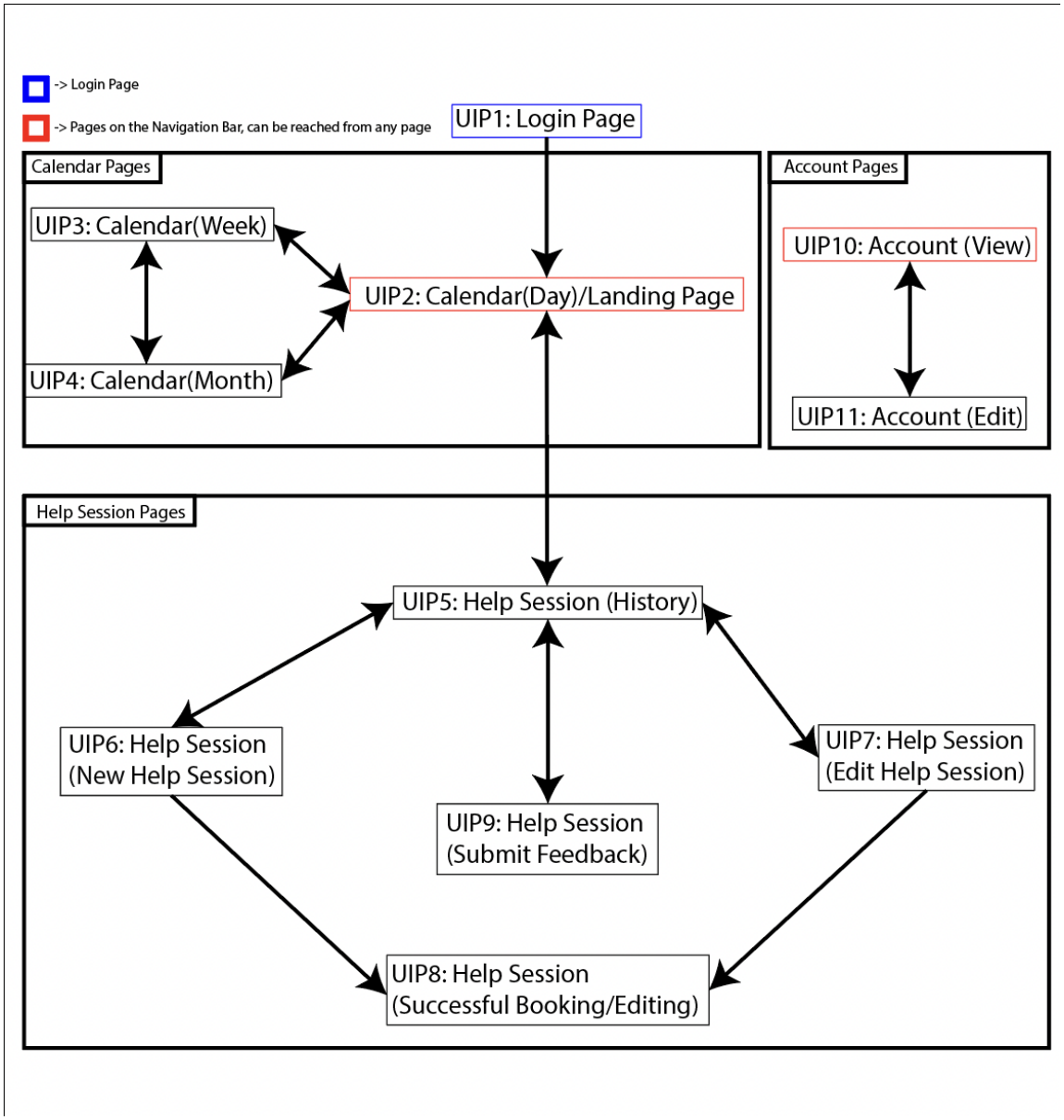


Description: In the event of an error, an appropriate error message will appear and direct the user back to the last correctly loaded page.

3.4.2 UI navigation

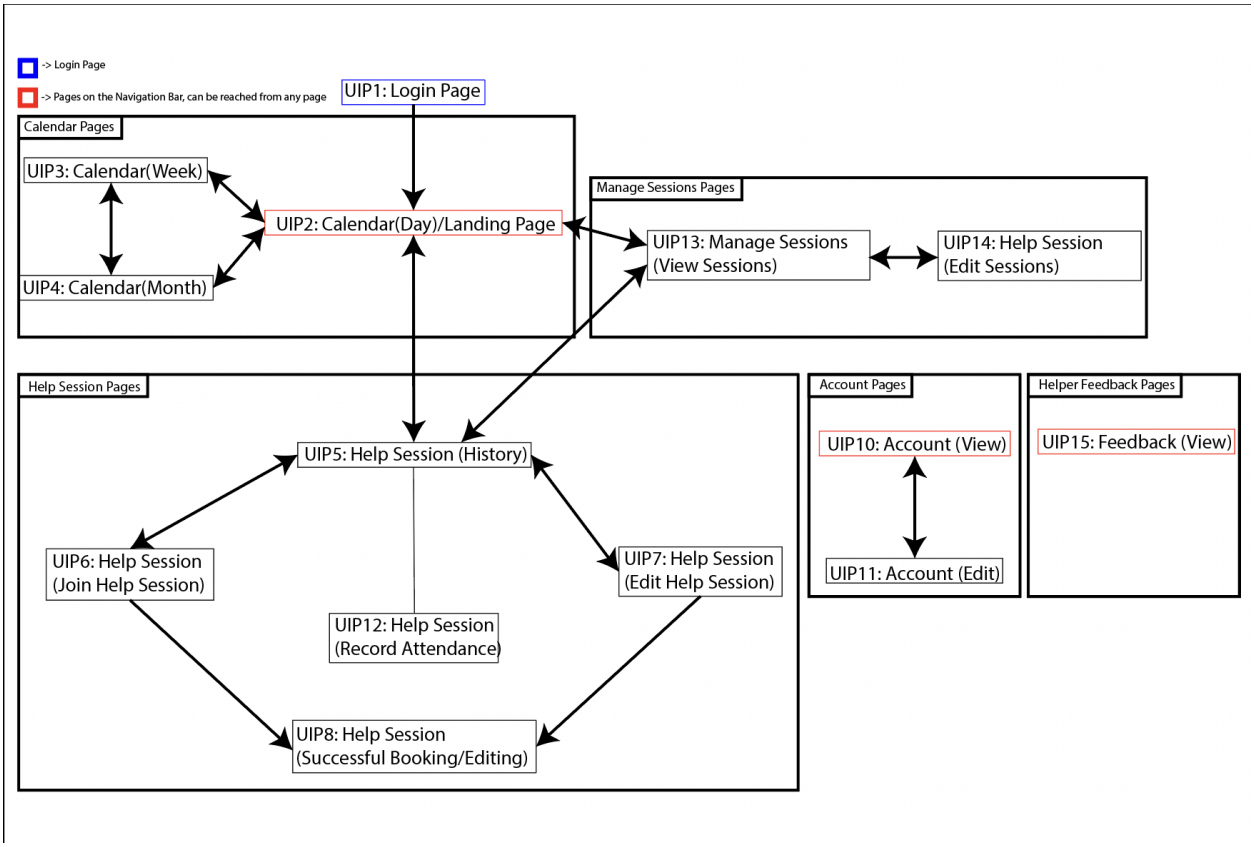
The User interface navigation consists of three sections, the experience as a student user, helper user, and admin user. There are small differences between the three, mostly having to do with permissions to data and scheduling features outlined in figures 3.7-3.9. The arrows indicate directions the user can move between each page. Pages highlighted in red are navigation pages and can be reached from any page. For a more in-depth, description of the pages and UI mockups, see Appendix B.

figure 3.7: Navigation Diagram Student View



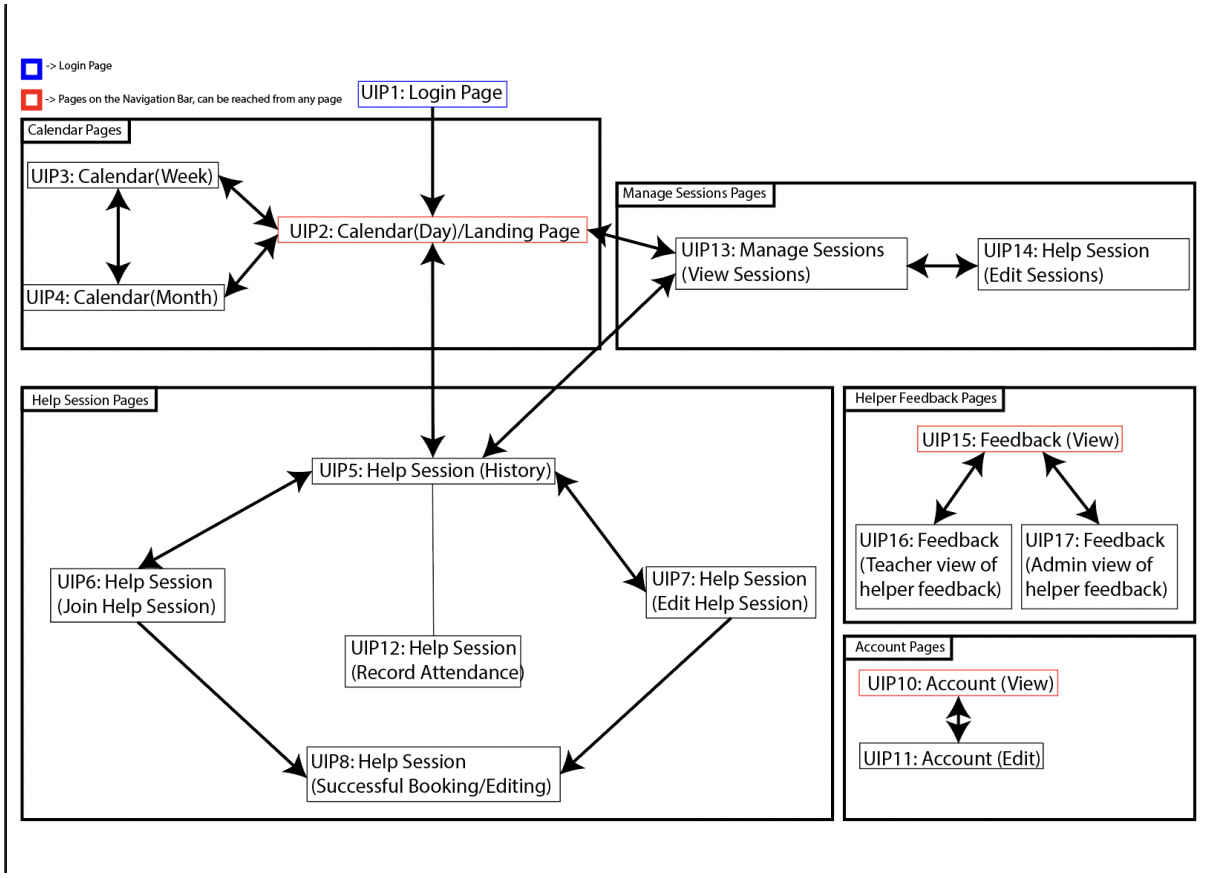
Description: Figure 3.7 shows the navigation paths through each page of the application as a general student user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

figure 3.8: Navigation Diagram Student View



Description: Figure 3.8 shows the navigation paths through each page of the application as a general helper user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

figure 3.9: Navigation Diagram Admin View



Description: Figure 3.9 shows the navigation paths through each page of the application as a general admin user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

4. Data & Report Generation

This section encapsulates the backend handling of the application in terms of data and report generation. Included is a description of the kinds of data the application will collect and how it will be used. Additionally, there are visualizations of the data to be collected, a prototype of the database, and examples of graphs that have been generated from test data.

4.1 Data Collection Overview

The data collected by the application will be useful for a variety of purposes such as scheduling an appropriate number of helpers during high flow periods, identifying which topics or courses students struggle with and setting up workshops for those topics, identifying students that may be struggling and finding them additional resources, or using student feedback to identify particularly useful teaching techniques used by helpers.

Data access will be based on user authorization level. Students will only be able to access reservations they have made and public help sessions, helpers will be able to access any reservation a student has made with them as well as any public help sessions, and administrators will be able to view all reservations and help sessions, as well as student feedback.

4.2 Report Generation Visualizations

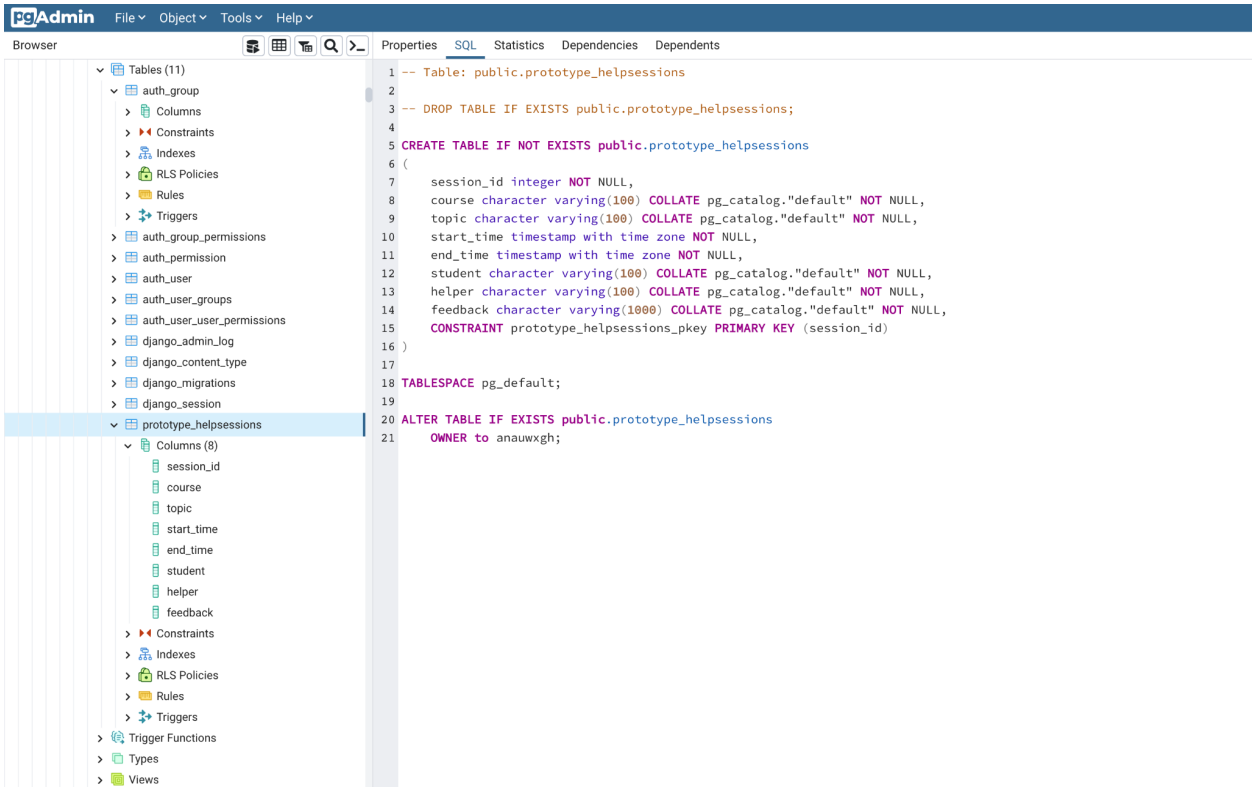
The table and figures in this section illustrate the data collection process from start to finish. The first table demonstrates the types of data that is collected during help session requests and user feedback. The next figure illustrates the database view of table creation using the data collected from the users. The last figure demonstrates reports that can be generated from the data saved in the database. These graphs were created using test data that simulates real application workflow. This information will be vital in tracking things like courses or subjects that students are having difficulty with, days in which the lab system is especially busy, and helpers that are especially popular among students.

table 4.1: Data Validation Description

ID	Data Type	Limits	Format
LogIn.username	String	none	Variable-Length Character Format
LogIn.password	String	length > 8	Variable-Length Character Format
Reservation.date	Date	current date - 4 months ahead	*ISO : yyyy-mm-dd
Reservation.time	Time	start time - end time	*ISO : hh.mm.ss
Reservation.method	String	none	Variable-Length Character Format
Reservation.recurring	Boolean	true, false	Indicator Format
HelpSession.course	String	none	Variable-Length Character Format
HelpSession.helper	String	none	Variable-Length Character Format
HelpSession.topic	String	none	Variable-Length Character Format
HelpSession.duration	Integer	in minutes	Integer Format
Feedback.rating	Integer	min value: 0, max value: 5	Integer Format
Feedback.comment	String	none	Variable-Length Character Format

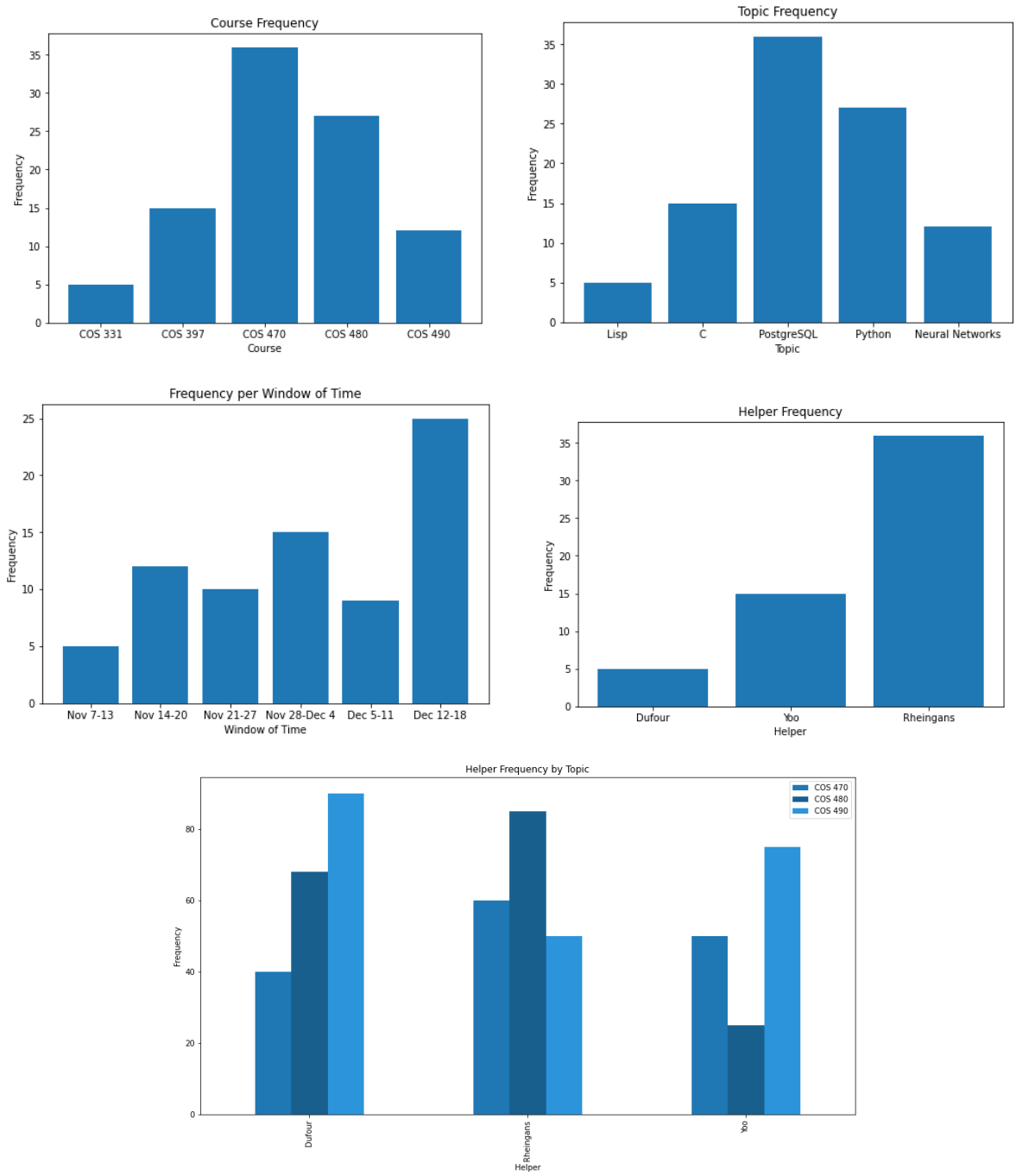
Description: Table 4.1 contains all data that will be entered into the system by a user. Login information will be entered before any other part of the site can be accessed. Reservation and Help Session information will be entered by users when scheduling or editing sessions. Feedback information will only be entered by student accounts after a help session has been attended.

figure 4.1: Help Session Data Collection Prototype



Description: figure 4.1 shows a prototype for the database used to create tables of usable information. Whenever a request for a help session is submitted, a table is created containing each of the above data points to be viewed and used later on by an admin.

figure 4.2: Reports Generated by MatPlotLib



Description: figure 4.2 demonstrates the kinds of reports that can be generated from data collected using the application using MatPlotLib. These reports were generated using test information that models the kind of data traffic that the system will encounter in actual use.

5. Testing

This section covers the testing associated with this application. Included are the types of tests that will be conducted, how they will be conducted, and why they are important.

5.1 Test Conditions

5.1.1 Unit Testing

Since this is a web application, multiple levels of unit testing will need to be written for each module. Almost all of the HTTP requests and template rendering tests will be handled by the Django testing framework while more granular tests for methods will be handled with assertion tests written with pytest. Since the Django testing framework allows for simulated requests, we will be writing unit test cases for each model in the database.

figure 5.1: Unit Test Example

```
from django.test import TestCase
from boardmanlab.models import HelpSession

class HelpSessionTestCase(TestCase):
    def setUp(self):
        HelpSession.objects.create(
            ID = 3a21b4441
            DateTime = "1/1/22 14:00:00",
            Helper = "Sam Morse",
            Topic = "COS125")

    def test_HelpSession_return_helper(self):
        # Help Session Helper can be identified
        0101221400_cos125_MLA = HelpSession.objects.get(ID="3a21b4441")
        self.assertEqual(010122_cos125_MLA.helper(), "Sam Morse")
```

Description: figure 5.1 is an example of a potential unit test that will be conducted in the second semester. This example checks a help session to see if the helper associated with the session is correct.

The test packages will be split into modules and will live in the application. They can be accessed and run with the *manage.py* utility. Modules can be tested individually by calling the test package of that particular module: ./manage.py test HelpSession or tests of a particular module can be tested using the appropriate path to the individual unit test. All tests will be organized by module, model, unit test.

5.1.2 Integration Testing

Integration testing will be done using the python module, Selenium, to render HTML pages. With Selenium, we will be able to write tests that simulate user interaction with the application. This will allow us to automate tests for all use cases. Examples of use case testing are: Google Oauth is working properly with our application, users are able to view and filter the calendar, helpers can create and edit instances of HelpSession.

5.1.3 System Testing

System testing will be done through the Django Test Framework and will focus on performance, stress, reliability, and regression tests. To avoid inefficient database querying, we will be testing the number of queries made by a single request and track requests with an open-source load tester such as LocustIO. Stress testing and reliability

testing will also be implemented using LocustIOl. The codebase will implement continuous integration with Github Actions. By using DevOps principles we will ensure that any updates to the code will be properly tested before any build occurs. System testing methods are subject to change during development.

5.2 Test Procedures

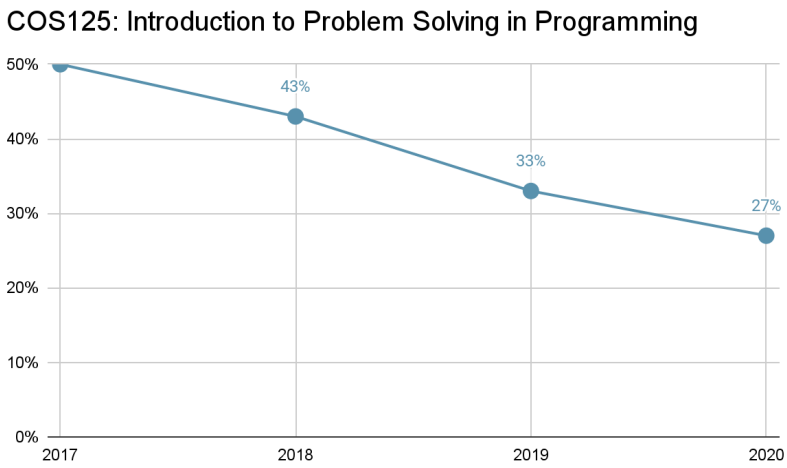
The application is going to be tested using the sandwich approach. Development and testing are going to start with the User Interface. The UI will be built out in HTML and CSS, while individual modules such as Calendar, Help Session, and Feedback will be scaffolded. Modules and methods will be unit tested and integration tested respectively as they are developed. Github actions will facilitate continuous integration and testing during the development process.

6. Conclusions

6.1 Analysis

Students benefit greatly from being able to interface directly with Teaching Assistants and Maine Learning Assistants in the Boardman Lab at the University of Maine. Since introducing the MLA program and adding the Boardman Lab, there has been a strong reduction in student failure rates for introductory courses.

Figure 6.1: Failure Rate Decline over three years



Description: Figure 6.1 illustrates the decline of failure rates in COS125 from 2017-2020. The reason for the slight decline in 2018 is due to COS120 becoming available to students with lower math aptitude scores. In 2019 MLAs and the Boardman Lab were added, leading to a sharp decrease in failure rates leading into 2020 at 27%.

By creating a tool that creates a greater vision on the schedule of office hours and help sessions, students would be able to leverage the Boardman Lab even more. Adding options to sign up for Office Hours and Help sessions and tracking participation and help topics would further inform instructors on how students are learning and what the problem areas are in their classes. Our goals for this application are to increase the number of students accessing the lab and allow instructors to get a better understanding of how their students are learning.

6.2 Conclusions

This application is designed to facilitate the improvement of the Boardman Computer Science lab at the University of Maine. We have developed a list of requirements that describe an application that allows students to access an up to date schedule of TA and MLA office hours in the Boardman lab, students to sign up for help sessions, and provide feedback to instructors, TAs, and MLAs to keep track of their help sessions and receive feedback, and instructors to track lab usage and get analytics about popular help topics and areas of concern for their students.

6.3 Recommendations

For continuous improvement of the wellbeing and academic performance of Computer Science students at the University of Maine, we recommend a web application to schedule and manage the Boardman Lab Help Sessions. A web-based application would allow access from anywhere there is an internet connection and portability to any internet-connected device with browser capabilities.

We recommend that the application is built on a framework that allows for modularity and tractability. The Django web framework is suited for this application because it has broad support in the development community and allows for expandability of applications. This is an important consideration because it will allow future developers to maintain and add features.

We recommend using a relational database to manage the data within the application. We advocate for a PostgreSQL database because it has strong support with our proposed framework and broad adoption within the development community.

During development, it is advised that a continuous integration approach is used to maintain the codebase in order to increase code coverage, decrease code review time, and decrease build time. By maintaining the codebase in GitHub, continuous integration can be achieved using the GitHub actions toolset.

In-house Operations is committed to getting this application launched in the next five months (5/1/22). We are planning to build in a top down approach starting with the User Interface. Work has already been done configuring the database and we hope to have the UI completed in the next four to five weeks (1/20/22). During the development of the UI, scaffolding will be constructed to take the place of our modules. Then it is a matter of building our modules and testing systematically with our team. We would like to implement a continuous integration approach during development meaning our team is fully prepared to take an AGILE stance on development - releasing early and often - so that our client is able to influence the crafting of this application from start to finish. It is essential that this application is built in conjunction with the feedback of our client and we plan on meeting with Chris Dufour regularly throughout development. This is going to be a challenging development process, and it is vital that we have strong communication channels with Umaine and our client for the utilization of Google Oauth which is ultimately going to enable seamless adoption from the users. Testing is of the utmost importance to us and we will be writing tests alongside each module to ensure that the end product is ready for deployment.

Appendix A – Back Matter

A.1 References

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, November 20). *In-house operations UIDD v.1.* Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/1ExOExrAWAgdNB8ay1yYSWllcMcLxWPr_ylOSGEBPz1U/edit?usp=sharing.

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, October 18). *In-house operations SRS*. Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/1YIFScQdYOcsTWcKpfTEac3g4aTRo3XtSTO1Uay2CQv4/edit?usp=sharing.

Bendo, K., Brisson, J., Landry, A., Morse, S., Schanck, A., & Swift, F. (2021, October 28). *In-house operations SDD v. 1.0*. Google Docs. Retrieved November 29, 2021, from https://docs.google.com/document/d/1qvNbDHLXdX5J8jHynGA8STqZW0qGLAZs1bayzoYAhoY/edit?usp=sharing.

Dufour, C., & Rheingans, P. (2021, September 16). dufour_help-Resource-Scheduling. Orono.

Morse, S. (2021, November 21). *Figma Design*. Retrieved December 12, 2021, from https://www.figma.com/file/YvE53I7e9N8KaubB23RP2m/Main(e)-Site?node-id=0%3A1.

Schanck, A. (2021, October 4). *Team 17 capstone proposal*. Google Docs. Retrieved December 13, 2021, from https://docs.google.com/document/d/19nm8LNdbCEEdSQNdVRj570LcsRJC7rjRumRwU4srtBE/edit?usp=sharing.

A.2 Bibliography

Division of Marketing and Communications. (2018). 2018 Branding Standards | University of Maine.

Django. (n.d.). Retrieved December 13, 2021, from https://www.djangoproject.com/.

Llazzaro, L. (2021, May 19). *Django-Scheduler/readme.md at Develop · Llazzaro/Django-Scheduler*. GitHub. Retrieved December 13, 2021, from https://github.com/llazzaro/django-scheduler/blob/develop/README.md.

OAuth Community Site. (n.d.). Retrieved December 13, 2021, from https://oauth.net/.

PostgreSQL as a Service. ElephantSQL. (n.d.). Retrieved December 13, 2021, from https://www.elephantsql.com/.

Slack. (n.d.). *Slack is where the future works*. Slack. Retrieved December 13, 2021, from https://slack.com/.

Your place to talk and hang out. Discord. (n.d.). Retrieved December 13, 2021, from https://discord.com/.

A.2 Acknowledgements

There are many who deserve acknowledgment for their help in our progress this first semester. First and foremost, our primary client Mr. Christopher Dufour and client Dr. Penny Rheingans have been helpful, patient, and knowledgeable throughout this process. Additionally, thank you to the students who participated in our survey who provided invaluable insight to use for how to proceed with this application. Finally thank you to Professor Yoo who has been a steady captain on the COS397 ship and has taught us many important lessons for our project and lives.

Appendix B – Prior Documentation

Software Requirements Specification Document

Boardman Computer Science Lab Web Portal

Client: Mr. Christopher Dufour

Development Company: In-House Operations

Development Team:

Klei Bendo
Jack Brisson
Alex Landry
Samuel Morse
Aaron Schanck
Forrest Swift

Date: 10/18/2021

Boardman Computer Science Lab Web Portal
System Requirements Specification

Table of Contents

1. Introduction

The Boardman Computer Science Lab Web Portal is an all encompassing tool for computer science students at the University of Maine. It is designed to ensure better help for students seeking aid in both specific inquiries, and broad subject areas at the Boardman Computer Science Lab through use of an interactive calendar, individual and group meeting scheduling, forum posting, and news updates. The Web Portal will make the Boardman Computer Science Lab more accessible and easy to use for University of Maine Computer Science Students.

1.1 Purpose of This Document

The purpose of this document is to define and describe the purposes, processes, and goals encapsulated within the development of the Boardman Computer Science Lab Web Portal project. It is intended for viewership by the client for confirmation of procedure for the development team, and other interested parties for official documentation of the application. This document includes background information on the project's development, requirements of the application both functional and non-functional, necessary future documentation and development, and signed agreements of both the development team and client for adequate directive of project scope and features.

1.2. References

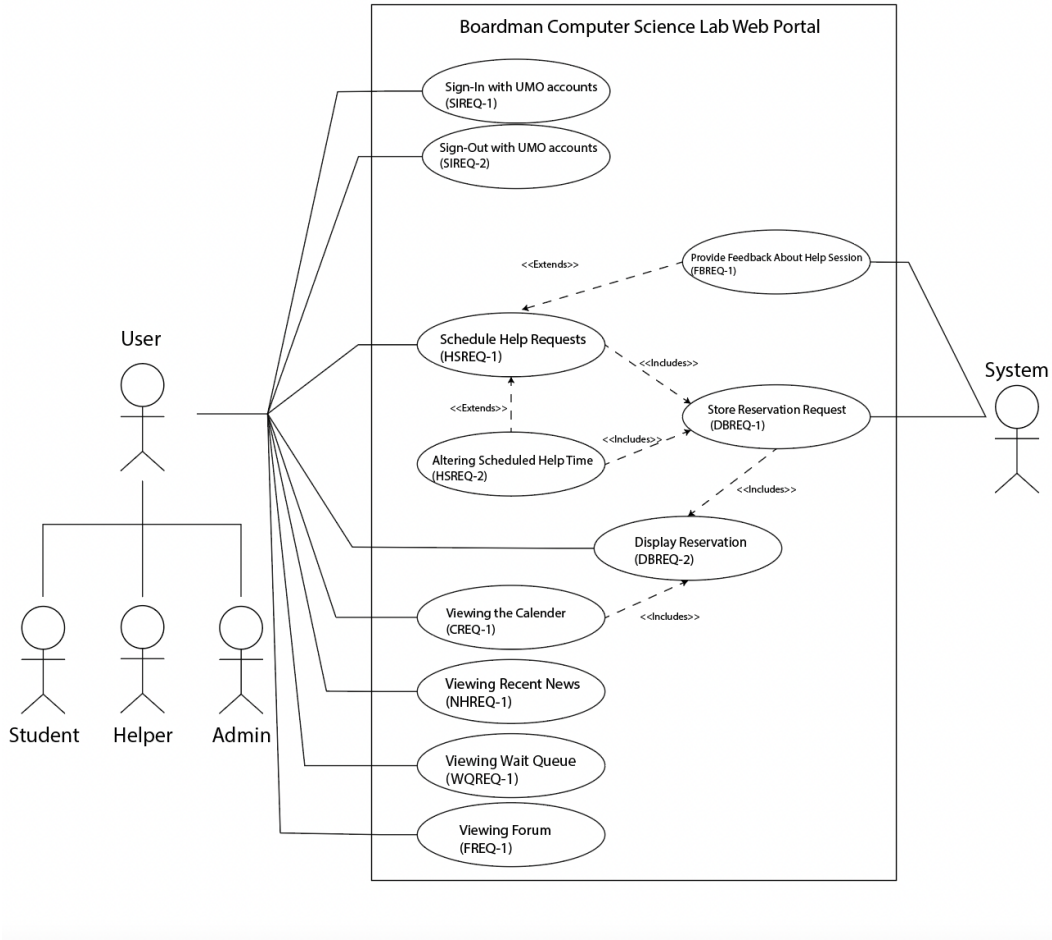
TBD - note: references will be added as they are created

1.3. Purpose of the Product

The Boardman Computer Science Lab is a long-standing resource for UMaine Computer Science students to receive help with computer science and other related issues, and as a meeting place for student groups. One common complaint facing the lab is a lack of clarity in regards to when lab helpers are available, and the areas in which these helpers are able to lend assistance. The Boardman Computer Science Lab Web Portal is a program aimed to both alleviate this complaint in a variety of ways, and to also track user data related to wait times and traffic within the labs to better address student needs moving forward.

1.4. Product Scope

This section identifies the boundary between the system under development and the outside world. To demonstrate this, a use case diagram is provided to display scope of functionality at the topmost level. This diagram displays the major features of the application and their interactions with the users and the system.



2. Functional Requirements

Included in this section are the functional requirements intended for the Boardman Computer Science Lab Web Portal. Each section is denoted by a major feature of the application. It contains a description and priority (1 = lowest, 5 = highest), stimulus/response sequences of the major requirements, the remainder of the associated functional requirements, and a use-case diagram of the feature. See the top-level use case diagram referred to in Section 1.4. for more context of the scope of this application.

2.1 Sign in / Sign out

2.1.1 Description and Priority

This feature will allow users to use their UMaine login credentials for the Boardman Lab website. This will allow the system to maintain security as well as store information about scheduled meeting times, requests for help and other information. This functionality is priority 5.

2.1.2 Stimulus/Response Sequence

p

Sign-In

Number	SIREQ-1
Name	Sign-In with UMO accounts
Summary	The system shall allow the user to enter the site with UMaine Login credentials
Priority	High
Preconditions	User has a umaine account.
Postconditions	User is signed into their umaine account and has access to the site
Primary Actors	Users
Secondary Actors	Google OAUTH, systems database
Triggers	Visiting the site
Main Scenario	1. User goes to the Boardman Lab website and selects “login” 2. User enters their UMaine credentials in the login screen 3. Using the OAUTH api, the system verifies the UMAINE user account and gives them access to the site.
Extensions	2.1 The user enters invalid credentials 2.2 The user is not allowed entry into the site
Open Issues	None

Sign-Out

Number	SIREQ-2
Name	Sign-Out of UMO account
Summary	The system shall allow the user to log out of the users account
Priority	High
Preconditions	Users are logged into their UMaine account on the boardman lab site.
Postconditions	User is signed out of their UMaine account and no longer have access to the site.

Primary Actors	Users
Secondary Actors	Google OAUTH, systems database
Triggers	Selecting “logout” button
Main Scenario	1. An onclick event calls signout function in OAuth API 2. System updates user authorization status 3. Site redirects user to the sign in page.
Extensions	1.1 User is signed in but inactive for 20 minutes 1.2 System calls signout function 2. -
Open Issues	None

2.1.3 Functional Requirements

SIREQ-3: The system shall utilize university of maine sign-in accounts for login.

SIREQ-4: The system shall utilize OAuth to manage logins.

SIREQ-5: The system shall allow the user to logout of their account.

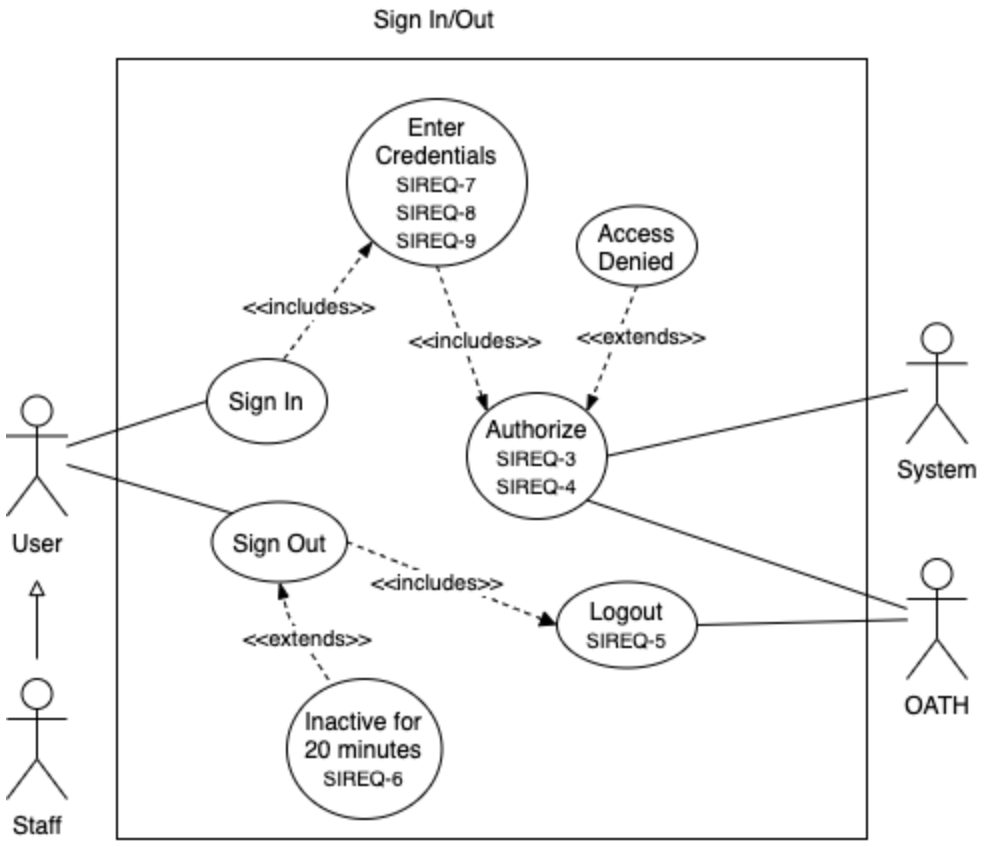
SIREQ-6: The system shall lock access to an account when not in use for a set amount of time.

SIREQ-7: The system shall have a user type for student.

SIREQ-8: The system shall have a user type for helper.

SIREQ-9: The system shall have a user type for administrator.

2.1.4 Use case diagram



2.2 Help Scheduling

2.2.1 Description and Priority

This feature will allow users to select time slots that are available on the calendar and create an instance of a help session. The user will be able to specify which class or topic they would like help with and be able to check availability of helpers. This functionality is priority 3.

2.2.2 Stimulus/Response Sequence

Schedule Help Request

Number	HSREQ-1
Name	The system shall allow users to Schedule help Requests
Summary	User selects time they would like to help and the system allows them to schedule that time with a helper either online or in person
Priority	Medium

Preconditions	User is signed in
Postconditions	System has created new item on the schedule for that student
Primary Actors	Users
Secondary Actors	System
Triggers	Selecting the “schedule help” button on the UI
Main Scenario	1. The user selects a date and time that they would like to get help by selecting an available time from the UI prompt. 2. The system checks to see if there is space available in the designated time slot wait queue.
Extensions	1.1 The user enters the class that they need help with in the prompt within the UI 1.2 The user selects the helper that they would like in the prompt within the UI 1.3 The user selects the mode of help: online or in person and one-on-one or group help. 2.1 If the selected time slot has too many students in the wait queue, the system displays a message saying that the selected time is busy. 2.2 If the selected time slot has space available in the wait queue, the system will add the student to the wait queue for that time slot.
Open Issues	None

Alter a Scheduled Help Request

Number	HSREQ-2
Name	The system shall allow the user to Alter a scheduled help time
Summary	User decided to cancel or move their scheduled time for a help session
Priority	High
Preconditions	User has a scheduled help session
Postconditions	User alters the scheduled help session
Primary Actors	Users
Secondary Actors	System

Triggers	User selects “Edit Scheduled Help Session”
Main Scenario	1. System displays the edit help session page
Extensions	1.1.1 User selects cancel help session 1.1.2 System cancels the help session and removes the user from the wait queue 1.2.1 User changes the scheduled help time 1.2.2 System cancels help session and removes user from the wait queue 1.2.3 System creates new scheduled help session with the new time.
Open Issues	None

2.2.3 Functional Requirements

HSREQ-3: The system shall allow requests for help at a specific date and time.

HSREQ-4: The system shall allow requests to schedule recurring meetings.

HSREQ-5: The system shall allow requests to search for available times matching a criteria.

HSREQ-6: The system shall allow requests to meet with a specific helper.

HSREQ-7: The system shall allow requests for help with a specific topic or course.

HSREQ-8: The system shall allow help reservations to have various modifications.

HSREQ-9: The system shall allow for remote or in person reservations.

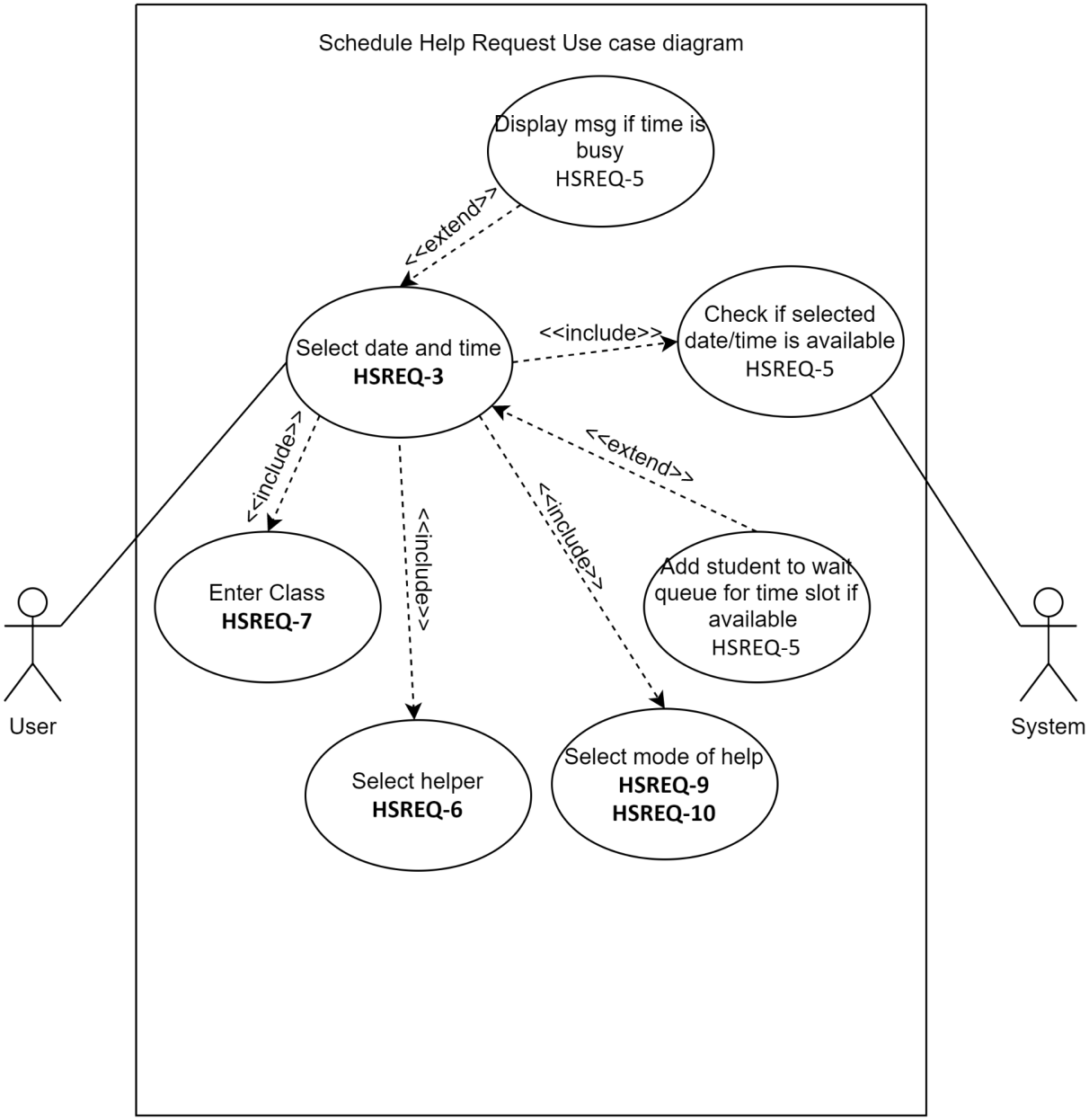
HSREQ-10: The system shall allow for solo or group help.

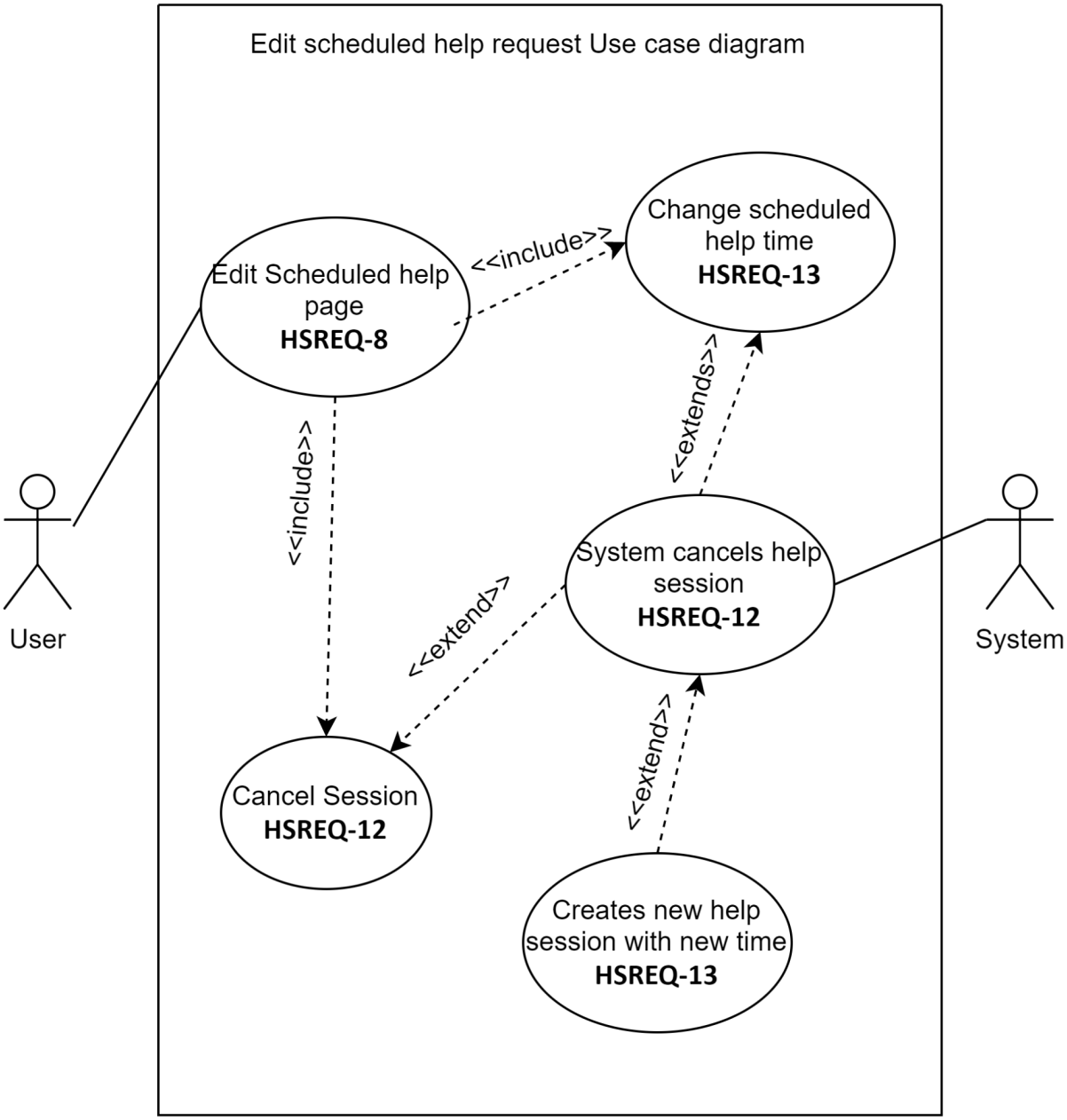
HSREQ-11: The system shall allow a limited number of students to join in a study group meeting

HSREQ-12: The system shall allow users to cancel a reservation.

HSREQ-13: The system shall allow users to move a reservation to a new valid time.

2.2.4 Use Case Diagrams


Schedule Help Request Use case diagram
Display msg if time is busy
HSREQ-5
<<extend>>
Select date and time
HSREQ-3
<<include>>
Check if selected date/time is available
HSREQ-5
<<include>>
<<extend>>
<<include>>
<<include>>
Enter Class
HSREQ-7
Add student to wait queue for time slot if available
HSREQ-5
Select helper
HSREQ-6
Select mode of help
HSREQ-9
HSREQ-10
User
System



2.3 Feedback

2.3.1 Description and Priority

The application will have opportunities for students to give feedback about their help sessions by accessing a history help session they have attended. These reviews will be accessible to the professors and TAs of any given class. This functionality is priority 4.

2.3.2 Stimulus/Response Sequence

Feedback for Help Session

Number	FBREQ-1
Name	Provide Feedback about Help Session
Summary	The system shall allow Students to give feedback on the helper who led a help meeting.
Priority	High
Preconditions	Student has attended a help session.
Postconditions	Feedback is saved and ready for review.
Primary Actors	Users
Secondary Actors	System
Triggers	Student selects 'Provide Feedback' for a help session in their history.
Main Scenario	1. System displays an embedded form within the website for the student to fill. 2. Student presses Submit. 3. System saves the response.
Extensions	1.1: Student enters name of help provider. 1.2: Student enters a short paragraph of what was done during the session. 1.3: Student enters the amount of time in minutes for how long the interaction lasted 1.4: Student enters an integer from 0-10 describing their satisfaction.
Open Issues	None

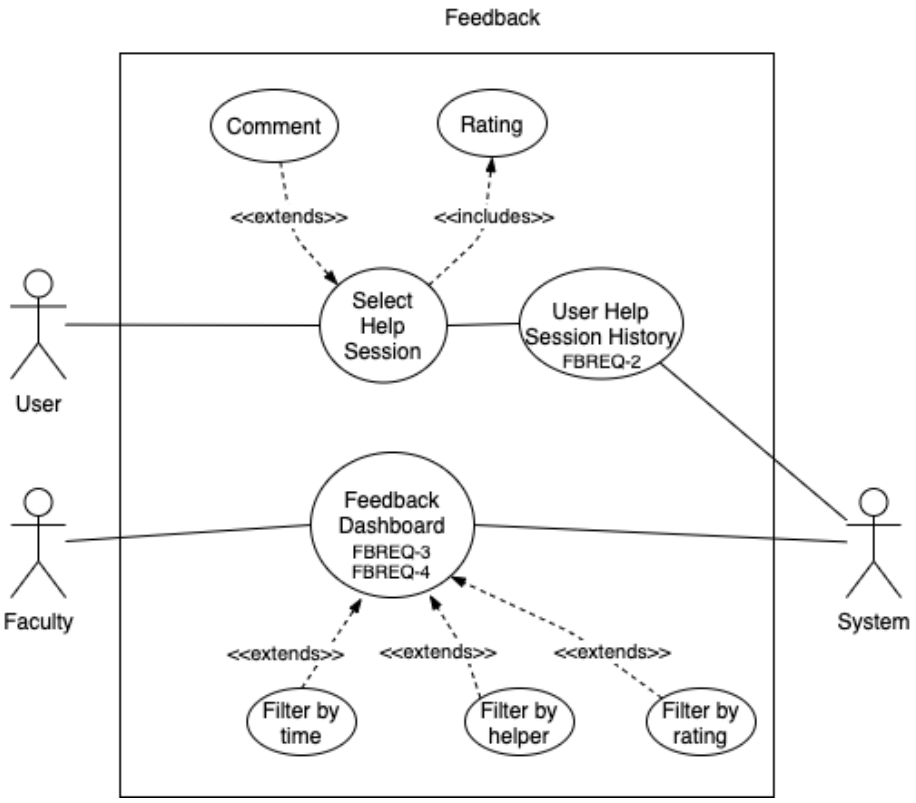
2.3.3 Functional Requirements

FBREQ-2: The system will display history of help sessions attended by the student.

FBREQ-3: The system shall allow for anonymous reviews of help sessions.

FBREQ-4: The system shall allow for anonymous reviews of helpers.

2.3.4 Use Case Diagram



2.4 Calendar

2.4.1 Description and Priority

The calendar view is a quick glance at who is in the lab at any given time and what they are most likely able to help with. This gives students and faculty vision on the help availability for any given day. The calendar is meant to be an interactive element with links to more information about a staff member. This functionality is priority 4.

2.4.2 Stimulus/Response Sequence

View Calendar

Number	CREQ-1
Name	Viewing the Calendar
Summary	The system shall allow a user to view a calendar of schedules and events
Priority	High
Preconditions	User is logged in
Postconditions	User has desired information about the calendar
Primary Actors	Users

Secondary Actors	System
Triggers	User clicks on the “Calendar” button
Main Scenario	1. The system displays a view of the calendar to the user 2. The user selects a time on the calendar 3. The user selects the week view 4. The user selects the month view
Extensions	2.1 The system displays the staff that are attending to the lab at that given time, their specialization, class they teach, and position 3.1 The system displays a summary for help availability for any given week 4.1 The system displays a summary for the help availability for any given month
Open Issues	None

2.4.3 Functional Requirements

CREQ-2: The system shall display a schedule of helpers and events.

CREQ-3: The system shall display the names of lab helpers that are available

CREQ-4: The system shall display the times that lab helpers are available.

CREQ-5: The system shall display the classes that the lab helper can assist with.

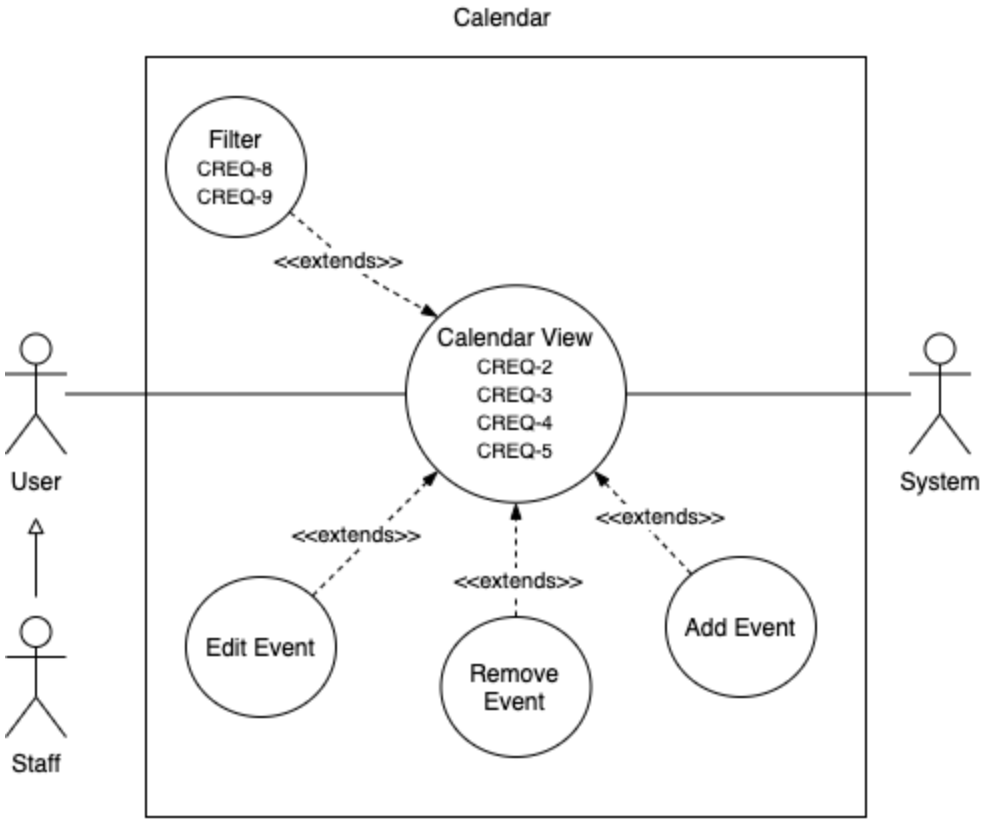
CREQ-6: The system shall display any scheduled group help sessions.

CREQ-7: The system shall display any scheduled refresher lectures.

CREQ-8: The system shall provide an option to view the calendar by week.

CREQ-9: The system shall provide an option to view the calendar by month.

2.4.4 Use Case Diagram



2.5 Database

2.5.1 Description and Priority

The database stores all pertinent information related to the calendar, scheduled help sessions, and user information. The database is also able to generate reports based on specified criteria for analytical purposes. This functionality is priority 5.

2.5.2 Stimulus/Response Sequence

Store Reservation Request

Number	DBREQ-1, 2, 3, 4
Name	Store Reservation Request
Summary	The system shall store user reservation requests.
Priority	High
Preconditions	New reservation has been created

Postconditions	Reservation is stored in a help time instance within the calendar
Primary Actors	User
Secondary Actors	System
Triggers	User submits reservation via the Schedule Help module
Main Scenario	1. The system adds an instance of a reservation to the desired help time object within the database 2. The system updates the calendar 3. The system stores the instance of the reservation to the user's profile. 4. The system displays the new number of users attending the designated help session
Extensions	1.1 The reservation is invalid (overlap). 1.2 The reservation is canceled and an error displayed.
Open Issues	

Edit Reservation

Number	DBREQ-5
Name	Edit Reservation
Summary	The system shall allow Administrators, Helpers, or Students to edit a previously established reservation.
Priority	High
Preconditions	An established reservation.
Postconditions	The reservation is changed according to user specifications.
Primary Actors	Administrator, Student
Secondary Actors	System
Triggers	User selects edit reservation option.
Main Scenario	1. A student selects the edit reservation option. 2. A list of all reservations made by the student is displayed. 3. The student selects a reservation.

	4. The student makes modifications to the reservation. 5. The reservation is replaced.
Extensions	1.1 An administrator selects the edit reservation option. 1.2 A list of all reservations is displayed.
Open Issues	

Store Help Session Data

Number	DBREQ-6, 7, 8, 9
Name	Store Help Session Data
Summary	The system shall store: name of student, the course that the student was helped with, name of staff member that helped the student, the topic and the duration of the session.
Priority	High
Preconditions	Information on help session has been submitted.
Postconditions	A new entry has been added to a table within the database.
Primary Actors	Administrator, Helper
Secondary Actors	System
Triggers	Data on a help session is submitted to the database.
Main Scenario	1. Information on a help session is entered by an administrator, helper, or recorded by the system. 2. A database entry is created recording the help session for future reports.
Extensions	
Open Issues	None

Display Reservation

Number	DBREQ-10, 11, 12, 13
Name	Display Reservation
Summary	The system shall send notifications and viewing options associated with help reservations to applicable

	users.
Priority	High
Preconditions	A valid help reservation has been submitted and processed
Postconditions	The reservation information is displayed to all applicable parties
Primary Actors	Users
Secondary Actors	System
Triggers	User submits reservation via the Schedule Help module
Main Scenario	1. The system adds an instance of a reservation 2. The system updates the calendar 3. The system stores the instance of the reservation to the user's profile. 4. The system displays the new number of users attending the designated help session
Extensions	1.1 The user enters an invalid meeting request 1.2 The system displays a warning to the user, asking them to change their entry.
Open Issues	None

Search Reservations

Number	DBREQ-14, 15, 16, 17, 18
Name	Search Reservation
Summary	The system shall display reservation instances based on search criteria such as: course, topic, time window, helper
Priority	High
Preconditions	Reservations table must be populated inside the Database.
Postconditions	A list of reservations is returned.
Primary Actors	Administrator
Secondary Actors	System
Triggers	One or multiple criteria have been selected, and the

	Search button has been pressed
Main Scenario	1. Administrator selects one or multiple search criterias. 2. Administrator enters text for each search criteria selected. 3. System retrieves all reservations that match the text from each of the selected criteria. 4. System displays the retrieved reservations.
Extensions	
Open Issues	None

Generate Report

Number	DBREQ-19, 20, 21, 22, 23
Name	Generate Report
Summary	The system shall allow the user to generate a variety of reports to analyze the data gathered.
Priority	High
Preconditions	A populated database
Postconditions	A report has been generated based on specifications provided.
Primary Actors	Administrator
Secondary Actors	
Triggers	The generate report option is selected.
Main Scenario	1. Administrator selects generate report 2. Administrator selects report mode (by course, by topic, by window of time, or by helper). 3. System retrieves all data that matches selected criteria. 4. System generates a formatted report of information for analytical use by administrators.
Extensions	
Open Issues	

2.5.3 Functional Requirements

Store Reservation Request

DBREQ-1: The system shall store user reservation requests.
DBREQ-2: The system shall manage reservation requests
DBREQ-3: The system shall check reservation requests for conflict.
DBREQ-4: The system shall store valid reservations.

Edit Reservation

DBREQ-5: The system shall allow for edits to reservations already in the database.

Store Help Session Data

DBREQ-6: The system shall store walk-ins data entered by administrators.
DBREQ-7: The system shall store the course the student was helped with.
DBREQ-8: The system shall store the topic the student was helped with.
DBREQ-9: The system shall store the duration of each help session.

Display Reservations

DBREQ-10: The system shall send notifications and viewing options associated with help reservations to applicable users.
DBREQ-11: The system shall display reservations**.**
DBREQ-12: The system shall display all reservations on a schedule to administrators.
DBREQ-13: The system shall display a student's own reservations to them.

Search Reservations

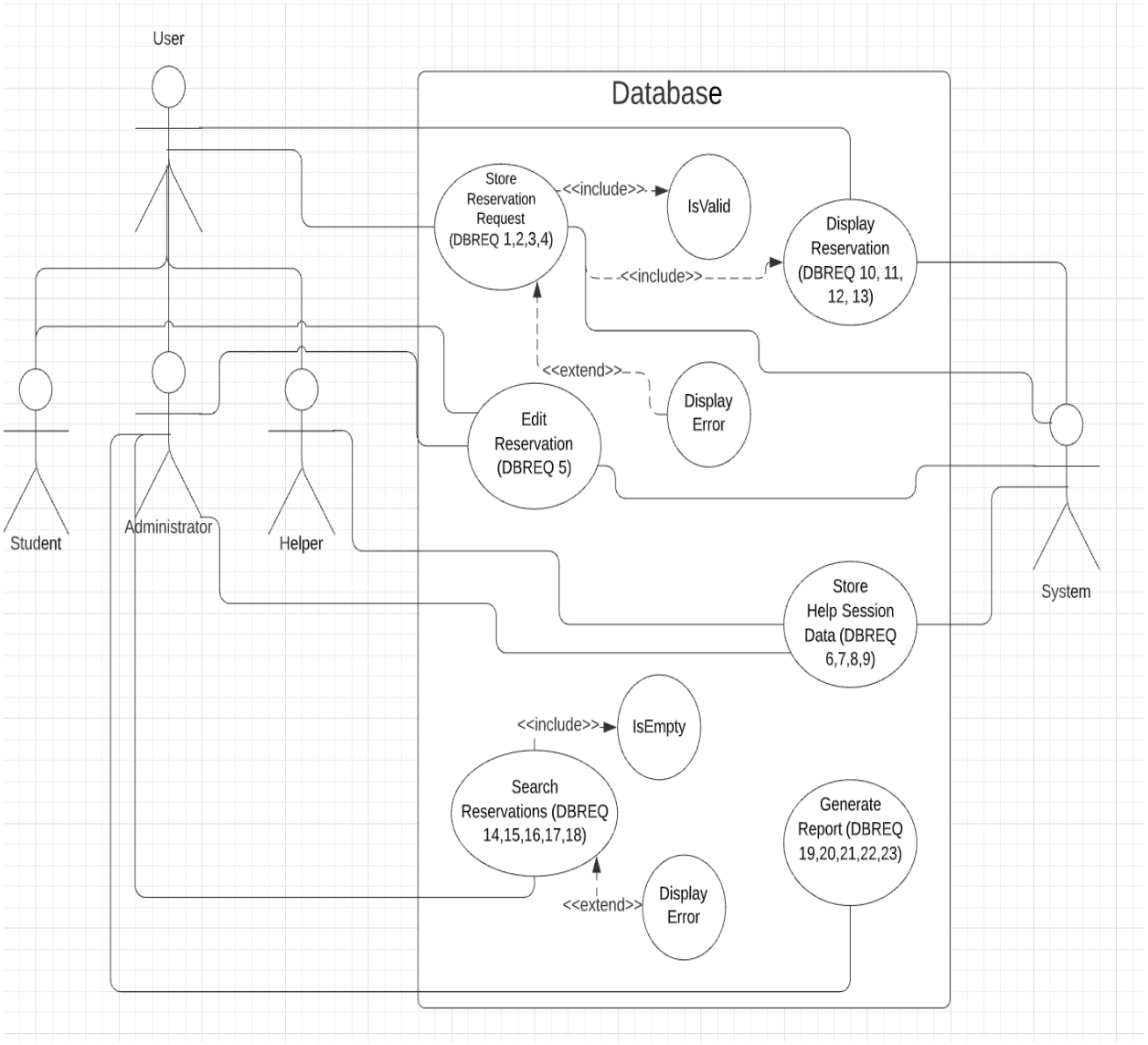
DBREQ-14: The system shall allow administrators to search reservations with criteria.
DBREQ-15: The system shall allow for searches by course.
DBREQ-16: The system shall allow for searches by topic.
DBREQ-17: The system shall allow for searches by window of time.
DBREQ-18: The system shall allow for searches by helper.

Generate Report

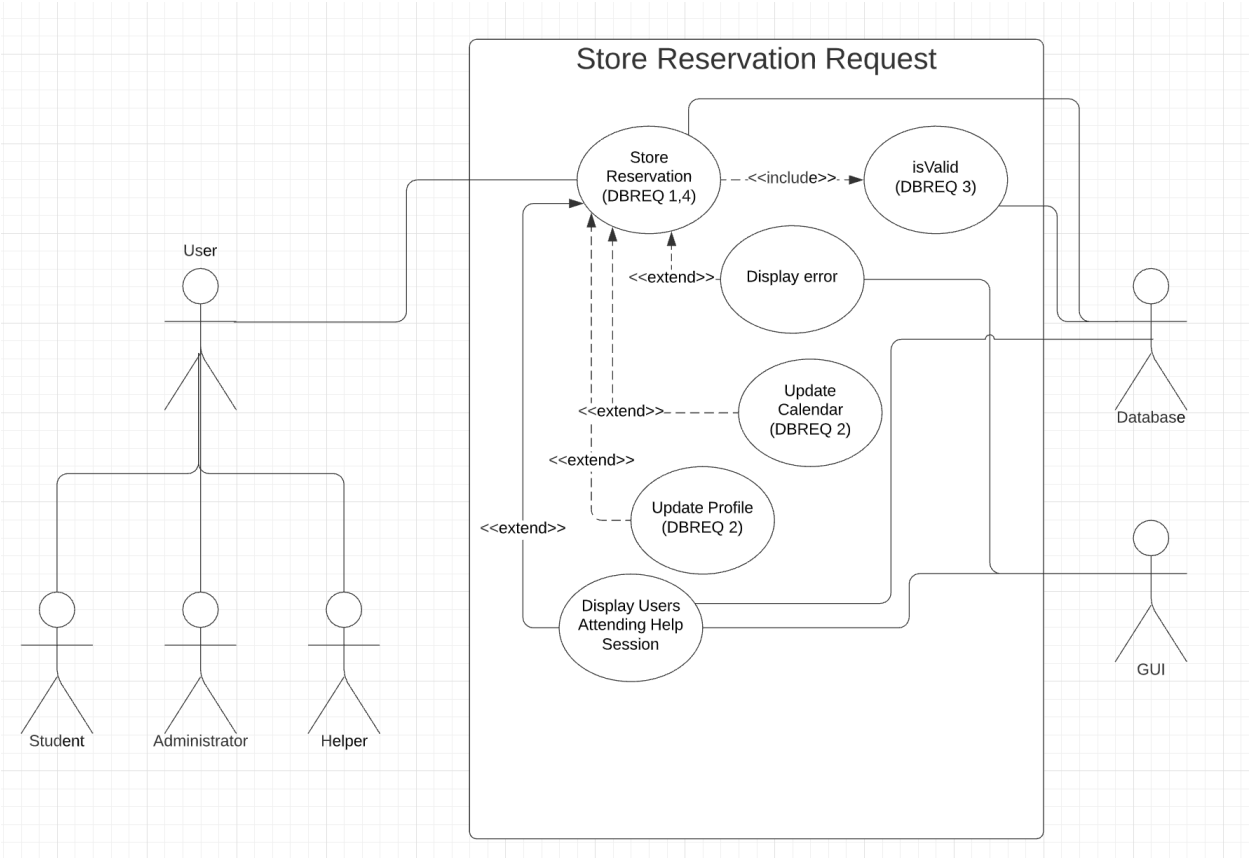
DBREQ-19: The system shall generate reports based on data from a specified window.
DBREQ-20: The system shall generate frequency reports based on course.
DBREQ-21: The system shall generate frequency reports based on topic.
DBREQ-22: The system shall generate frequency reports based on windows of time.
DBREQ-23: The system shall generate frequency reports based on helper.

2.5.4 Use Case Diagram

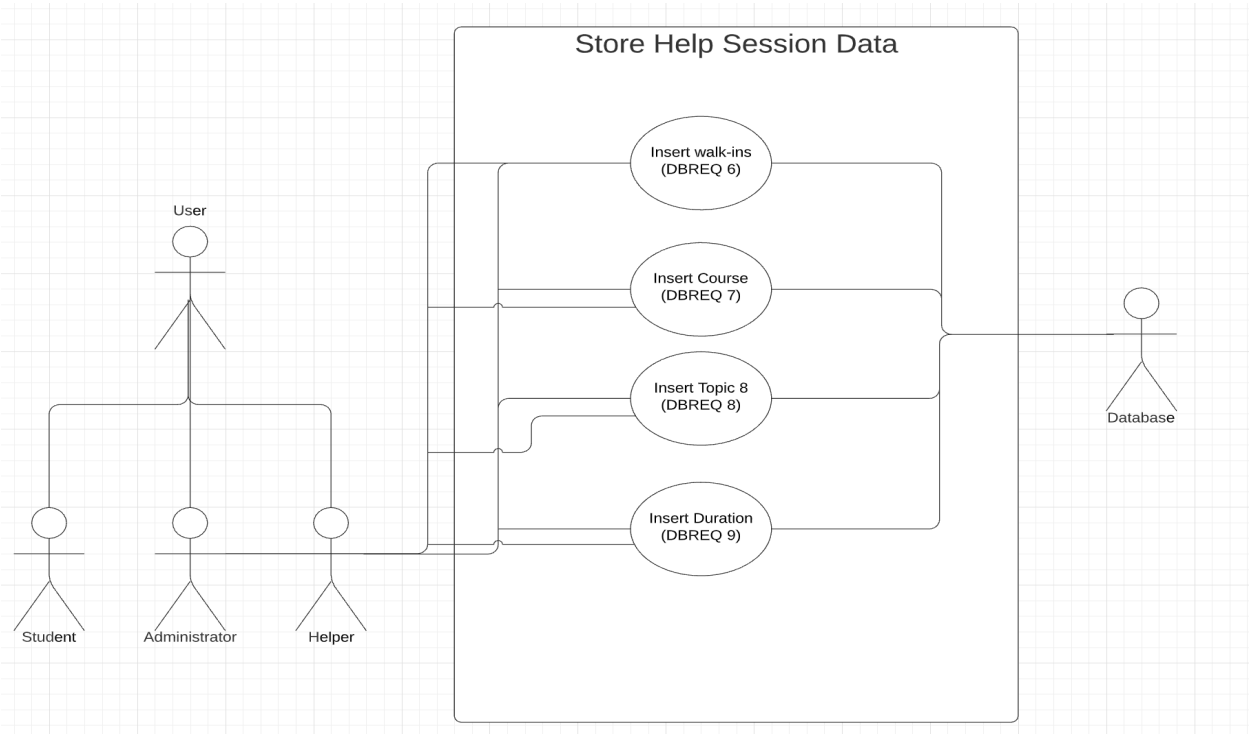
2.5.4.1 Top Level View



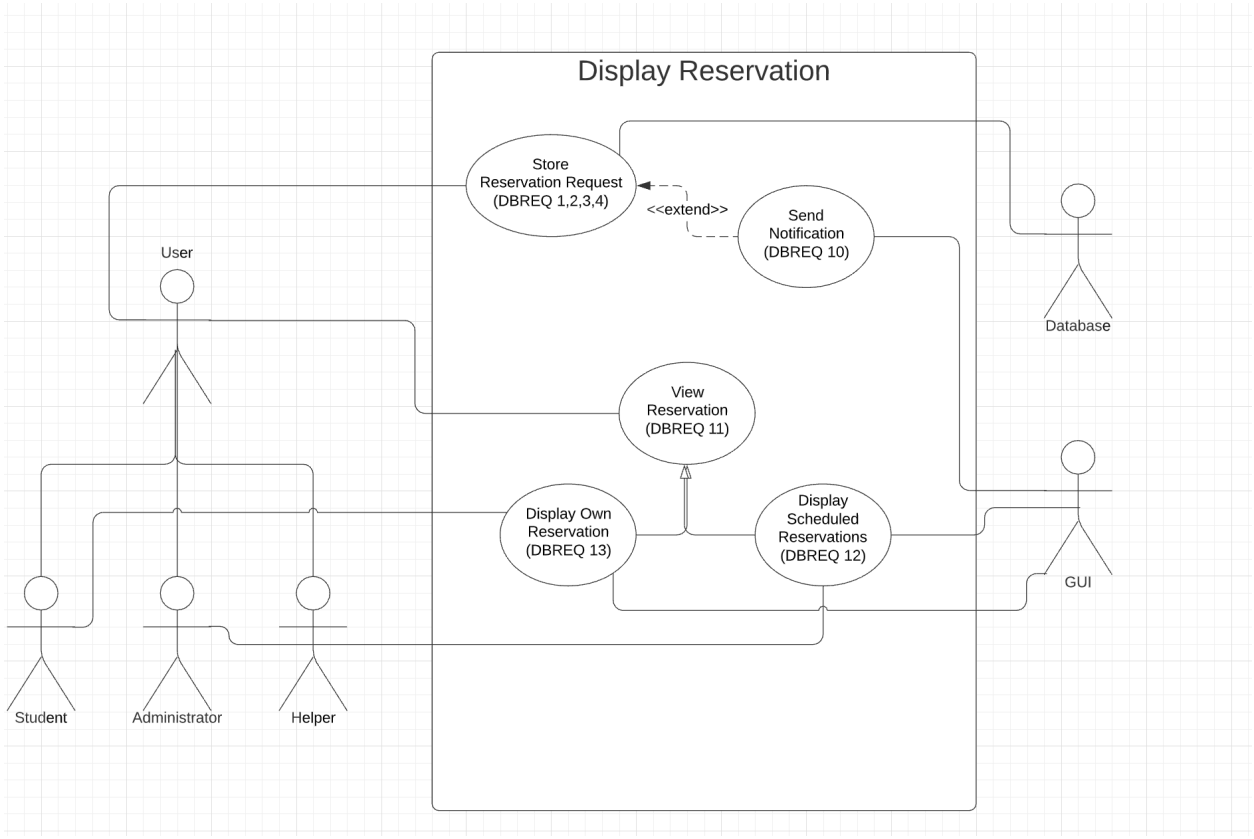
2.5.4.2 Store Reservation Request



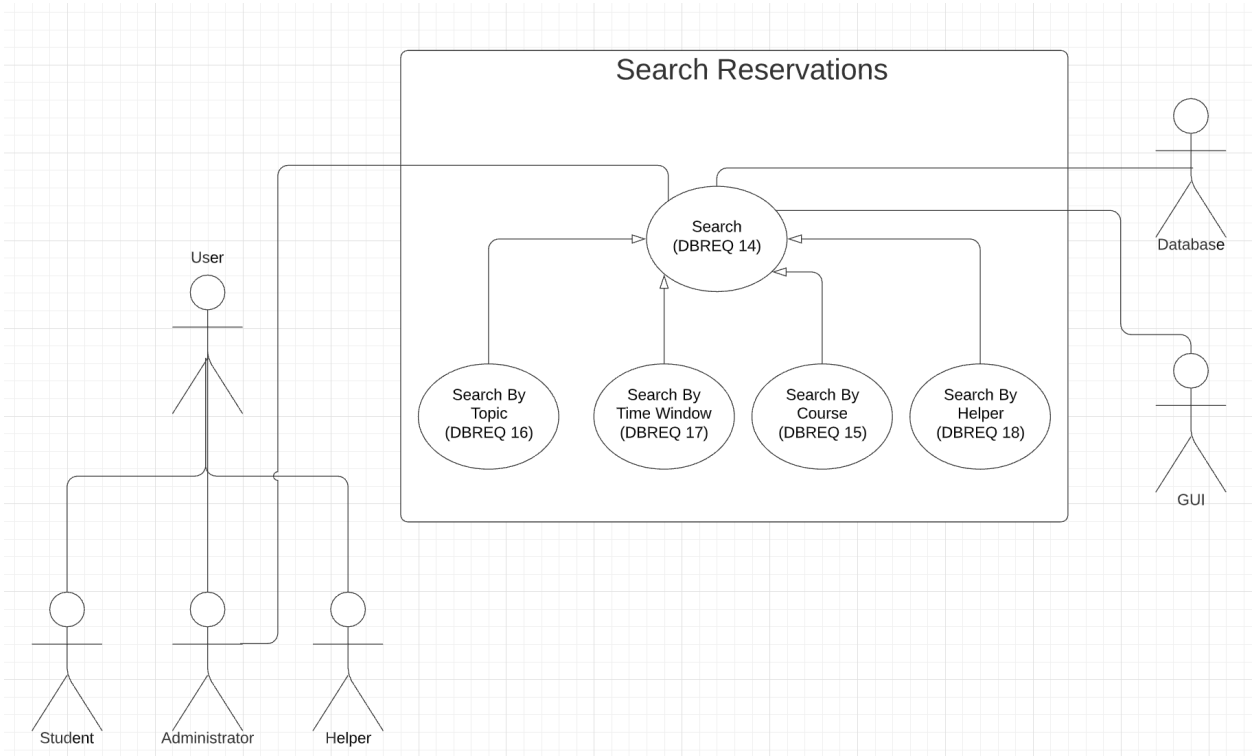
2.5.4.3 Store Help Session Data



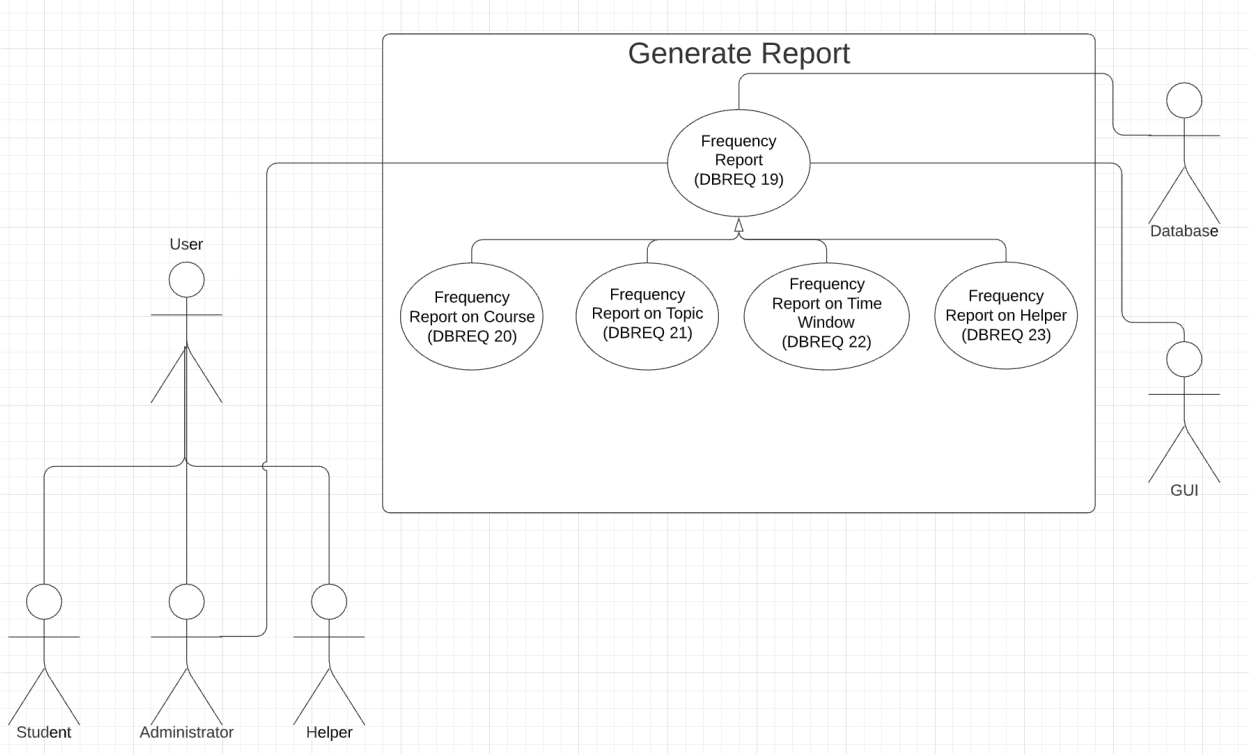
2.5.4.4 Display Reservation



2.5.4.5 Search Reservations



2.5.4.6 Generate Report



2.6 Stretch Goals

The functional requirements listed in this section are identified as ideas for potential development if time permits, but are not necessarily required for the client's specifications. If the previously listed requirements are met before the final deadline, this section will be filled out and development will begin on these requirements. These functionalities are priority 1.

2.6.1 News / Homepage

2.6.1.1 Description and Priority

TBD

2.6.1.2 Stimulus/Response Sequence

USE CASE

TBD

2.6.1.3 Functional Requirements

NHREQ-1: The system shall allow the user to view recent news.
NHREQ-2: The system shall display a list of recent computer science updates.
NHREQ-3: The system shall allow helpers to make new posts to the news and updates page.

NHREQ-4: The system shall filter recent news and updates by class.

2.6.2 Wait Queue

2.6.2.1 Description and Priority

TBD

2.6.2.2 Stimulus/Response Sequence

USE CASE

TBD

2.6.2.3 Functional Requirements

WQREQ-1: The system shall allow the user to view a wait queue of students waiting to receive help.
WQREQ-2: The system shall display a number of students currently receiving help.
WQREQ-3: The system shall display the current average wait time of being helped.

2.6.3 Forum

2.6.3.1 Description and Priority

TBD

2.6.3.2 Stimulus/Response Sequence

TBD

2.6.3.3 Functional Requirements

FREQ-1: The system shall display a forum of student asked questions.
FREQ-2: The system shall allow students to post questions to the forum page.
FREQ-3: The system shall allow students to post replies to forum questions
FREQ-4: The system shall allow Helpers to post replies to forum questions
FREQ-5: The system shall allow students to filter the forum by class tag

3. Non-Functional Requirements

Included in this section are the non-functional requirements intended for the Boardman Computer Science Lab Web Portal including NFRs specific to product requirements, organizational requirements, and external requirements. Each NFR is organized by use of a unique identifier, priority scale (1 = lowest, 5 = highest), a brief description of the NFR, and the thest(s) associated with the NFR.

3.1. Non-Functional Requirements Expanded

NFR1: The system shall not be down for more than 5 mins at a time between the hours of 6am and 10pm EST 96% of the time.
Priority: 4
Description: Major application maintenance is only expected to occur between the hours of 10pm and 6am EST, so server downtime is required to be less than 5 minutes outside this window.
Test(s): Test every 5 minutes for server availability for 24 hours, if more than one consecutive ping fails, test fails.

NFR2: The system shall be able to handle at least 500 requests per second 95% of the time.
Priority: 5
Description: At any given point in time, the server will not experience overload for up to 500 requests in a second.
Test(s): Make 500 requests in a second, if the system is able to handle these requests without crashing, test success.

NFR3: The system shall be able to process a request within 10 seconds 99% of the time.
Priority: 4
Description: At any given point in time, the system will receive and execute user requests within 10 seconds.
Test(s): Time the process time in milliseconds, a value over 10000 is failure.

NFR4: The system shall be able to load pages within 2 seconds when ping < 1000ms 96% of the time.
Priority: 3
Description: At any given point in time, new pages will load within 2 seconds upon user request if their ping is less than 1000ms.
Test(s): Test on a connection with a ping of less than 1000ms and time the page load in milliseconds. A value of over 2000 fails the test.

NFR5: The system shall allow no more than three errors per second 99% of the time.
Priority: 3
Description: At any given point in time, the system will experience a maximum of 3 errors in a second.
Test(s): Test all functions repeatedly for an hour. If the number of errors in the hour is greater than 10,000, test fails.

NFR6: The system shall not share account information with other users.
Priority: 5
Description: Login information and user data will not be shared with users other than the associated user, likewise, user data will not be used for any other purpose than internal analysis.
Test(s): Attempt to access another user's account without the correct password, attempt to access another user's reservations without the correct password.

NFR7: The system shall account for a list of popular resolutions (1920x1080, 1366x768, 1280x720, 360x640, 414x896, 1536x864, 375x667).
Priority: 2
Description: The website will fit accurately within most common screen size resolutions.
Test(s): Attempt to display the website on each of the listed resolutions. If there are graphical distortions or errors, the test fails.

NFR8: The system shall use 16 px font size for the body text.
Priority: 1
Description: All body text in the website will use size 16 font for purposes of uniformity.
Test(s): Search for font size declarations in document, each body text declaration should point to 16pt font.

NFR9: The system shall be able to store reservations for any point from 1970-2099.
Priority: 3
Description: When a student or helper reserves a meeting time, dates and time will be available for selection from years 1970 - 2099 so that updating will likely be unnecessary in perpetuity.
Test(s): Make a reservation for each year from 1970-2099. If any years are missing from the reservation table, the test fails.

NFR10: The system shall be compatible with mobile and web based devices.
Priority: 2
Description: The website will be usable for both desktop (latest ubuntu, Windows 10, and MACOS) and mobile devices (latest android, latest IOS).
Test(s): Test the system on each of the listed devices. If there are any graphical distortions or errors, the test fails.

NFR11: The system shall be able to run on Google Chrome, Firefox, Microsoft Edge, and safari.
Priority: 2
Description: The website will be usable for all common web browsers.
Test(s): Test the system on latest Chrome, Firefox, Edge, and Safari browsers. If there are any graphical distortions or errors, the test fails.

NFR12: The system shall be scalable, to accommodate for growth in the number of users (up to 1 million users).
Priority: 5
Description: The website will be able to accommodate any number of new users.
Test(s): Attempt to add 1 million users to the database system. If there are 1 million users in the database, the test succeeds.

NFR13: The system shall be able to file a new meeting time in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request for a new meeting is received, the request will be processed within 3 minutes.

Test(s): Attempt to file a new meeting, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR14: The system shall be able to file a change in a meeting in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request for a meeting change is received, the request will be processed within 3 minutes.
Test(s): Attempt to file a change to a meeting, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR15: The system shall be able to update the calendar in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request to update the calendar is received, the request will be processed within 3 minutes.
Test(s): Attempt to update the calendar, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR16: The system shall be able to update a forum request in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request to post to the forum is received, the request will be processed within 3 minutes.
Test(s): Attempt to update a forum request, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR17: The system shall be able to upload a feedback request in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request to upload a feedback form is received, the request will be processed within 3 minutes.
Test(s): Attempt to upload a feedback form, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR18: The system shall be able to update the walkin-wait times in less than 3 minutes 98% of the time.
Priority: 4
Description: When a request to update the wait times for help is received, the request will be processed within 3 minutes.
Test(s): Attempt to update walkin wait times, time the response. If the response occurs in less than 3 minutes, the test succeeds.

NFR19: The system shall be able to report wait time data within 3 minutes of a new entry 98% of the time.
Priority: 4
Description: When there is a change in wait time, that data will be saved for analysis within 3 minutes
Test(s): Attempt to update walkin wait times, time the response for the walkin wait update. If the response occurs in less than 3 minutes, the test succeeds.

NFR20: The system shall be able to report help request data within 3 minutes of a new entry 98% of the time.
Priority: 4
Description: When there is a request for help, the subject matter of request and time of request will be saved for analysis within 3 minutes
Test(s):
Test the time required that it takes for the request to be saved after it has been submitted. If less than 3 minutes, then the test succeeded.

4. User Interface

This section describes and explains the user interface intended for this project, or a reference to exterior documentation that better describes the user interface.

4.1. User Interface Expanded

See “User Interface Design Document” for the Boardman Computer Science Lab Web Portal.

5. Deliverables

This section includes a list of all deliverable items for this project. Additionally, it includes the tentative date of completion and the format that it will be submitted in.

5.1. Deliverables Expanded

Hard copies of each of the following:

- Systems Requirement Specification
  Date Expected: 10/25
- System Design Document
  Date Expected: 11/10
- User Interface Design Document
  Date Expected: 11/29
- User Manual
  Date Expected: Second Semester
- Administrator Manual
  Date Expected: Second Semester
- Copies of all Biweekly Status Reports
  Date Expected: Second Semester

An electronic file containing the following:

- Systems Requirement Specification
  Date Expected: 10/25
- System Design Document
  Date Expected: 11/10
- User Interface Design Document

Date Expected: 11/29

- User Manual
  Date Expected: Second Semester
- Administrator Manual
  Date Expected: Second Semester
- All source code
  Date Expected: Second Semester
- The executable program
  Date Expected: Second Semester
- Any other software required for installation and execution of the delivered program
  Date Expected: Second Semester

6. Open Issues

This section displays Issues that have been raised and do not yet have a conclusion. These issues will be addressed later in the development process.

6.1. Open Issues Expanded

TBD - note: Open Issues will be added as they are created

Appendix A – Agreement Between Customer and Contractor

This section denotes that both the client and the development team have agreed upon the information contained within this document. It will be used as both a guideline and as an end goal in terms of the requirements needed for the application to function to the clients vision.
In the case that an addition or edit be needed after the completion and signing of this document, the change or addition must be agreed upon by both client and development team and included in **Appendix D - Document Additions** with the title of the addition, date, brief description, and signature from both parties.

-Client-
Name: Mr. Christopher Dufour
Date:
Signature:

-Development Team-
Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Client Comments (Continues on next page if needed):

Client Comments Cont.

Appendix B – Team Review Sign-off

This section denotes that all members of the In-House Operations development team have reviewed this document and agree on its content and format. If any minor disagreements in content and format are present, they are listed below the development team signatures.

Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Minor Disagreements in Content and Format (if any):

Appendix C – Document Contributions

This section denotes the contributions of each team member to this document. It includes the sections each member worked on and their percentage contributed in parentheses.

Name: Klei Bendo
Sections worked on (percentage contributed):
Section 2 - Functional Requirements (15%) (Database section, use-cases and diagram)
Section 3 - Non-Functional Requirements (20%) (requirement and test writing)

Name: Jack Brisson
Sections worked on (percentage contributed):
Section 2 - Functional Requirements (15%) (Sign-In/Sign-Out section, use-cases and diagram)
Section 3 - Non-Functional Requirements (20%) (requirement and test writing)

Name: Alex Landry
Sections worked on (percentage contributed):
Section 2 - Functional Requirements (15%) (Help Scheduling section, use-cases and diagram)
Section 3 - Non-Functional Requirements (20%) (requirement and test writing)

Name: Samuel Morse
Sections worked on (percentage contributed):
Section 2 - Functional Requirements (35%) (calendar and feedback sections, use-cases and diagram, and most of the Stimulus/Response diagrams)
Section 3 - Non-Functional Requirements (5%) (requirement and test writing)

Name: Aaron Schanck
Sections worked on (percentage contributed):
Section 1 - Introduction (100%)
Section 2 - Functional Requirements (5%) (Introduction and layout)
Section 3 - Non-Functional Requirements (5%) (Introduction and layout)
Section 4 - User Interface (100%)
Section 5 - Deliverables (100%)
Section 6 - Open Issues (100%)
Appendices (100%)

Name: Forrest Swift
Sections worked on (percentage contributed):
Section 2 - Functional Requirements (15%) (List of Functional Requirements, Database section use-cases)
Section 3 - Non-Functional Requirements (20%) (requirement and test writing)

Appendix D – Document Additions

No Document additions to date.

System Design Document

Boardman Computer Science Lab Web Portal

Client: Mr. Christopher Dufour

Development Company: In-House Operations

Development Team:

Klei Bendo
Jack Brisson
Alex Landry
Samuel Morse
Aaron Schanck
Forrest Swift

Date: 10/28/2021

Version 1.0

Boardman Computer Science Lab Web Portal
System Design Document

Table of Contents

1. Introduction

The Boardman Computer Science Lab Web Portal is an all encompassing tool for computer science students at the University of Maine. It is designed to ensure better help for students seeking aid in both specific inquiries, and broad subject areas at the Boardman Computer Science Lab through use of an interactive calendar, individual and group meeting scheduling, forum posting, and news updates. The Web Portal will make the Boardman Computer Science Lab more accessible and easy to use for University of Maine Computer Science Students.

1.1 Purpose of This Document

The purpose of this document is to outline and describe the design of the Boardman Computer Science Lab Web Portal in terms of its system architecture and database. It details how each class is connected and the relationships between them. It is intended primarily for the client of the project and the development team to keep true to requirements and the project's intended design moving forward and for posterity of reference. This document is also intended for any other interested parties for official documentation of the application. It includes diagrams and descriptions of the system architecture with a decomposition description, diagrams and descriptions of the database, and a matrix of requirements as they relate to the different class diagrams.

1.2 References

Bendo, Klei, et al. "In-House Operations SRS." *Google Docs*, Google, 18 Oct. 2021, https://docs.google.com/document/d/1YIFScQdYOcsTWcKpfTEac3g4aTRo3XtSTO1Uay2CQv4/edit?usp=sharing.

Bendo, Klei, et al. "Ui Design Ideas." *Google Docs*, Google, 4 Nov. 2021, https://docs.google.com/document/d/1UTH4vEWQyTghzSD2DuvfVMSrA-7srWsTkONHvFu4Suo/edit?usp=sharing.

Dufour, Christopher, and Penny Rheingans. "dufour_help-Resource-Scheduling." 16 Sept. 2021.

Schanck, Aaron. "Team 17 Capstone Proposal." *Google Docs*, Google, 4 Oct. 2021, https://docs.google.com/document/d/19nm8LNdbCEEdSQNdVRj570LcsRJC7rjRumRwU4srtBE/edit?usp=sharing.

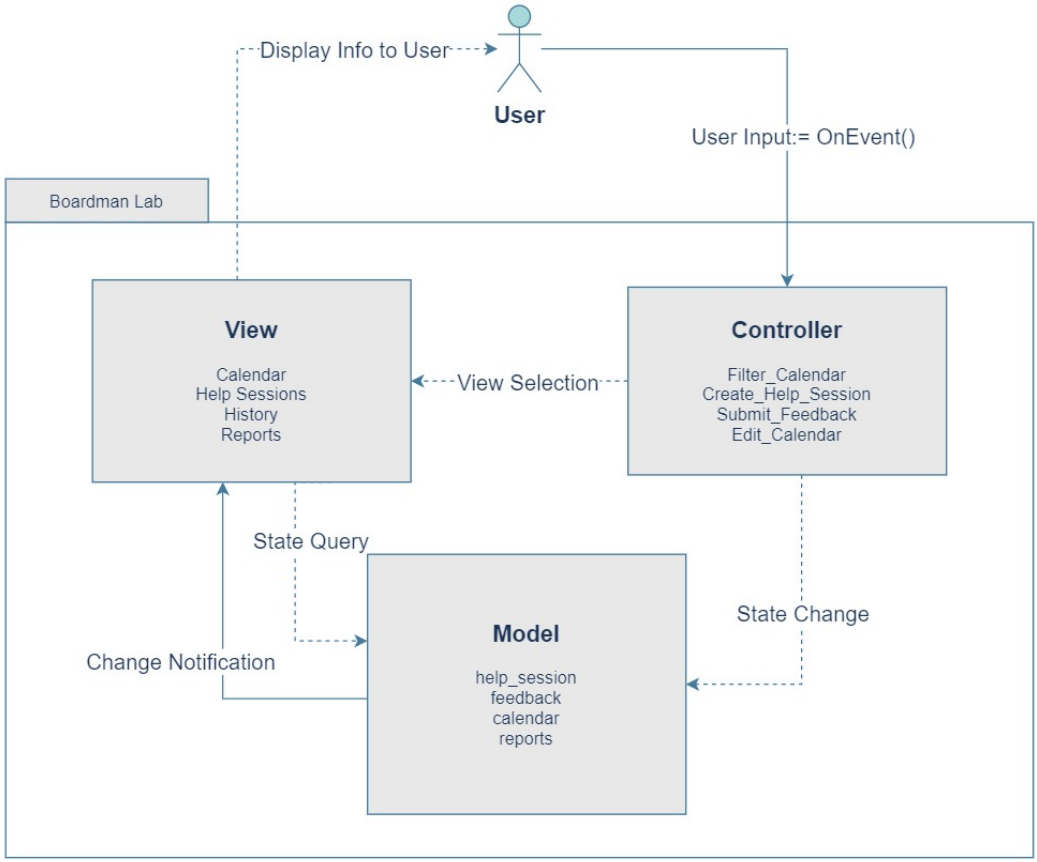
2. System Architecture

Included in this section are the class diagrams associated with the Boardman Computer Science Lab Web Portal including diagrams for Sign in/Sign Out, Help Scheduling, Feedback, and calendar. Additionally, this section includes diagrams depicting the decomposition of the components of the system. Overall it describes and overviews the front end design of the system architecture. For a complete description of which class diagrams associate with which requirements, see Section 4.

2.1 Architectural Design

For the system architecture, we are going to be using the model view controller design pattern. The backend will be using the Django framework with an SQL database (at this time we are between either mySQL or PostGreSQL) and the frontend will use a NodeJS framework. Django will handle the queries to the database (controller and models) and NodeJS will handle display and user interaction (view). We are going to be implementing a single sign-in using Google Oauth and Umaine login information to streamline user interactions and allow us to not store sensitive user information on our database. This means that we will not be storing identifiable user data - only an encrypted email address that will keep track of help sessions, attendance metrics, calendars and feedback.

Figure 1: Logical Architecture (Model, View, Controller Design Pattern)*
Showing a generalized architecture for the Boardman Lab Web Application.



*Please note that classes contained in 'views' and 'models' and methods within 'controller' do not represent the full scope of the application and are examples to assist in the understanding of the design pattern.

The Model View Controller design pattern is very useful for web applications because it allows for a highly dynamic user interface. Since we will have different user types, it is important that we manage permissions across user types. Typically, user permissions will be dictated by the authority prescribed via the administrator control panel, but depending on the information we can get from Google Oauth,

permissions also might be handled directly by the umaine account system (for example, instructors and student aids have specific designations within their account).

Models are essentially the classes that are described below and shown in their respective class diagrams. The relationships between these classes will be represented in the database and their methods will be written in Django. The various views of the application are going to be coded in a combination of HTML, CSS and javascript while utilizing the Node.JS framework that will allow us to

2.2 Decomposition Description

2.2.1 Sign In/Out

Description (Sign In): When a user attempts to login to their profile, the system shall submit the imputed credentials to OAUTH for validation. If the attempt is successful, a token is generated containing the user's Maine.edu email name and the user is permitted to access the Boardman Computer Science Lab Web Portal.

Description (Sign Out): When a user clicks a UI element to sign out of their profile or if they have been inactive for a period of time, they will be brought to the login screen, and the token containing the user's Maine.edu email address will be removed.

Figure 2: Sign In Class Diagram

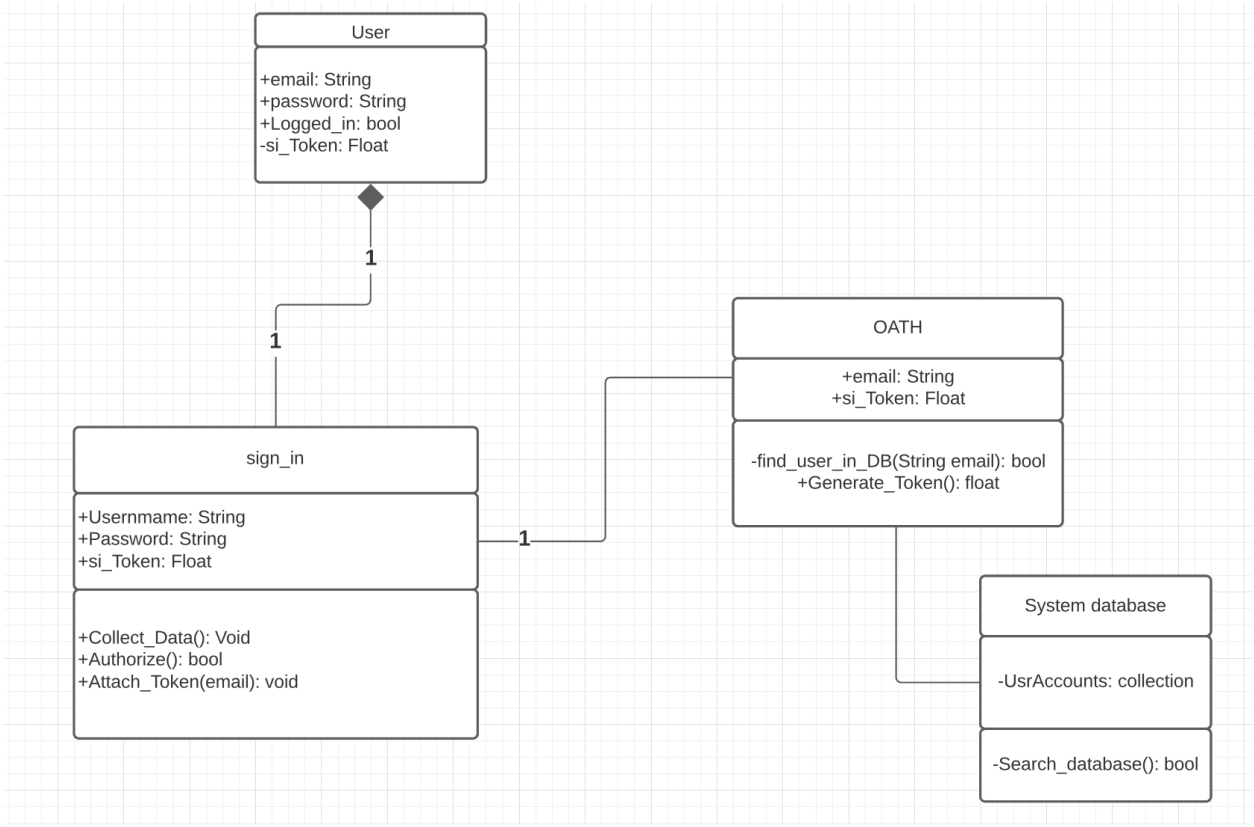
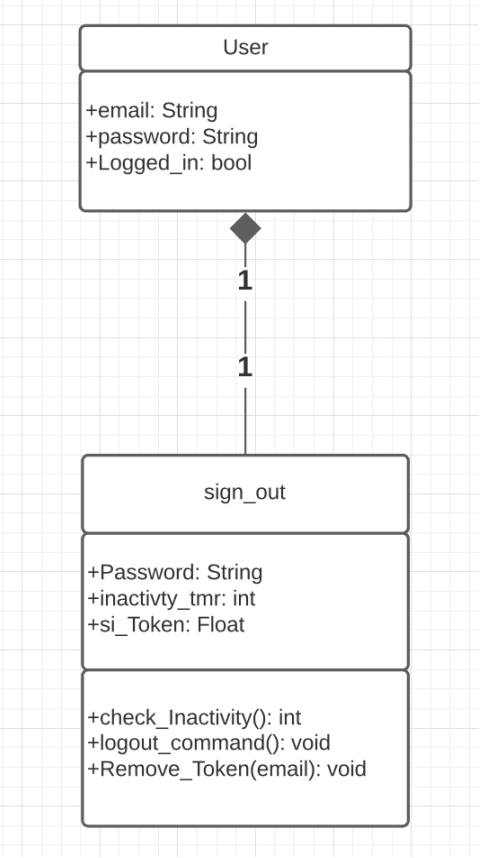


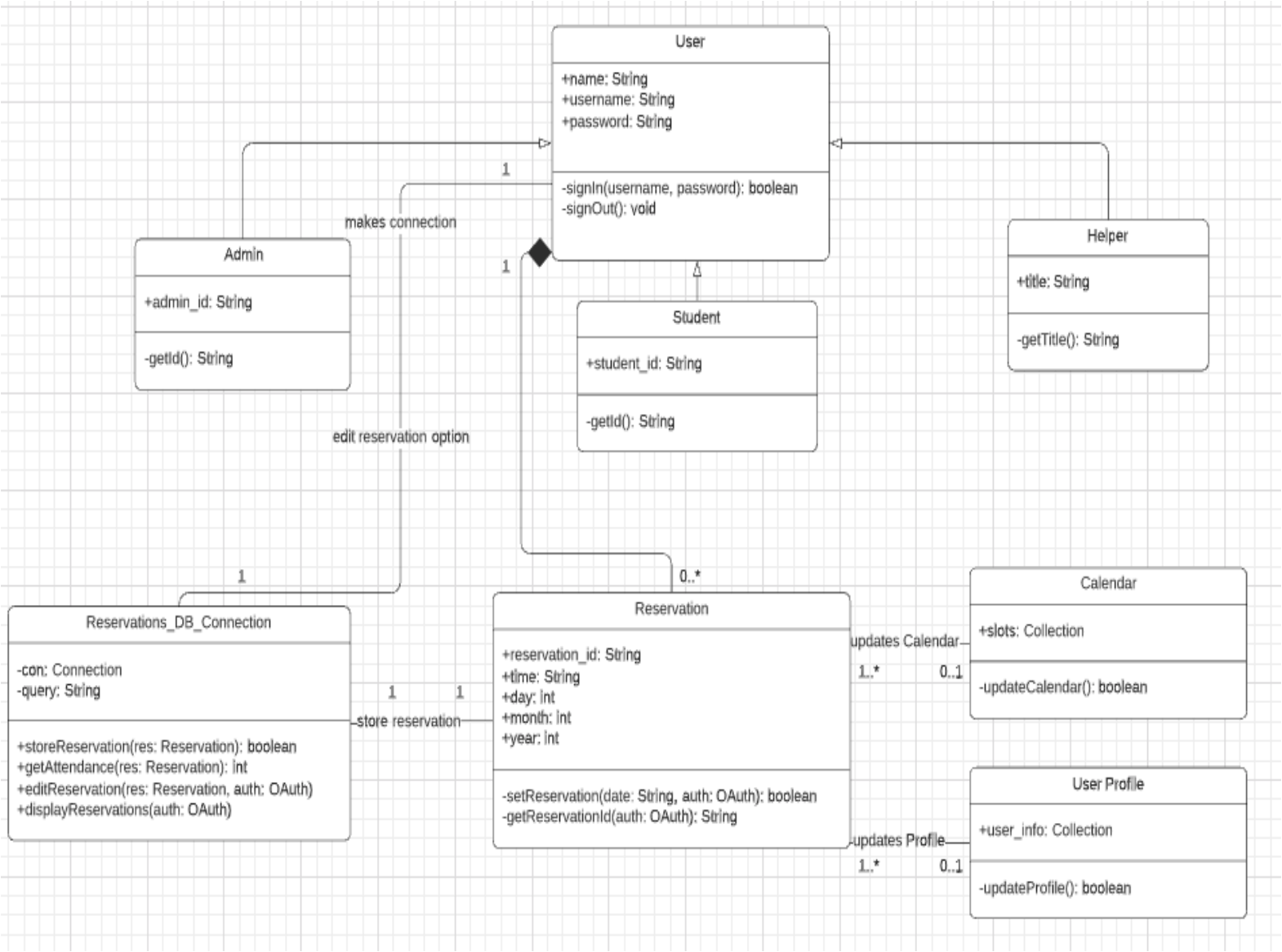
Figure 3: Sign Out Class Diagram



2.2.2 Store Reservation Request/ Edit Reservation

Description: When making a new reservation, or changing an existing one, the system shall be able to update the calendar and the user profile. It shall also create a new entry in the Reservations table of the Database. The system shall be able to display their own reservations to the student user, display the reservations they will help in to the helper user, and display all reservations to the administrator user. An administrator or a student shall be able to edit a reservation.

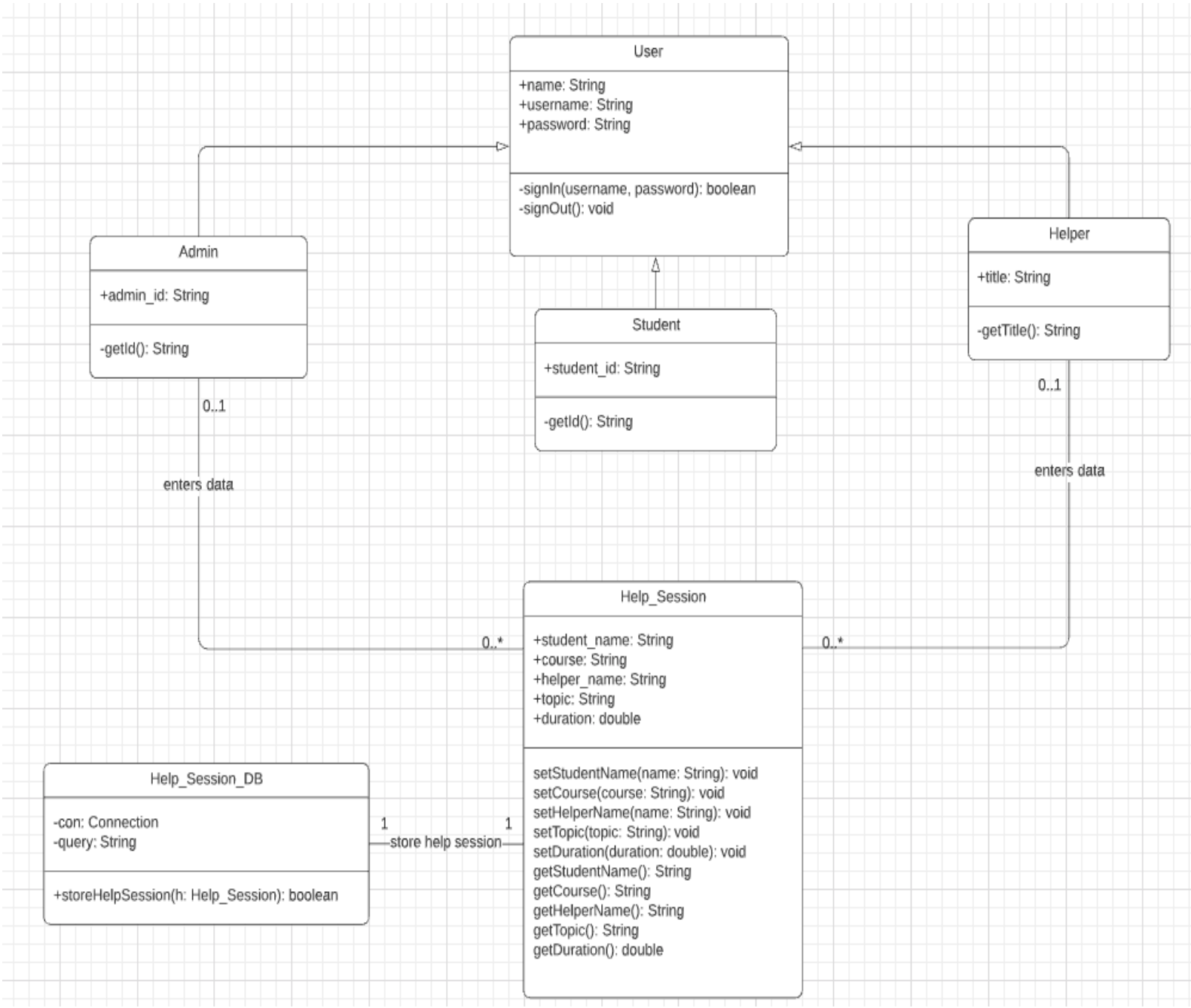
Figure 4: Store Reservation Request/ Edit Reservation Class Diagram



2.2.3 Store Help Session Data

Description: After a student provides feedback, an administrator or a helper shall be able to enter information about a particular help session. The information about the session will then be stored into the Database.

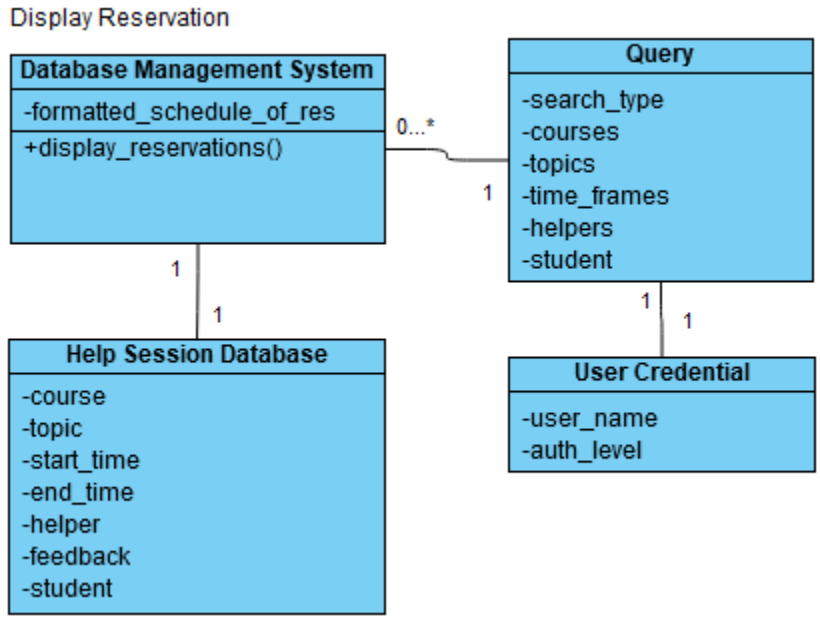
Figure 5: Store Help Session Data Class Diagram



2.2.4 Display Reservation

Description: Provides the data required to display reservation information according to a user's authorization level and query criteria.

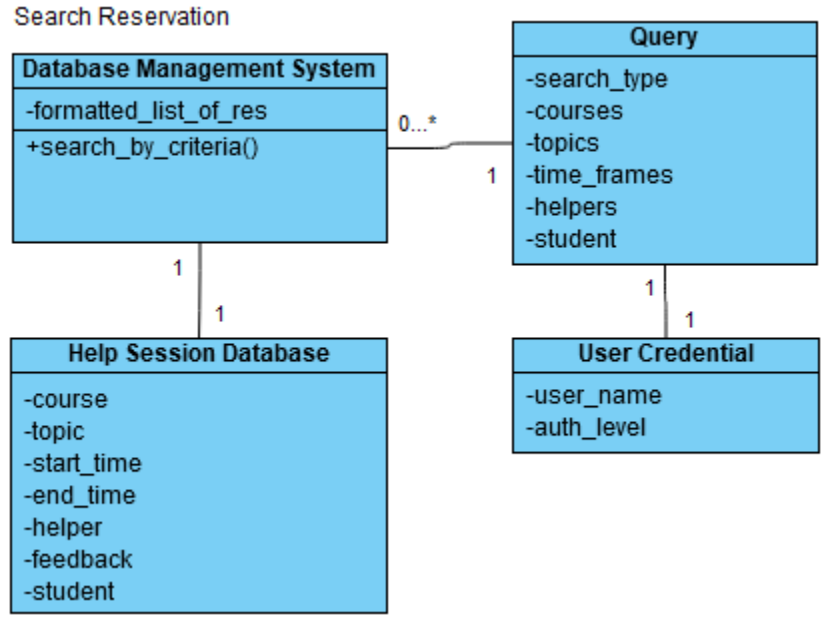
Figure 6: Display Reservation Class Diagram



2.2.5 Search Reservation

Description: Provides the data required to display all reservations that meet a specified criteria. This functionality is accessible only by administrators.

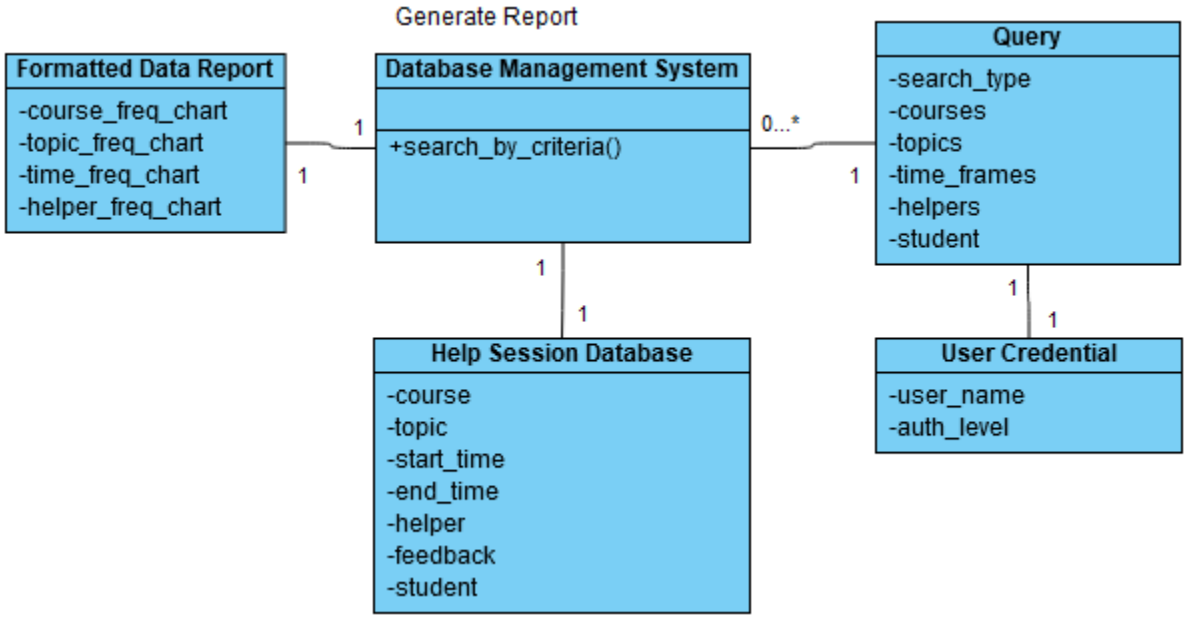
Figure 7: Search Reservation Class Diagram



2.2.6 Generate Report

Description: Generates several frequency charts to show distribution of student help sessions by course, topic, time, and helper. Generated reports sample from reservations that meet a specified criteria. This functionality is accessible only by administrators.

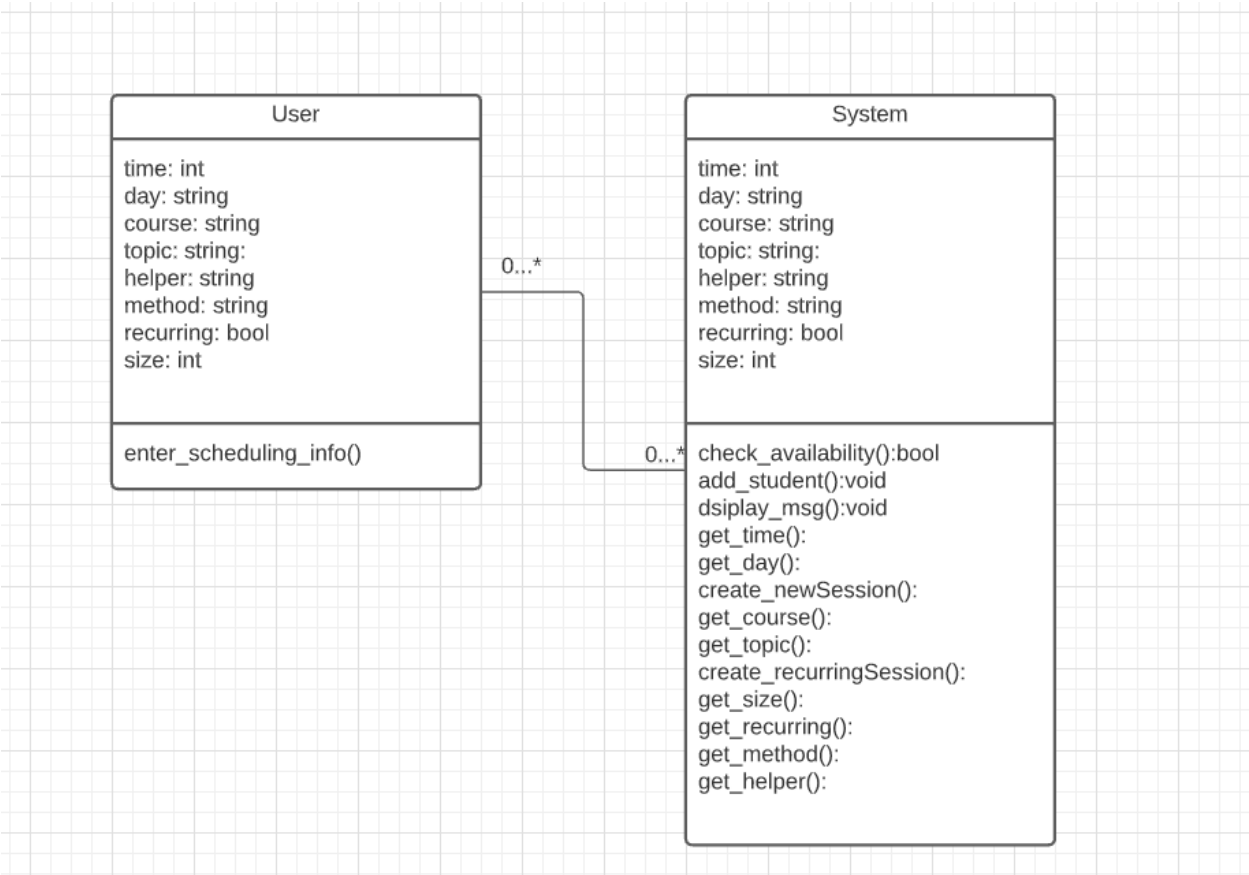
Figure 8: Generate Report Class Diagrams



2.2.7 Schedule Help Request

Description: This class diagram allows the user to create a help session request. The user enters time, day, method, size, course, topic, help, and whether it is supposed to be a recurring meeting. The system then checks availability and either places the student into the time slot or notifies the user that their time slot is not available.

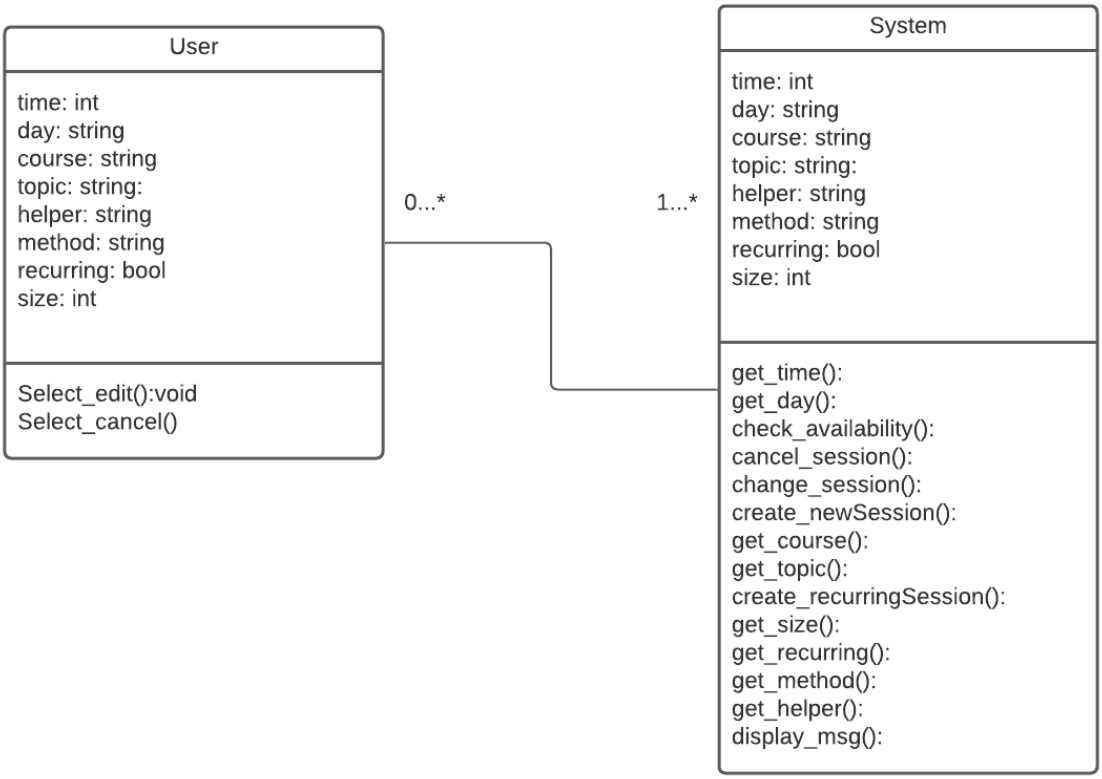
Figure 9: Schedule Help Request Class Diagrams



2.2.8 Edit Help Request

Description: This class diagram illustrates how a user would go about editing a help request. The User will be able to request changes to their help request by changing the time, day, method, size, course, topic, help, and whether it is supposed to be a recurring meeting. This Diagram shows the basic front end of this system.

Figure 10: Edit Help Request Class Diagrams



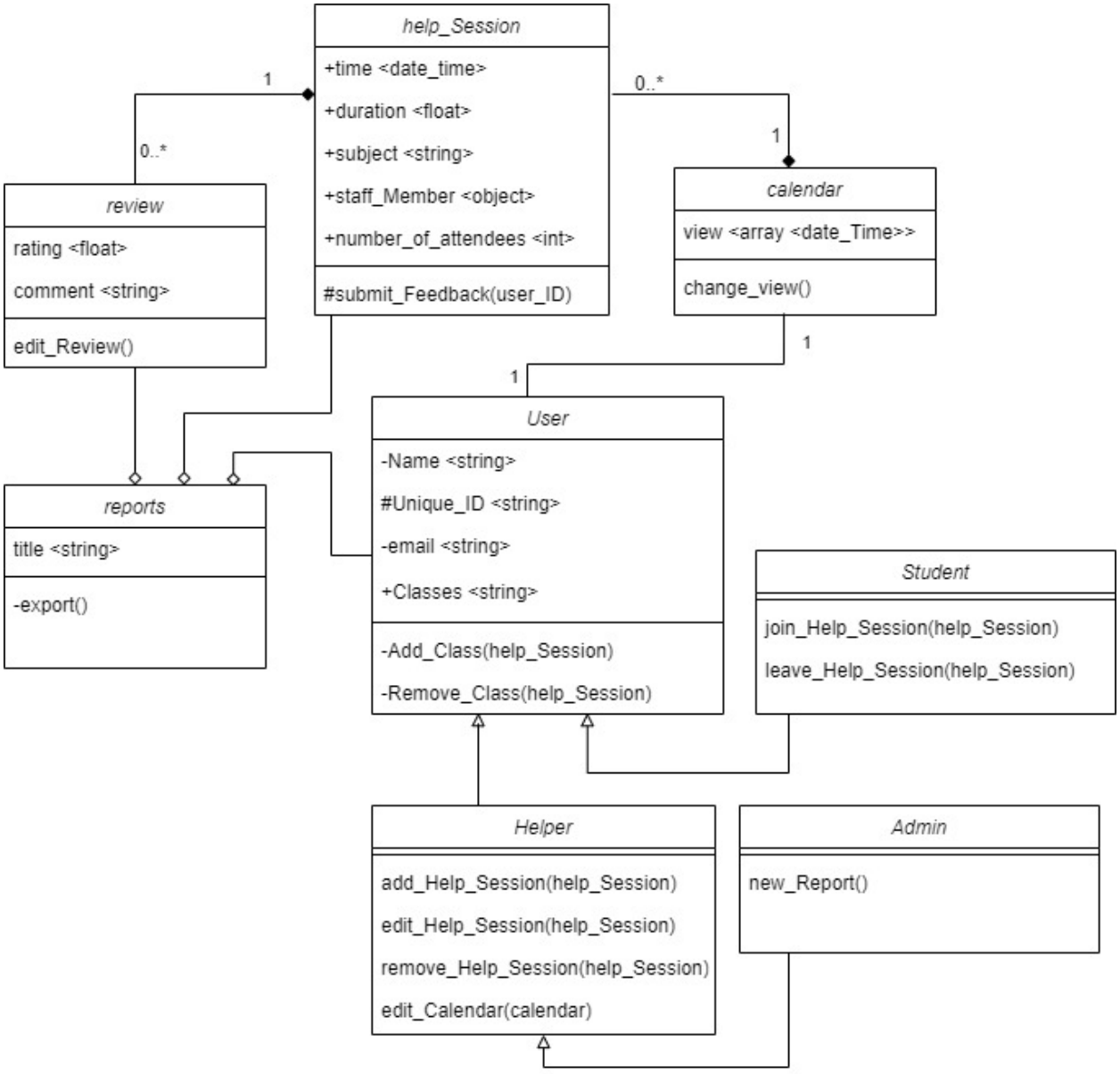
2.2.9 Submit Feedback/ User Hierarchy / Calendar View

Description: The purpose of this diagram is to expand on the front end features of the application by showing the interactions and relationships between the class models. The diagram shows the calendar, help sessions feedback (reviews) and reports as well as the inheritance hierarchy of 'user'. It also shows how reporting is an aggregation of many different classes including feedback, users and help sessions.

A help session is a composition of reviews - and reviews/feedback cannot exist without a corresponding help session. This is also the case for the relationship between the calendar and help sessions. The calendar holds all of the help sessions that are active in the system and organizes them within the calendar view (see Architectural Design Section: 2.1).

Admin inherits from Helper and both Helper and Student inherits from User. This separation is important for the permission hierarchy of the application. Admins will have the ability to generate reports and gather data from the help sessions while helpers will be able to alter their own schedules in the lab and manage their individual help sessions that they own. Students will be able to sign up for help sessions, cancel a reservation for help and submit feedback on help sessions that they have attended.

Figure 11 Submit Feedback / User Hierarchy / Calendar View Class Diagrams



3. Persistent Data Design

Included in this section are both descriptions of the proposed database for the Boardman Computer Science Lab Web Portal and class diagrams depicting the relations between each larger area of the database as a whole. If there were files that were important to include for this project they would also be included in this section. Moreover this section provides a description of the back end processes of the application.

3.1 Database Descriptions

- Our database will consist of a single table of reservations.
- Users with student authorization will only see reservations they have made.

- Users with helper authorization will be able to see any reservation they are a helper in or any reservations they have made.
- Users with administrator authorization will be able to see all reservations.
- Start and End times are in UTC with the format YYYY-MM-DD HH:MM:SS
- Student and Helper identifiers are in the format firstname.lastname

Figure 12: Database Table Fields and Data Types

ID	Course	Topic	Start Time	End Time	Student	Helper	Feedback
integer	text	text	text	text	text	text	text

3.2 File Descriptions

No files are used in the production or use of this application.

4. Requirements Matrix

This section outlines which system components satisfy each of the functional requirements from the SRS. Each relationship is outlined using a tabular format including the associated system component, the diagram it's located in, the requirement ID, and Requirement name.

4.1 Requirements Matrix Expanded

Figure 13: Requirements Matrix

System Componant (function/Method)	Diagram	Requirement ID	Requirement Name
sign_in	Sign In/Out	SIREQ-1	The system shall allow the user to enter the site with UMaine Login credentials
sign_out	Sign In/Out	SIREQ-2	The system shall allow the user to log out of the users account
OAUTH	Sign In/Out	SIREQ-3:	The system shall utilize university of maine sign-in accounts for login.
OAUTH	Sign In/Out	SIREQ-4:	The system shall utilize OAuth to manage logins.
sign_out	Sign In/Out	SIREQ-5:	The system shall allow the user to logout of their account.
sign_out	Sign In/Out	SIREQ-6:	The system shall lock access to an account when not in use for a set amount of time.
Student	Submit Feedback/ User Hierarchy / Calendar View	SIREQ-7:	The system shall have a user type for student.
Helper	Submit Feedback/ User Hierarchy / Calendar View	SIREQ-8:	The system shall have a user type for helper.
Admin	Submit Feedback/ User Hierarchy / Calendar View	SIREQ-9:	The system shall have a user type for administrator.
System	Schedule Help Request	HSREQ-1	The system shall allow users to Schedule help Requests
System	Edit Help Request	HSREQ-2	The system shall allow the user to Alter a scheduled help time
System	Schedule Help Request	HSREQ-3	The system shall allow requests for help at a specific date and time.
System	Schedule Help Request	HSREQ-4:	The system shall allow requests to schedule recurring meetings.
System	Schedule Help Request	HSREQ-5:	The system shall allow requests to search for available times matching a criteria.
System	Schedule Help Request	HSREQ-6:	The system shall allow requests to meet with a specific helper.
System	Schedule Help Request	HSREQ-7:	The system shall allow requests for help with a specific topic or course.
System	Edit Help Request	HSREQ-8:	The system shall allow help reservations to have various modifications.
System	Schedule Help Request	HSREQ-9:	The system shall allow for remote or in person reservations.
System	Schedule Help Request	HSREQ-10:	The system shall allow for solo or group help.
System	Schedule Help Request	HSREQ-11:	The system shall allow a limited number of students to join in a study group meeting
System	Edit Help Request	HSREQ-12:	The system shall allow users to cancel a reservation.
System	Edit Help Request	HSREQ-13:	The system shall allow users to move a reservation to a new valid time.
help_session	Submit Feedback/ User Hierarchy / Calendar View	FBREQ-1	The system shall allow Students to give feedback on the helper who led a help meeting.
help_session	Submit Feedback/ User Hierarchy / Calendar View	FBREQ-2:	The system will display history of help sessions attended by the student.
review	Submit Feedback/ User Hierarchy / Calendar View	FBREQ-3:	The system shall allow for anonymous reviews of help sessions.
review	Submit Feedback/ User Hierarchy / Calendar View	FBREQ-4:	The system shall allow for anonymous reviews of helpers.
calendar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-1	The system shall allow a user to view a calendar of schedules and events
calendar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-2:	The system shall display a schedule of helpers and events.
help_session	Submit Feedback/ User Hierarchy / Calendar View	CREQ-3:	The system shall display the names of lab helpers that are available
help_session	Submit Feedback/ User Hierarchy / Calendar View	CREQ-4:	The system shall display the times that lab helpers are available.
help_session	Submit Feedback/ User Hierarchy / Calendar View	CREQ-5:	The system shall display the classes that the lab helper can assist with.
calandar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-6:	The system shall display any scheduled group help sessions.
calandar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-7:	The system shall display any scheduled refresher lectures.
calendar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-8:	The system shall provide an option to view the calendar by week.
calandar	Submit Feedback/ User Hierarchy / Calendar View	CREQ-9:	The system shall provide an option to view the calendar by month.
Reservation_DB_Connection	Store Reservation Request, Edit Reservation	DBREQ-1:	The system shall store user reservation requests.
Reservation_DB_Connection	Store Reservation Request, Edit Reservation	DBREQ-2:	The system shall manage reservation requests
Reservation_DB_Connection	Store Reservation Request, Edit Reservation	DBREQ-3:	The system shall check reservation requests for conflict.
Reservation_DB_Connection	Store Reservation Request, Edit Reservation	DBREQ-4:	The system shall store valid reservations.
Reservation_DB_Connection	Store Reservation Request, Edit Reservation	DBREQ-5:	The system shall allow for edits to reservations already in the database. Store Help Session Data.
Help_Session_DB	Store Help Session Data	DBREQ-6:	The system shall store walk-ins data entered by administrators.
Help_Session_DB	Store Help Session Data	DBREQ-7:	The system shall store the course the student was helped with.
Help_Session_DB	Store Help Session Data	DBREQ-8:	The system shall store the topic the student was helped with.
Help_Session_DB	Store Help Session Data	DBREQ-9:	The system shall store the duration of each help session.
Database Management System	Display Reservation	DBREQ-10:	The system shall send notifications and viewing options associated with help reservations to applicable users.
Database Management System	Display Reservation	DBREQ-11:	The system shall display reservations.
Database Management System	Display Reservation	DBREQ-12:	The system shall display all reservations on a schedule to administrators.
Database Management System	Display Reservation	DBREQ-13:	The system shall display a student's own reservations to them.
Database Management System	Search Reservations	DBREQ-14:	The system shall allow administrators to search reservations with criteria.
Help Session Database	Search Reservations	DBREQ-15:	The system shall allow for searches by course.
Help Session Database	Search Reservations	DBREQ-16:	The system shall allow for searches by topic.
Help Session Database	Search Reservations	DBREQ-17:	The system shall allow for searches by window of time.
Help Session Database	Search Reservations	DBREQ-18:	The system shall allow for searches by helper.
Formatted Data Report	Generate Report	DBREQ-19:	The system shall generate reports based on data from a specified window.
Help Session Database	Generate Report	DBREQ-20:	The system shall generate frequency reports based on course.
Help Session Database	Generate Report	DBREQ-21:	The system shall generate frequency reports based on topic.
Help Session Database	Generate Report	DBREQ-23:	The system shall generate frequency reports based on helper.

Appendix A – Agreement Between Customer and Contractor

This section denotes that both the client and the development team have agreed upon the information contained within this document. It will be used as both a guideline and as an end goal in terms of the requirements needed for the application to function to the clients vision.
In the case that an addition or edit be needed after the completion and signing of this document, the change or addition must be agreed upon by both client and development team and included in **Appendix D - Document Additions** with the title of the addition, date, brief description, and signature from both parties.

-Client-
Name: Mr. Christopher Dufour
Date:
Signature:

-Development Team-
Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Client Comments (Continues on next page if needed):

Client Comments Cont.

Appendix B – Team Review Sign-off

This section denotes that all members of the In-House Operations development team have reviewed this document and agree on its content and format. If any minor disagreements in content and format are present, they are listed below the development team signatures.

Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Minor Disagreements in Content and Format (if any):

Appendix C – Document Contributions

This section denotes the contributions of each team member to this document. It includes the sections each member worked on and their percentage contributed in parentheses.

Name: Klei Bendo
Sections worked on (percentage contributed):
Section 2: 15% (Sections 2.2.2, 2.2.3)
Section 3: 10% (Edits and final lookover of the content of section 3)

Name: Jack Brisson
Sections worked on (percentage contributed):
Section 2: 15% (Section 2.2.1)

Name: Alex Landry
Sections worked on (percentage contributed):
Section 2: 15% (sections 2.2.7, 2.2.8)

Name: Samuel Morse
Sections worked on (percentage contributed):
Section 2: 35% (secitons 2.1, 2.2.9)
Section 3: 10% (additional formatting and edits)

Name: Aaron Schanck
Sections worked on (percentage contributed):
Section 1: 100%
Section 2: 5% (introduction)
Section 3: 5% (introduction)
Section 4: 100%
Appendices: 100%

Name: Forrest Swift
Sections worked on (percentage contributed):
Section 2: 15% (sections 2.2.4, 2.2.5, 2.2.6)
Section 3: 75% (description and table)

Appendix D – Document Additions

No Document additions to date.

User Interface Design Document

Boardman Computer Science Lab Web Portal

Client: Mr. Christopher Dufour

Development Company: In-House Operations

Development Team:
Klei Bendo
Jack Brisson
Alex Landry
Samuel Morse
Aaron Schanck
Forrest Swift

Date: 11/20/2021
Version 1.0

Boardman Computer Science Lab Web Portal
User Interface Design Document

Table of Contents

1. Introduction

The Boardman Computer Science Lab Web Portal is an all encompassing tool for computer science students at the University of Maine. It is designed to ensure better help for students seeking aid in both specific inquiries, and broad subject areas at the Boardman Computer Science Lab through use of an interactive calendar, individual and group meeting scheduling, forum posting, and news updates. The Web Portal will make the Boardman Computer Science Lab more accessible and easy to use for University of Maine Computer Science Students.

1.1 Purpose of This Document

The purpose of this document is to outline and describe the User Interface design of the Boardman Computer Science Lab Web. It is intended primarily for the client of the project and the development team to keep true to requirements and the project's intended design moving forward and for posterity of reference. This document is also intended for any other interested parties for official documentation of the application. This document details the formatting standards used by this application that ensure consistency throughout, a diagram (figure 1.1-1.3) that describes the transitions between UI elements, mockups of the application's pages, and a full description of all data items that can be entered into the system by the user.

1.2 References

Dufour, Christopher, and Penny Rheingans. "dufour_help-Resource-Scheduling." 16 Sept. 2021.

In-House Operations. "In-House Operations SRS." *Google Docs*, Google, 18 Oct. 2021, https://docs.google.com/document/d/1YIFScQdYOcsTWcKpfTEac3g4aTRo3XtSTO1Uay2CQv4/edit?usp=sharing.

In-House Operations. "Ui Design Ideas." *Google Docs*, Google, 4 Nov. 2021, https://docs.google.com/document/d/1UTH4vEWQyTghzSD2DuvfVMSrA-7srWsTkONHvFu4Suo/edit?usp=sharing.

In-House Operations. "In-House Operations SDD v. 1.0." *Google Docs*, Google, 28 Oct. 2021, https://docs.google.com/document/d/1qvNbDHLXdX5J8jHynGA8STqZW0qGLAZs1bayzoYAhoY/edit?usp=sharing.

Morse, Samuel "Figma Design" 21 Nov. 2021 https://www.figma.com/file/YvE53I7e9N8KaubB23RP2m/Main(e)-Site?node-id=0%3A1

Schanck, Aaron. "Team 17 Capstone Proposal." *Google Docs*, Google, 4 Oct. 2021, https://docs.google.com/document/d/19nm8LNdbCEEdSQNdVRj570LcsRJC7rjRumRwU4srtBE/edit?usp=sharing.

University of Maine. "Brand Standards" 21 Nov 2021 https://umaine.edu/marketingandcommunications/wp-content/uploads/sites/209/2020/06/Brand-Standards-FINAL-web.pdf

2. User Interface Standards

The user interface standards include an overview of the design standards used to maintain consistency in the user interface throughout the system. This incompasses layout/design, components, navigation components, navigation, and error handling. These standards can be further observed in Section 3 figures 2.1-2.22.

2.1 User Interface Standards Expanded

2.1.1 Layout and Design

The general layout of the application is simple, with a navigation bar on the left hand side of the display and content in the center (justified right). We worked on creating a layout that would be usable for mobile devices, with the option to hide the navigation bar for optimal viewing. Each page has tabs that are specific to user privilege level, and maintain a simple design standard.

A dark color palette was used to both reduce eye strain and save energy. Colors included are brought directly from the University of Maine 2018 Branding Standards to ensure continuity with the user's workflow.

There are no soft edges in the design (except for special buttons) to portray a utilitarian application - the design focuses on use while maintaining a clean organized appearance. We were influenced by the design of other applications that students commonly engaged with such as Slack and Discord.

University of Maine Design Standards:
https://umaine.edu/marketingandcommunications/wp-content/uploads/sites/209/2020/06/Brand-Standards-FINAL-web.pdf

2.1.2 Components

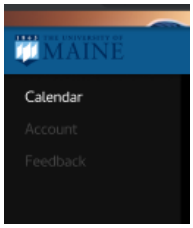
Buttons

Green = Creating and Saving
Red = Removing and Deleting
Light Blue = Editing and Altering
Dark Blue = Navigation Link

Interactive regions of the application are highlighted in brighter colors than other elements, though the navigation pane remains darker than the content itself. Buttons are color coded based on their purpose

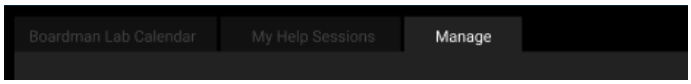
2.1.3 Navigation

Left Side Navigation Pane



The left side navigation pane is reserved for larger scope navigation such as account view, calendar view and feedback. This navigation should be a hideable entity for the web version of the application.

Tabs in Content

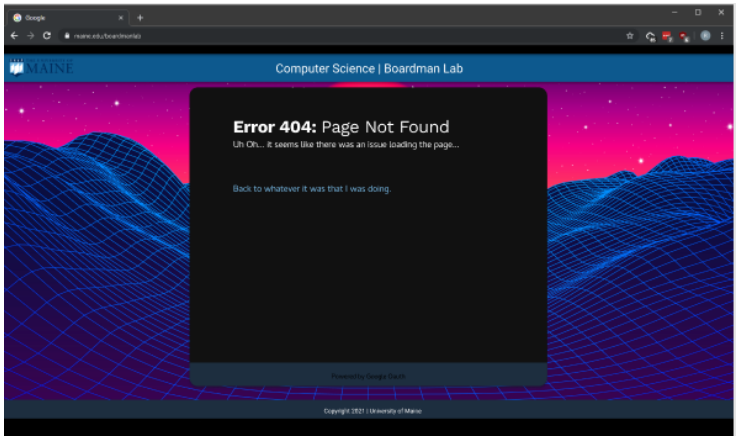


Each content page may have multiple navigation tabs to enable other features of the application to be more easily accessible to the user. Currently selected tabs are highlighted while the other tabs are darkened.

Button Links

There are a series of buttons that will direct the user to other content that is contained within the current section, such as the

2.1.4 Error Handling



In the event of an error, an appropriate error message will appear and direct the user back to the last correctly loaded page.

3. User Interface Walkthrough

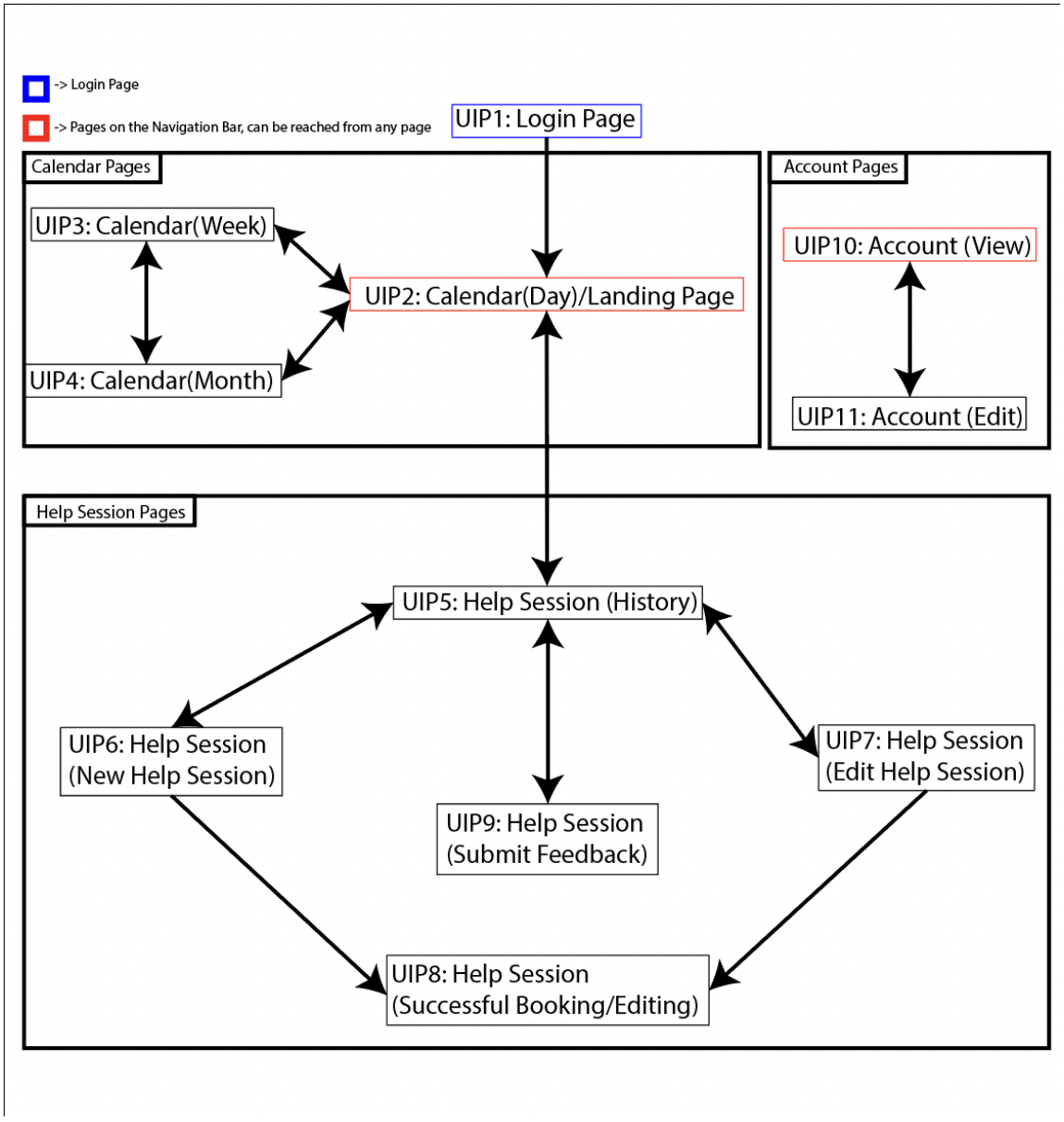
The User interface Walkthrough consists of three navigation diagrams and a series of User Interface Design Mockups. The navigation diagrams are split into 3 sections, the experience as a student user, helper user, and admin user. There are small differences between the three, mostly having to do with permissions to

data and scheduling features outlined in figures 1.1-1.3. The User Interface Design Mockups show a visual of the actual pages of the application, the UI elements, and their purposes.

3.1 Navigation Diagrams

3.1.1 Student View

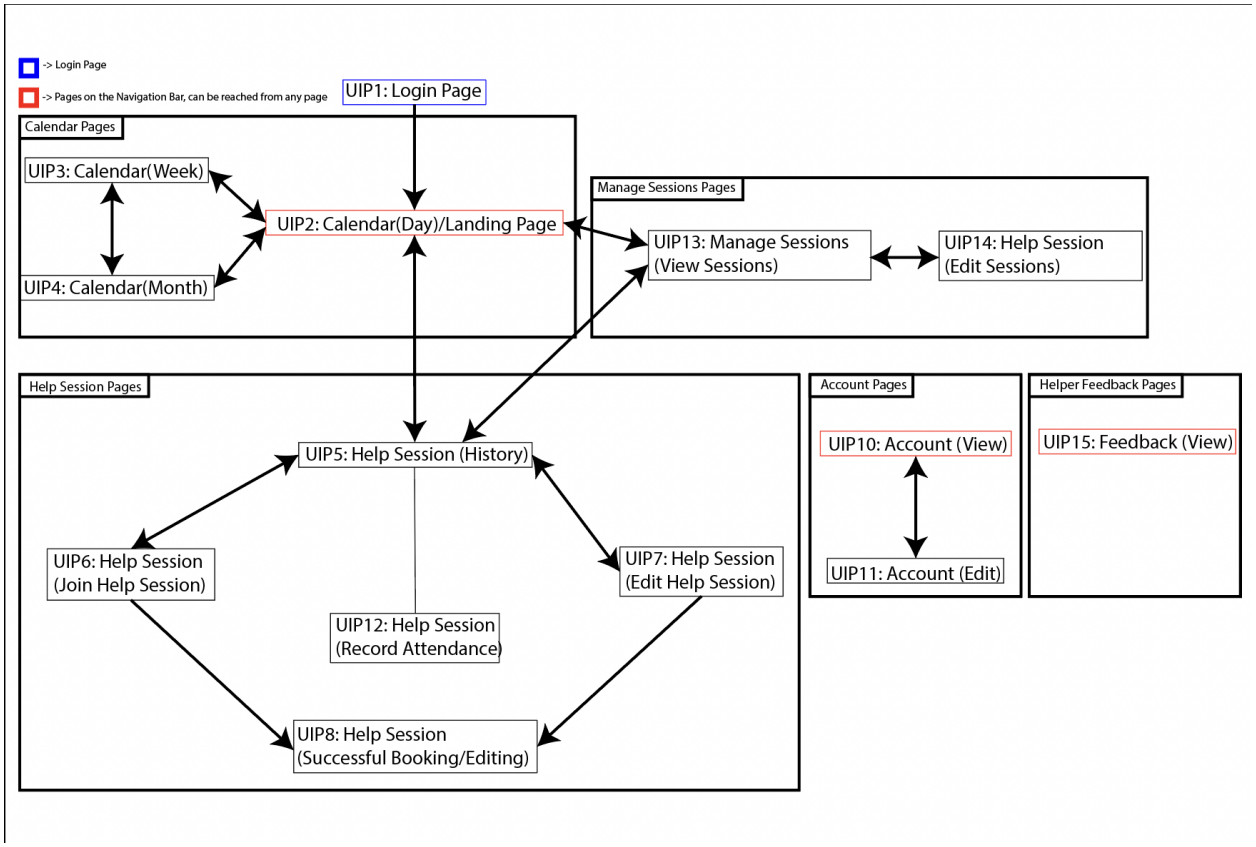
figure 1.1: Navigation Diagram Student View



Description: Figure 1.1 shows the navigation paths through each page of the application as a general student user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

3.1.2 Helper View

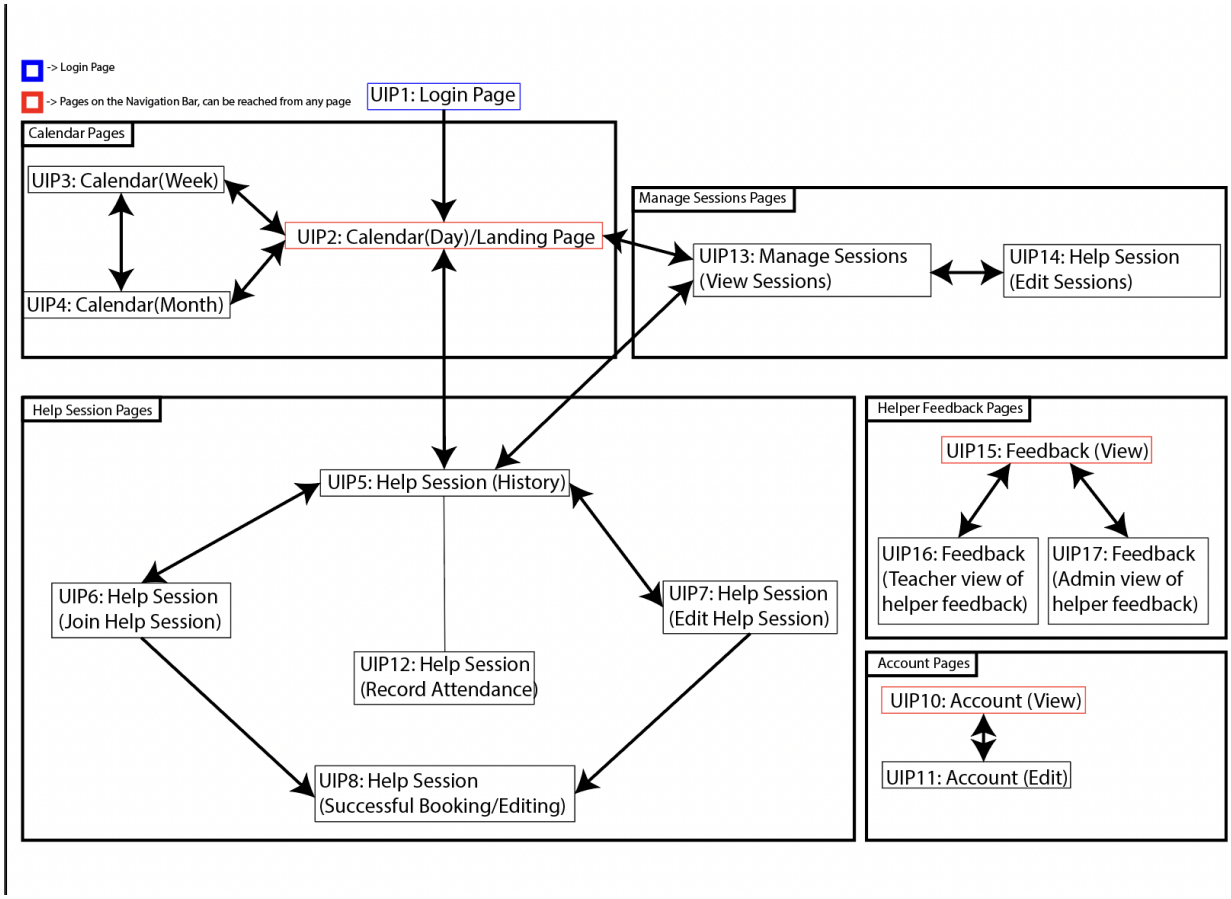
figure 1.2: Navigation Diagram Student View



Description: Figure 1.2 shows the navigation paths through each page of the application as a general helper user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

3.1.3 Admin View

figure 1.3: Navigation Diagram Admin View

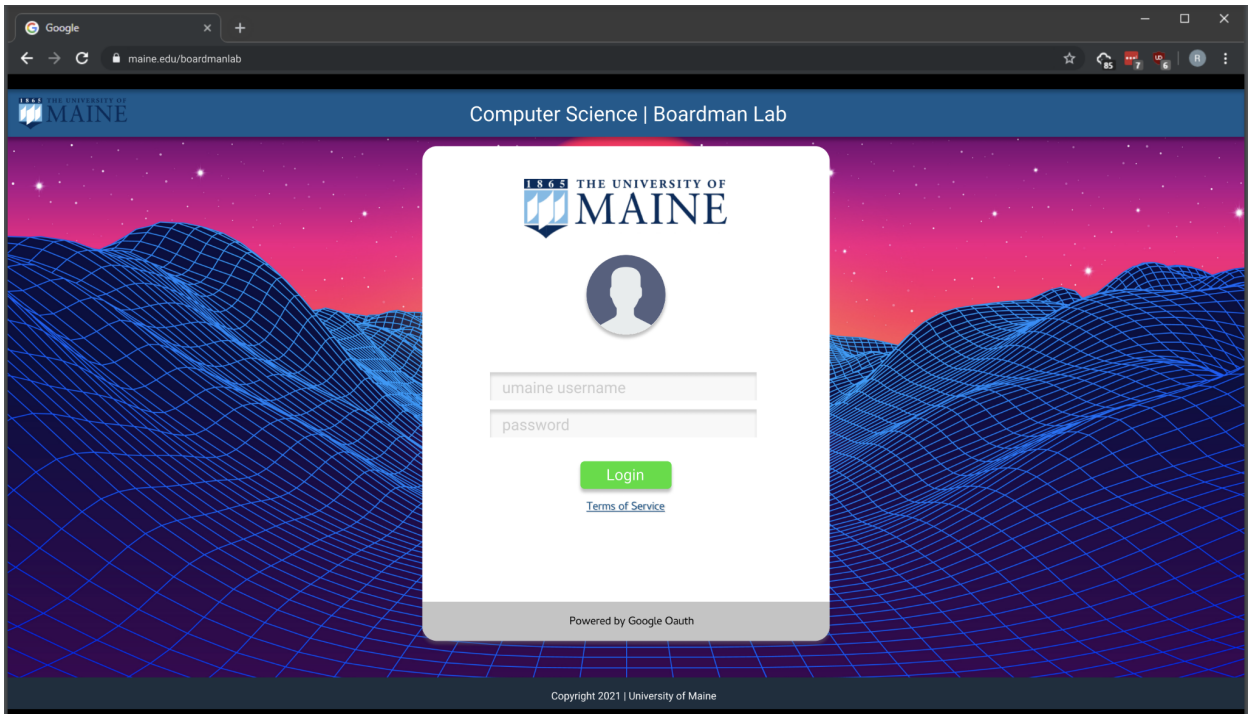


Description: Figure 1.3 shows the navigation paths through each page of the application as a general admin user. Arrows denote a direction a user can take from one page to another, and the black boxes separate general page groupings in the application.

3.2 User Interface Design Mockups

3.2.1 UIP1: Login

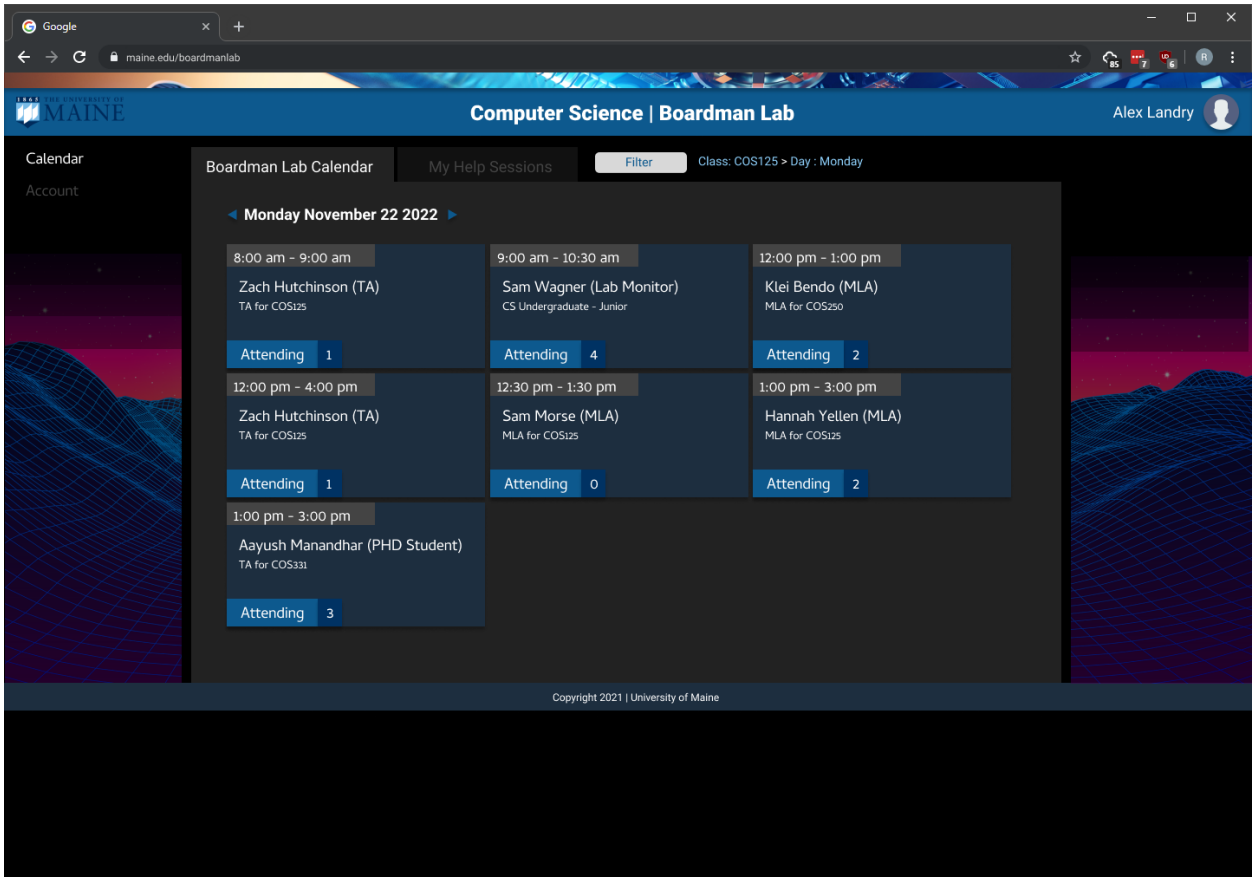
figure 2.1: Login Screen



Description: UIP1 is the main login page for all users. The user interaction in UIP1 occurs in the main window in the center of the screen. The user is able to enter strings into the two text boxes for username and password. The user can also click the buttons for Login, which will submit the imputed strings for validation and send the user to UIP2, and the Terms of Service button which will take the user to the terms of Service page of the application.

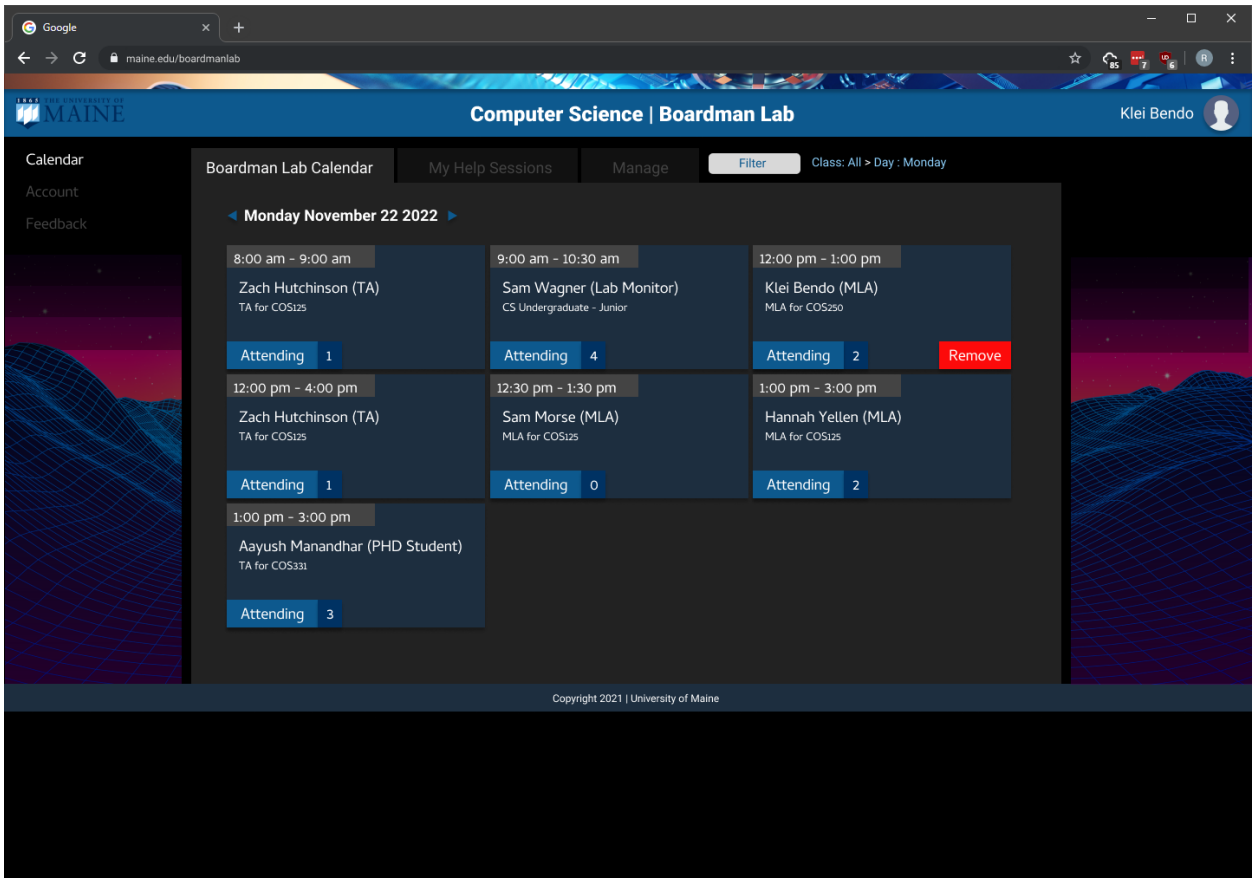
3.2.2 UIP2: Calendar(Day)/Landing Page

figure 2.2: Calendar (Day)/Landing Page Student View



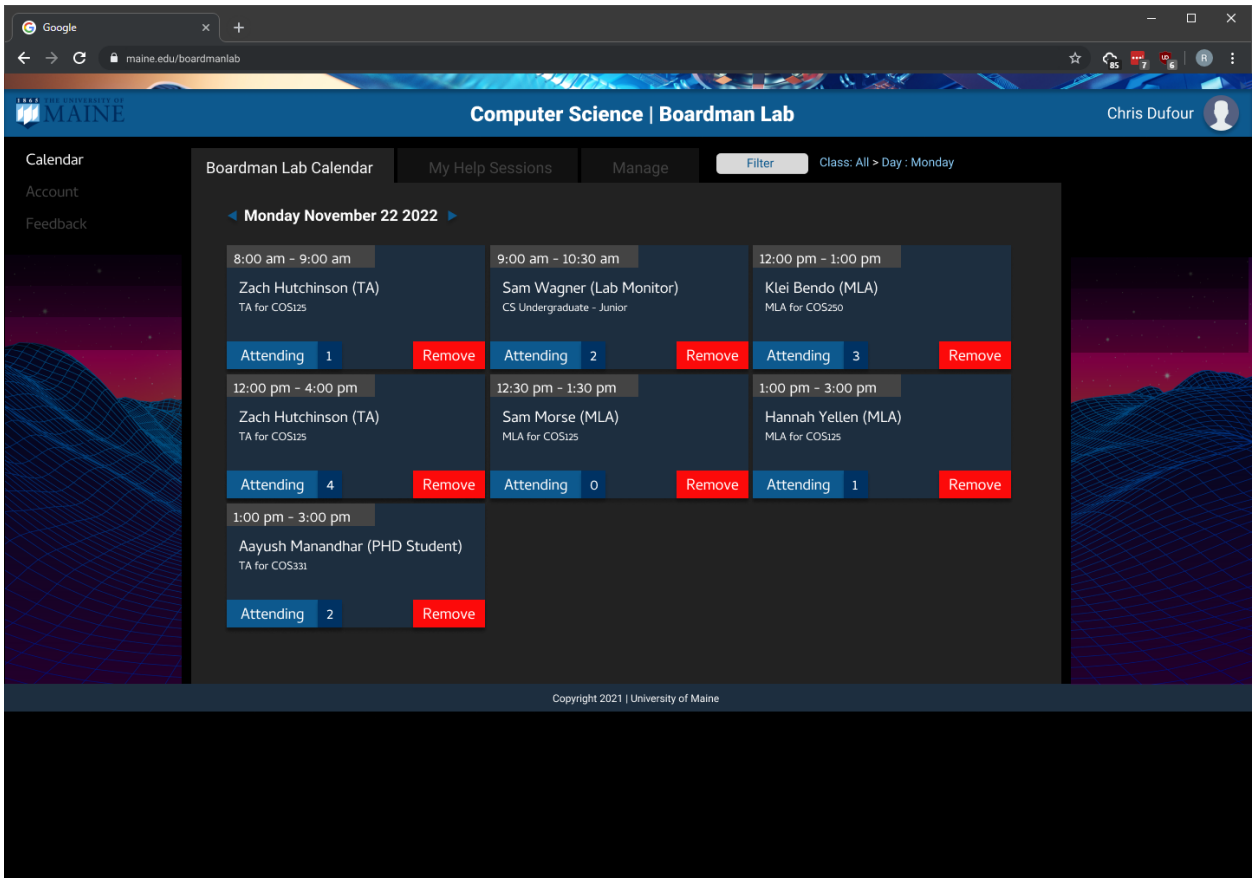
Description: UIP2 is the landing page for the user once they log in. Additionally it is the day view of the calendar. On this page, users will see help sessions scheduled at specified times during the day as well as hours for helpers. This page has a different look for different types of users: student, helper, and admin. The student view (seen above, figure 2.2) has no interactive elements except moving to the my help page

figure 2.3: Calendar (Day)/Landing Page Helper View



Description: Figure 2.3, seen above, is the helper view of the day calendar. It has the same elements as the student day calendar in figure 2.2, but helpers are able to remove their own help session by clicking the “remove” element in red, removing their help session from their calendar and the attendee’s calendars.

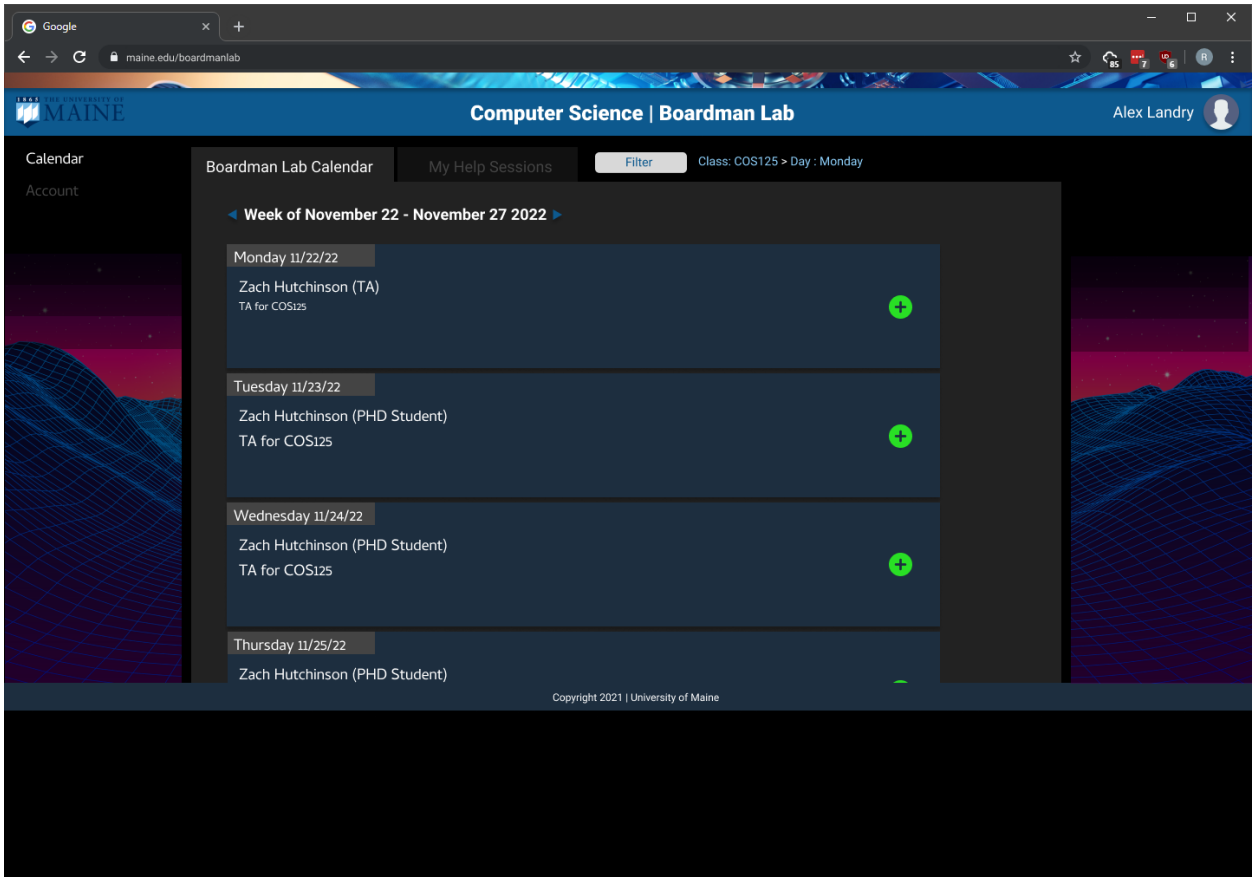
figure 2.4: Calendar (Day)/Landing Page Admin View



Description: In the image the admin can view and remove all of the times that have been scheduled to have a helper in the lab. The information they can view include the name of the helper and what they are, eg: Ta for 125 or MLA for 250. The admin is able to view how many people are attending for the time that the helper has scheduled. The admin also has the option to remove any of the scheduled times as well. The page also has a filter that allows the use of two factors. The first is by class, with this the admin can view all the helpers for a certain class, eg COS125. The other factor that can be used is the date, in which the admin can view the scheduled times just for the selected date. In order to remove a time from the calendar the admin can select the “remove” button on the lower right corner on any of the times.

3.2.3 UIP3: Calendar(Week)

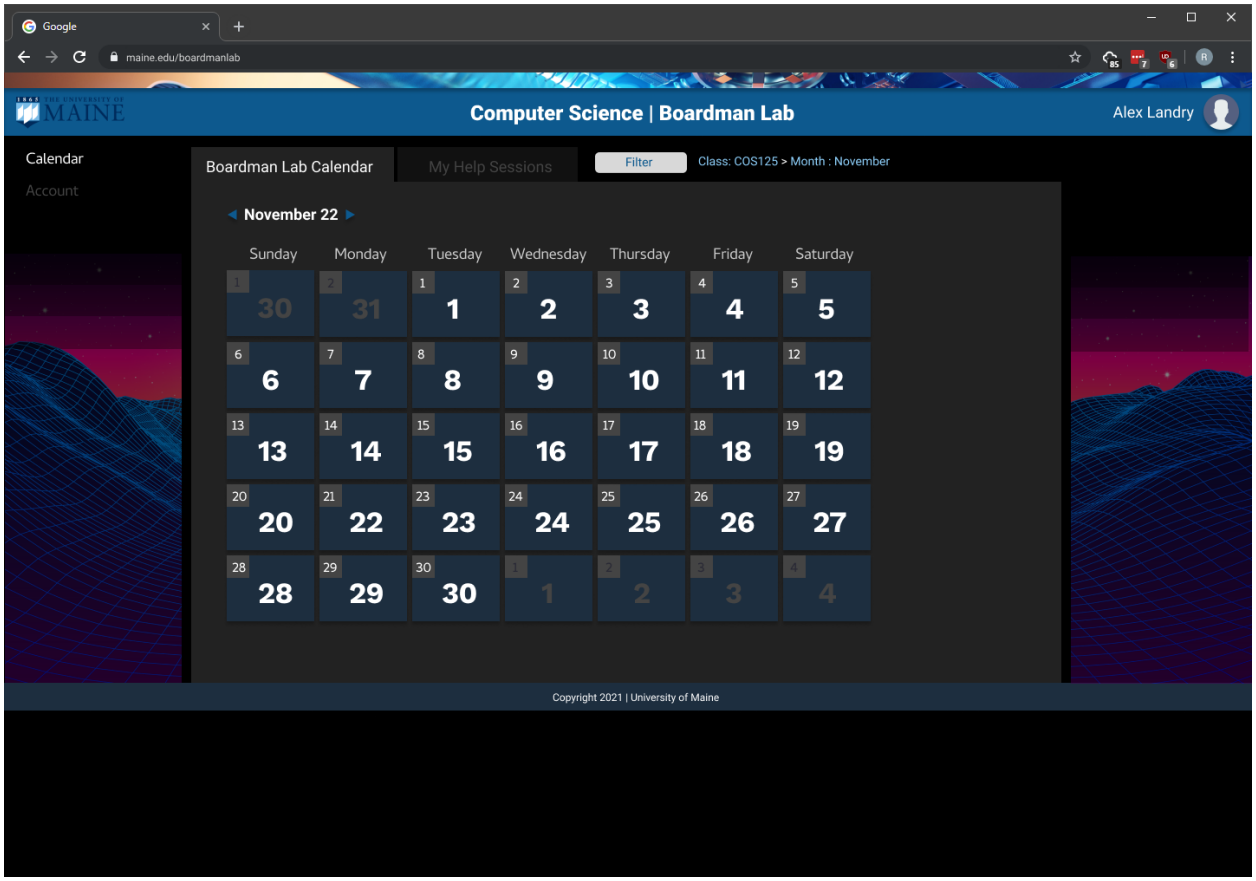
figure 2.5: Calendar (Week) student



Description: In the image the student is presented with the options for lab help throughout the entire week. They have the option to schedule any of the given times that have been declared as available by the helper which is displayed in the boxes that contain the date, the helper's name and what class they can help with. The student also has the option to use the filter, which can be found at the top of the page, which can be used to determine what date that the student wants help. The filter also has an option to determine what class the helper is associated with thus allowing the student to find appropriate help for a specific class.

3.2.4 UIP4: Calendar(Month)

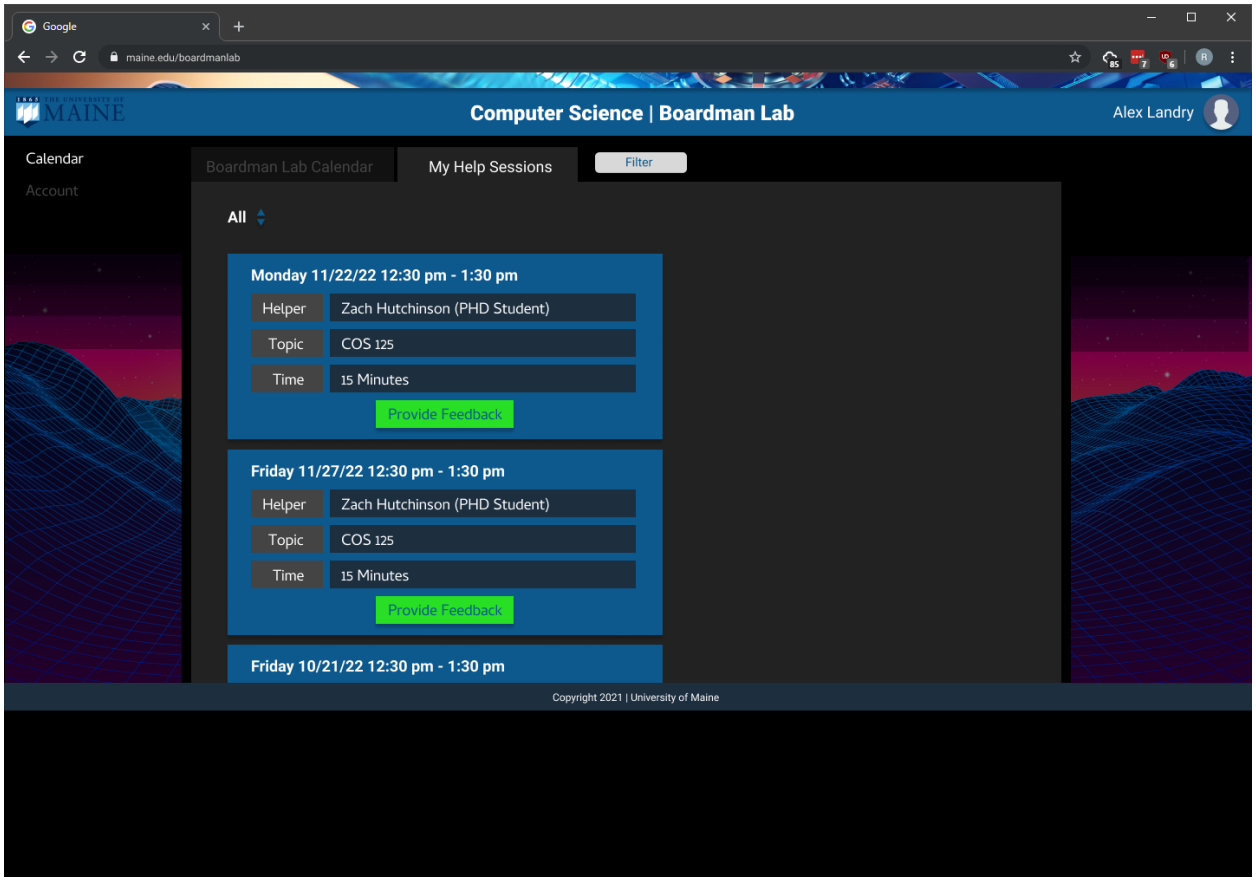
figure 2.6: Calendar (Month) Student



Description: In the image the student can view all the days within the month. Each date within the UI is a button that can be selected. After selecting a date the app will display all of the available times for help on that date. The page that the student will be taken too will be the image that was shown earlier 3.2.2 UIP2: Calendar(Day).

3.2.5 UIP5: Help Session (History)

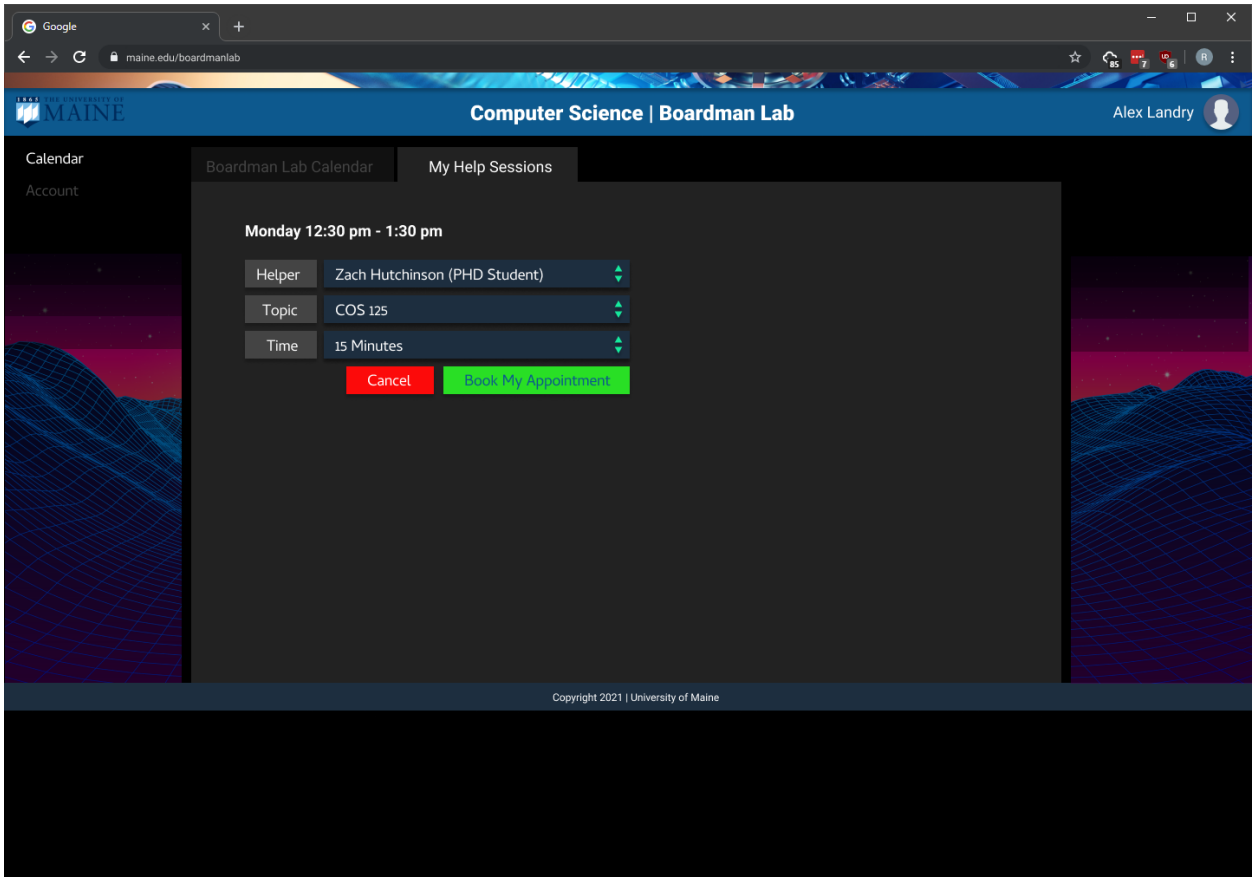
figure 2.7: Help Session (History)



Description: UIP5 shows the student's help session history. Whenever a student attends a help session, that session appears in their history. The windows for each session include the date of the session, the associated helper, topic and time. The user is able to select the "provide feedback" element to bring the user to UIP9. The user is also able to select the filter UI element to filter by helper, topic, time, and date.

3.2.6 UIP6: Help Session (New Help Session)

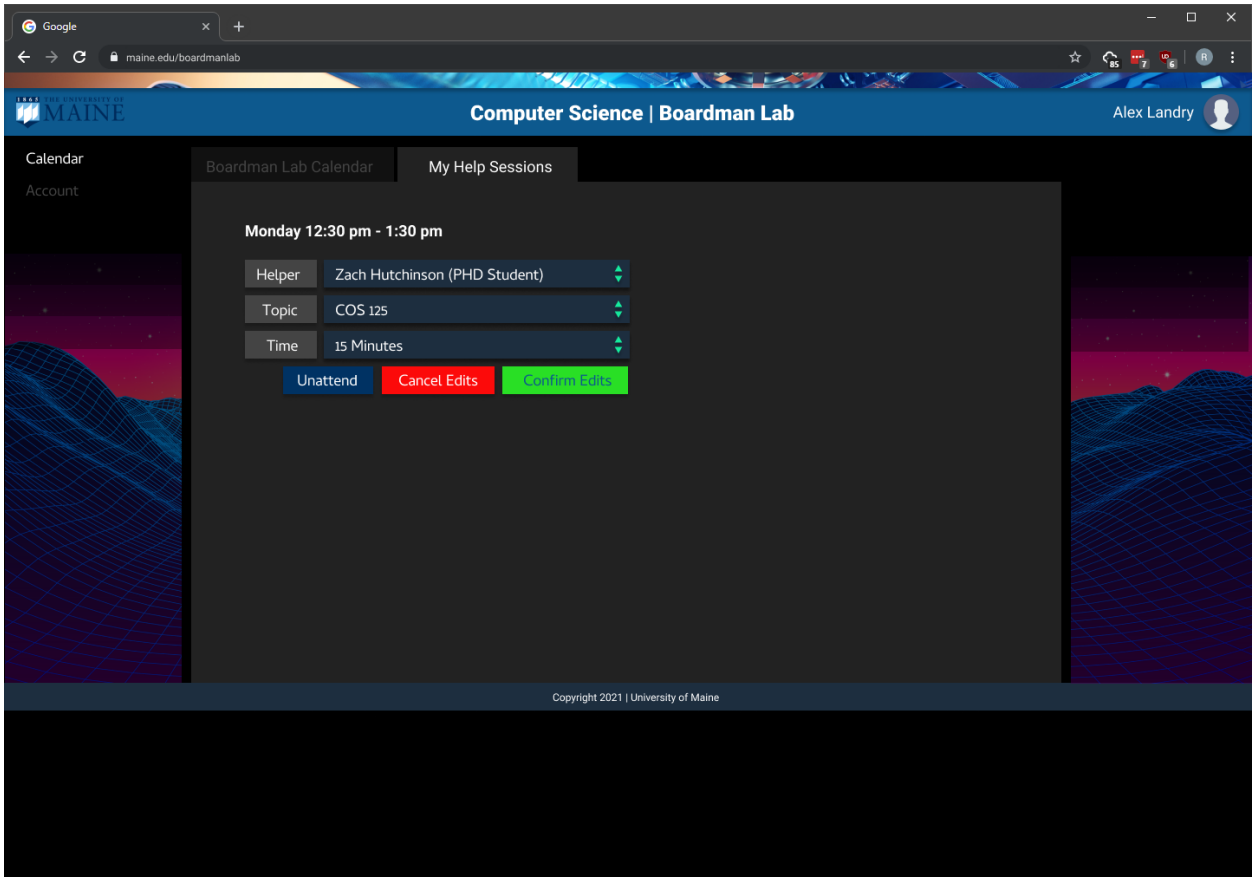
figure 2.8: Help Session (New Help Session) Student View



Description: UIP6 shows the process of a student booking an appointment with a specific helper. The user selects a time/date, a helper, topic of help, and the time frame needed. There are two elements below this to cancel the current action, and to confirm the selections.

3.2.7 UIP7: Help Session (Edit Help Session)

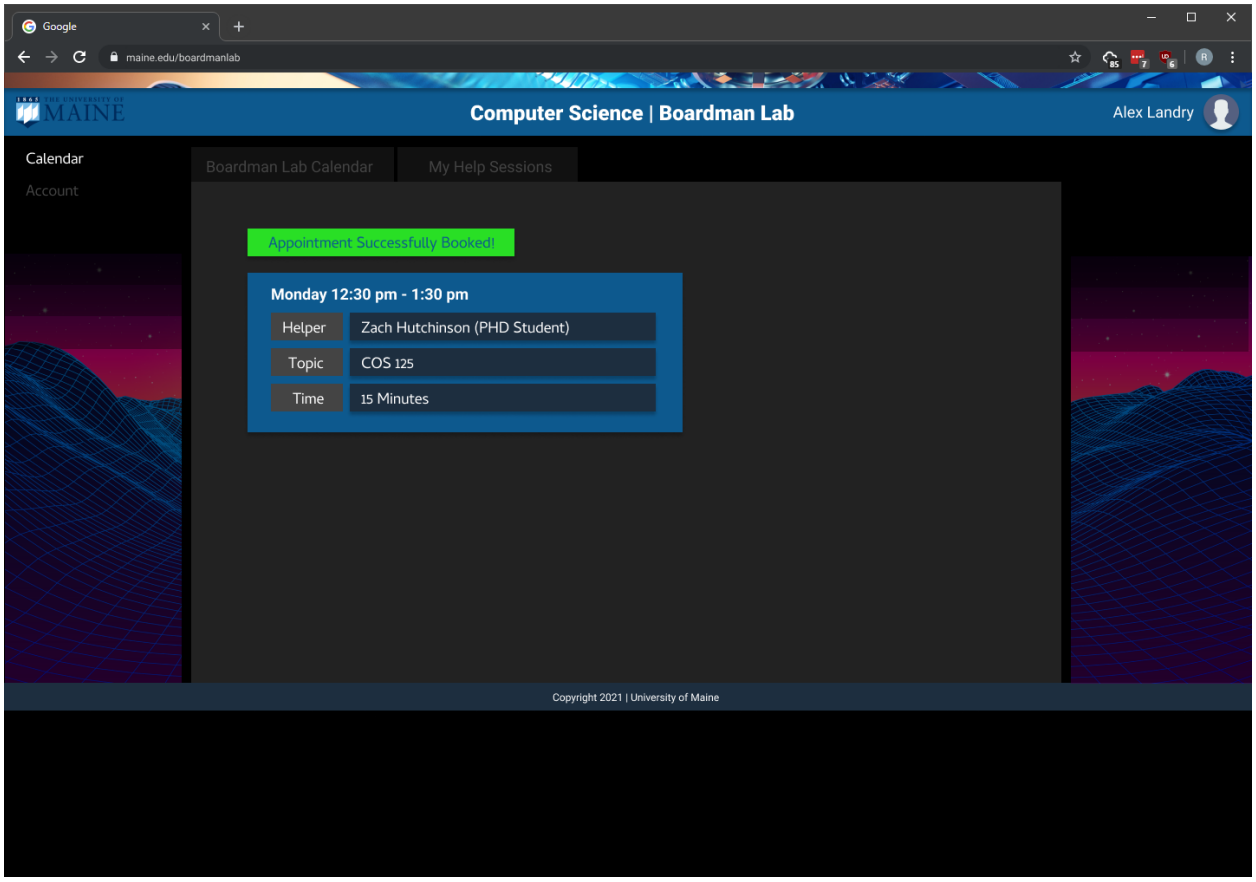
figure 2.9: Help Session (Edit Help Session) Student View



Description: Figure 2.9 shows what a student user would see when editing their help session. They are given the options to change helper, topic, and time with drop down menus. The three options you see under the previously mentioned features are allowing the user to confirm their changes, cancel their changes, and unattend which would mean they won't be able to make it.

3.2.8 UIP8: Help Session (Successful Booking/Editing)

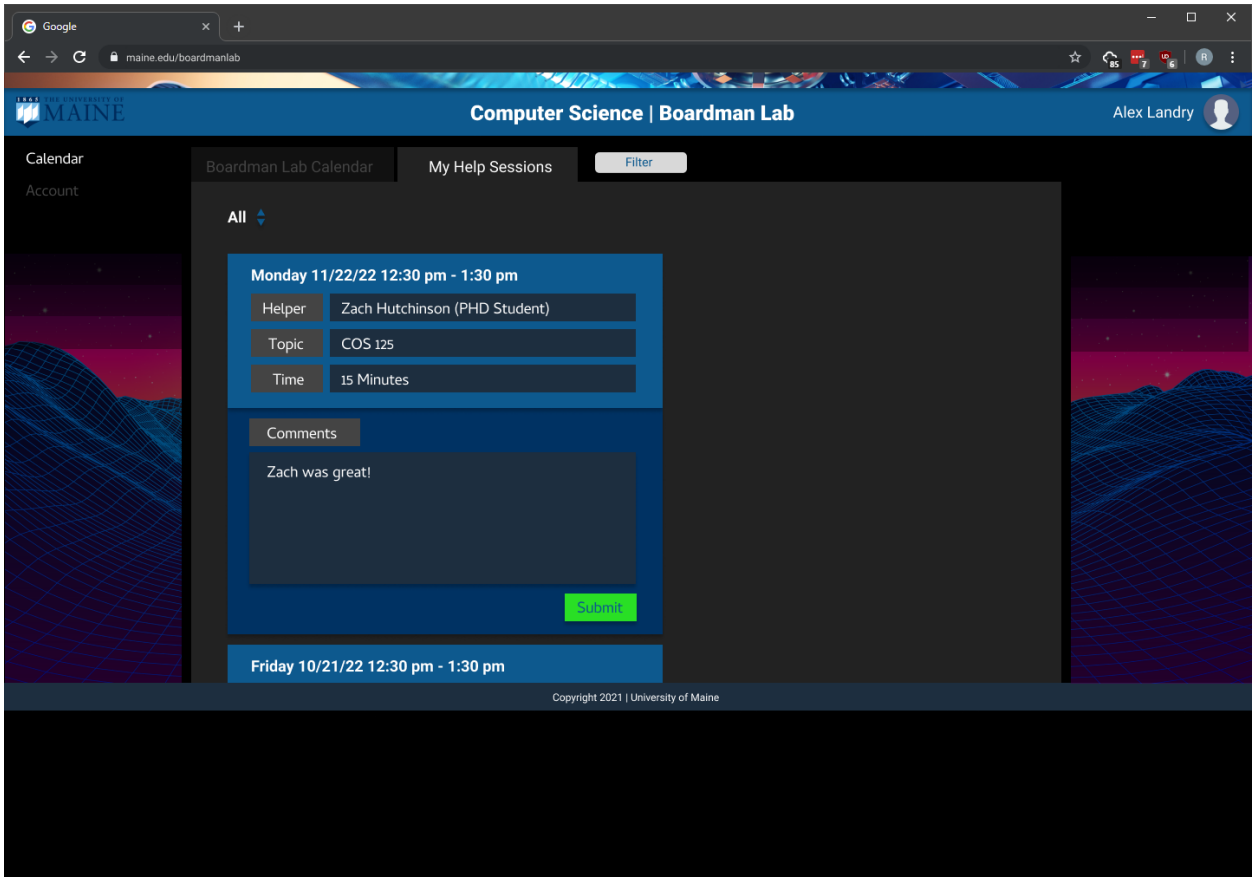
figure 2.10: Help Session (Successful Booking/Editing)



Description: Figure 2.10 shows the confirmation screen a student user will see after successfully booking a help session.

3.2.9 UIP9: Help Session (Submit Feedback)

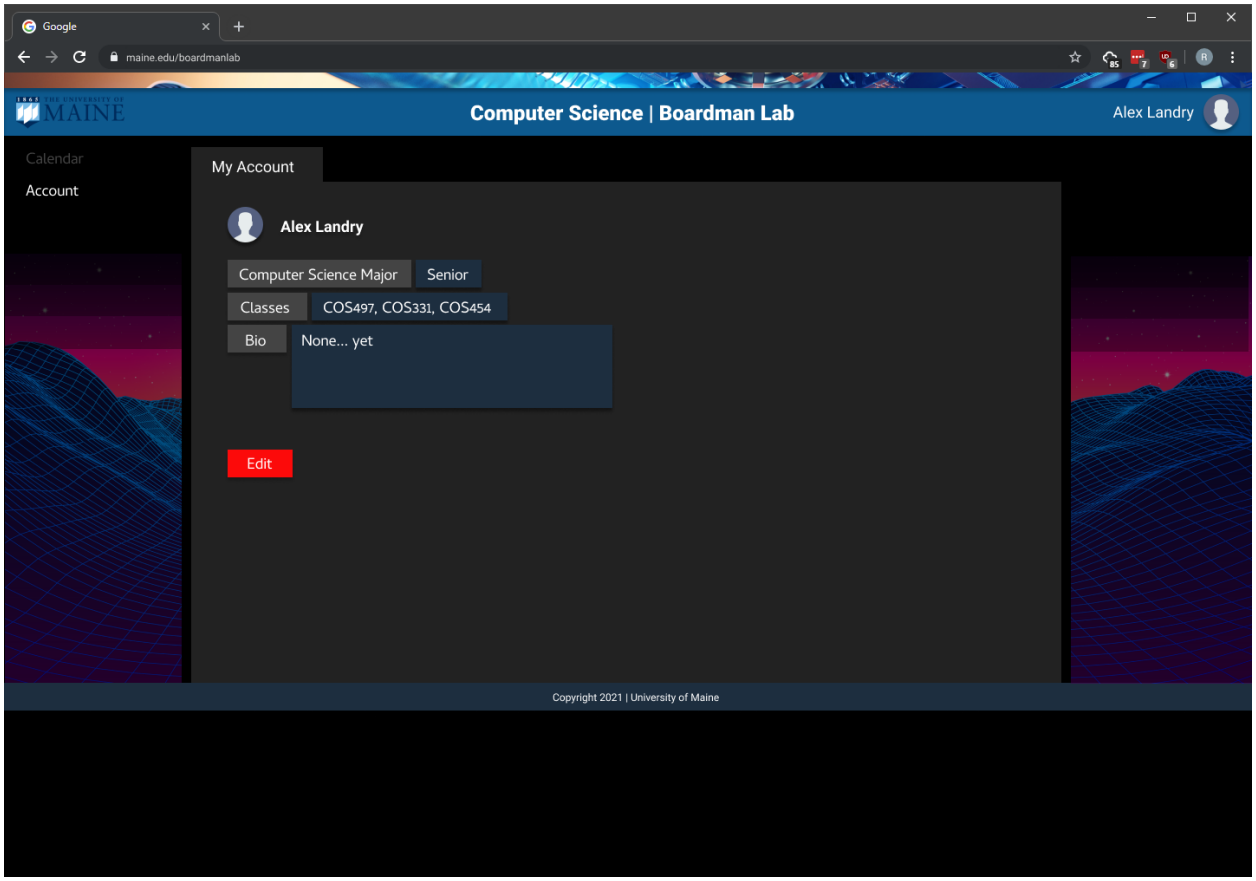
figure 2.11: Help Session (Submit Feedback)



Description: Figure 2.11 shows what screen a user will see when checking past help sessions. The user is given an option to rate their session on a 1-5 star scale. This information will only be able to be seen by the helper themself, the student who wrote it, and administrators. Under the star rating the student is given the option to leave any comments which the helper will be able to see. There is also a submit button in the bottom right of the feedback box. This will submit the feedback to the helper.

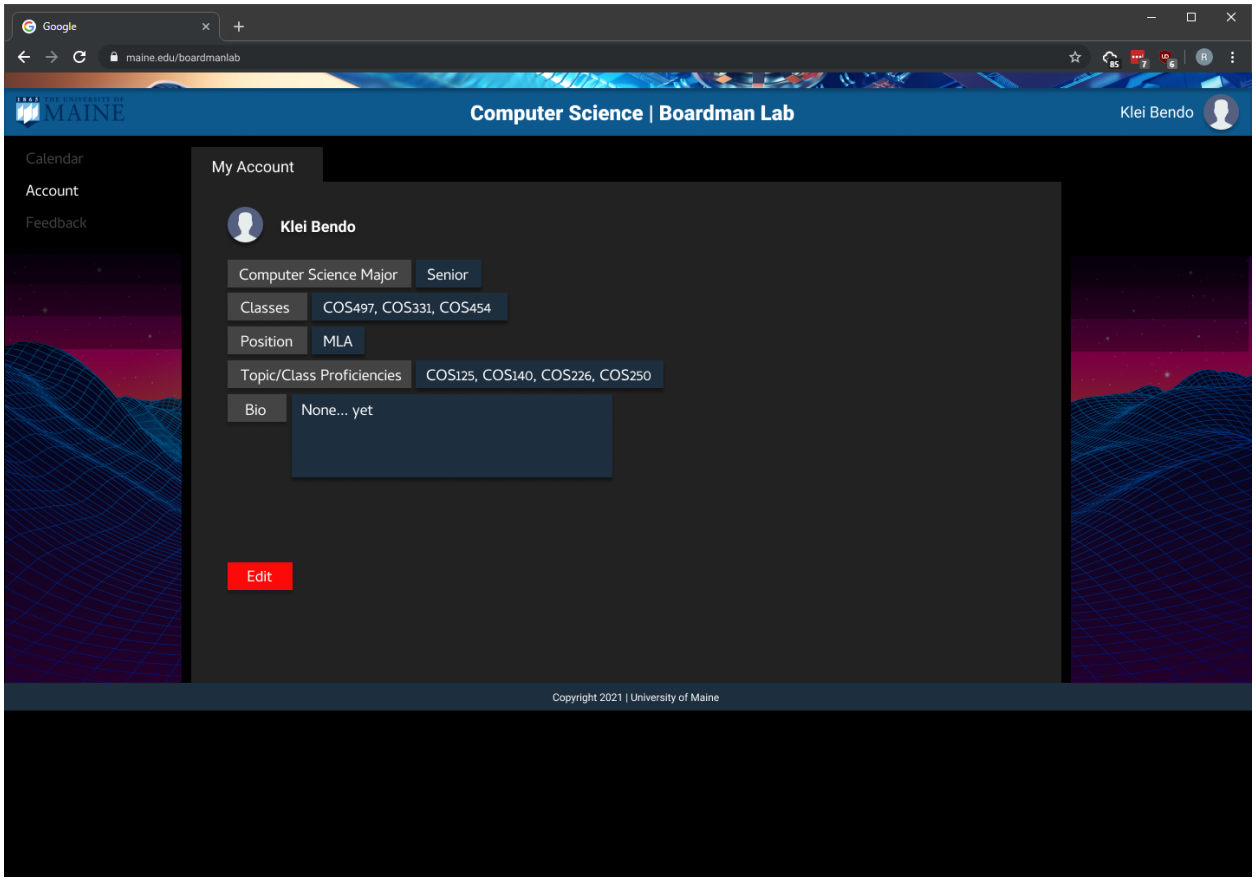
3.2.10 UIP10: Account (View)

figure 2.12: Account (View) Student View



Description: Figure 2.12 shows a view of the 'Student Account', which consists of a student's academic year, classes, and bio. There is an edit button, which when pressed allows the student to edit their profile. On the left of the screen there is a navigation bar that they can use to navigate between the account view and the calendar view.

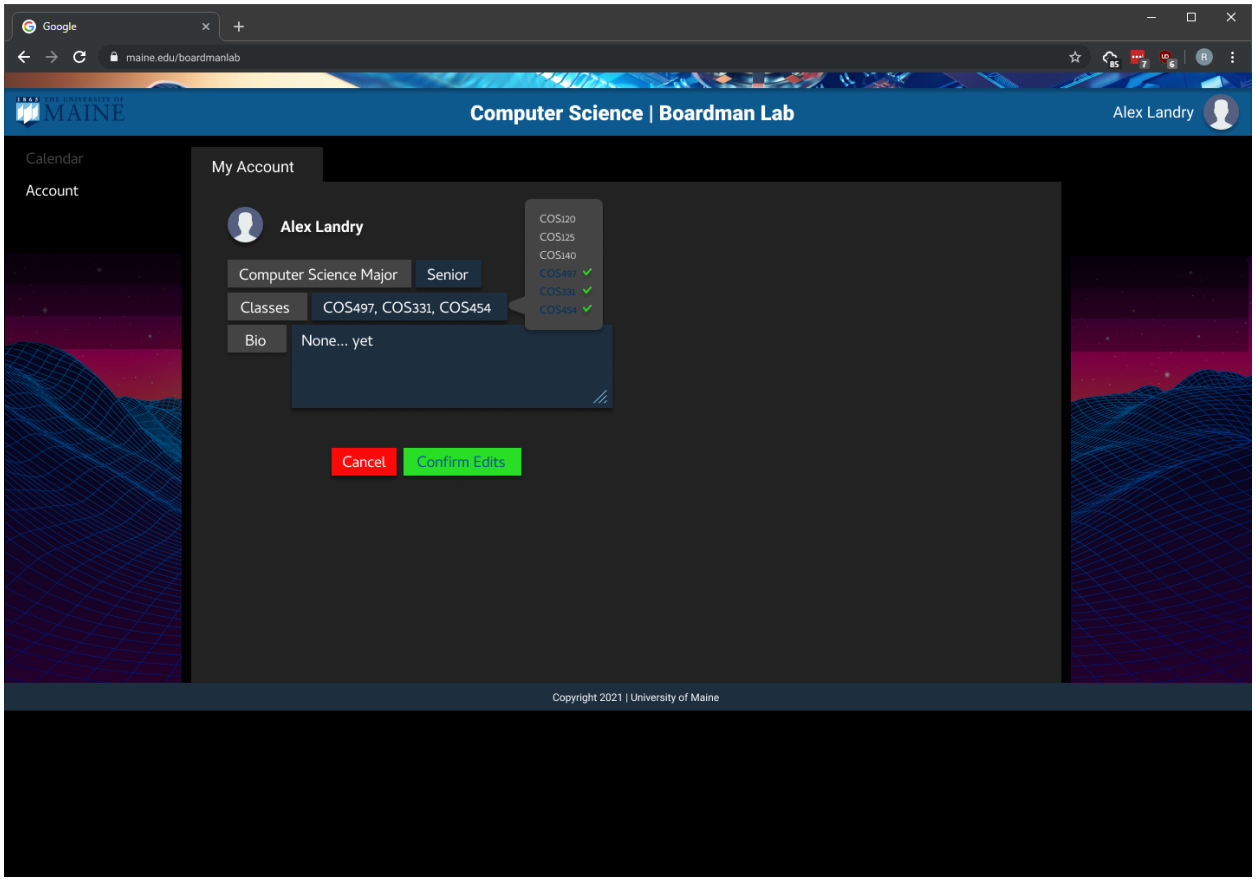
figure 2.13: Account (View) Helper View



Description: Figure 2.13 shows a view of the 'Helper Account', which consists of a helper's academic year, classes, position, topic/class proficiencies, and bio. There is an edit button, which when pressed allows them to edit their profile. On the left of the screen there is a navigation bar that the helper can use to navigate between the account view, calendar view, and the feedback view.

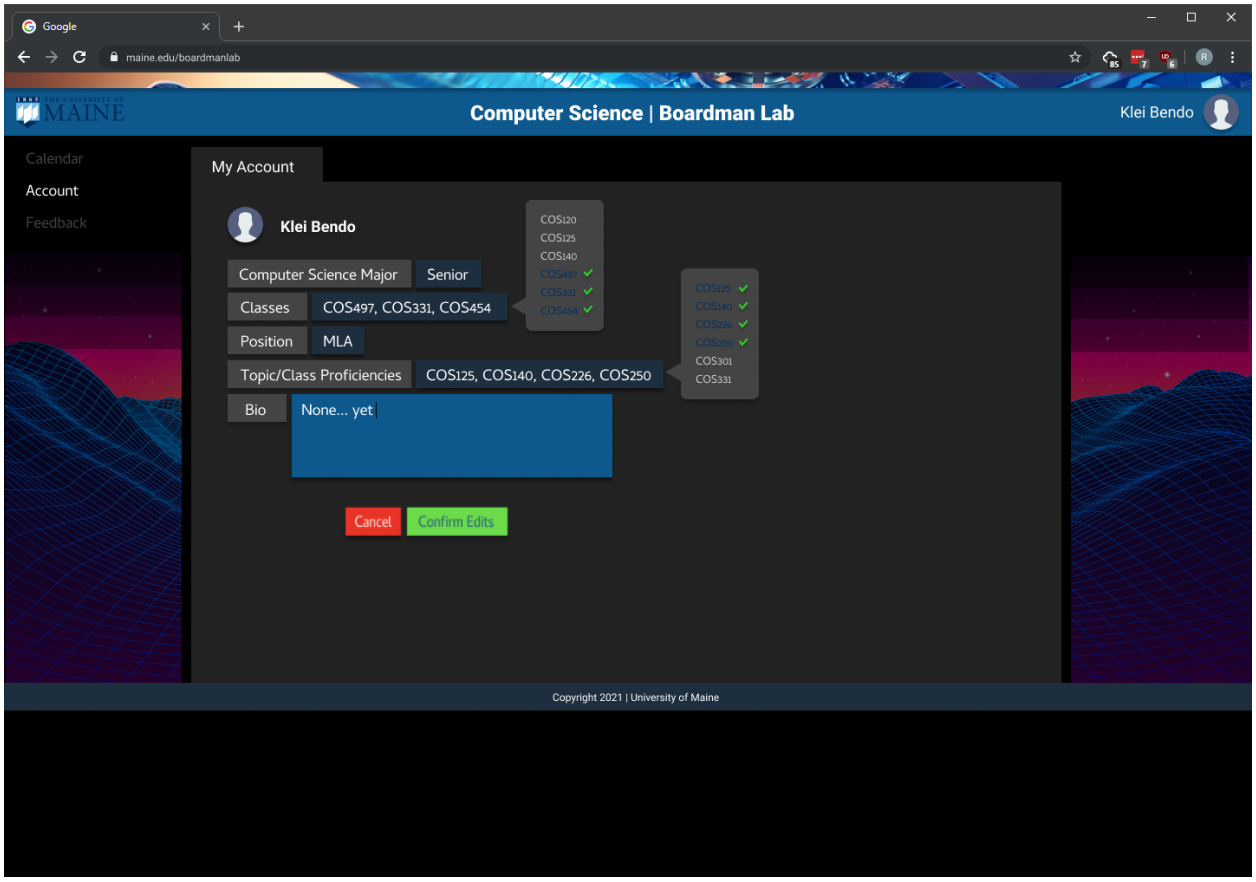
3.2.11 UIP11: Account (Edit)

figure 2.14: Account (Edit) Student View



Description: Figure 2.14 shows the student view of the 'Edit Account' user interface. The student is able to add or remove classes from a given list, as well as change their bio. There is a 'Cancel' button which redirects them to the 'View Account' user interface and discards their changes. There is also a 'Confirm edits' button which saves the changes made.

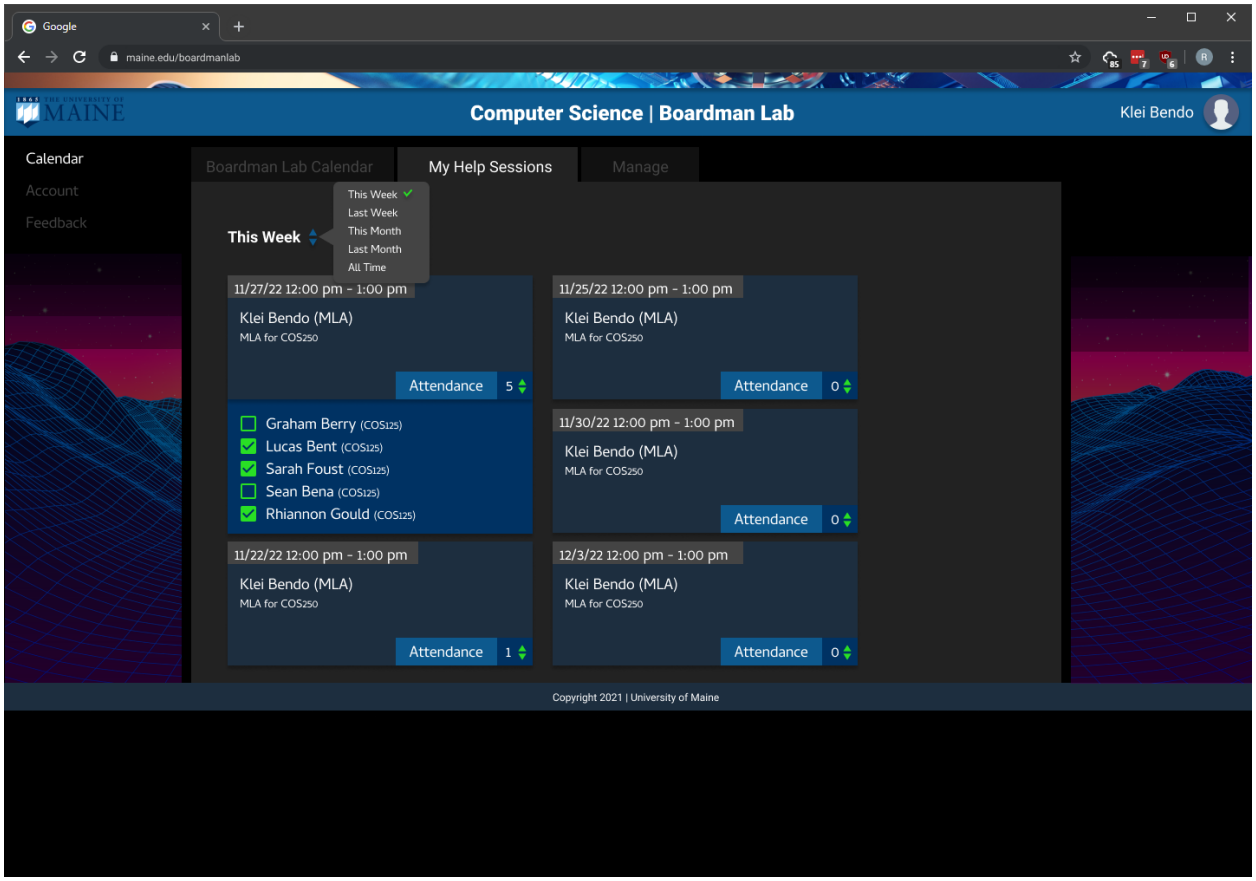
figure 2.15: Account (Edit) Helper View



Description: Figure 2.15 shows the helper view of the 'Edit Account' user interface. The helper is able to add or remove classes and topic/class proficiencies from a given list, as well as change their bio. There is a 'Cancel' button which redirects them to the 'View Account' user interface and discards their changes. There is also a 'Confirm edits' button which saves the changes made.

3.2.12 UIP12: Help Session (Record Attendance)

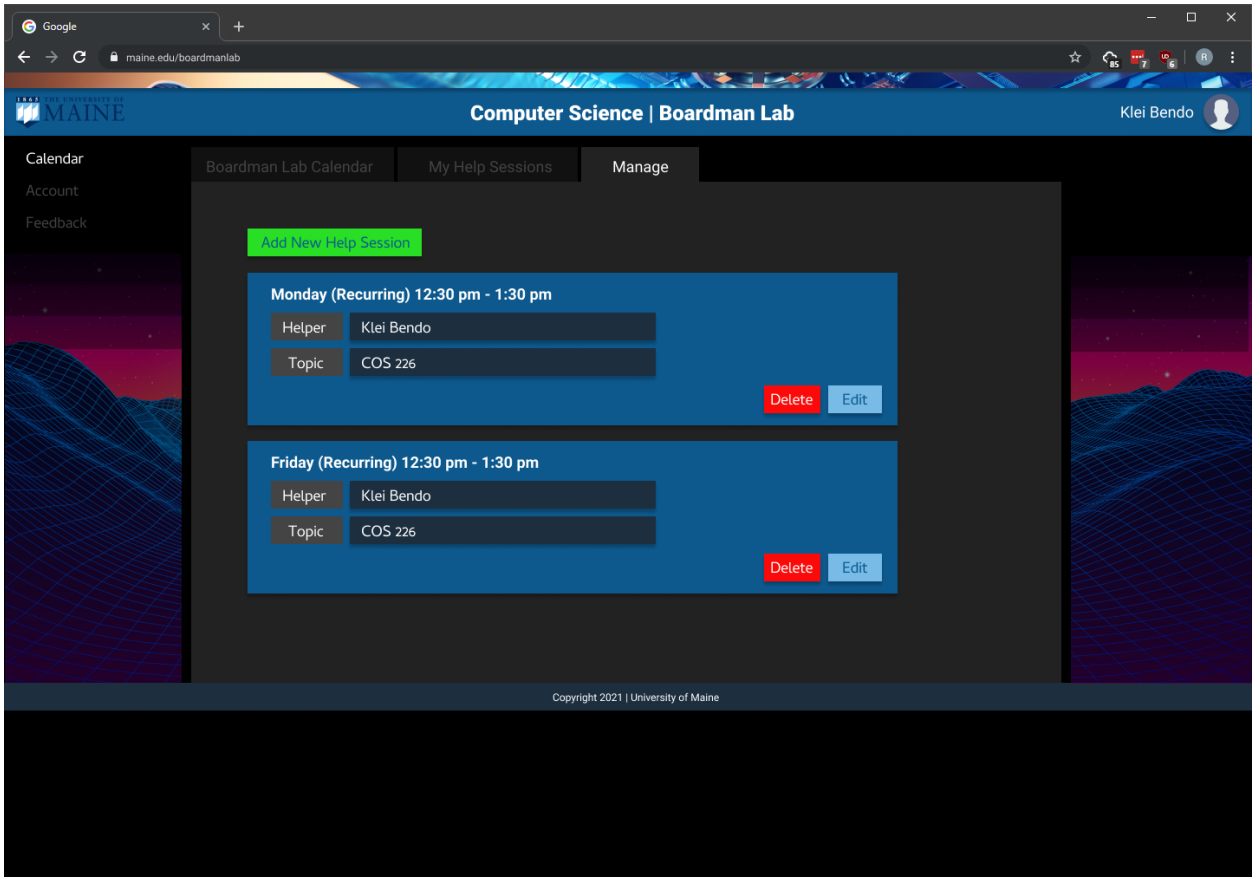
figure 2.16: Help Session (Record Attendance)



Description: Figure 2.16 shows the user interface that a helper can interact with to record attendance for each of their help sessions. The sessions can be filtered by this week, last week, this month, last month, and all time. The helper can view which students signed up for a particular help session, and click the checkboxes for those who actually attended the session.

3.2.13 UIP13: Manage Sessions (View Sessions)

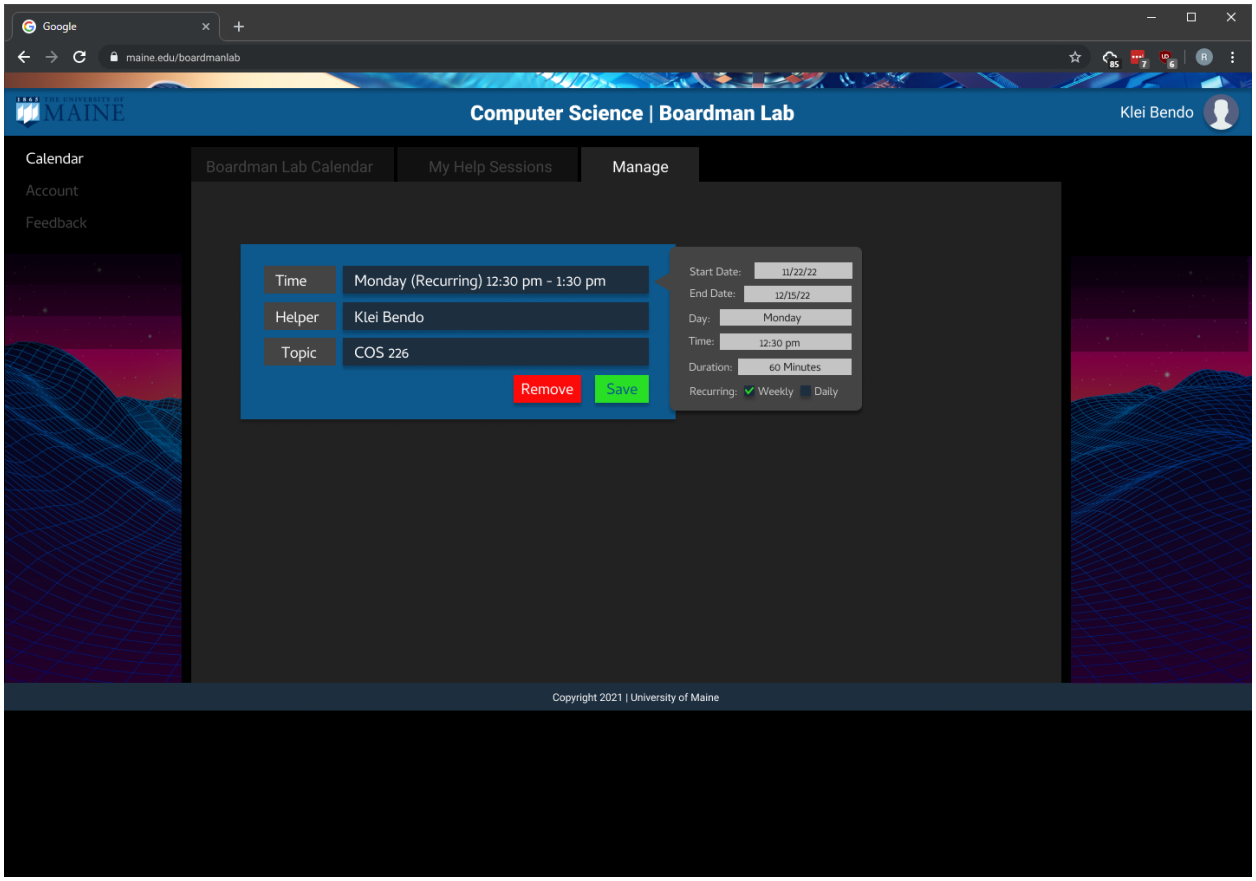
figure 2.17: Manage Sessions (View Sessions)



Description: UIP13 displays all help sessions a helper has scheduled, and will show times, date, and topic information for each help session. UIP13 will also allow users to delete or edit any help sessions they have access to.

3.2.14 UIP14: Manage Sessions (Edit Sessions)

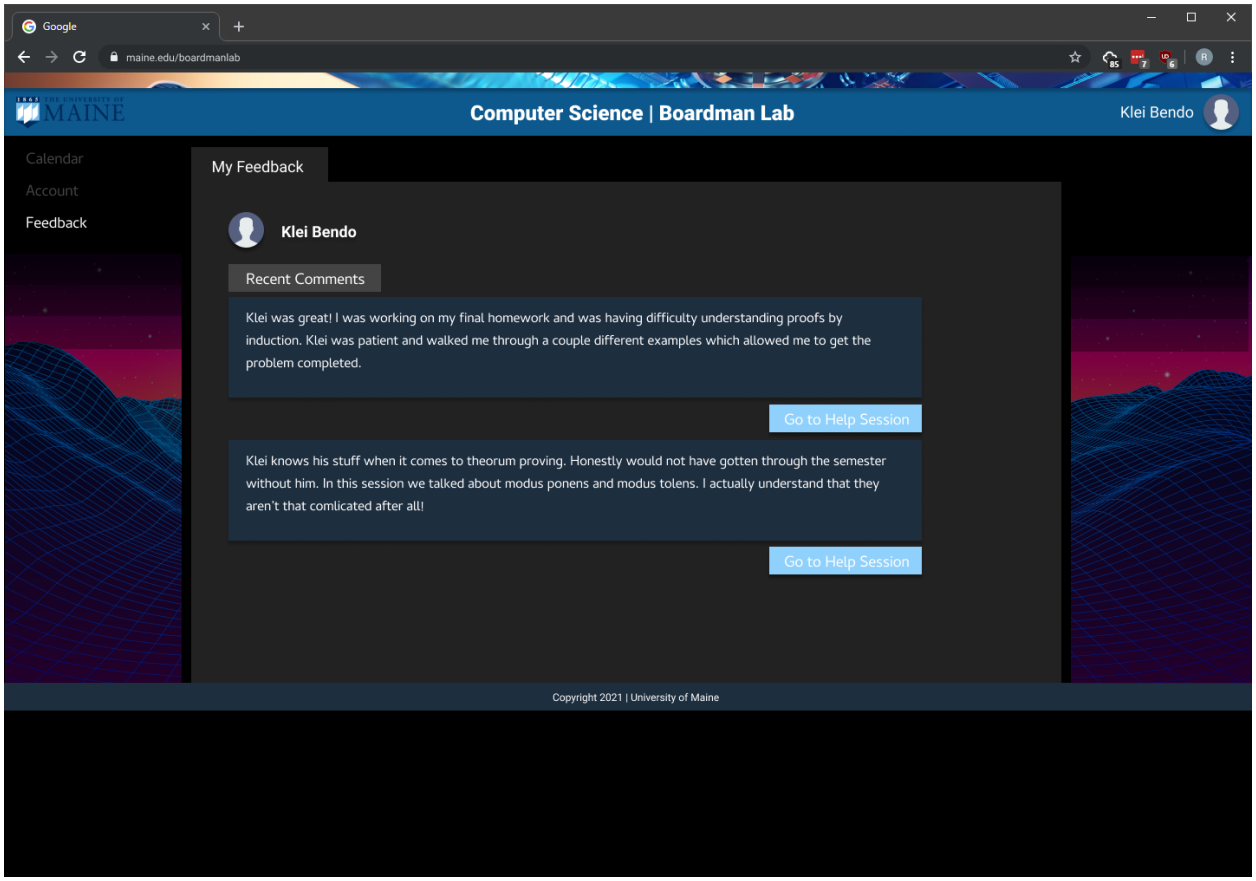
figure 2.18: Manage Sessions (Edit Sessions)



Description: UIP14 displays the edit options available after clicking the edit button on a help session in UIP13. The options for editing include: Start Date, End Date, Day, Time, Duration, and whether the session is recurring. Changes may then be saved or discarded by selecting the appropriate button.

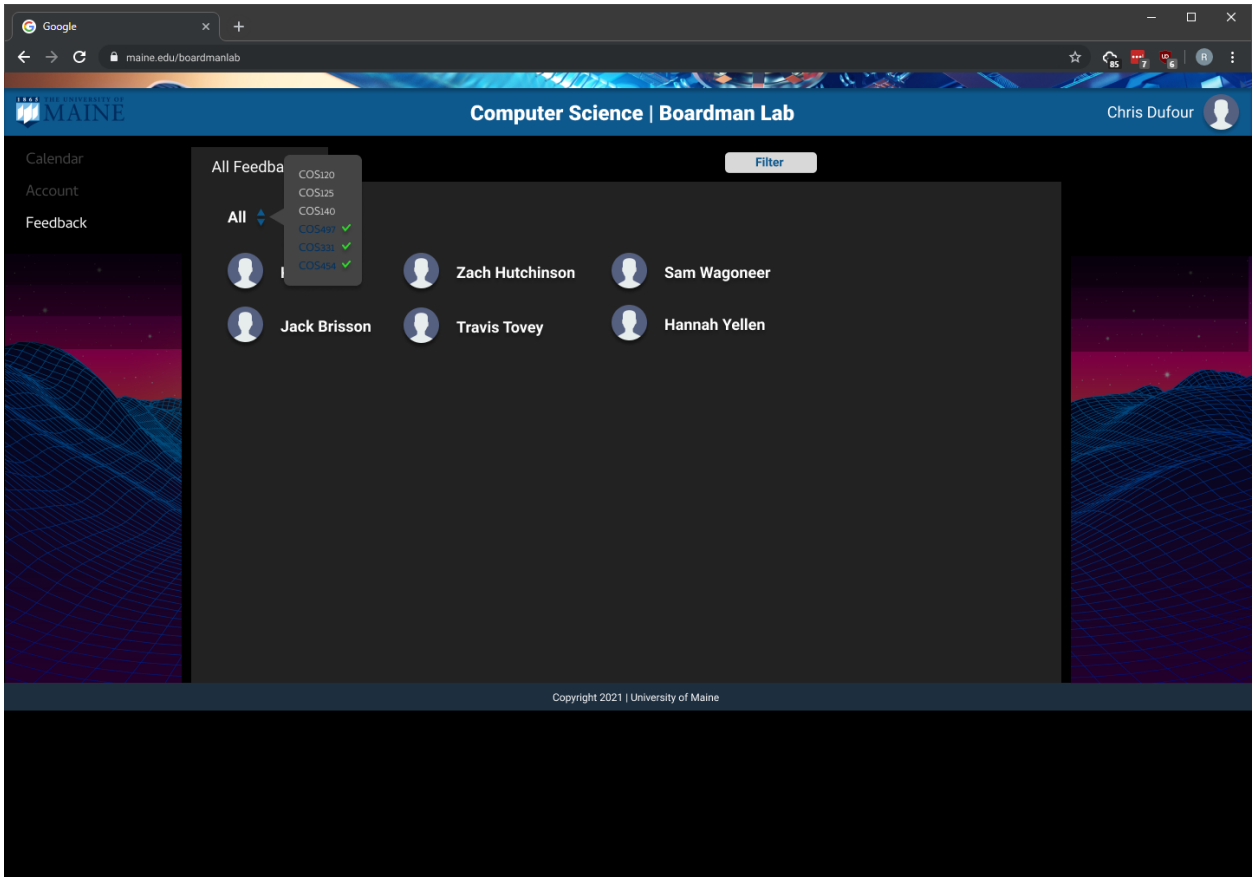
3.2.15 UIP15: Feedback (View)

figure 2.19: Feedback (View) Helper View



Description: UIP15 displays anonymous feedback from students on help sessions. The Helper will see a list of feedback comments as well as an feedback rating averaged from all student feedback.

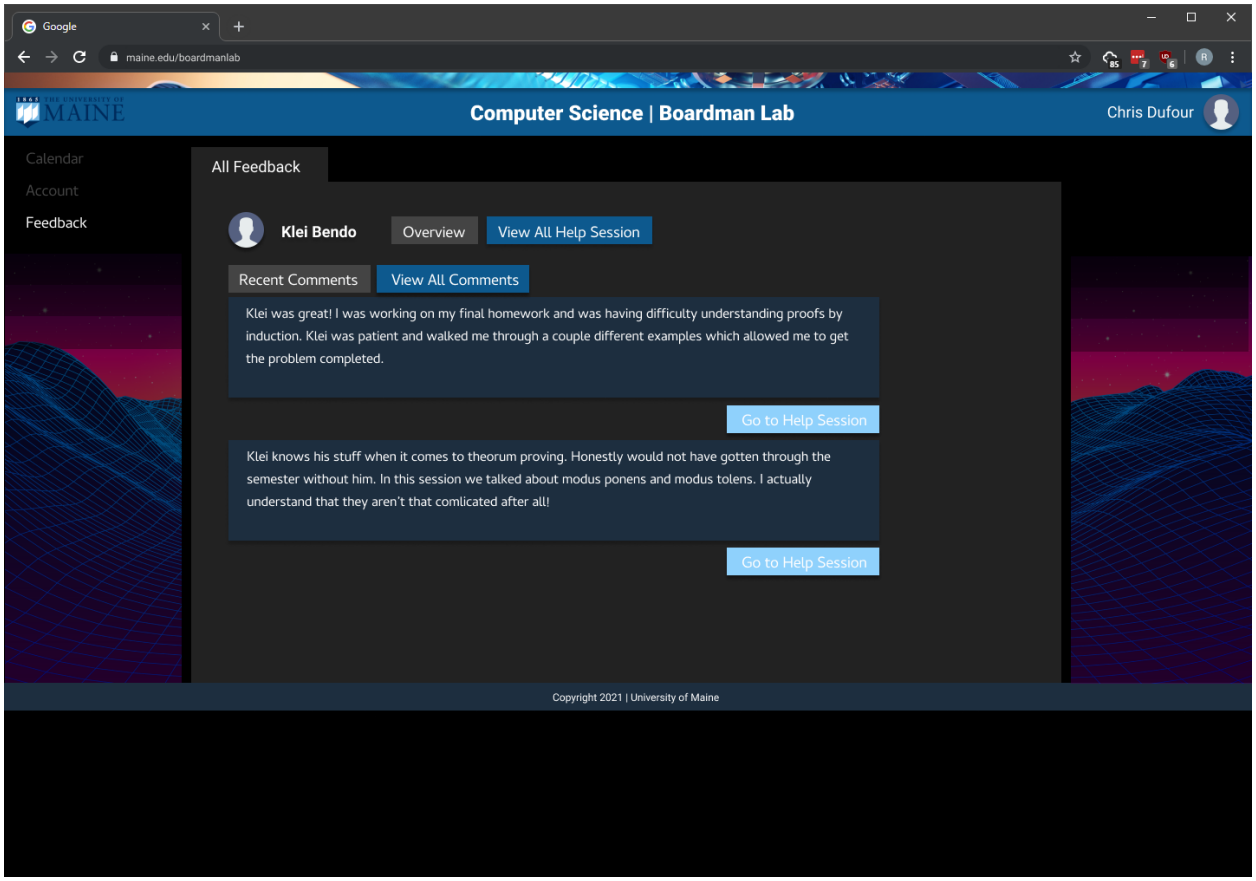
figure 2.20: Feedback (View) Admin View



Description: In the Administrator view, a list of all helpers can be selected from to view feedback for that helper. Administrators may select courses from a list, and only Helpers who have given help with that course will be displayed.

3.2.16 UIP16: Feedback (Teacher View of Helper Feedback)

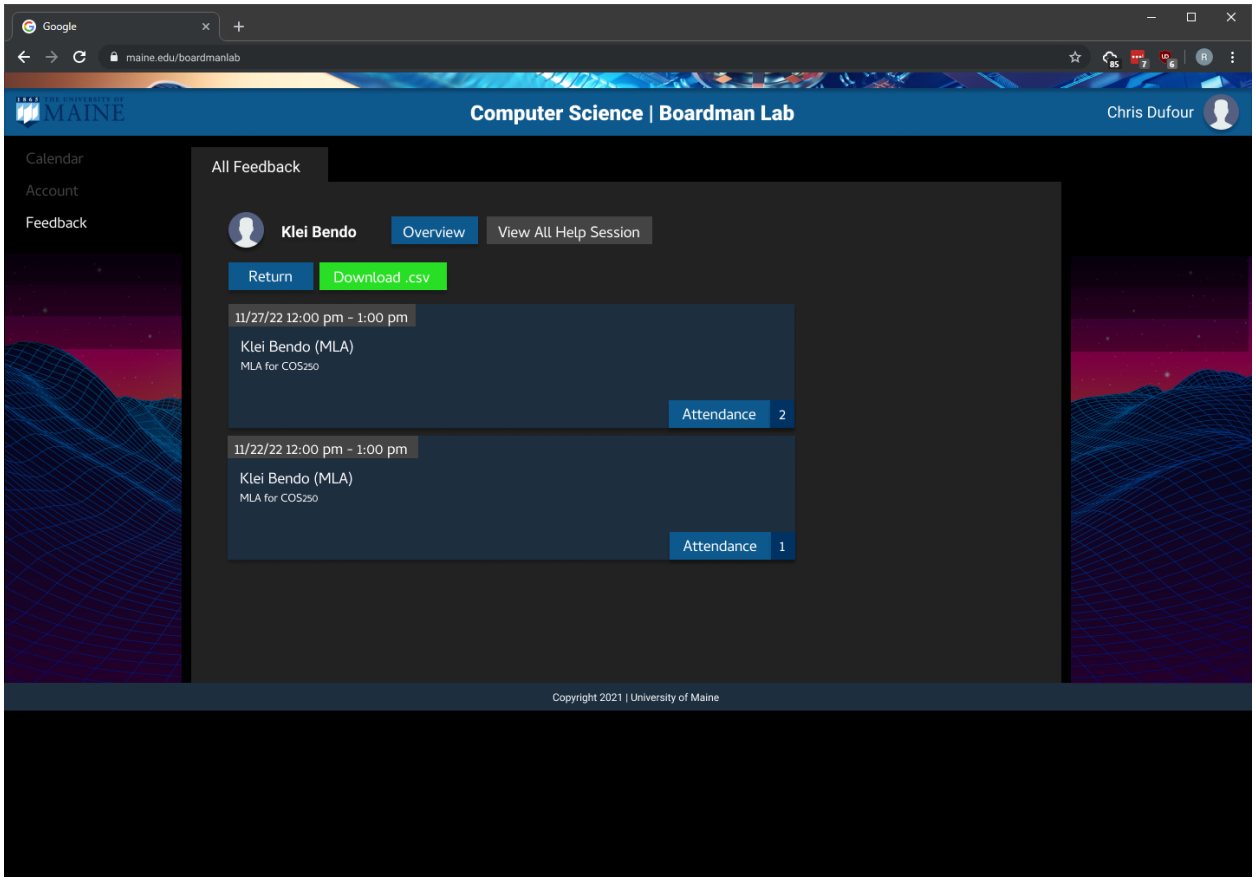
figure 2.21: Feedback (Teacher View of Helper Feedback)



Description: Once the Admin clicks on a particular helper, they can view an overview of the helper's rating and recent comments made. They can choose to view all Help Sessions by that particular helper, go to the Help Session of a particular comment or view all comments made to that helper.

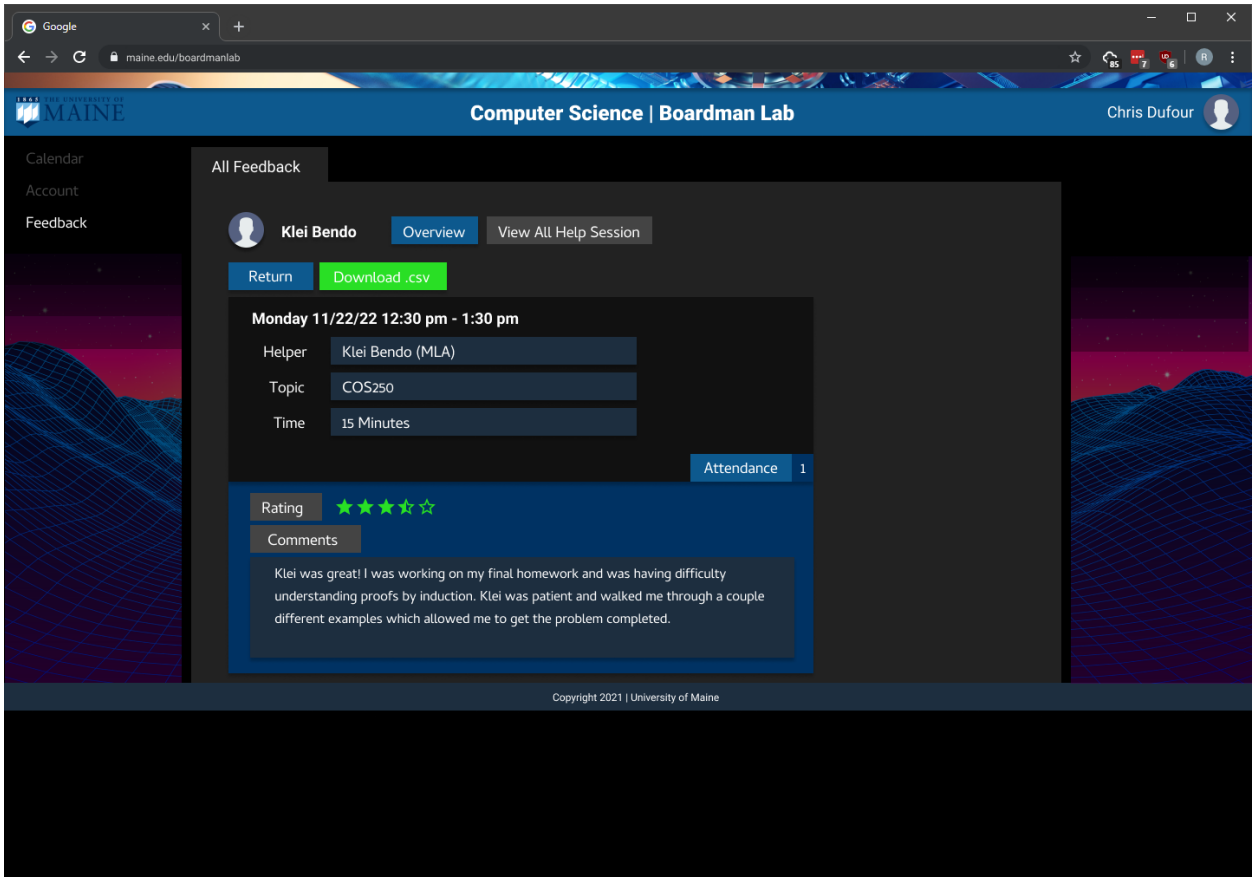
3.2.17 UIP17: Feedback (Admin View of Helper Feedback)

figure 2.22: Feedback (Teacher View of Helper Feedback)



Description: This page is generated when the admin clicks on “View All Help Sessions”. On this page, a list of all help sessions that this particular helper has hosted will be displayed with options to inspect each session. The admin can also choose to return to the list of all helpers, go back to the overview of that particular helper or download a csv file with all of the data contained in each of that helper’s help sessions.

figure 2.22: Feedback (Teacher View of Helper Feedback)



Description: This page is generated when the admin clicks on a help session that is in the list of help sessions for a particular helper. A detailed view of the help sessions is displayed including any comments made to the helper, a list of people who attended the help session and a list of people who signed up for the help session. The admin can also choose to return to the list of all help sessions from that particular helper, go back to the overview of that particular helper or download a csv file with the data contained in the selected help session.

4. Data Validation

Figure 3 contains all data that will be entered into the system by a user. Login information will be entered before any other part of the site can be accessed. Reservation and Help Session information will be entered by users when scheduling or editing sessions. Feedback information will only be entered by student accounts after a help session has been attended.

4.1 Data Validation Expanded

figure 3: Data Validation Description

ID	Data Type	Limits	Format
LogIn.username	String	none	Variable-Length Character Format
LogIn.password	String	length > 8	Variable-Length Character Format
Reservation.date	Date	current date - 4 months ahead	*ISO : yyyy-mm-dd
Reservation.time	Time	start time - end time	*ISO : hh.mm.ss
Reservation.method	String	none	Variable-Length Character Format
Reservation.recurring	Boolean	true, false	Indicator Format
HelpSession.course	String	none	Variable-Length Character Format
HelpSession.helper	String	none	Variable-Length Character Format
HelpSession.topic	String	none	Variable-Length Character Format
HelpSession.duration	Integer	in minutes	Integer Format
Feedback.rating	Integer	min value: 0, max value: 5	Integer Format
Feedback.comment	String	none	Variable-Length Character Format

Description: Figure 3 contains all data that will be entered into the system by a user. Login information will be entered before any other part of the site can be accessed. Reservation and Help Session information will be entered by users when scheduling or editing sessions. Feedback information will only be entered by student accounts after a help session has been attended.

Appendix A – Agreement Between Customer and Contractor

This section denotes that both the client and the development team have agreed upon the information contained within this document. It will be used as both a guideline and as an end goal in terms of the requirements needed for the application to function to the clients vision.
In the case that an addition or edit be needed after the completion and signing of this document, the change or addition must be agreed upon by both client and development team and included in **Appendix D - Document Additions** with the title of the addition, date, brief description, and signature from both parties.

-Client-
Name: Mr. Christopher Dufour
Date:
Signature:

-Development Team-
Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Client Comments (Continues on next page if needed):

Appendix B – Team Review Sign-off

This section denotes that all members of the In-House Operations development team have reviewed this document and agree on its content and format. If any minor disagreements in content and format are present, they are listed below the development team signatures.

Name: Klei Bendo
Date:
Signature:

Name: Jack Brisson
Date:
Signature:

Name: Alex Landry
Date:
Signature:

Name: Samuel Morse
Date:
Signature:

Name: Aaron Schanck
Date:
Signature:

Name: Forrest Swift
Date:
Signature:

Minor Disagreements in Content and Format (if any):

Appendix C – Document Contributions

This section denotes the contributions of each team member to this document. It includes the sections each member worked on and their percentage contributed in parentheses.

Name: Klei Bendo
Sections worked on (percentage contributed to document):
Sections 3 & 4 (Mockup work, 3.2 descriptions, data validation for section 4)
percentage contributed to document: 17.5%

Name: Jack Brisson
Sections worked on (percentage contributed to document):
Section 3 (Mockup work, 3.2 descriptions)
percentage contributed to document: 10%

Name: Alex Landry
Sections worked on (percentage contributed to document):
Section 3 (Mockup work, 3.2 descriptions)
percentage contributed to document: 10%

Name: Samuel Morse
Sections worked on (percentage contributed to document):
Section 2 & 3 (Section 2.1, Mockup work, 3.2 descriptions)
percentage contributed to document: 25%

Name: Aaron Schanck
Sections worked on (percentage contributed to document):
Sections 1, 2, 3, & 4 (Section 1, Section 2 intro, Section 3.1, 3.2 descriptions mockup work, Appendices)
percentage contributed to document: 20%

Name: Forrest Swift
Sections worked on (percentage contributed to document):
Sections 3 & 4 (Mockup work, 3.2 descriptions, data validation and intro for section 4)
percentage contributed to document: 17.5%

Appendix D – Document Additions

No Document additions to date.