# Software Design Document

for

# Advising Appointment Scheduling System

Release 0.1

Prepared by AASS Team

# Contents

# List of Figures

# List of Tables

# Revision History

| Date | Description | Revised by |
|---|---|---|
| 04/12/24 | Initial draft | AASS Team |

# 1 INTRODUCTION

## 1.1 Purpose

This software design document describes the architecture, system, and component design of the Advising Appointment Scheduling System. This document shall serve as the blueprint by which the AASS software will be developed. Therefore, the contents of this document are pertinent to all members of the AASS team as well as all stakeholders.

## 1.2 Scope

The AASS shall be responsible for the creation and management of advising meetings between university faculty and students. The AASS shall also provide an interface for administrative users to create, modify, and delete faculty, student, and scheduling data as needed to facilitate proper system performance for other actors.

## 1.3 Overview

See Page 2 for Table of Contents.

## 1.4 Definitions and Acronyms

Table 1.1: Definitions & Acronyms

| Term | Definition |
|---|---|
| Advising Appointment Scheduling System (AASS) | The software product described in this design document |
| Unified Modeling Language (UML) | The modeling standard by which the following diagrams were created |
| Comma Separated Values (CSV) | A common file format that shall be used in the AASS for uploading large amounts of data |
| Object Relational Mapping (ORM) | Programming library that contains implementation of object relational mapping techniques for communication between a database and object oriented programming language |

# 2 SYSTEM OVERVIEW

The AASS is a dedicated, Internet-based software system to better facilitate and organize communication between university faculty and students for advising meetings. Faculty members are allowed to present available times to students for meetings, and students are able to utilize the system to select the available time that works best for both parties. Administrative users are able to edit system information, including the modification, creation, and deletion of student, faculty, and scheduling information.

# 3 SYSTEM ARCHITECTURE

## 3.1 Architectural Design

The high-level architecture of the AASS will follow the model-view pattern. The view shall be responsible for rendering content for the client, handling user input, and communicating with the model to fetch and manipulate application data. The model shall be responsible for communication with the database, storing application data, and communicating with the view for presentation of data to the client. The database shall responsible for persistent storage of application data and accepting requests from the model to create, read, update, and delete data. See Figure 3.1.

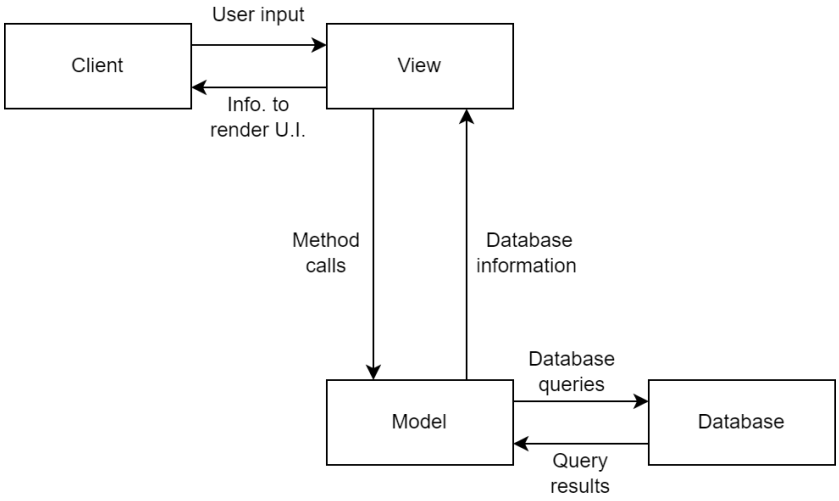


Figure 3.1: Top-Level Architecture Diagram

## 3.2 Decomposition Description

### 3.2.1 View

The view shall be composed of a series of pages that will display information and provide interaction capabilities for the user. As the user interacts with the view, the data stored in the model (and by extension the database) will be updated to reflect the user's actions. See Figure 3.2 for a behavioral diagram detailing how the user may navigate from one page to another.

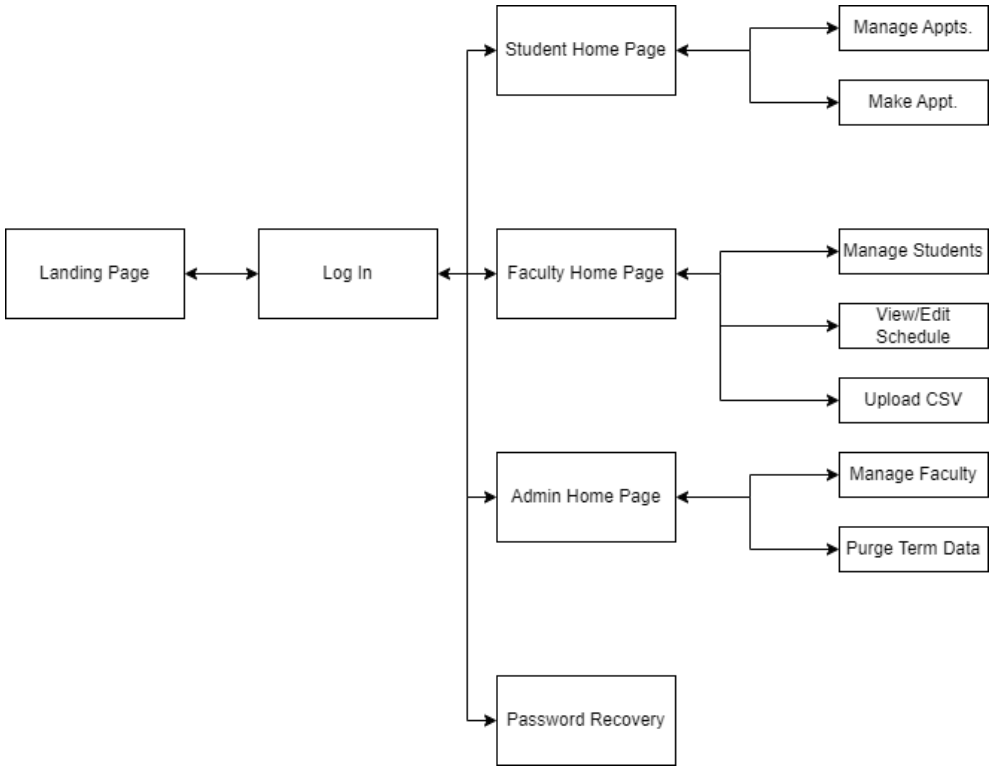


Figure 3.2: View Behavior Diagram

### 3.2.2 Model

The model shall be composed of multiple classes that will represent data stored in the system's database. The four classes are as follows: Profile, Appointment, AdvisorPairing, and Term. The Profile class serves as a representation of a user. See Figure 3.4 for more Profile class information. The Appointment class serves as a representation of all data pertinent to a single scheduled appointment. See Figure 3.5 for more Appointment class information. The AdvisorPairing class serves as a representation of an entry in the advisor table stored in the system's database. See Figure 3.6 for more AdvisorPairing class information. The Term class serves as a representation of a single entry in the term table stored in the system's database. See Figure 3.7 for more Term class information. Since the classes all represent data items stored in the system's database, no classes are directly dependent on another; therefore, a class diagram depicting dependencies has not been used in representing the above classes. An entity-relationship diagram representing the relationships and structure of the described objects is shown in Figure 3.3.

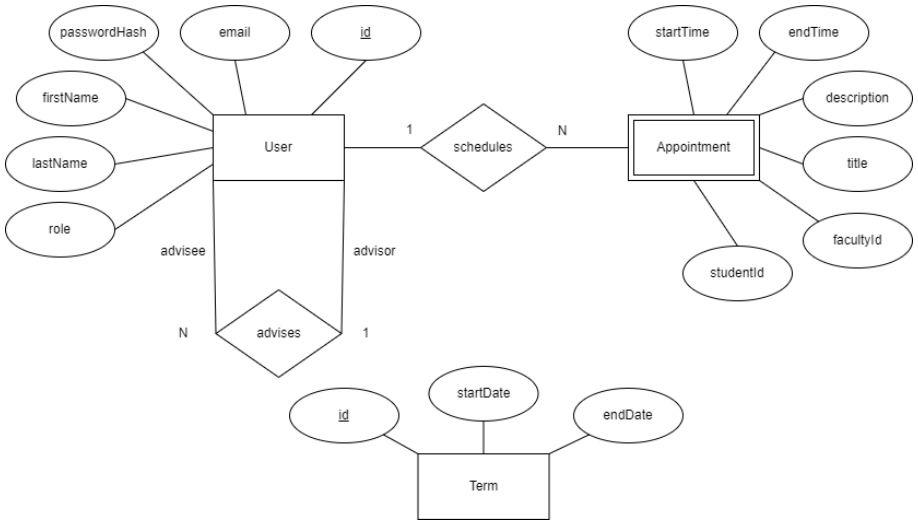


Figure 3.3: Entity-Relationship Diagram for the AASS

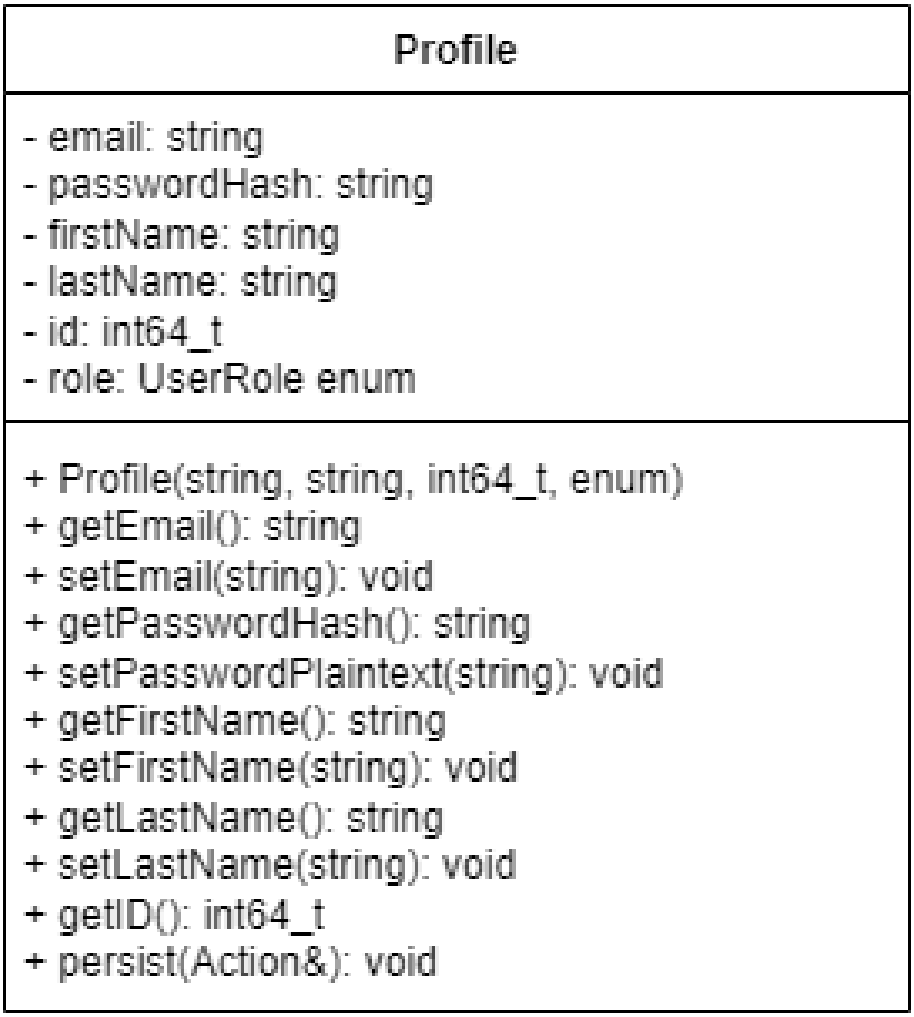


Figure 3.4: Profile Class Diagram


Figure 3.5: Appointment Class Diagram

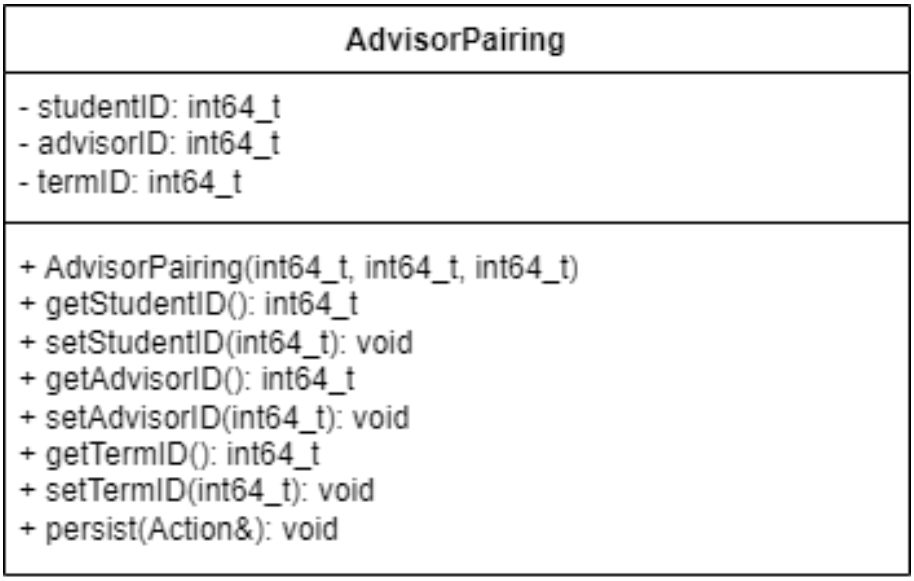


Figure 3.6: AdvisorPairing Class Diagram

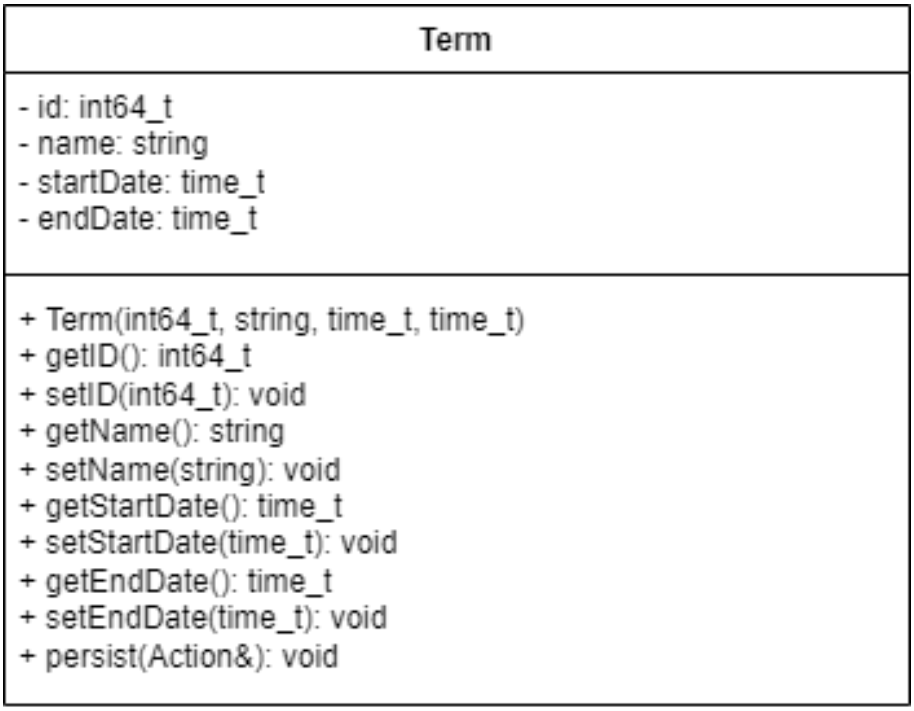


Figure 3.7: Term Class Diagram

## 3.3 Design Rationale

The model-view architecture was selected due to it being utilized by the Wt web framework. Since the Wt framework was specified in the AASS requirements, no other architectural considerations were made.

# 4 DATA DESIGN

## 4.1 Data Description

The information domain of the AASS revolves around advising appointments and the various profiles of possible users. All data structures shall be stored in the database, represented by the model, and processed by the view. See Section 4.2 for a list of data entities.

## 4.2 Data Dictionary

Table 4.1: Data Dictionary

| Item Category | Item | Description |
|---|---|---|
| Profile | | Stores data pertinent to users (Student, Faculty, or Admin) |
| | Email | String that must have a length in the range [9,50] that must end in "@una.edu" |
| | Password Hash | Stores a hashed and salted string corresponding to the user's password |
| | First Name | String that must have a length in the range [1,30] |
| | Last Name | String that must have a length in the range [1,30] |

| | | |
|---|---|---|
| | ID | A positive integer in the range $[0,(2^{63}-1)]$ that must be universally unique |
| | Role | Can be STUDENT, FACULTY, or ADMIN |
| Appointment | | Stores data pertinent to each appointment created |
| | Title | String that must have a length in the range [1,30] |
| | Description | String that must have a length in the range [0,100] |
| | Start Time | Stores a UNIX timestamp representing the start time of the appointment |
| | End Time | Stores a UNIX timestamp representing the end time of the appointment |
| | Student ID | Stores the profile ID of the student that will be attending the meeting (See ID data item) |
| | Faculty ID | Stores the profile ID of the faculty member that will be attending the meeting (See ID data item) |
| Term Table | | Stores data related to academic terms in which appointments can be scheduled |
| | ID | A positive integer in the range $[0,(2^{63}-1)]$ that must be universally unique |
| | Name | Stores a short string with a length in the range [1,30] that is used to identify terms (e.g. "Fall 2024") |

| | | |
|---|---|---|
| | Start Date | Stores a UNIX timestamp representing the start of the term |
| | End Date | Stores a UNIX timestamp representing the end of the term |
| Advisor Table | | Stores IDs to indicate what faculty advisor is assigned to each student |
| | Student ID | Stores the ID of the student in the advisor pairing (See ID data item) |
| | Advisor ID | Stores the ID of the faculty member in the advisor pairing (See ID data item) |
| | Term ID | Stores the ID of the term in which the pairing is valid (See ID data item in Term class) |

# 5 COMPONENT DESIGN

## 5.1 View

The following section describes the functionality of each page that may be presented by the view.

### 5.1.1 Landing Page

This is the first page that all users will encounter. The only functionality provided by the landing page is navigation to the login page. For the design of this page, see Figure 6.1

### 5.1.2 Login Page

The Login Page shall provide two data fields: one for the user's email and the user's password. Once the user has been authenticated, the user will be sent to the home page that corresponds to their role. If the user needs to use the password recovery feature, he may navigate to the password recovery page by clicking the link labeled "Forgot your password?". For the design of this page, see Figure 6.4

### 5.1.3 Student Home Page

The Student Home Page shall provide navigation to two different pages: one to manage appointments and one to make a new appointment. For the design of this page, see Figure 6.12

### 5.1.4 Student: Manage Appointments

The user shall be shown all of their currently booked appointments upon navigating to the page. Upon clicking an appointment, the user shall be able to either edit or delete the given appointment. The main page design is detailed in Figure 6.14, and a focused view of the screen shown to the user upon editing an appointment is shown in Figure 6.15.

### 5.1.5 Student: Make Appointment

The user may make an appointment from this page. The following information must be entered by the user: The date, the start time, and the end time of the appointment. The faculty member shall be selected automatically based on the student's advisor pairing. For the design of this page, see Figure 6.13

### 5.1.6 Faculty Home Page

The Faculty Home Page shall provide navigation to three different pages: one to manage students, one to view and edit the user's schedule, and one to upload CSV files. For the design of this page, see Figure 6.8

### 5.1.7 Faculty: Manage Students

The user may perform three different actions from the Manage Students page: create a new student profile, edit an existing user profile, and delete an existing student profile. For the design of this page, see Figure 6.9

### 5.1.8 Faculty: View/Edit Schedule

The user may perform three different actions from the View/Edit Students page: view all existing appointments in the given term, edit the user's own appointments, and designate certain times in which the user shall be considered unavailable. For the design of this page, see Figure 6.10

### 5.1.9 Faculty: Upload CSV

The user may perform two different actions from the Upload CSV page: upload a CSV file for student profile data and upload a CSV file for scheduling data. If an uploaded CSV file is formatted incorrectly, an error message will display informing the user of the issue. For the design of this page, see Figure 6.11

### 5.1.10 Admin Home Page

The Admin Home Page shall provide navigation to two different pages: one to manage faculty profiles and one to purge term data. For the design of this page, see Figure 6.5

### 5.1.11 Admin: Manage Faculty

The user may perform three different actions from the Manage Faculty page: create new faculty profiles, edit existing faculty profiles, and delete existing faculty profiles. For the design of this page, see Figure 6.6

### 5.1.12 Admin: Purge Term Data

The user may delete all data associated with a term that is not the current term. Therefore, the user will be presented with a list of all existing terms stored in the database, from which they may select any past term to delete all data associated with that term's ID. For the design of this page, see Figure 6.7

### 5.1.13 Password Recovery Page

The Password Recovery Page shall accept a user's email. Once the user has submitted their email, a text box will confirm the submission. For the design of this page, see Figure 6.3

### 5.1.14 Password Change Page

This page shall only be accessible from the link provided by a password recovery request. The page shall accept a new password from the user, ask the user to confirm the new password, and then send a request to edit that user's password to reflect the requested password. Since this page cannot be navigated to during normal use of the software, it is not included in the view's behavioral diagram. For the design of this page, see Figure 6.2

## 5.2 Model

The following tables describe the functionality provided by each class's methods. For the Profile class, see Table 5.1. For the Appointment class, see Table 5.2. For the AdvisorPairing class, see Table 5.3. For the Term class, see Table 5.4.

Table 5.1: Methods for the Profile class

| Method | Parameters | Return Value | Method Description |
|---|---|---|---|
| Profile | User's email, user's plain-text password, id for the user, and user's role | | Constructs the Profile class for the given user. The password provided in the constructor is hashed and salted before being persisted in the database. |
| getEmail | | User's email | Returns email for the specified user |
| setEmail | New email for user | | Sets email for the specified user |
| getPasswordHash | | Hashed password for user | Returns the user's hashed password. |

| | | | |
|---|---|---|---|
| setPasswordPlaintext | Plain text password for user | | Sets user's password. The password is hashed and salted before being stored. |
| getFirstName | | User's first name | Returns the first name of the specified user |
| setFirstName | New first name for the user | | Changes the user's first name to the string passed in |
| getLastName | | User's last name | Returns the last name of the specified user |
| setLastName | New last name for the user | | Changes the user's last name to the string passed in |
| getID | | The user's ID | Returns the user's ID |
| persist | A reference to a templated variable provided by the Wt framework | | Provides the Wt ORM with class fields that need to be stored when the object is persisted as a row in a database table. |

Table 5.2: Methods for the Appointment Class

| Method | Parameters | Return Value | Method Description |
|---|---|---|---|
| Appointment | The title of the appointment, the description of the appointment, the time the appointment should begin, the time the appointment should end, the ID of the attending student, and the ID of the attending faculty member | | This is the only constructor for the Appointment class |
| getTitle | | Name of the appointment | Returns the name of the specified appointment. |
| setTitle | Name of the appointment | | Sets the title of the specified appointment. |
| getDescription | | Description of the appointment | Returns a description of the specified appointment. |
| setDescription | Description of the appointment | | Sets the description of the specified appointment. |
| getStartTime | | The start time of the appointment | Returns the start time of the specified appointment. |

| | | | |
|---|---|---|---|
| setStartTime | The start time of the appointment | | Sets the start time of the specified appointment. |
| getEndTime | | The end time of the appointment | Returns the end time of the specified appointment. |
| setEndTime | The end time of the appointment | | Sets the end time of the specified appointment. |
| getStudentID | | The ID of the student associated with the appointment | Returns the ID of the student associated with the specified appointment. |
| setStudentID | The ID of the student associated with the appointment | | Sets the ID of the student associated with the specified appointment. |
| getFacultyID | | The ID of the faculty member associated with the appointment | Returns the ID of the faculty member associated with the specified appointment. |
| setFacultyID | The ID of the faculty member associated with the appointment | | Sets the ID of the faculty member associated with the specified appointment. |

| | | | |
|---|---|---|---|
| persist | A reference to a templated variable provided by the Wt framework | | Provides the Wt ORM with class fields that need to be stored when the object is persisted as a row in a database table. |

Table 5.3: Methods for the AdvisorPairing Class

| Method | Parameters | Return Value | Method Description |
|---|---|---|---|
| AdvisorPairing | The student's ID, the faculty member's ID, and the term ID | | This is the only constructor for the AdvisorPairing class. |
| getStudentID | | The student ID | Returns the student ID stored in the object |
| setStudentID | The intended student ID | | Changes the student ID to the value passed in |
| getAdvisorID | | The faculty ID | Returns the student ID stored in the object |
| setAdvisorID | The intended faculty ID | | Changes the faculty ID to the value passed in |
| getTermID | | The term ID | Returns the ID of the term in which the advisor pairing is valid |
| setTermID | The intended term ID | | Changes the term ID to the value passed in |
| persist | A reference to a templated variable provided by the Wt framework | | Provides the Wt ORM with class fields that need to be stored when the object is persisted as a row in a database table. |

Table 5.4: Methods for the Term Class

| Method | Parameters | Return Value | Method Description |
|---|---|---|---|
| Term | The unique ID for the term, the name of the term, the term's start date, and the term's end date | | This is the only constructor for the Term class |
| getID | | The ID associated with the term | Returns the term ID |
| setID | The intended ID for the term | | Changes the term ID to the value passed in |
| getName | | The name associated with the term | Returns the term name |
| setName | The intended name for the term | | Changes the term name to the string passed in |
| getStartDate | | The start date for the term | Returns the term's start date |
| setStartDate | The intended start date for the term | | Changes the term's start date to the value passed in |
| getEndDate | | The end date for the term | Returns the term's end date |

| | | | |
|---|---|---|---|
| setEndDate | The intended end date for the term | | Changes the term's end date to the value passed in |
| persist | A reference to a templated variable provided by the Wt framework | | Provides the Wt ORM with class fields that need to be stored when the object is persisted as a row in a database table. |

# 6 HUMAN INTERFACE DESIGN

## 6.1 Overview of User Interface

## 6.2 Screen Designs


Figure 6.1: AASS Landing Page

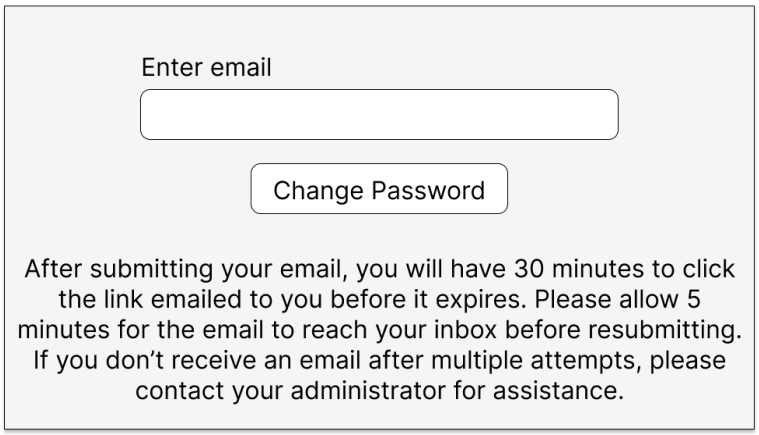


Figure 6.2: Password Change Page

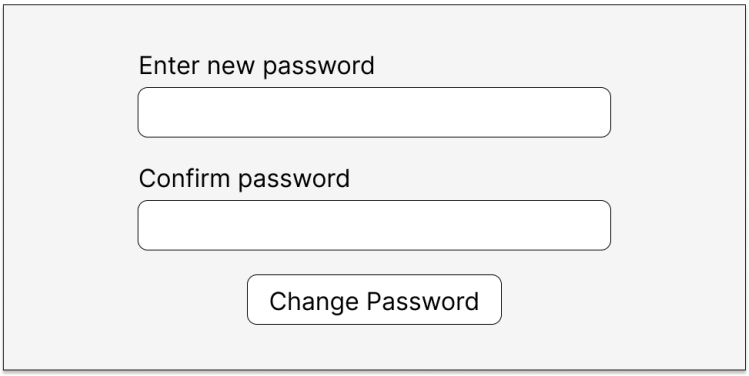


Figure 6.3: Password Recovery Page

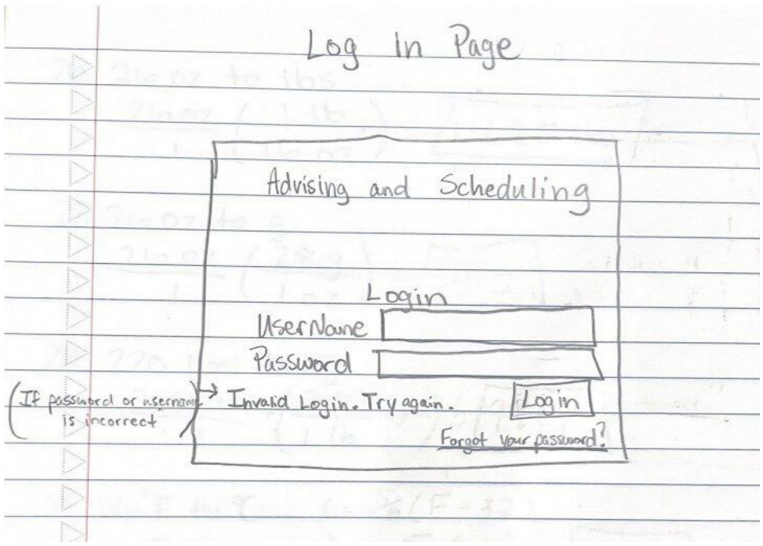


Figure 6.4: Log-In Page

### 6.2.1 Admin UI Designs

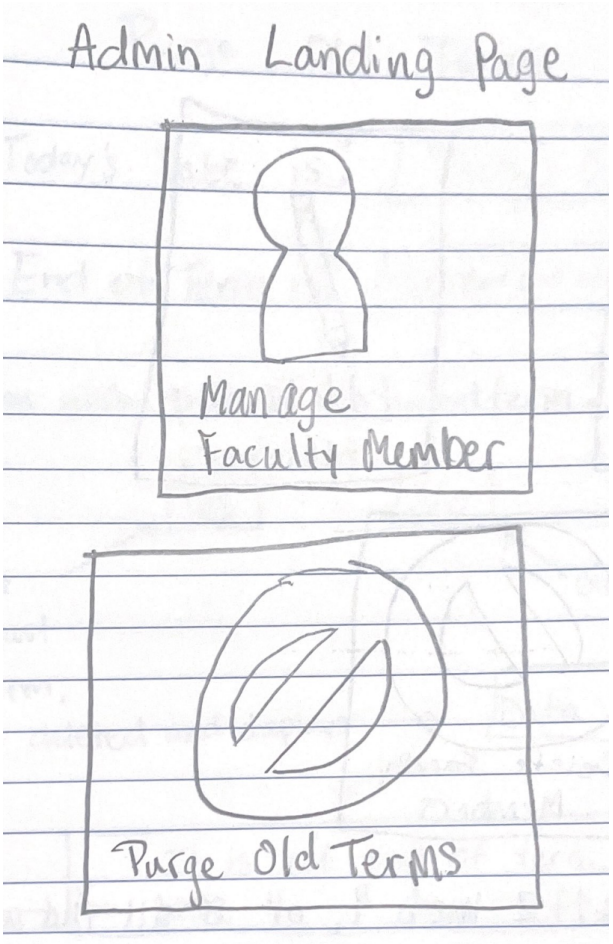


Figure 6.5: Admin Home Page

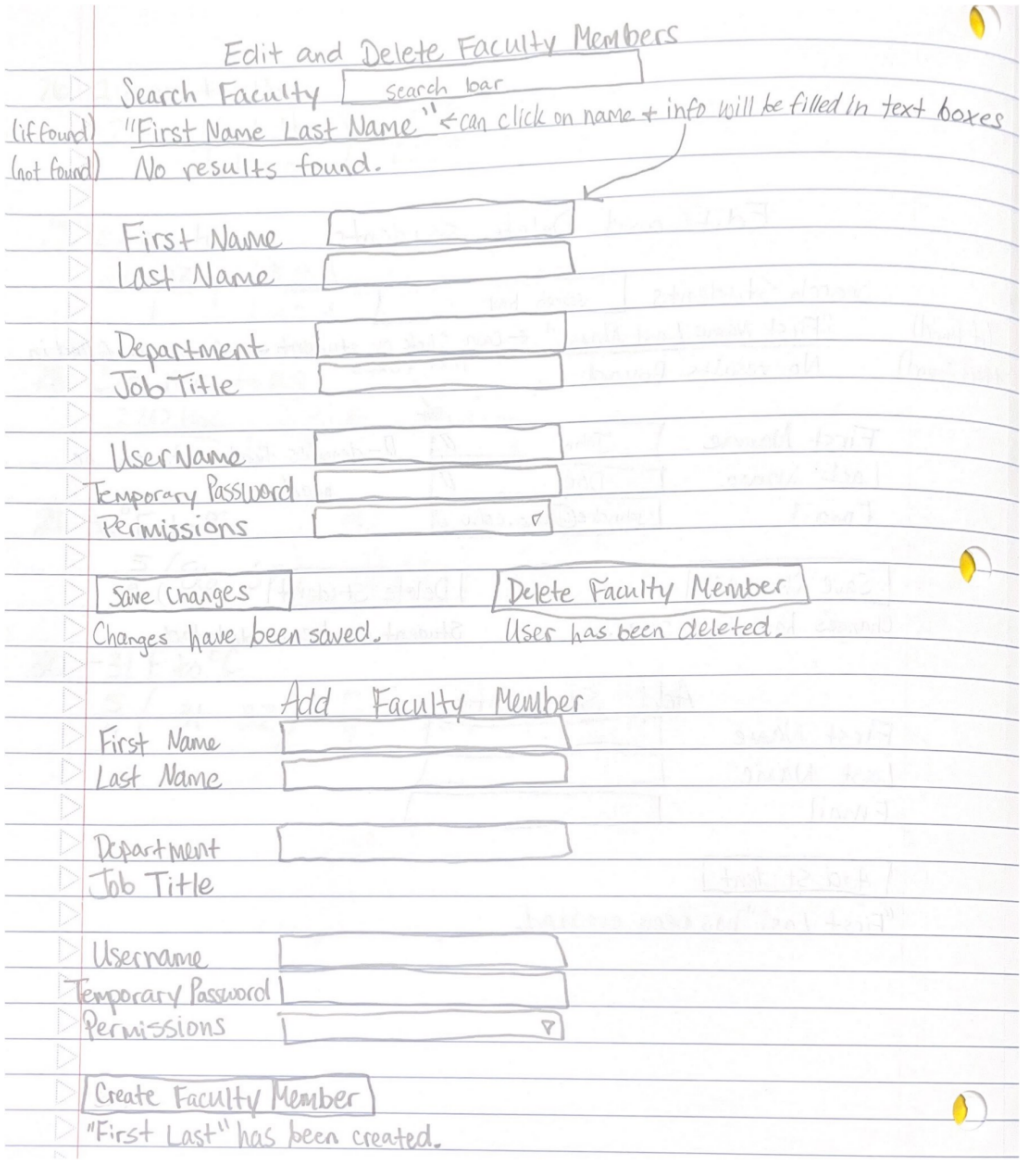


Figure 6.6: Manage Faculty Page

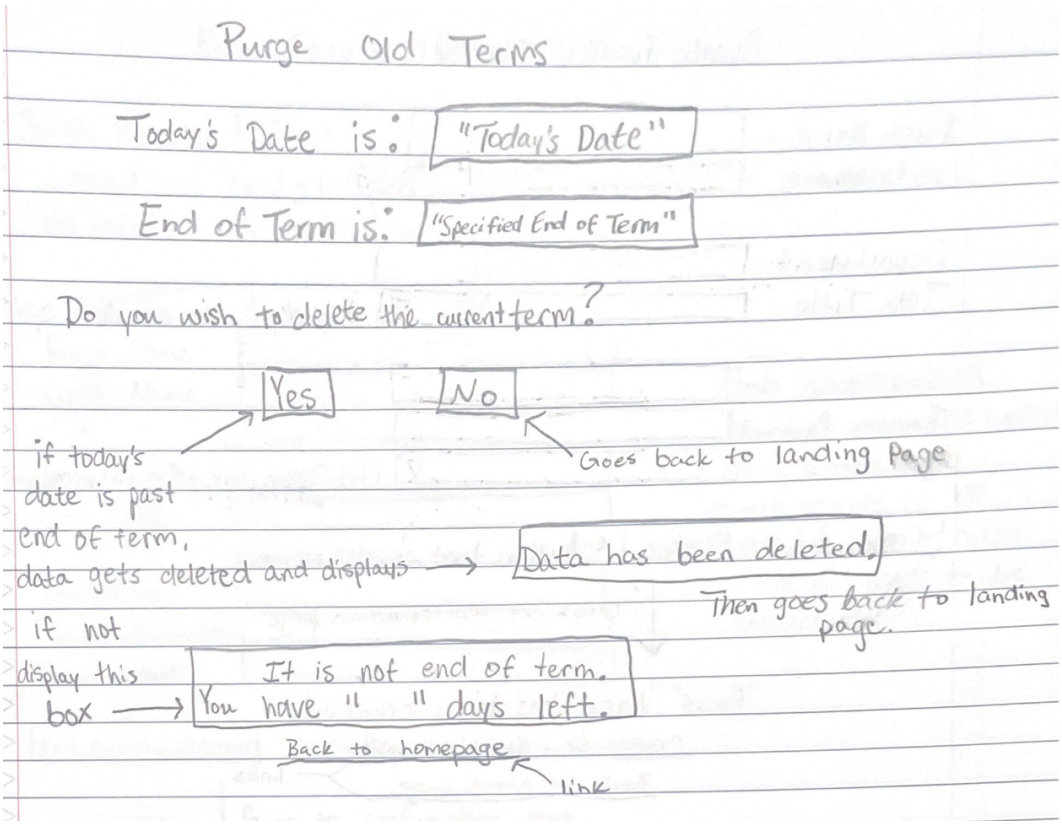


Figure 6.7: Purge Old Term Data

### 6.2.2 Faculty UI Designs

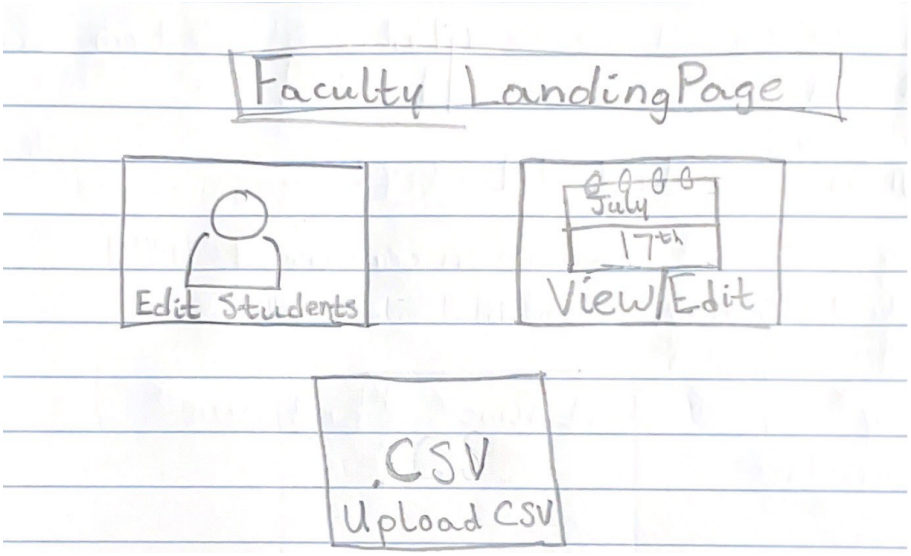


Figure 6.8: Faculty Home Page

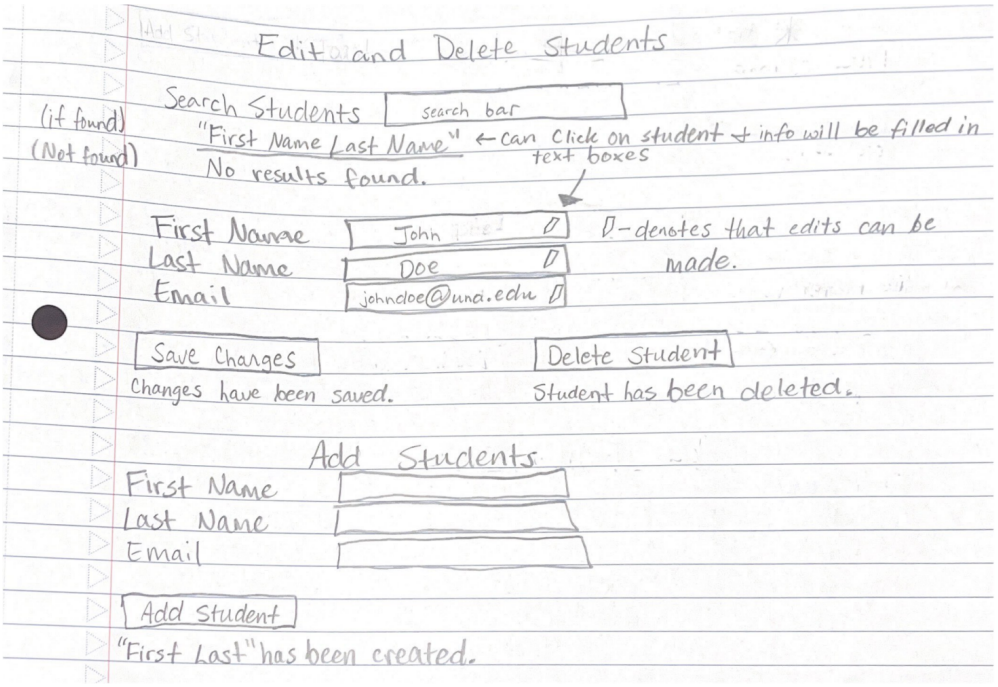


Figure 6.9: Student Change Related Buttons

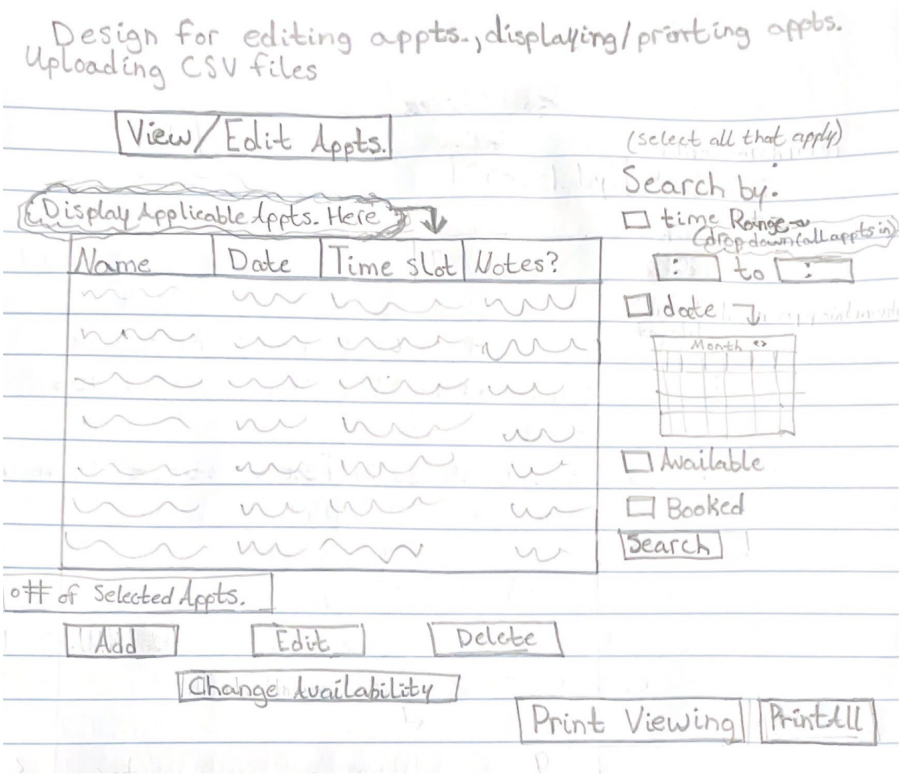


Figure 6.10: View/Edit/Print Appointments Page Appearance

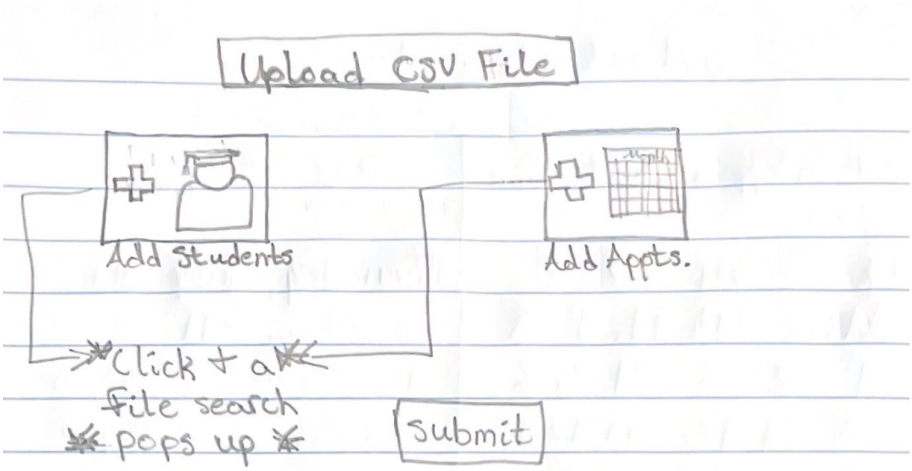


Figure 6.11: Covers the Bulk Upload of Student Info and Bulk Upload of Available Appointments Requirements.

### 6.2.3 Student UI Designs

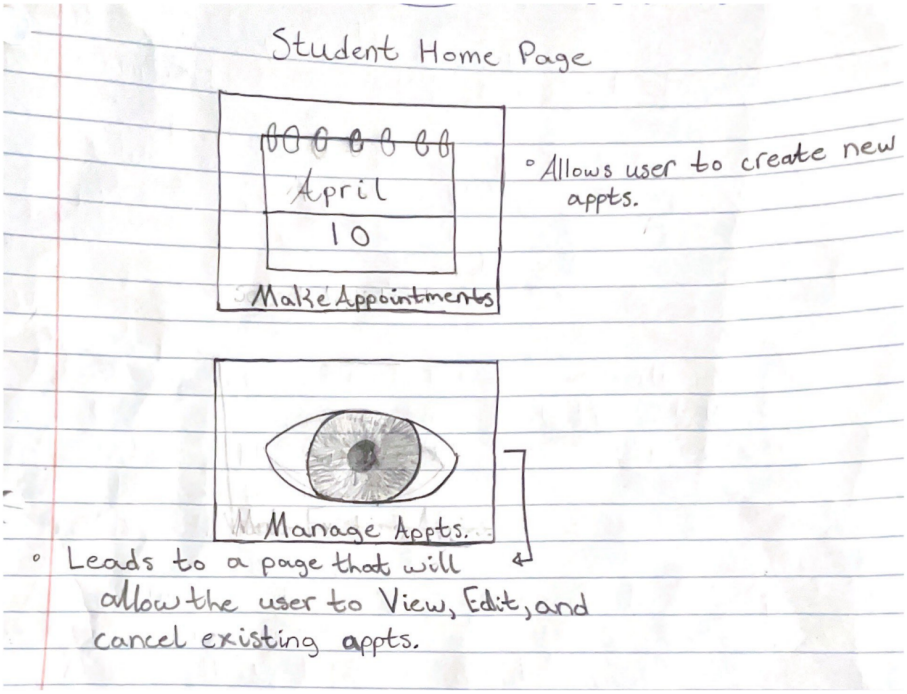


Figure 6.12: Student Home Page

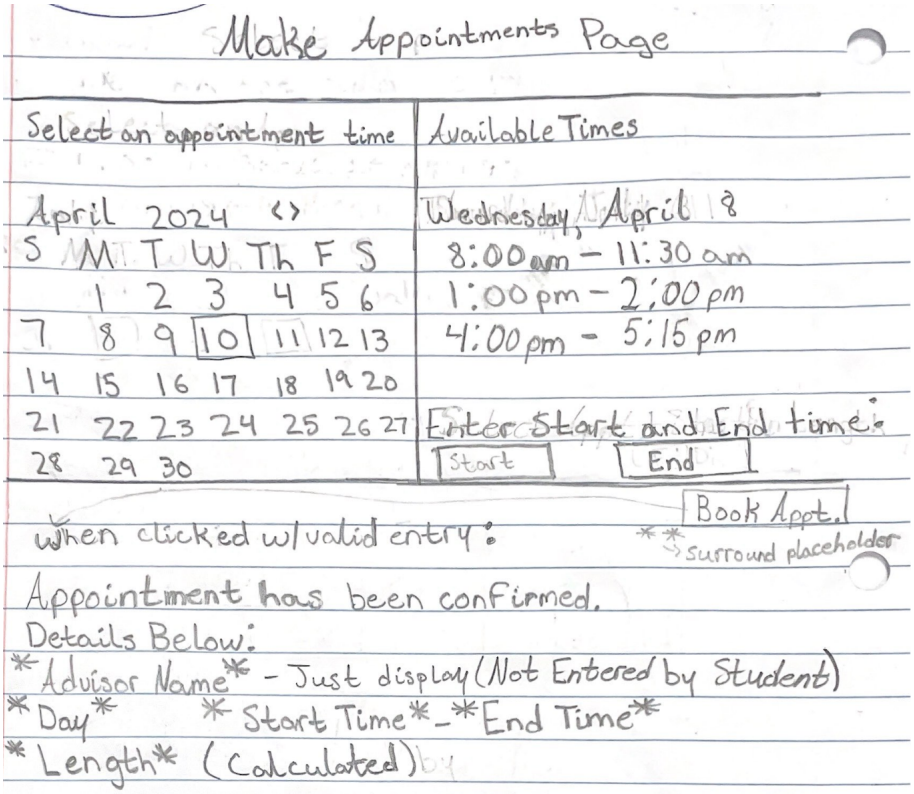


Figure 6.13: Select Appointment Page

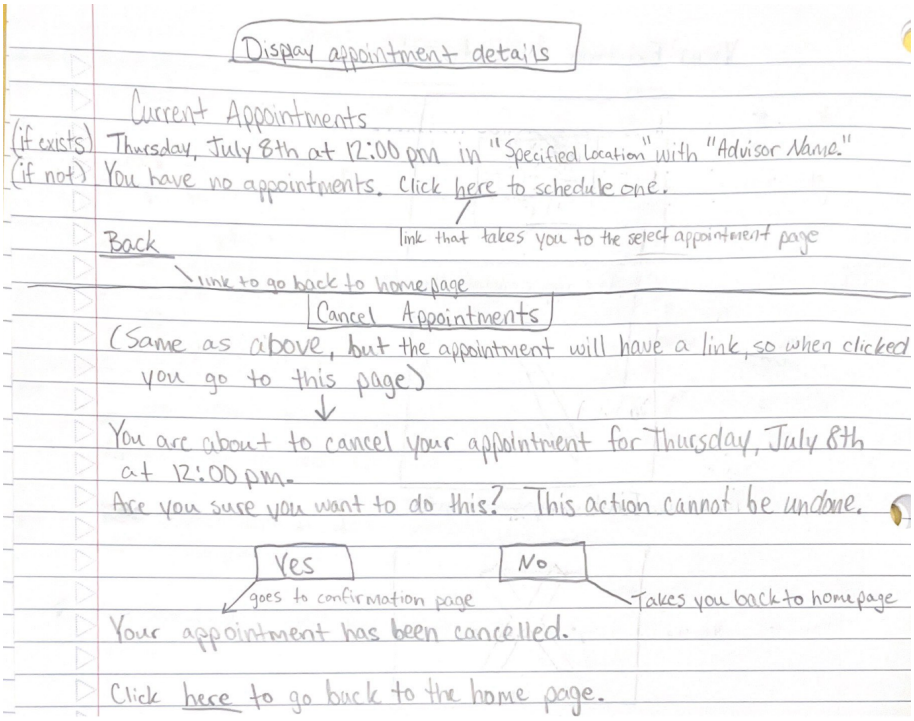


Figure 6.14: Display Appointment Details and Cancellation of Appointments Pages

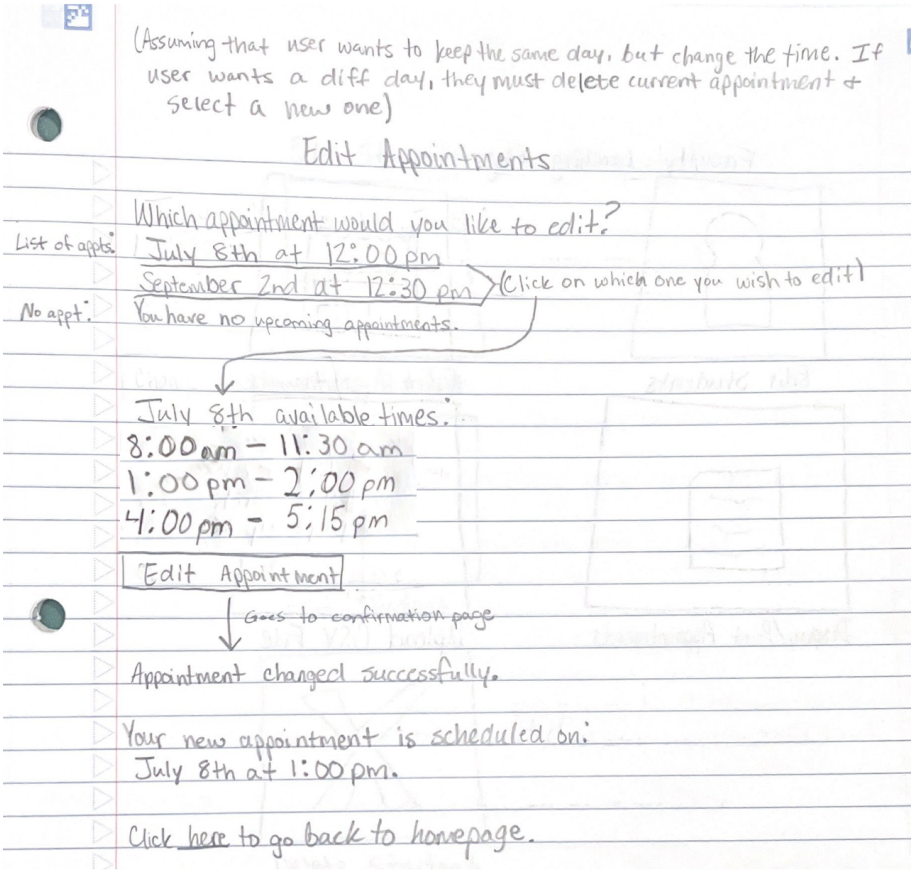


Figure 6.15: Edit Appointments Page

# 7 REQUIREMENTS TRACEABILTY MATRIX

Table 7.1 assigns abbreviated identification numbers to all requirements for use in the Requirements Traceability Matrix:

Table 7.1: Requirements Identification

| ID | Requirement |
|---|---|
| R1 | Student Authentication |
| R2 | Student Password Recovery |
| R3 | Faculty Authentication |
| R4 | Faculty Password Recovery |
| R5 | Admin Authentication |
| R6 | Admin Password Recovery |
| R7 | Student Appointment Creation |
| R8 | Student Appointment Viewing |
| R9 | Student Appointment Editing |
| R10 | Student Appointment Deletion |
| R11 | Bulk Upload of Student Info |
| R12 | Bulk Upload of Available Appointments |
| R13 | Student Notification System |
| R14 | Faculty Viewing All Appointments |
| R15 | Faculty Viewing Appointments by Date |
| R16 | Faculty Viewing Entire Schedule |
| R17 | Faculty Marking Times Unavailable |
| R18 | Faculty Adding Students |

| | |
|---|---|
| R19 | Faculty Editing Students |
| R20 | Faculty Deleting Students |
| R21 | Faculty Adding Appointments |
| R22 | Faculty Editing Appointments |
| R23 | Faculty Deleting Appointments |
| R24 | Admin Adding Faculty |
| R25 | Admin Editing Faculty |
| R26 | Admin Deletion of Faculty |
| R27 | Purging Old Data |

Table 7.2 assigns abbreviated identification numbers to all product features for use in the Requirements Traceability Matrix:

Table 7.2: Features Identification

| ID | Feature |
|---|---|
| F1 | Landing Page |
| F2 | Log-In Page |
| F3 | Admin Home Page |
| F4 | Admin Manage Faculty Page |
| F5 | Admin Purge Term Data Page |
| F6 | Student Home Page |
| F7 | Student Manage Appointments Page |
| F8 | Student Make Appointment Page |
| F9 | Faculty Home Page |
| F10 | Faculty Manage Students Page |
| F11 | Faculty View/Edit Schedule Page |
| F12 | Faculty Upload CSV Page |
| F13 | Password Recovery Page |
| F14 | Password Change Page |
| F15 | Profile Class |
| F16 | Appointment Class |
| F18 | Term Class |
| F19 | AdvisorPairing Class |

Tables 7.3 and 7.4 form the Requirements Traceability Matrix using the above IDs.

Table 7.3: Requirements Traceability Matrix for R1 - R14

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | X | | X | | X | | | | | | | | | |
| F2 | X | | X | | X | | | | | | | | | |
| F3 | | | | | | | | | | | | | | |
| F4 | | | | | | | | | | | | | | |
| F5 | | | | | | | | | | | | | | |
| F6 | | | | | | | X | X | X | X | | | | |
| F7 | | | | | | | | X | X | X | | | | |
| F8 | | | | | | | X | | | | | | | |
| F9 | | | | | | | | | | | X | X | | X |
| F10 | | | | | | | | | | | | | | |
| F11 | | | | | | | | | | | | | | X |
| F12 | | | | | | | | | | | X | X | | |
| F13 | | X | | X | | X | | | | | | | | |
| F14 | | X | | X | | X | | | | | | | | |
| F15 | X | | X | | X | | X | | | | X | | X | X |
| F16 | | | | | | | X | X | X | X | | X | X | X |
| F17 | | X | | X | | X | | | | | | | | |
| F18 | | | | | | | | | | | | | | |
| F19 | | | | | | | X | X | X | | | | | X |

Table 7.4: Requirements Traceability Matrix for R15 - R27

| | R15 | R16 | R17 | R18 | R19 | R20 | R21 | R22 | R23 | R24 | R25 | R26 | R27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | | | | | | | | | | | | | |
| F2 | | | | | | | | | | | | | |
| F3 | | | | | | | | | | X | X | X | X |
| F4 | | | | | | | | | | X | X | X | |
| F5 | | | | | | | | | | | | | X |
| F6 | | | | | | | | | | | | | |
| F7 | | | | | | | | | | | | | |
| F8 | | | | | | | | | | | | | |
| F9 | X | X | X | X | X | X | X | X | X | | | | |
| F10 | | | | X | X | X | | | | | | | |
| F11 | X | X | X | | | | | | | | | | |
| F12 | | | | | | | | | | | | | |
| F13 | | | | | | | | | | | | | |
| F14 | | | | | | | | | | | | | |
| F15 | X | X | | X | X | X | | | | X | X | X | |
| F16 | X | X | X | | | | X | X | X | | | | |
| F17 | | | | | | | | | | | | | |
| F18 | | | | | | | | | | | | | X |
| F19 | | | X | X | X | X | X | X | X | X | X | | |