**Software Requirements Specification Template**

The following annotated template shall be used to complete the Software Requirements Specification (SRS). The instructor must approve any modifications to the overall structure of this document.

**Template Usage:**

Text contained within angle brackets (‘<’, ‘>’) shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either ‘Smart Home’ or ‘Sensor Network’.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

**Acknowledgements:**

Sections of this document are based upon the IEEE Guide to Software Requirements Specification (ANSI/IEEE Std. 830-1984).

Course Advising Application

Software Requirements Specification

1.0

10th February , 2020

David Herrington

Team Leader/Programmer

Simran Bhamra

Designer/Programmer

Michael Fishler

Programmer/Researcher

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 3/2/2020 | Version 0.0.1 | David, Micheal, Simran | User stories completed |
| 10/2/2020 | Version 1 | David, Micheal, Simran | Completed version 1 |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
| Simran Bhamra | Simran Bhamra | Programmer/ Designer | 10/2/2020 |
| David Herrington | David Herrington | Team Leader/Programmer | 10/2/2020 |
| Michael Fishler | Michael Fishler | Programmer/Researcher | 10/2/2020 |

**Table of Contents**

[**1. Introduction**](#_heading=h.8uky3y4vjkig) **9**

[1.1 Purpose](#_heading=h.3znysh7) 9

[1.2 Scope](#_heading=h.2et92p0) 9

[1.3 Definitions, Acronyms, and Abbreviations](#_heading=h.tyjcwt) 9

[1.4 Overview](#_heading=h.3dy6vkm) 9

[**2. General Description**](#_heading=h.1t3h5sf) **9**

[2.1 Product Perspective](#_heading=h.4d34og8) 10

[2.2 Product Functions](#_heading=h.17dp8vu) 10

[2.3 Users and Characteristics](#_heading=h.26in1rg) 10

[2.4 General Constraints](#_heading=h.35nkun2) 10

[2.5 Assumptions and Dependencies](#_heading=h.1ksv4uv) 10

[2.6 Operating Environment](#_heading=h.44sinio) 10

[**3. Specific Requirements**](#_heading=h.2jxsxqh) **11**

[3.1 External Interface Requirements](#_heading=h.z337ya) 11

[3.1.1 User Interfaces](#_heading=h.3j2qqm3) 11

[3.1.2 Hardware Interfaces](#_heading=h.1y810tw) 11

[3.1.3 Software Interfaces](#_heading=h.4i7ojhp) 11

[3.1.4 Communications Interfaces](#_heading=h.2xcytpi) 11

[3.2 Functional Requirements](#_heading=h.1ci93xb) 11

[3.2.1 Log in via Google Account](#_heading=h.3whwml4) 11

[3.2.1.1 Description and Priority](#_heading=h.xqx774wvw4gl) 11

[3.2.1.2 Stimulus/Response Sequences](#_heading=h.v20xpa9io2ji) 11

[3.2.1.3 Functional Requirements](#_heading=h.dr6ht3n13l0p) 11

[3.2.2 Students can see and book appointments with advisors in their major/minor.](#_heading=h.3z49ojkkxury) 12

[3.2.2.1 Description and Priority](#_heading=h.95qm6wno561y) 12

[3.2.2.2 Stimulus/Response Sequences](#_heading=h.rn4w3450nxlt) 12

[3.2.2.3 Functional Requirements](#_heading=h.il6iwpb3h552) 12

[3.2.3 Exporting mock schedule to PDF.](#_heading=h.yzo1vcigss1f) 12

[3.2.3.1 Description and Priority](#_heading=h.3pcc5rvny7fc) 12

[3.2.3.2 Stimulus/Response Sequences](#_heading=h.m17y9rqtnifm) 12

[3.2.3.3 Functional Requirements](#_heading=h.kyrx5jppnx39) 13

[3.2.4 Generating course suggestions based on major/minor.](#_heading=h.siye8d4cc65n) 13

[3.2.4.1 Description and Priority](#_heading=h.8v9aoeqd2paq) 13

[3.2.4.2 Stimulus/Response Sequences](#_heading=h.pyhun9cebwty) 13

[3.2.4.3 Functional Requirements](#_heading=h.k2zl7xc4yuk4) 13

[3.2.5 Push appointment reminders to students and advisors.](#_heading=h.4bv06ctkmlpn) 13

[3.2.5.1 Description and Priority](#_heading=h.36xf9ruzc3sl) 13

[3.2.5.2 Stimulus/Response Sequences](#_heading=h.1fjj4tanf9yr) 13

[3.2.5.3 Functional Requirements](#_heading=h.gdubz9lb2zw2) 13

[3.2.6 Creating appointment availability](#_heading=h.m9qv0b5nwhxc) 13

[3.2.6.1 Description and Priority](#_heading=h.2cpxxlwevdwf) 14

[3.2.6.2 Stimulus/Response Sequences](#_heading=h.23434o1l9te7) 14

[3.2.6.3 Functional Requirements](#_heading=h.os7t81xp1672) 14

[3.2.7 Deletion of appointments](#_heading=h.4oifh3x8ao5u) 14

[3.2.7.1 Description and Priority](#_heading=h.1a4d6o5sykiu) 14

[3.2.7.2 Stimulus/Response Sequences](#_heading=h.braulvw1tcmr) 14

[3.2.7.3 Functional Requirements](#_heading=h.ijstgx7v9uyb) 14

[3.2.8 Creating a mock schedule.](#_heading=h.lctf4qhxcrbk) 14

[3.2.8.1 Description and Priority](#_heading=h.t1wvd37hicrc) 14

[3.2.8.2 Stimulus/Response Sequences](#_heading=h.tik8fn5mwh7) 14

[3.2.8.3 Functional Requirements](#_heading=h.2jcxbzdh8ofc) 15

[3.2.9 Students should only see availability for advisors in their department.](#_heading=h.lndlramt4rm9) 15

[3.2.9.1 Description and Priority](#_heading=h.j55b5qh3o49q) 15

[3.2.9.2 Stimulus/Response Sequences](#_heading=h.pkuybpn3ypbh) 15

[3.2.9.3 Functional Requirements](#_heading=h.hh0h9obm09j4) 15

[3.2.10 Students will be able to select classes they have already taken.](#_heading=h.2pf4az2ew6rg) 15

[3.2.10.1 Description and Priority](#_heading=h.c5jx3ura8vot) 15

[3.2.10.2 stimulus/Response Sequences](#_heading=h.j41hclh1ktso) 15

[3.2.10.3 Functional Requirements](#_heading=h.e7ab2yv1wc5) 15

[3.2.11 Advisors will be able to see appointments scheduled for a day/week.](#_heading=h.ixhm0ilxllz5) 15

[3.2.11.1 Description and Priority](#_heading=h.tebruye24n7o) 16

[3.2.11.2 Stimulus/Response Sequences](#_heading=h.ynbcxssajqpn) 16

[3.2.11.3 Functional Requirements](#_heading=h.y0ff0oyn9ch7) 16

[3.2.12 Modification of appointments](#_heading=h.uonkg8x4xlgb) 16

[3.2.12.1 Description and Priority](#_heading=h.m96ow6qi0hh) 16

[3.2.12.2 Stimulus/Response Sequences](#_heading=h.frj236nevhdy) 16

[3.2.12.3 Functional Requirements](#_heading=h.2upjh637c29w) 16

[3.3 Use Cases](#_heading=h.3as4poj) 16

[3.3.1 Use Case #1](#_heading=h.1pxezwc) 16

[3.3.2 Use Case #2](#_heading=h.2p2csry) 17

[3.3.3 Use Case #3](#_heading=h.mdi17v6ceqk0) 18

[3.3.4 Use Case #4](#_heading=h.fbbkd4d9xj3w) 18

[3.3.5 Use Case #5](#_heading=h.ggso27nsbcad) 19

[3.3.6 Use Case #6](#_heading=h.dyf0lz7g6uru) 20

[3.3.7 Use Case #7](#_heading=h.2ddajbocxkz5) 20

[3.3.8 Use Case #8](#_heading=h.6ey5wc5y9tbn) 21

[3.3.9 Use Case #9](#_heading=h.8o7ph7hbaev1) 22

[Students should only see availability for advisors in their department.](#_heading=h.s4rjaet472ax) 22

[3.3.10 Use Case #10](#_heading=h.e09fy9adwgc2) 22

[Creating a mock schedule.](#_heading=h.wvzzrjflh9ln) 22

[3.3.11 Use Case #11](#_heading=h.7seq74g36xy2) 23

[Push appointment reminders to students and advisors.](#_heading=h.l4tgbj9ac650) 23

[3.3.10 Use Case #12](#_heading=h.l99jy7c2grah) 23

[Students can see and book appointments with advisors in their major/minor.](#_heading=h.el77yxpthxf0) 23

[3.4 Non-Functional Requirements](#_heading=h.147n2zr) 24

[3.5.1 Performance](#_heading=h.3o7alnk) 24

[3.5.2 Reliability](#_heading=h.23ckvvd) 24

[3.5.3 Availability](#_heading=h.ihv636) 24

[3.5.4 Security](#_heading=h.32hioqz) 24

[3.5.5 Maintainability](#_heading=h.1hmsyys) 25

[3.5.6 Portability](#_heading=h.41mghml) 25

[3.5 Design Constraints](#_heading=h.2grqrue) 25

[3.6 Logical Database Requirements](#_heading=h.vx1227) 25

[**4. Analysis Models**](#_heading=h.1v1yuxt) **25**

[Sequence Diagrams](#_heading=h.4f1mdlm) 25

[**5. Change Management Process**](#_heading=h.3tbugp1) **27**

[**References**](#_heading=h.28h4qwu) **27**

[**Appendices**](#_heading=h.nmf14n) **27**

[A.1 Appendix 1](#_heading=h.37m2jsg) 27

[A.2 Appendix 2](#_heading=h.1mrcu09) 27

**List of Figures**

[Figure 1: Data Flow Diagram of Use Case 1](#_heading=h.2u6wntf) 23

[Figure 2: Data Flow Diagram of Use Cases 3 & 7](#_heading=h.2u6wntf) 24

[Figure 3: Data Flow Diagram of Use Cases 4 & 8](#_heading=h.2u6wntf) 24

[Figure 4: Data Flow Diagram of Use Case 5](#_heading=h.2u6wntf) 25

[Figure 5: Data Flow Diagram of Use Cases 6 & 10](#_heading=h.2u6wntf) 25

**List of Tables**

[U](#_heading=h.3as4poj)se Case Tables 14

Table 1: [Use Case #1](#_heading=h.1pxezwc) 15

Table 2: [Use Case #2](#_heading=h.2p2csry) 15

Table 3: [Use Case #3](#_heading=h.mdi17v6ceqk0) 16

Table 4: [Use Case #4](#_heading=h.fbbkd4d9xj3w) 16

Table 5: [Use Case #5](#_heading=h.ggso27nsbcad) 17

Table 6: [Use Case #6](#_heading=h.dyf0lz7g6uru) 18

Table 7: [Use Case #7](#_heading=h.2ddajbocxkz5) 18

Table 8: [Use Case #8](#_heading=h.6ey5wc5y9tbn) 19

Table 9: [Use Case #9](#_heading=h.8o7ph7hbaev1) 20

Table 10: [Use Case #10](#_heading=h.e09fy9adwgc2) 20

Table 11: [Use Case #11](#_heading=h.7seq74g36xy2) 21

Table 12: [Use Case #12](#_heading=h.l99jy7c2grah) 21

# 1. Introduction

In this Software Requirement Specification (SRS) document, we describe the features, requirements, use cases, and data flows of our Course Advising Application. Our application will provide a more efficient way for students and advisors to schedule appointments, create mock schedules, and improve overall communication.

## 1.1 Purpose

The purpose of this SRS document is to outline all of the software requirements for our Course Advising Application. The intended audience for this document is the university who wishes to adopt our program, and the advisors or students using the program.

## 1.2 Scope

* The software product will be a program that simplifies the student advising process.
* At launch this program will only be suited for mathematics and computer science majors.
* The software will allow the students to schedule appointments with advisors from their major.
* Will allow mock schedules to be created.
* Will generate course suggestions based on the students major when creating the mock schedules.
* Advisors will be able to set available time slots for appointments.
* Students will be able to select courses they have already taken, which will take them out of the course suggestions list.
* Advisors will be able to see scheduled appointments in a calendar format.

## 1.3 Definitions, Acronyms, and Abbreviations

CAA - Course Advising Application.

SQL - Structured Query Language

## 1.4 Overview

This document is broken down into; functional requirements, use cases, and data flow diagrams. The functional requirement section describes the individual features we plan to use to make the program function. Use cases describe how we plan to implement these features and data flow diagrams are a visual representation of these use cases.

# 2. General Description

Our product is aimed to make the advising process more efficient for university students as well as advisors. Our web application will make scheduling advising appointments easier for the students while automatically organizing these appointments into an easy to read schedule for the advisor. Along with setting up appointments, the application will assist students with choosing courses by generating course suggestions. These suggestions are based on the students major/minor and the selected classes already taken by the student. These course suggestions can be organized into a mock schedule which can be sent to the advisor for approval.

## 2.1 Product Perspective

The idea to create this product was inspired by DeSales University’s current system for setting up advising appointments. The current system for setting up advising appointments is for the student to sign their name next to an available time slot, on a sheet of paper taped to the advisors office door. Our product aims to make this process more efficient by creating a web application with added features to improve the advising experience for both students and advisors.

## 2.2 Product Functions

* Advisor Functions
  + Create time slots where students can schedule appointments.
  + Generate calendar with all currently scheduled appointments.
* Student Functions
  + Set up Appointments.
  + Enter classes they have already taken.
  + Generate course suggestions based on selected major/minor.
  + Only see availability for advisors in their department.
* Universal Functions
  + Log in with a Google account.
  + Export mock schedule to PDF.
  + Delete appointments.
  + Push appointment reminders.
  + Modification of appointments.

## 2.3 Users and Characteristics

The users for this application will mainly be for mathematics and computer science majors as well as the advisors for these two departments. The students will be at a college education level and will be granted a lower level access to the program. The main abilities of the student will be to set up appointments with advisors, and create mock schedules with course suggestions from the application. The advisors will be granted a higher level access to the application, giving them the ability to set up available time slots for students, as well as, see a calendar with all scheduled appointments where the advisor can view, modify, or delete any of the appointments.

## 2.4 General Constraints

For this program our first main constraint will be time. We plan to implement features for mathematics and computer science departments only due to the limited amount of time we have to complete this program. The second main constraint will be money. We cannot use any third party services that will require a monthly or one time fee.

## 2.5 Assumptions and Dependencies

For this product we are assuming that the web platform will be available for our product when it is released. Our product depends on the Firebase web development platform to be reliable when hosting our product. If we run into an issue using Firebase to host our web application, it will cause us to seek out a new hosting solution.

## 2.6 Operating Environment

This program will be built for the web environment. This way students and advisors will not have to download any software to their personal devices. The program will also be created using Firebase, this improves the reliability of our app and allows the app data to be stored on the cloud.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 Log in via Google Account

**ID: FR1**

#### 3.2.1.1 Description and Priority

This feature will allow both students and advisors to log in with a Google account or email link, this will

simplify the sign in process. This feature is high priority (9) as a user must sign in to use the program and use the

push notifications.

#### 3.2.1.2 Stimulus/Response Sequences

**Option 1 - Google login**

Stimulus: User opens the application and is prompted to “Sign in with Google”.

Response: The user is taken to the Google sign in page and asked to fill in their details.

Stimulus: The user is authenticated with the application via the Google login.

Response: The user is taken to their homepage, if this is the first time the user has logged on then the user will be

taken to the course survey or professor home based on the account type they have selected.

**Option 2 - Email login**

Stimulus: User opens the application and does not have a Google account and selects “Sign-in with email”.

Response: The user enters their email and a link is sent to the email account.

Stimulus: The user logins into the account via the link in their account and is sent to the survey or the professor

home based on the account type they have selected.

#### 3.2.1.3 Functional Requirements

* In order for the end user to use this feature they must have a Google account or access to an email address and a stable internet connection.
* To execute the above use cases, Google Firebase authentication must be used both for Google account logins and email link logins.
* There is no case where users will not be able to login as each student and professor is associated with a DeSales University email address. However, unexpected errors in logins will be handled by the Google authenticator.

### 3.2.2 Students can see and book appointments with advisors in their major/minor.

**ID: FR2**

#### 3.2.2.1 Description and Priority

Students are allowed to schedule time during advising week with their advisors for various purposes. Students are only to see advisors within their major/minor, depending on what the student selected as his/her major at the initialization stage. This is a high (9) priority.

#### 3.2.2.2 Stimulus/Response Sequences

Stimulus: User requests to schedule an appointment.

Response: System shows advisor’s availability

Stimulus: User selects an open time slot

Response: System shows confirmation window with additional meeting configuration settings

Stimulus: User enters in reason for and confirms the appointment

#### 3.2.2.3 Functional Requirements

* Point and click UI for carrying out task - this is the only input for this task
* Calendar views for displaying schedule/time slots
* Pop-up windows for confirming appointments
* Text input for entering appointment reason

### 3.2.3 Exporting mock schedule to PDF.

**ID: FR3**

#### 3.2.3.1 Description and Priority

The user should be able to download a PDF copy of their appointments and class schedule. Student users should

be able to download the PDF version of the current schedule that they have made while professor users should be able to download PDF versions of their appointments for the day/week depending on what view they have picked and select students schedules. This function is low priority (3).

#### 3.2.3.2 Stimulus/Response Sequences

**Option 1 - Student schedule download**

Stimulus: The student user clicks to export their class schedule.

Response: The PDF version of the class schedule is downloaded to the device.

**Option 2 - Professor appointment download**

Stimulus: The professor user selects their preferred view (day/week) in the calendar and clicks to export.

Response: The PDF version of the appointment schedule is downloaded to the device.

**Option 3 - Professor student schedule download**

Stimulus: The professor user selects the student they want to view and clicks to export their class schedule.

Response: The PDF version of the appointment schedule is downloaded to the device.

#### 3.2.3.3 Functional Requirements

* To use this feature the user must allow downloads from the web application.
* Software capabilities required are TBD.

### 3.2.4 Generating course suggestions based on major/minor.

**ID: FR4**

#### 3.2.4.1 Description and Priority

After a student inputs his/her major and/or minor, and after the student inputs completed courses, the program will display courses that the student needs to take in order to graduate, and/or courses being offered for the next semester.

#### 3.2.4.2 Stimulus/Response Sequences

Stimulus: User requests to see suggested courses

Response: System displays list of courses that user needs/may like to take.

* This functional requirement depends on the action generated by other functional requirements (inputting major/minor and completed courses.)

#### 3.2.4.3 Functional Requirements

* Point and click UI for selecting feature

### 3.2.5 Push appointment reminders to students and advisors.

**ID: FR5**

#### 3.2.5.1 Description and Priority

Since the user is logging in with their Google account, the user can opt-in for appointment reminders sent to their email the morning before the appointment.

#### 3.2.5.2 Stimulus/Response Sequences

Stimulus: Trigger set to go off morning of the appointment.

Response: Email will be sent to users gmail with appointment reminder information.

#### 3.2.5.3 Functional Requirements

* When making an appointment, we’ll need a check box for sending reminders.
* Program will have to send an email the morning of the appointment with reminder information.

##### 

### 

### 3.2.6 Creating appointment availability

**ID: FR6**

#### 3.2.6.1 Description and Priority

The professor user should be able to set dates and time sessions where they are available for meetings as well as setting a time limit for the meeting. For example, the professor would be able to set each slot as 15 minutes as well as limit how many sessions each student can have. This is a high(9) priority function.

#### 3.2.6.2 Stimulus/Response Sequences

Stimulus: The user enters information via the UI to specifying, dates, time blocks, meeting time limits, and meeting

amount limits they would like.

Response: The software will create a calendar where available appointments become viewable to the students and

they can select a meeting time.

Stimulus: A student user has selected and confirmed an appointment.

Response: The appointment time block on the professors calendar will be shown as confirmed (by the change in

color) and show the students name and student ID.

#### 3.2.6.3 Functional Requirements

* The user and professor must enter/ select valid appointment times.
* The UI will be designed so that the professor cannot enter invalid times or dates with the use of drop down menus.
* If the error occurs that a student user requires more meetings than are allowed by the professor then they will be prompted to privately email the professor.
* If a student cannot make any time slot provided the student user will be prompted by the system to privately email the professor to work out a suitable appointment time.
* The software must provide a fully functional calendar system for both users.

### 3.2.7 Deletion of appointments

**ID: FR7**

#### 3.2.7.1 Description and Priority

Both the professor and student user should be able to delete appointments, this will be a high (6) priority feature.

#### 3.2.7.2 Stimulus/Response Sequences

Stimulus: A user deletes an appointment.

Response: The appointment will be deleted from the student and professor calendar.

#### 3.2.7.3 Functional Requirements

* To execute the use case the calendaring system must work.
* The professor and student appointment calendars must be able to sync, specific technical requirements are TBD.

##### 

### 3.2.8 Creating a mock schedule.

**ID: FR8**

#### 3.2.8.1 Description and Priority

Both the professor and student user should be able to make mock schedules, being able to send the mock schedule to the advisor or to the student. This will be a high (7) priority feature.

#### 3.2.8.2 Stimulus/Response Sequences

Stimulus: A user selects courses from the catalogue for the next semester.

Response: The courses are organized into a schedule template.

#### 3.2.8.3 Functional Requirements

* The course selection catalog must work.
* Must have a function that takes the course number and time slot data from the course catalogue and converts it to a weekly calendar view.

### 3.2.9 Students should only see availability for advisors in their department.

**ID: FR9**

#### 3.2.9.1 Description and Priority

In account settings, an option will allow them to select their major/minor. Doing this will only show advisors in their department. This prevents the student from creating appointments with advisors from the wrong departments. This feature has medium priority (5).

#### 3.2.9.2 Stimulus/Response Sequences

Stimulus: A student select their major/minor in account settings.

Response: When making an appointment or communicating with the advisor, the program will only show advisors in the correct department.

#### 3.2.9.3 Functional Requirements

* Must have a setting that allows the user to select their major/minor.
* Must have a function that takes students selected major/minor and filters through the complete list of advisors, only showing the correct advisors for the specific major/minor.

### 

### 3.2.10 Students will be able to select classes they have already taken.

**ID: FR10**

#### 3.2.10.1 Description and Priority

In account settings, an option will allow the student to select and save any classes that they have already taken. This will filter the classes needed for the students major and exclude any classes that have been marked as taken. This will be a high (7) priority feature.

#### 3.2.10.2 stimulus/Response Sequences

Stimulus: A student select classes that they have already taken in settings.

Response: When a student is searching for classes, the classes that have been marked as taken will not show up in the course search.

#### 3.2.10.3 Functional Requirements

* The course selection catalog must work.
* Must have the ability to save completed courses for the student, that allows the program to reference when searching for classes.

### 3.2.11 Advisors will be able to see appointments scheduled for a day/week.

**ID: FR11**

#### 3.2.11.1 Description and Priority

The advisor will be able to see an organized calendar view of appointments scheduled within a specified period of time. He/she will be able to click on appointments to display additional information like the appointment meeting. This requirement has a medium priority (6).

#### 3.2.11.2 Stimulus/Response Sequences

Stimulus: The advisor requests to view his/her schedule

Response: The program generates a list of appointments for the selected time frame.

Stimulus: The advisor toggles the time frame

Response: The program regenerates the list of appointments in the selected view.

Stimulus: The advisor clicks on an appointment.

Response: The program displays meeting information like time and reason.

#### 3.2.11.3 Functional Requirements

* Multiple calendar views for each time structure in the advisor would like to view his/her schedule
* Communication between students and advisor to populate advisor schedule with created appointments
* Point and Click UI for navigating through the task

### 3.2.12 Modification of appointments

**ID: FR12**

#### 3.2.12.1 Description and Priority

Both the professor and student user should be able to modify appointments, this will be a high (6) priority feature.

#### 3.2.12.2 Stimulus/Response Sequences

Stimulus: A user modifies an appointment.

Response: The appointment will be modified on the student and professor calendar.

#### 3.2.12.3 Functional Requirements

* To execute the use case the calendaring system must work.
* The professor and student appointment calendars must be able to sync, specific technical requirements are TBD.

## 3.3 Use Cases

### 3.3.1 Use Case #1

|  |  |
| --- | --- |
| **Use Case Name** | Advisor sets up appointments |
| **Reference** | Section 3.2.6, FR 6 |
| **Trigger** | The user clicks to add new appointments. |
| **Precondition** | The calendar for the advisor. |
| **Basic Path** | 1. The user is on the home page and right clicks the floating plus button. 2. User right clicks option to create appointments. 3. A dialog box will appear on the screen and the user right-clicks on the day they would like to set an appointment slot. 4. In the next dialog box, the user enters information to add a block of appointments or a singular appointment. 5. If the user clicks to make a block of appointments they are sent to a survey on the next dialog box. 6. On the survey, the user picks the time blocks they would like to set and then the duration of each appointment. 7. The user is presented with a confirmation screen and right click to confirm or can click the go back button or exit out. 8. The user clicks to confirm then the appointments are populated on the user's calendar. |
| **Alternative Paths** | In step 4, if the user selects to make a singular appointment they will be taken to a different dialog screen.  5. The user can select the date and then the time and duration of the meeting once they are done they right click continue or go back.  6. The user clicks to confirm then the appointments are populated on the user's calendar.  At any point.   1. At any point during the survey on the dialog boxes the user can right click the cross in the right hand corner of the screen to exit out of the appointment making mode. |
| **Postcondition** | The advisor has a set of appointments available. |
| **Exception Paths** | The advisor can choose to exit the appointment making. |
| **Other** | The appointment time slot will default to 15 minutes each. |

*Table 1: Use case 1*

### 3.3.2 Use Case #2

|  |  |
| --- | --- |
| **Use Case Name** | Advisors can view appointments. |
| **Reference** | Section 3.2.11, FR 11 |
| **Trigger** | The user is on their homepage and has a calendar view up. Alternatively the user right clicks into the calendar. |
| **Precondition** | The home page is displayed with the calendar. |
| **Basic Path** | 1. The user views the calendar and sees a breakdown of their appointments. |
| **Alternative Paths** | In step 1, if the user right clicks anywhere on the calendar the full calendar screen will appear.   1. The user can view the calendar in a monthly, weekly or daily view by right clicking the view icon. 2. The user selects the view they would like based on the drop down menu 3. The calendar with appointments is shown. |
| **Postcondition** | The user views appointments. |
| **Exception Paths** | The user may abandon the calendar at any time by selecting the cross while in the calendar view but on the home page they will remain in the homepage. |
| **Other** | To view appointments the user must have appointments. |

*Table 2: Use case 2*

### 3.3.3 Use Case #3

|  |  |
| --- | --- |
| **Use Case Name** | Delete Appointments |
| **Reference** | Section 3.2.7, FR7 |
| **Trigger** | User clicks “delete appointment.” |
| **Precondition** | Scheduling window displays option for deleting appointment. |
| **Basic Path** | 1. User opens the scheduling/appointments window. 2. The reader selects previously scheduled appointments. 3. Program provides the user with an option to delete an appointment. 4. Users can select “Delete” or “Cancel” from the dialogue box. |
| **Alternative Paths** | In step 4, if the user selects “Delete”.   1. The appointment will be deleted from the users and advisors schedule. 2. The time slot will become available for other students.   In step 4, if the user selects “Cancel”.   1. The appointment will not be deleted from the users and advisors schedule. |
| **Postcondition** | The appointment is deleted. |
| **Exception Paths** | The user may abandon the deletion process at any time. |
| **Other** | N/A |

*Table 3: Use case 3*

### 3.3.4 Use Case #4

|  |  |
| --- | --- |
| **Use Case Name** | The student enters their major/ minor information. |
| **Reference** | Section 3.2.4, FR 4 |
| **Trigger** | The student logs in for the first time or the student goes into their settings on their profile. |
| **Precondition** | Survey window appears the first time the user logs in to the application, or the user goes to their settings. |
| **Basic Path** | 1. The student logs in for the first time. 2. The student is presented with a survey screen where they can select their primary major through a drop down menu. 3. The student right clicks confirm. 4. The next screen has an option to add another major via another drop down box, if the user does not have a double major they can right click the skip button. 5. Once the student user right clicks confirm or skip another page is loaded which asks the user if they would like to add a minor. 6. The student selects their minor from the drop down menu and clicks confirm and finish or the student right clicks skip and finish. |
| **Alternative Paths** | Instead of logging in for the first time to access this information, the student major/minor information may have changed in which case they can edit via their settings.   1. The student right clicks to go to their profile. 2. The student right clicks the edit button to edit their profile. 3. Once the edit is selected the student can select their new major, double major or minor from the drop down menu. 4. The student right clicks save to save and commit all changes.    1. Or the student may right click the cancel button to discard any changes made, before they click save. |
| **Postcondition** | The students major, double major, minor is changed. |
| **Exception Paths** | The student may go back at any time to a previous question in the survey. |
| **Other** | The student will be asked to enter their new advisor if their primary major is changed. |

*Table 4: Use case 4*

### 3.3.5 Use Case #5

|  |  |
| --- | --- |
| **Use Case Name** | User logs in using Google account |
| **Reference** | Section 3.2.1, FR1 |
| **Trigger** | Start screen pops with a request for a username and password |
| **Precondition** | Student must have a registered Google account, one is automatically assigned if the user is a DeSales student |
| **Basic Path** | 1. Open the application; the first screen that appears will be this log in screen |
| **Alternative Paths** | 1. If a user chooses to log out of the current account, the first log in screen will appear   In step 2, if the user selected to log in with email.   1. The user selects login with email. |
| **Postcondition** | The student will have access to his/her class schedule and the ability to schedule appointments. |
| **Exception Paths** | If the user does not have a Google account (advisors are not automatically given an account), he/she may sign up using another email |
| **Other** | The categories list is generated from the information provided when articles are published and not predefined in the Online Journal database. |

### 3.3.6 Use Case #6

|  |  |
| --- | --- |
| **Use Case Name** | Student downloads a copy of his/her compiled schedule |
| **Reference** | Section 3.2.3, FR3 |
| **Trigger** | A button or a menu option that reads “Export.” |
| **Precondition** | Students must have created a sample schedule with at least one chosen class. |
| **Basic Path** | 1. On the schedule screen, the student clicks the button titled “Export.” 2. A popup screen appears that allows the user to select the destination for the PDF file. 3. A window appears indicating whether the export succeeded or failed. |
| **Alternative Paths** | N/A |
| **Postcondition** | The user’s schedule in PDF format will be saved to the user’s file system. |
| **Exception Paths** | The user can close the destination popup window. |
| **Other** | The categories list is generated from the information provided when article are published and not predefined in the Online Journal database. |

### 3.3.7 Use Case #7

|  |  |
| --- | --- |
| **Use Case Name** | Students can view/modify appointments. |
| **Reference** | Section 3.2.11, FR11, Section 3.2.12, FR12 |
| **Trigger** | User selects view/modify on an existing appointment. |
| **Precondition** | Scheduling window displays option for viewing/modifying appointment. |
| **Basic Path** | 1. User opens the scheduling/appointments window.  2. The reader selects previously scheduled appointments.  3. Program provides the user with an option to view/modify an appointment. |
| **Alternative Paths** | In step 3, if the user selects “Modify”.   1. Displays preview of the appointment, the user can now modify appointment time and date, and reason for appointment.   In step 3, if the user selects “Cancel”.   1. The view/modify window for the selected appointment will close. |
| **Postcondition** | The appointment has been viewed and or modified by the user. |
| **Exception Paths** | The user may abandon the viewing/modifying process at any time. |
| **Other** | N/A |

*Table 7: Use case 7*

### 3.3.8 Use Case #8

|  |  |
| --- | --- |
| **Use Case Name** | Students will be able to select classes they have already taken. |
| **Reference** | 3.2.10 FR 10 |
| **Trigger** | Students log in for the first time or student enters their settings. |
| **Precondition** | Students must have entered major/double major/ minor information. |
| **Basic Path** | 1. The student logs in for the first time. 2. The student enters their major/double major/ minor information. 3. A list of classes is generated for the students major. 4. The student is presented the classes in categories as it appears on the major requirement paper, here students will right click to select classes they have taken. 5. The student right clicks the next button to continue to the next section of classes. 6. The student has a final option to add classes that are not in their major requirement as a write in submission. 7. The student goes through all the sections and right clicks finish. |
| **Alternative Paths** | During steps 4-6 the student can go back and edit their previous selections.  5. The student right clicks the back arrow to go back to the last section to edit the classes taken  6. The student right clicks the next button to continue to the next section of classes.  7. The student has a final option to add classes that are not in their major requirement as a write in submission.   1. The student goes through all the sections and right clicks finish.   The student may go into settings on their profile and edit their classes taken.   1. The student right clicks their profile. 2. The student right clicks the settings. 3. The student scrolls down to edit classes. 4. The survey screen is presented where the student goes through the sections again to edit classes taken, again with a tick box driven survey. 5. The student right clicks the back arrow to edit any changes that were mistakenly made. 6. The student completes and edits the survey and right clicks finish to complete the survey. 7. The student is taken back to their settings page. 8. At the bottom of the page the student clicks to confirm the changes. |
| **Postcondition** | Classes taken are added to the students profile and used to suggest new classes. |
| **Exception Paths** | The user may abandon this process at any time. |
| **Other** | N/A |

*Table 8: Use case 8*

### 3.3.9 Use Case #9

|  |  |
| --- | --- |
| **Use Case Name** | Students should only see availability for advisors in their department. |
| **Reference** | Section 3.2.9, FR 9 |
| **Trigger** | The user right clicks to view advisors availability. |
| **Precondition** | The student must have a major selected as well as an advisor. |
| **Basic Path** | 1. The user right clicks the button to see advisors availability on the homepage. 2. The user is taken to their advisors appointment schedule. |
| **Alternative Paths** | Instead of viewing the advisor though the home page the user can right click the menu button and right click ‘myAdvisor’   1. The user is taken to the advisors profile where they can view the advisors appointment schedule. |
| **Postcondition** | The student will see the advisors availability and make an appointment. |
| **Exception Paths** | If the user has no advisor selected they will be shown an error and not be taken to the advisor appointment schedule. |
| **Other** | N/A |

*Table 9: Use case 9*

### 3.3.10 Use Case #10

|  |  |
| --- | --- |
| **Use Case Name** | Creating a mock schedule. |
| **Reference** | Section 3.2.8 , FR 8 |
| **Trigger** | The student selects classes they want to attend. |
| **Precondition** | The student has completed the survey to detail what classes they have taken as well as their major/minor. |
| **Basic Path** | 1. The user right clicks the button to create a timetable/schedule. 2. The user is taken to a time table page where the classes they should take are in the suggested area on the right side of the page. 3. The user selects classes that they would like to take and it is populated in the time table. 4. The user right clicks the button to save. |
| **Alternative Paths** | N/A |
| **Postcondition** | The time table is created. |
| **Exception Paths** | At step 3 the user can delete classes they no longer want to take.   1. The user right clicks the delete icon on the bottom of the class in the time table. 2. A dialogue box appears to confirm with the user. 3. The user right clicks confirm to delete the appointment. |
| **Other** | N/A |

*Table 10: Use case 10*

### 3.3.11 Use Case #11

|  |  |
| --- | --- |
| **Use Case Name** | Push appointment reminders to students and advisors. |
| **Reference** | Section 3.2.5 , FR 5 |
| **Trigger** | Upcoming appointment date will trigger an email reminder to be sent to the students email. |
| **Precondition** | The student must have already made an appointment. |
| **Basic Path** | 1. A student will create an appointment with an advisor. 2. A function will check the appointment date and time and create an automated reminder email. 3. A trigger will be set to activate on the morning of the appointment at 12:00am. 4. When the trigger is activated the reminder email will be sent to the student. |
| **Alternative Paths** | N/A |
| **Postcondition** | An appointment reminder will have been sent to the student. |
| **Exception Paths** | N/A |
| **Other** | N/A |

*Table 11: Use case 11*

### 3.3.10 Use Case #12

|  |  |
| --- | --- |
| **Use Case Name** | Students can see and book appointments with advisors in their major/minor. |
| **Reference** | Section 3.2.2, FR 2 |
| **Trigger** | The user right clicks to see the advisors availability. |
| **Precondition** | The student user must have a major/ minor and advisor confirmed. |
| **Basic Path** | 1. The user right clicks the button to see advisors availability on the homepage. 2. The user is taken to their advisors appointment schedule. 3. The user right clicks the arrows on the calendar to view their selected week. 4. The user right clicks on the appointment slot they want. 5. A dialogue box and the user right clicks confirm to confirm the appointment. |
| **Alternative Paths** | Instead of viewing the advisor though the home page the user can right click the menu button and right click ‘myAdvisor’   1. The user is taken to the advisors profile where they can view the advisors appointment schedule.   At step 5, the user can choose to abandon the appointment.  5. The user right clicks the cancel button to be taken back to the appointment page. |
| **Postcondition** | The student has an appointment booked. |
| **Exception Paths** | If no major/minor or advisor is confirmed a dialogue box will show to prompt the user to go into settings and add the required information. |
| **Other** | The user can abandon the process at any time by changing pages. |

*Table 12: Use case 12*

## 3.4 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc.).

### 3.5.1 Performance

The application should perform well in both a mobile setting and computer setting as the application should be built to be a responsive web application. Furthermore, we aim for 90% of the transactions of the application to be completed within 1-5 seconds at the database will not be extremely large since the application will only cover computer science and mathematics majors.

### 3.5.2 Reliability

The application should be available throughout the business week and weekend. In order to perform any maintenance that the application may have to go through users will be alerted and changed will be applied on Sunday at 2am to make sure the users are as uninterrupted as possible.

### 3.5.3 Availability

To ensure the applications availability, CAA will be a web based application meaning that students and advisors can reach the application wherever and whenever they have a stable internet connection and they do not have to download anything.

### 3.5.4 Security

The program will be hosted using Google Firebase therefore, security will be configured using the Firebase security rules. In addition to the security rules given by Firebase we will add our own such as requiring the user sign in with a ‘@desales.edu’ email address to ensure that only the intended users have access to the application. The users will have write access to edit their appointment availability and change their class information.

### 3.5.5 Maintainability

In order to maintain the application all team members will review the application periodically, every month and if an issue is reported by the user then immediately set up a plan of action to solve the issues. To apply changes and carry out other maintenance the application will experience down time on Sundays at 2am to make sure the users are as uninterrupted as possible. We will be using the tools that Google Firebase contains to monitor the application as well as user issues and periodically checking the application ourselves.

### 3.5.6 Portability

As mentioned previously CAA will be a web based application therefore the application is extremely portable, we also plan to make the application a responsive web application so that users can use the application on their mobiles.

## 3.5 Design Constraints

The biggest limitation of this program is that only computer science and mathematics majors will be catered to as there is not enough time to cater to all the major, double major, and minor combinations available at DeSales University. The software will also be designed to follow the login protocols of the university by ensuring that the email the users user end in ‘@desales.edu’ to ensure only members of the university have access to the information.

## 3.6 Logical Database Requirements

In order to ensure all the classes and requirements for each major and or minor are present we will use a SQL database. This database will hold the required classes as well as the catalogue of classes offered for that semester. In CAA we will only cater for computer science majors and mathematics majors so not all classes offered by the school will be present for the user. Lastly, a database will be used to store the professor users appointment availability and to save the student users mock schedule. To ensure data integrity most data that the user enters will be from a drop down selection menu and produce errors if the user enters an invalid data type.

# 4. Analysis Models

## Sequence Diagrams

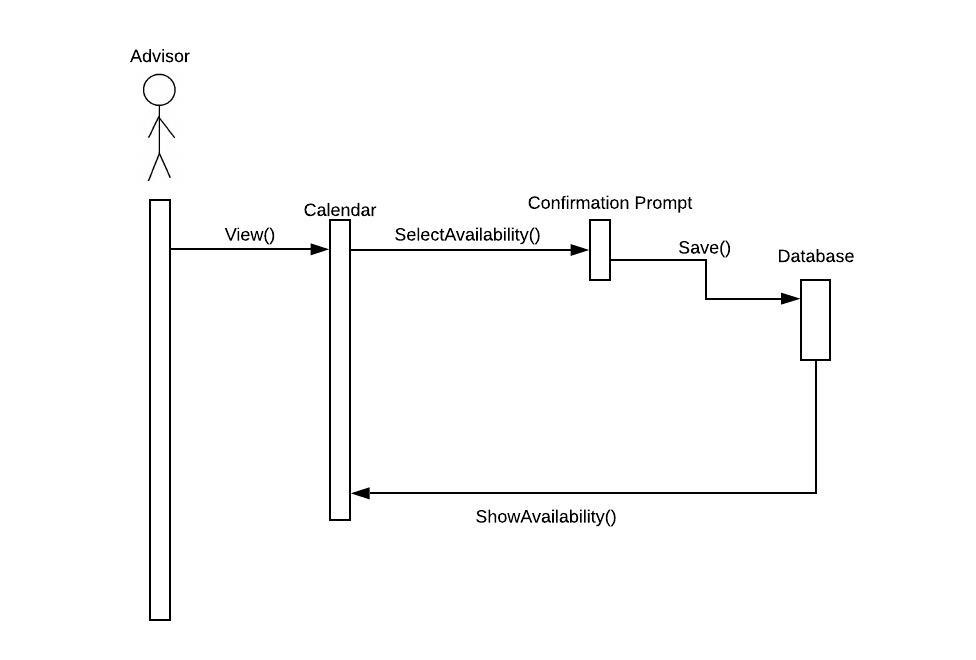


Figure 1: Data Flow Diagram of Use Case 1

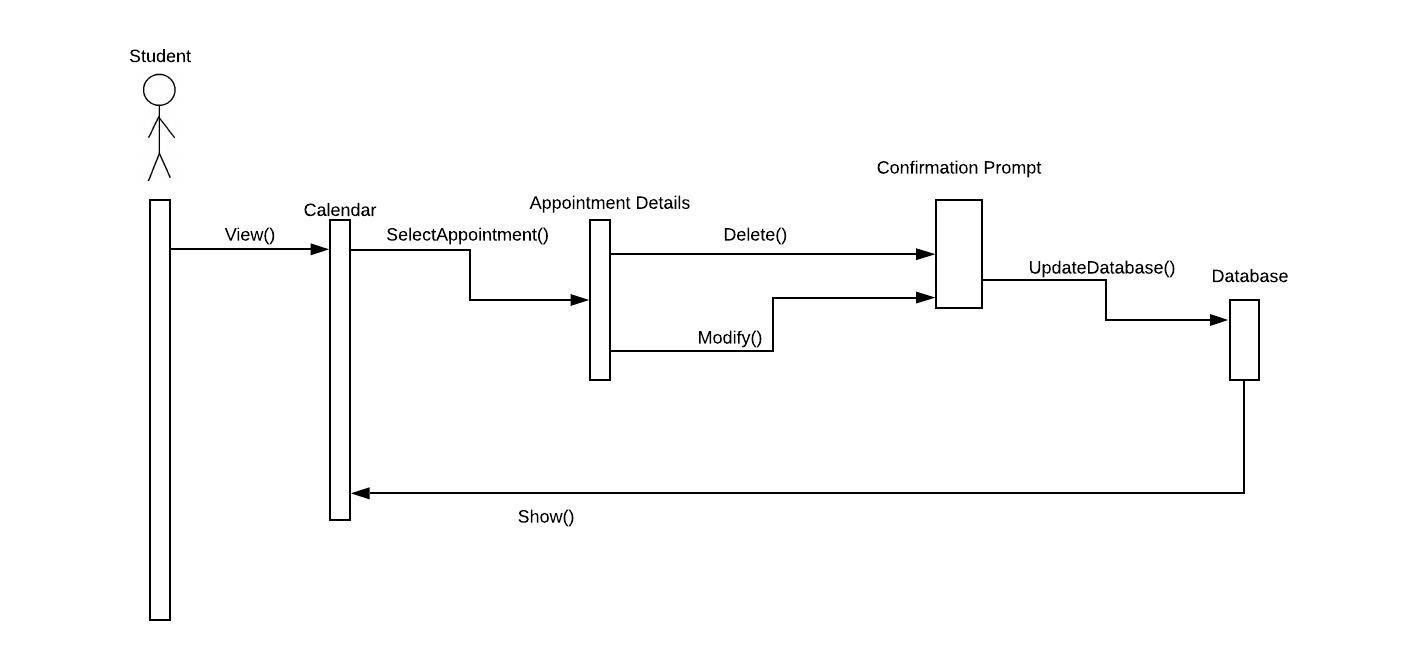


Figure 2: Data Flow Diagram of Use Cases 3 & 7

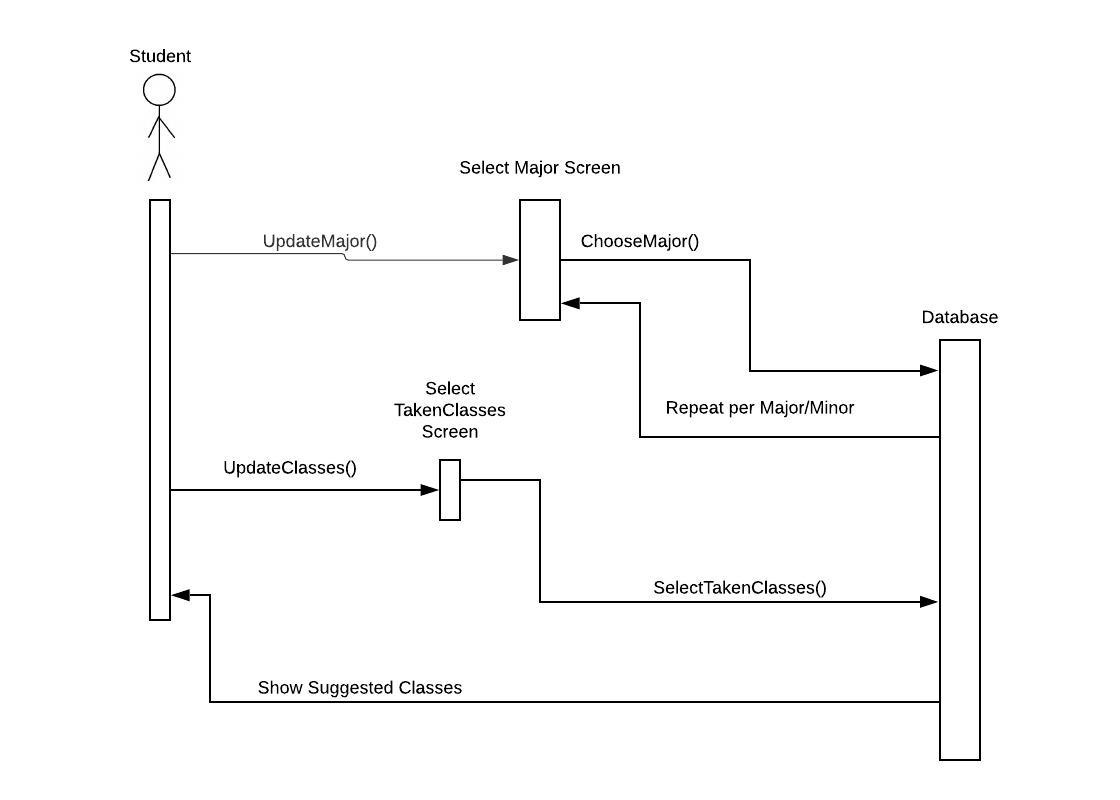


Figure 3: Data Flow Diagram of Use Cases 4 & 8

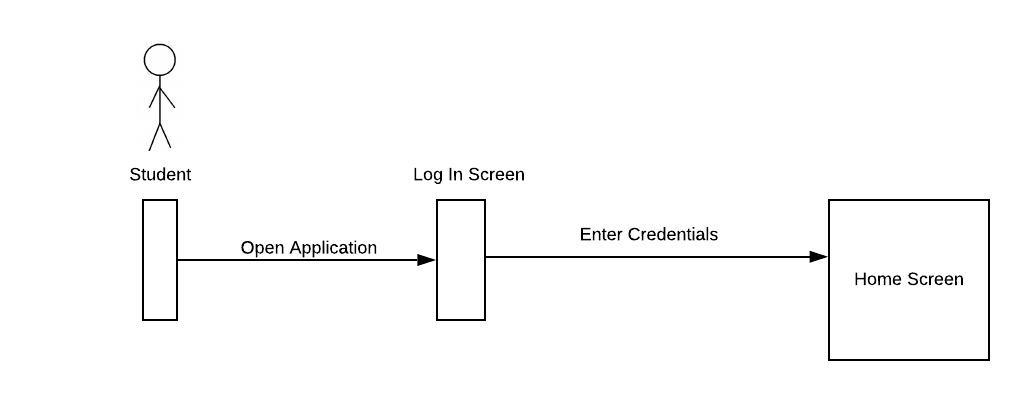


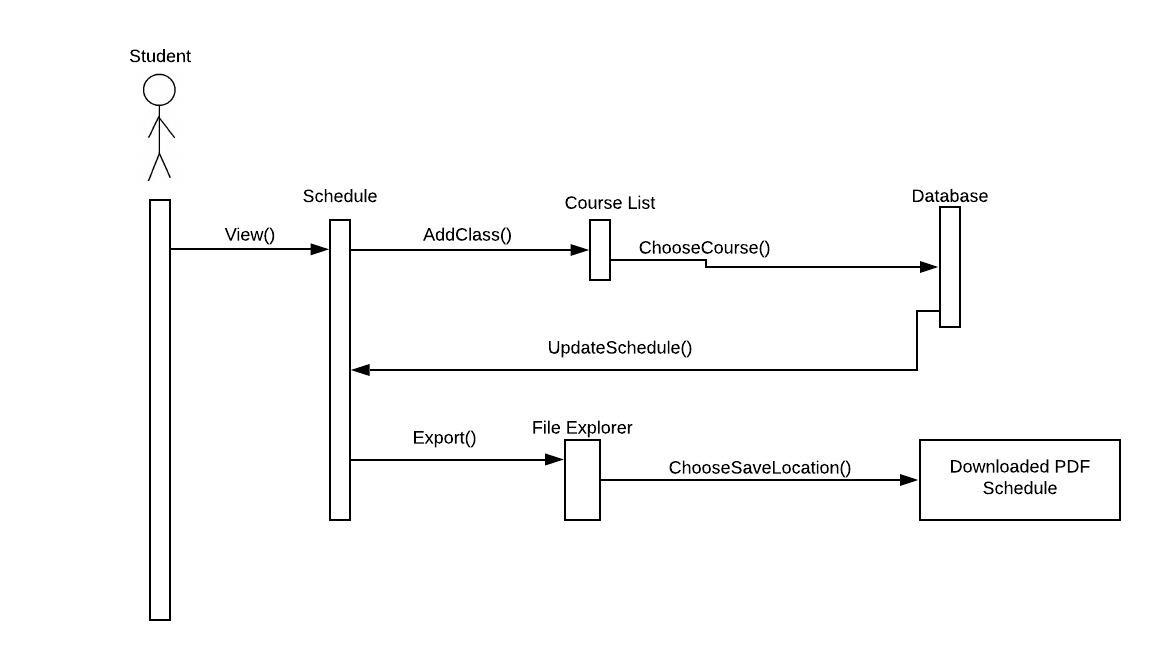
Figure 4: Data Flow Diagram of Use Case 5

Figure 5: Data Flow Diagram of Use Cases 6 & 10

# 5. Change Management Process

To ensure version control the SRS document will be uploaded to GitHub once a version is confirmed by all members of the team. As the project continues changes to the document will be completed by one of the team members and sent to the others for review then uploaded to GitHub with the version information updated.

# References

# Appendices

## A.1 Appendix 1

## A.2 Appendix 2