**Software Design Document**

**Version 1.6**

**December 15th, 2017**

**Student Calendar Integration Application**

**BYU-Idaho CS364 Software Engineering**

# Authors

## Project Manager

Michael Carey

## Team Leads

Landon Shumway  
Robert Nelson  
Adam Shumway  
Daniel Craig

## Team Members

Austin Golding

Marcus Rhodes

Jonathan Comeau

Dallin Snell

Kylor Kersavage

Aaron Shore

Brian Robertson

Joseph Keene

Jacob Keene

Zane West

Rex Nesbit

John Memmott

Jefferson Santos

Keith Wheeler

Tyler Sorenson

Brandon Hartshorn

Micah Wilson

Jason Catmull

Sebastian Calle

Fernando Gomez

# Revision History

| **Name** | **Date** | **Reason for Changes** | **Version** |
| --- | --- | --- | --- |
| Initial | 11/4/2017 | First Draft | 1.0 |
| Draft | 11/12/2017 | Second Draft | 1.1 |
| Non-Required Draft | 11/18/2017 | Third Draft | 1.2 |
| Use Case Focus | 11/25/2017 | Fourth Draft | 1.3 |
| Whole Document Work | 12/2/2017 | Fifth Draft | 1.4 |
| Whole Document Work | 12/9/2017 | Rework of the SDD based on Mapping to IEEE Spec. | 1.5 |
| Final Submission | 12/15/2017 | Document Complete | 1.6 |

Table of Contents

[Authors 2](#_Toc501008406)

[Project Manager 2](#_Toc501008407)

[Team Leads 2](#_Toc501008408)

[Team Members 2](#_Toc501008409)

[Revision History 4](#_Toc501008410)

[List of Figures 8](#_Toc501008411)

[1.0 Introduction 10](#_Toc501008412)

[1.1 Purpose 10](#_Toc501008413)

[1.2 Scope 10](#_Toc501008414)

[1.3 Context 10](#_Toc501008415)

[1.4 Overview 12](#_Toc501008416)

[1.5 References 12](#_Toc501008417)

[2.0 Glossary 13](#_Toc501008418)

[3.0 Use Cases 15](#_Toc501008419)

[3.1 Identified Stakeholders and Design Concerns 15](#_Toc501008420)

[3.1.1 Calendar Manager 15](#_Toc501008421)

[3.1.2 Developer 16](#_Toc501008422)

[3.1.3 Tester 16](#_Toc501008423)

[3.1.4 Technical Support 16](#_Toc501008424)

[3.2 Use Case Tables and Descriptions 17](#_Toc501008425)

[3.2.1 Calendar Display 18](#_Toc501008426)

[3.2.2 Calendar Event Notifications 36](#_Toc501008427)

[3.2.3 To-Do List 47](#_Toc501008428)

[3.2.4 Calendar Integrations 56](#_Toc501008429)

[3.2.5 User Profile Settings 61](#_Toc501008430)

[3.2.6 Import Assignments from I-Learn 70](#_Toc501008431)

[3.2.7 Log-In Use case 78](#_Toc501008432)

[4.0 Design Overview 87](#_Toc501008433)

[4.1 System Architecture 88](#_Toc501008434)

[4.1.0 Architecture Description 89](#_Toc501008435)

[4.2.0 UML Class Descriptions 90](#_Toc501008436)

[4.2.1 View Classes 90](#_Toc501008437)

[4.2.x To-Do List View 90](#_Toc501008438)

[4.2.x Calendar Page Main View 92](#_Toc501008439)

[4.2.x Login Page View 97](#_Toc501008440)

[4.2.2 Controller Classes 100](#_Toc501008441)

[4.2.3 Model Classes 119](#_Toc501008442)

[4.3 System Interfaces 131](#_Toc501008443)

[4.3.1 User Interface 131](#_Toc501008444)

[4.3.2 Software Interfaces 131](#_Toc501008445)

[4.4 Constraints and Assumptions 132](#_Toc501008446)

[4.4.1 Constraints 132](#_Toc501008447)

[4.4.2 Assumptions 132](#_Toc501008448)

[4.5 Error Handling 133](#_Toc501008449)

[4.5.1 Database Facade Class Commit Errors 133](#_Toc501008450)

[4.5.2 Invalid Data Errors 133](#_Toc501008451)

[4.5.3 I-Learn Importing Errors 133](#_Toc501008452)

[4.5.4 UI Controller Class Errors 133](#_Toc501008453)

[5.0 Data Design 134](#_Toc501008454)

[5.1 Data Description 134](#_Toc501008455)

[5.2 Entity Relationship Diagram 135](#_Toc501008456)

[5.3 Data Dictionary 136](#_Toc501008457)

# List of Figures

[Figure 1 - Projected User Base Chart 11](#_Toc501008164)

[Figure 2 - Calendar Display UML Communication Diagram 20](#_Toc501008165)

[Figure 3 - Calendar Home Screen Mockup 22](#_Toc501008166)

[Figure 4 - Event Creation Screen Mockup 26](#_Toc501008167)

[Figure 5 - Assignment Access Screen Mockup 29](#_Toc501008168)

[Figure 6 - Holiday Display Screen Mockup 33](#_Toc501008169)

[Figure 7 - Event Editing Screen Mockup 35](#_Toc501008170)

[Figure 8 - Calendar Event Notifications UML Communication Diagram 38](#_Toc501008171)

[Figure 9 - Notification Settings Screen Mockup 40](#_Toc501008172)

[Figure 10 - Reminder Window Screen Mockup 43](#_Toc501008173)

[Figure 11 - To Do Assignment List UML Communication Diagram 49](#_Toc501008174)

[Figure 12 - To Do Assignment List Screen Mockup 51](#_Toc501008175)

[Figure 13 - Calendar Integrations UML Communication Diagram 58](#_Toc501008176)

[Figure 14 - Import Calendar Screen Mockup 60](#_Toc501008177)

[Figure 15 - User Profile Settings diagram 63](#_Toc501008178)

[Figure 16 - Import Assignments from I-Learn Feature Diagram 72](#_Toc501008179)

[Figure 17 - Import Assignments from I-Learn: Manual Sync Mockup 74](#_Toc501008180)

[Figure 18 - Import Assignments from I-Learn: Automatic Sync Mockup 78](#_Toc501008181)

[Figure 19 - Log-In Use Case Diagram 80](#_Toc501008182)

[Figure 20 - System Architecture Diagram 91](#_Toc501008183)

[Figure 21 - Entity Relationship Diagram 140](#_Toc501008184)

# 1.0 Introduction

## 1.1 Purpose

The purpose of this Software Design Document is to define the design elements of the Student Calendar Integration Application. The document provides sufficient detail to implement the software product to stakeholder specifications.

## 1.2 Scope

This document supplies details for all features required in the initial release of the Student Calendar Integration Application. Specifically, internal data models, user interface, overall design views, and relevant use cases are defined.

## 1.3 Context

The Student Calendar Integration Application is an application to ease the tracking of class assignments for college students at BYU-Idaho. Teachers, class development staff, and technical support will also have access to the administrative interface of the application. It will be integrated with the OLP (Online Learning Platform) currently in use by the school, Brainhoney. The application will be accessed via mobile or web interface. The purpose is to assist students with meeting deadlines to maximize learning outcomes.

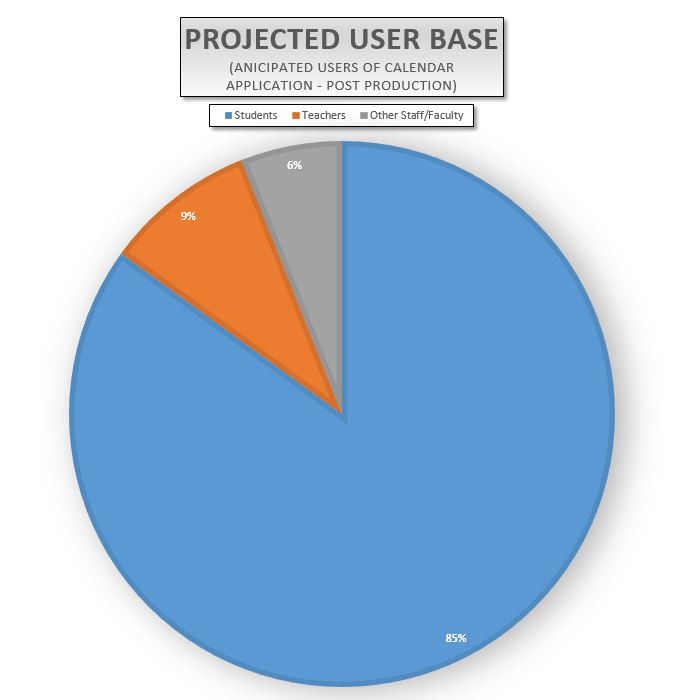
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/piChart.jpg)

Figure 1 - Projected User Base Chart

## 1.4 Overview

This document is divided into 6 sections:

1. Introduction
2. Glossary
3. Use Cases
4. Design Overview
5. Data Design
6. User Interface Design

## 1.5 References

[1] IEEE, "IEEE Standard for Information Technology — Systems Design — Software Design Descriptions" pp. 1-35, Jul. 2009. [Online] Available: <http://ieeexplore.ieee.org/servlet/opac?punumber=5167253>

[2] Student Calendar Integration Application Software Requirements Specification

* Version 1.8, Last Updated Oct 23, 2017

# 2.0 Glossary

| **Terms** | **Definitions** |
| --- | --- |
| Actor | (in a UML context) someone or something that interacts with a given system |
| API | application programming interface; a set of subroutine definitions, protocols, and tools for building application software |
| Boolean | a binary variable, having two possible values called "true" and "false" |
| Facade | a simplified interface |
| Getter | a function in a software program that acts as a communicator to read from private data |
| Pseudocode | an informal high-level description of the operating principle of a computer program |
| Sanitize | when referring to user input: input is "sanitized" in that any malicious actions that may be attempted through the input are safely removed |
| Setter | a function in a software program that acts as a communicator to write to private data |
| Stakeholder | a person or entity who has a voice in the development requirements |
| SQL Injection | an attack on a system that involves "injecting" the attacker's own SQL code to perform unintended queries to a given database |
| UI | user interface; the space where interactions between humans and machines occur |
| UML | Unified Modeling Language; it is a modeling language intended to provide a standard way to visualize the design of a system |
| Use case | a list of actions or event steps |
| UX | User Experience; it refers to a person's emotions and attitudes about using a particular product, system or service |

# 3.0 Use Cases

This section addresses the identified stakeholders and their design concerns in 3.1. In section 3.2 we provide several context viewpoints including UML communication diagrams to provide an overview of each viewpoint, and use cases that are design entities that pertain to the corresponding context viewpoint.

Each UML communication diagram displays the function calls that are associated with each use case. The function names are labeled with a preceding number indicating which use case the function call corresponds to (ex. the function labeled "0.1 addToDoAssignments()" located in the To-Do Assignment List diagram(3.2.3), corresponds to use case 3.2.3.0 "Load List" while "1.1 markAssignmentCompleted(Assignment)" corresponds to use case 3.2.3.1 "Mark Assignment Done") Each of these functions are defined in section 4.2 within their respective classes.

## 3.1 Identified Stakeholders and Design Concerns

### 3.1.1 Calendar Manager

The Calendar Manager is a user who works with student tasks, events, assignments, reminders, etc. There are several users who fall into the Calendar Manager category such as teachers, event coordinators, and students. Each user can perform similar actions, but for different reasons. For example, a teacher may want to highlight a test deadline for his or her students. A student may want to set reminders about a test or homework deadline throughout the week. Another term for calendar manager would be "end users."

#### 3.1.1.1 Design Concerns

* Calendar managers should be able to view a calendar with their scheduled assignments and events.
* Calendar managers should be able to set reminders for their assignments, so they won't forget.
* Calendar managers should be able to see a prioritized list of their assignments to increase their productivity.
* Calendar managers should be able to use their existing calendar application in conjunction with ours.
* Calendar managers should be able to change settings according to their preference.
* Calendar managers should be able to import assignments from ILearn.
* Calendar managers should have their data be secure and only accessible by them.

### 3.1.2 Developer

Developers are those involved in the development process of the calendar application.

#### 3.1.2.1 Design Concerns

* Developers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Developers should know what patterns and classes to be used.
* Developers should know how the data is stored in the database.

### 3.1.3 Tester

Testers are those that make test cases and assist in testing the calendar application.

#### 3.1.3.1 Design Concerns

* Testers should know of the expected inputs and outputs.
* Testers should understand the processes associated to test all aspects of the product.

### 3.1.4 Technical Support

Technical Support are those that will be helping calendar managers to troubleshoot problems that may arise with using the calendar application.

#### 3.1.4.1 Design Concerns

* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

## 3.2 Use Case Tables and Descriptions

Use cases provide design viewpoints that show the interactions between the actors or personas and the system that are required. The following table is employed as a template to describe use cases in section 3.4. The left side of the table remains the same, while the right provides a description of the text that will be filled in according to the particular use case.

#### 3.2.0.0 Use Case Template

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | The Screenshot/Mockup will contain a graphical illustration which will show an overlay of the use case under consideration. |
| Page Title: | The page title is the title of the web page corresponding to the use case or "N/A" if not applicable. |
| Author: | Robert Nelson |
| Type: | Process. |
| Purpose: | A description of why the element exists. The purpose attribute shall provide the rationale for the creation of the element. |
| Parent User Story: | The Parent User Story links to the higher-level use case that is parent to a use case. |
| Stakeholders: | The stakeholders that the use case pertains to. |
| Pre-conditions/Product(s) Required: | The conditions, procedures, and requirements to be completed before the use case may be attempted. |
| Post-conditions/Product(s) Produced: | The conditions, procedures, and requirements to be completed upon termination of the use case. |
| Links: | Additional references either inside this document or external references. |
| SRS Document | The reference to the section in Software Requirements Specification pertaining to the use case. |
| Description/Notes: | A complete description of the process and interactions that take place in the use case. These interactions include design concerns, requirements, and design attributes. The description/notes section also contains any additional notes and requirements pertaining to the use case. |

#### 3.2.0.1 Error Handling

As a general policy any input received from the user interface will be validated to prevent injection attempts. Any unexpected input will display an error message that explains the error encountered. Use cases will explain specific error handling situations.

### 3.2.1 Calendar Display

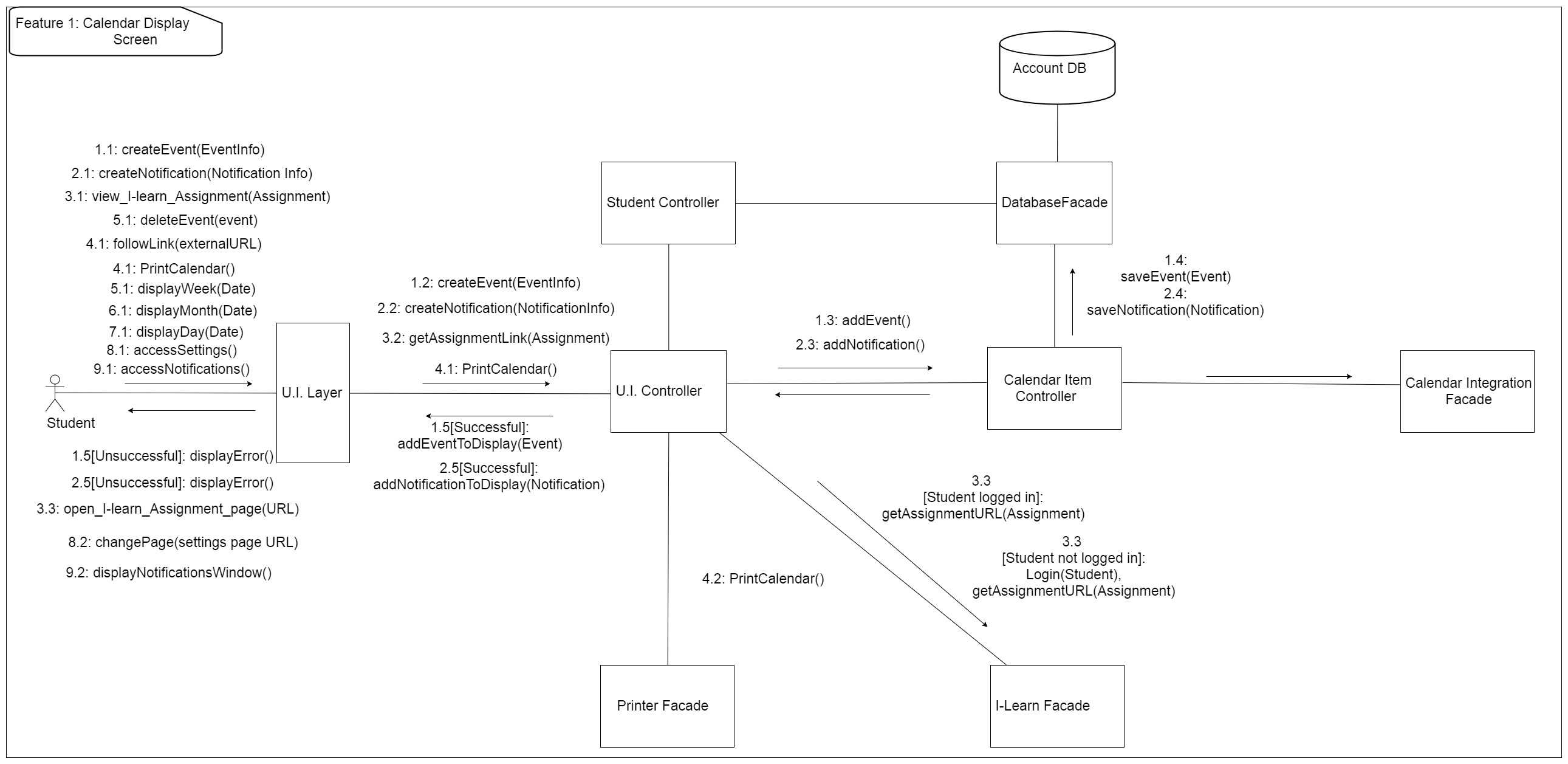
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/Calendar%20Display%20UML%20Communication%20Diagram.jpg)

Figure 2 - Calendar Display UML Communication Diagram

Design Concerns Addressed:

* Calendar managers should be able to view a calendar with their scheduled assignments and events.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.1.0 Calendar Display: Load Home Screen

| **Data** | **Description** |
| --- | --- |
| ScreenShot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.4.1.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.4.1.0.png)  Figure 3 - Calendar Home Screen Mockup |
| Page Title: | Home Page |
| Parent User Story: | N/A |
| Actor(s)/Persona(s): | Calendar Users |
| Pre-conditions/Product(s) Required: | A compatible web browser such as Google Chrome, Firefox, Chromium, Safari, or Microsoft Edge. |
| Post-conditions/Product(s) Produced: | The Home Screen is displayed. |
| Links: | None. |
| SRS Document | 3.2.2 |
| Description/Notes: | 1. The calendar user will navigate to the URL of the calendar website. 2. The website's home page will be loaded and displayed. a. The home page will load the user's specific calendar if they are logged in. b. If the user has not logged in, then a default calendar will be loaded on screen. c. If the user is not logged in, then a small modal box will pop up prompting the user to sign in or register an account.  d. If the user clicks the sign in button they will be redirected to the log-in screen.  e. If the user clicks the new user button they will be redirected to the account registration page. 3. If the web browser is not supported by the web site, then it will display a message explaining the compatibility issue. 4. Pages that users try to access but do not exist on the web site domain will show an error message and have a link back to the home page. |

#### 3.2.1.1 Calendar Display: Checking the Schedule

| **Data** | **Description** |
| --- | --- |
| ScreenShot/Mockup: |  |
| Page Title: | Checking the Schedule |
| Parent User Story: | N/A |
| Actor(s)/Persona(s): | Calendar Users |
| Pre-conditions/Product(s) Required: | User is logged in. |
| Post-conditions/Product(s) Produced: | Calendar is up on display. |
| Links: | None. |
| SRS Document | 3.2.2 |
| Description/Notes: | 1. When the user goes to the home page the calendar will be shown with the events the user has on their calendar.  a. Events will be displayed as user has scheduled.  b. School assignments will be displayed as synced with I-Learn.  2. The user will be able to switch between daily, weekly, and monthly views. See 3.4.1.3.  3. The assignment titles and due time will be displayed and resized to fit on the screen. a. If the assignment titles won't fit on a calendar day then the calendar will be resized to be larger. 3. The user can click the assignment to view its details. |

#### 3.2.1.2 Calendar Display: Creating an Event

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | ￼  Figure 4 - Event Creation Screen Mockup |
| Page Title: | Creating an Event |
| Author: | Jacob Keene |
| Type: | Process. |
| Purpose: | Creating event in the calendar allows the user to schedule future events to assist in organization and productivity. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. |
| Post-conditions/Product(s) Produced: | Event is created and displays correctly on calendar. |

#### 3.2.1.3 Calendar Display: Switching Views

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Switching Views |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Switching views allows the user to have alternative ways to view their calendar and increase productivity. |
| Parent User Story: | 3.2.1.1 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. User is on home page. |
| Post-conditions/Product(s) Produced: | Calendar is displayed in the corresponding daily, weekly, or monthly view with the user's events and assignments shown. |

#### 3.2.1.4 Calendar Display Screen: Creating an Alert

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Creating an Alert |
| Author: | Zane West |
| Type: | Process. |
| Purpose: | An alert allows the user to be notified as to what action is needed for different events. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. An application event requiring an alert occurs.  2. An error occurs that requires immediate user action or notification. |
| Post-conditions/Product(s) Produced: | 1. The user is presented with the recommended course of action or the option to ignore. |

#### 3.2.1.5 Calendar Display: Access Assignment

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.1.6.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.1.6.png)  Figure 5 - Assignment Access Screen Mockup |
| Page Title: | Access Assignment |
| Author: | Joseph Keene |
| Type: | Process. |
| Purpose: | The user is able to click links of assignments in the calendar view to be directed to each assignment. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is Logged in  2. Assignments have been imported. |
| Post-conditions/Product(s) Produced: | Assignment is displayed correctly to the user. |
| Links: | None. |
| SRS Document | 3.2.8 |
| Description/Notes: | Accessing An Assignment. 1. Assignments are displayed to the User on the Calendar as links. a. Users clicks on one of the links. b. The correct assignment is pulled up on screen for the user. 2. Assignment details button is shown. a. User clicks assignments details button. b. User is able to interact with different features for the event. |

#### 3.2.1.6 Calendar Display: Show Assignment Details

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Show Assignment Details |
| Author: | Joseph Keene |
| Type: | Process. |
| Purpose: | The assignment details are provided to the user for easier access and productivity while using the calendar. |
| Parent User Story: | 3.2.1.5 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in 2. Assignments have been imported. |
| Post-conditions/Product(s) Produced: | Assignments details are shown and displayed correctly to the user. |
| Links: | None. |
| SRS Document | 3.2.6 |
| Description/Notes: | Show Assignment Details 1. Show Details Button is displayed on the Calendar App by the Assignments. a. User clicks Details Button. 2. Details are displayed to the User. a. An option to collapse the details is shown as well. |

#### 

#### 3.2.1.7 Calendar Display: Show Holidays

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.1.8.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.1.8.png)  Figure 6 - Holiday Display Screen Mockup |
| Page Title: | Show Holidays, |
| Author: | Joseph Keene |
| Type: | Process. |
| Purpose: | Holidays will be shown on the users' calendars to help the users plan accordingly. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. Calendar is loaded. |
| Post-conditions/Product(s) Produced: | The user will see a text label on national holidays. |
| Links: | None. |
| SRS Document | 3.2.26 |
| Description/Notes: | Show Holidays 1. The user chooses to view the calendar by Day, by Week, and by Month. 2. The user will see a text label on national holidays. |

#### 

#### 3.2.1.8 Calendar Display: Edit an Event

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.1.8.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.1.8.png)  Figure 7 - Event Editing Screen Mockup |
| Page Title: | Edit Event |
| Author: | Brian Robertson |
| Type: | Process. |
| Purpose: | Users will be able to edit events to reflect the most current event information and respond to changes in the details of an event. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. |
| Post-conditions/Product(s) Produced: | Event is updated and displays correctly on calendar. |
| Links: | None. |
| SRS Document | 3.2.16 |
| Description/Notes: | Editing an Event  1. Edit Event button is displayed on the Calendar App.  a. User clicks Edit Event button.  b. User prompted to update selected time(s) for the event.  c. User enters / edits description of event.  2. Event is updated and displayed on User's Calendar.  a. User can interact with Event on Calendar. |

### 3.2.2 Calendar Event Notifications

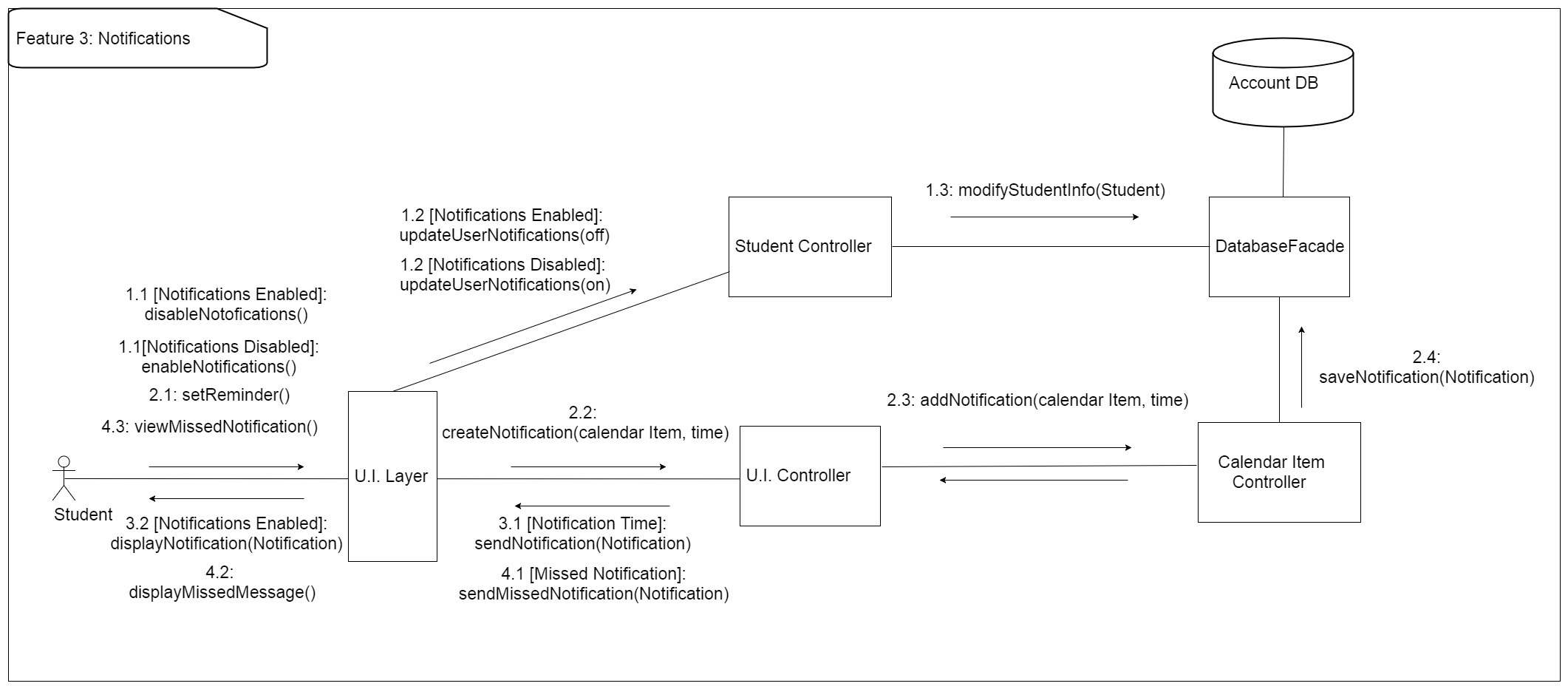
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/Notification%20UML%20Communication%20Diagram.jpg)

Figure 8 - Calendar Event Notifications UML Communication Diagram

Design Concerns Addressed:

* Calendar managers should be able to set reminders for their assignments, so they won't forget.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.2.0 Calendar Event Notifications: Enable/Disable Notifications

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.2.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.2.0.png)  Figure 9 - Notification Settings Screen Mockup |
| Page Title: | Enable/Disable Notifications |
| Author: | Zane West |
| Type: | Process. |
| Purpose: | Calendar event notifications will help remind the user of upcoming events, activities, and holidays. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Notification permissions are enabled. |
| Post-conditions/Product(s) Produced: | 1. User will be able to toggle notifications.  2. When notifications are disabled, a disabled notifications icon will be present within the application to remind the user that the application will not remind them of upcoming events. |
| Links: | 3.2.5.0 |
| SRS Document | 3.2.20 |
| Description/Notes: | Enable/Disable Notifications  1. Notifications will be enabled by default.  2. The notification toggle will be found in the parent settings page.  3. Warnings  a. If notifications or notification permissions are blocked by the operating system, a prompt will notify the user of this situation.  b. The notification will have the option to redirect the user to the permissions page of their operating system or browser to correct the error.  c. The notification will have the option to never allow the permission and never remind the user of the error. |

### 

#### 3.2.2.1 Calendar Event Notifications: Set Reminders

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.2.1.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.2.1.png)  Figure 10 - Reminder Window Screen Mockup |
| Page Title: | Set Reminders |
| Author: | Zane West |
| Type: | Process. |
| Purpose: | Reminders will be available to the user to help remind them of upcoming events, activities and holidays. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. An event has been created. |
| Post-conditions/Product(s) Produced: | 1. A reminder is prepared.  2. User is returned to their last viewed calendar. |
| Links: | None. |
| SRS Document | 3.2.22 |
| Description/Notes: | 1. The user will have options as to how they would like to be reminded about their upcoming events.  a. The user can select the timing of the notification, e.g.: 10 minutes before, 1 hour before, 1 day before.  b. The user can choose the means by which they will be reminded, e.g.: operating system reminder, e-mail or text message. |

#### 3.2.2.2 Calendar Event Notifications: Send Notification

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Send Notification |
| Author: | Zane West |
| Type: | Process. |
| Purpose: | The application will send notifications to the users invited to each event to remind them of upcoming events, activities and holidays. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. An event has been created.  3. Notification permissions are allowed.  4. Notifications are enabled in the settings. |
| Post-conditions/Product(s) Produced: | 1. A notification is prepared.  2. User is returned to their last viewed calendar.  3. Notification is sent to user based on their preference selected during the event creation. |
| Links: | None. |
| SRS Document | 3.2.22 |
| Description/Notes: | Sent Notification  1. Notification  a. Based on preference, an initial notification can be prepared and sent to the user informing them of the event details.  b. Based on the application preferences of each event guest, an additional reminder will also be prepared and reserved according to their reminder and notification preferences.  2. Warning  a. An alert should warn the host that some invited guests may have reminders disabled and may not see the event.  b. If notification permissions are not allowed, this feature will not work. |

#### 3.2.2.3 Calendar Event Notifications: View Missed Notifications

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | View Missed Notifications |
| Author: | Jacob Keene |
| Type: | Process. |
| Purpose: | The application will send reminders about missed notifications to help ensure the user doesn't miss important events. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. A notification has been created.  3. Users have been sent a notification. |

#### 3.2.2.4 Calendar Event Notifications: Set Notification Methods

| Screenshot/Mockup: |  |
| --- | --- |
| Page Title: | Set Notification Methods |
| Author: | Jacob Keene |
| Type: | Process. |
| Purpose: | To allow the user to decide which notifications they would like to receive. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. User goes to notification settings. |
| Post-conditions/Product(s) Produced: | 1. User's notification methods have been updated.  2. User is returned to their last viewed calendar. |
| Links: | None. |

### 3.2.3 To-Do List

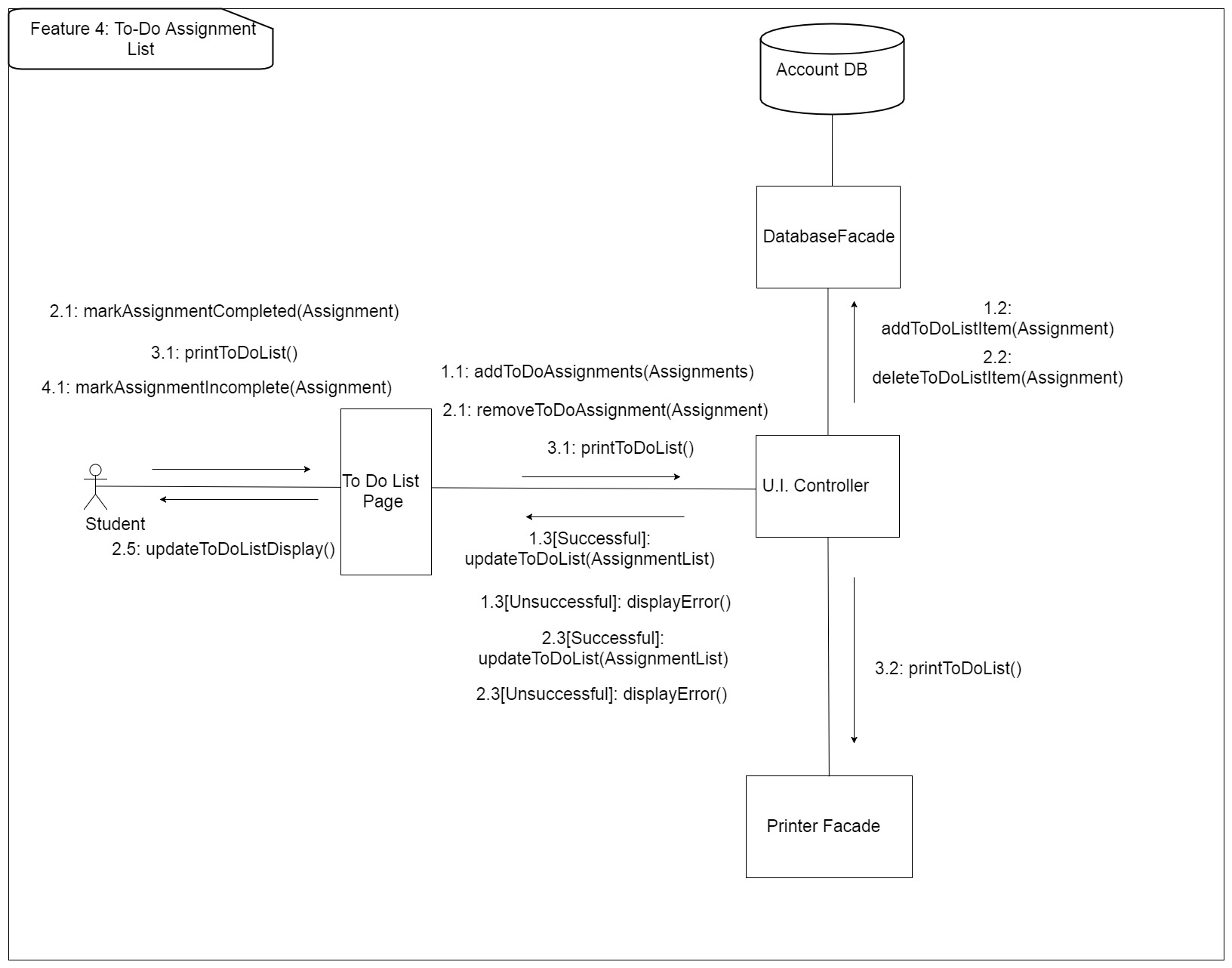
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/Assignment%20To-Do%20List%20UML%20Communication%20Diagram.jpg)

Figure 11 - To Do Assignment List UML Communication Diagram

Design Concerns Addressed:

* Calendar managers should be able to see a prioritized list of their assignments to increase their productivity.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.3.0 To-Do List: Load List

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.3.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.3.0.png)  Figure 12 - To Do Assignment List Screen Mockup |
| Page Title: | Load To-Do List |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Allow the user to see their list they have created of objectives they are trying to complete. |
| Parent User Story: | 3.2.1.1 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. User is on home page. |
| Post-conditions/Product(s) Produced: | To-Do list is shown with the user's assignments and events shown. |
| Links: | None. |
| SRS Document | 3.2.24 |
| Description/Notes: | There will be a button to access and launch the To-Do List found near the views tabs.  1. To-Do List launched will show events displayed to the user has scheduled for the active week. a. School assignments will be displayed as synced with I-Learn for the active week. b. The To-Do list will have sort (3.2.3.2) and filter (3.2.3.3) options. c. The To-Do list will be sorted according to due date by default, with assignments that are coming due soonest first. |

#### 3.2.3.1 To-Do List: Mark Assignment Done

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Mark Assignment Done |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Allow the user to keep track of assignments done by checking them as completed. |
| Parent User Story: | 3.2.3.0 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. User is on To-Do List. |
| Post-conditions/Product(s) Produced: | To-Do list is displayed with the user's incomplete events and assignments shown. |
| Links: | None. |
| SRS Document | 3.2.24 |
| Description/Notes: | 1. To-Do List launched will show events displayed to the user has scheduled for the active week. a. School assignments will be displayed as synced with I-Learn for the active week. 2. The To-Do item can be marked as completed. In order to accomplish this, the user will simply click on the item, and then select 'Mark as Completed' from the dialogue that appears.  a. The selected assignment will be removed so that the user will be able to focus on the next To-Do List item.  b. The item will be removed from the calendar once marked as completed. |

#### 3.2.3.2 To-Do List: Sort

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Sort |
| Author: | Keith Wheeler |
| Type: | Process. |
| Purpose: | To provide the user with a way to organize their assignments. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. The user's courses have been added into the calendar.  3. The user is on To-Do List View. |
| Post-conditions/Product(s) Produced: | The assignments on the To-Do list page are sorted by the user's selected method. |
| Links: | None. |
| SRS Document | 3.2.33 |
| Description/Notes: | The user will be presented with a variety of ways to sort their assignment, by class, by due date, or by priority. Once the user makes their selection the To-Do list page will be re-populated with their sorted results. There they will also have the option to view sorted results in an ascending or descending manner. |

#### 

#### 3.2.3.3 To-Do List: Filter

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Filter |
| Author: | Zane West |
| Type: | Process. |
| Purpose: | To provide the user with a way to filter assignments based on course, due-date, or priority. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. The user's courses have been added into the calendar.  3. The user is on To-Do List View. |
| Post-conditions/Product(s) Produced: | 1. A filtered To-Do list is presented based on the options selected by the user. |
| Links: | None. |
| SRS Document | 3.2.33 |
| Description/Notes: | 1. Filter a. The user opens the calendar and opens To-Do List.  b. The user selects the filter option and a pop-up box with the available filtering options will show by which to filter.  c. The user selects the course(s) they wish to view, or the date range they wish to view, or to filter by priority assignments.  d. The calendar view updates to reflect only assignments meeting the filter criteria.  2. Warning  a. The filter will require coursework be entered into the calendar for at least one course to be able to filter. If there are no courses then it will show "No courses" in the pop-up box after clicking filter.  b. The filter shall reset with each Calendar Manager session to prevent the user from missing filtered assignments. |

#### 

### 3.2.4 Calendar Integrations

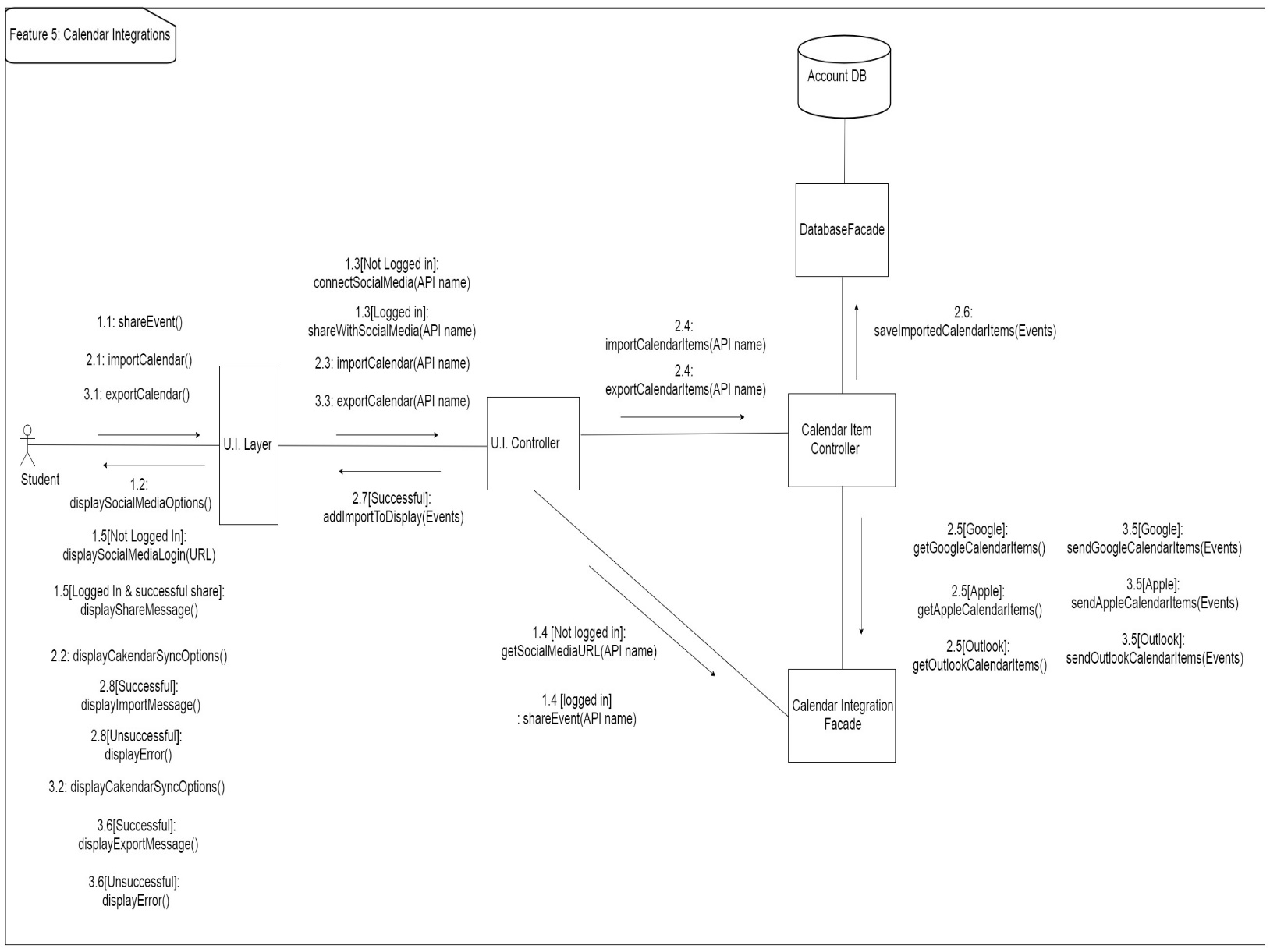
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/Calendar%20Integrations%20UML%20Communication%20Diagram.jpg)

Figure 13 - Calendar Integrations UML Communication Diagram

Design Concerns Addressed:

* Calendar managers should be able to use their existing calendar application in conjunction with ours.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.4.0 Calendar Integrations: Import Calendar

|  |  |
| --- | --- |
| **Data** | **Description** |
| ScreenShot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/Cal_upload_mock.PNG](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/Cal_upload_mock.PNG)  [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/Cal_name_mock.PNG](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/Cal_name_mock.PNG)  Figure 14 - Import Calendar Screen Mockup |
| Page Title: | Third Party Calendar Integration |
| Author: | Tyler Sorenson, Robert Nelson |
| Type: | Process. |
| Purpose: | Allow users to connect other supporting calendars to have everything in one place, instead of having to get on multiple calendars to check different objectives. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured. 2. User has made an account for the Calendar application with a registered e-mail address.  3. User has logged into the Calendar Portal.  4. User has exported their third-party calendars using the standard iCalendar format. |
| Post-conditions/Product(s) Produced: | 1. The user will be directed to upload their iCalendar File.  2. The uploaded iCalendar will be verified.  3. If an error is encountered with the iCalendar file the user will be prompted to verify file is correct and re-upload.  4. Upon successful upload the user's events will be displayed in the calendar. |

#### 3.2.4.1 Calendar Integrations Export Calendar

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Export Calendar |
| Author: | Tyler Sorenson, Robert Nelson |
| Type: | Process. |
| Purpose: | Allow the user to export the calendar to work with other supporting calendars. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured. 2. User has made an account for the Calendar application with a registered e-mail address.  3. User has logged into the Calendar Portal. |
| Post-conditions/Product(s) Produced: | 1. The user will download the calendar's events using the iCalendar format conforming to the IETF iCalendar standard. |
| Links: | 1. Standard for iCalendar: <https://tools.ietf.org/html/rfc5545> |
| SRS Document | 3.2.47-3.2.49 |
| Description/Notes: | 1. Supported third party calendar applications include: a. Google Calendar  b. Apple Calendar  c. Outlook Calendar  d. Any calendar that supports importing an iCalendar file  2. Exporting iCalendar  a. The user will click export on the calendar page.  b. The system will create an iCalendar file.  c. The iCalendar file will attempt to be downloaded.  d. The iCalendar file may then be imported into the supported third-party calendar applications by the user. |

### 3.2.5 User Profile Settings

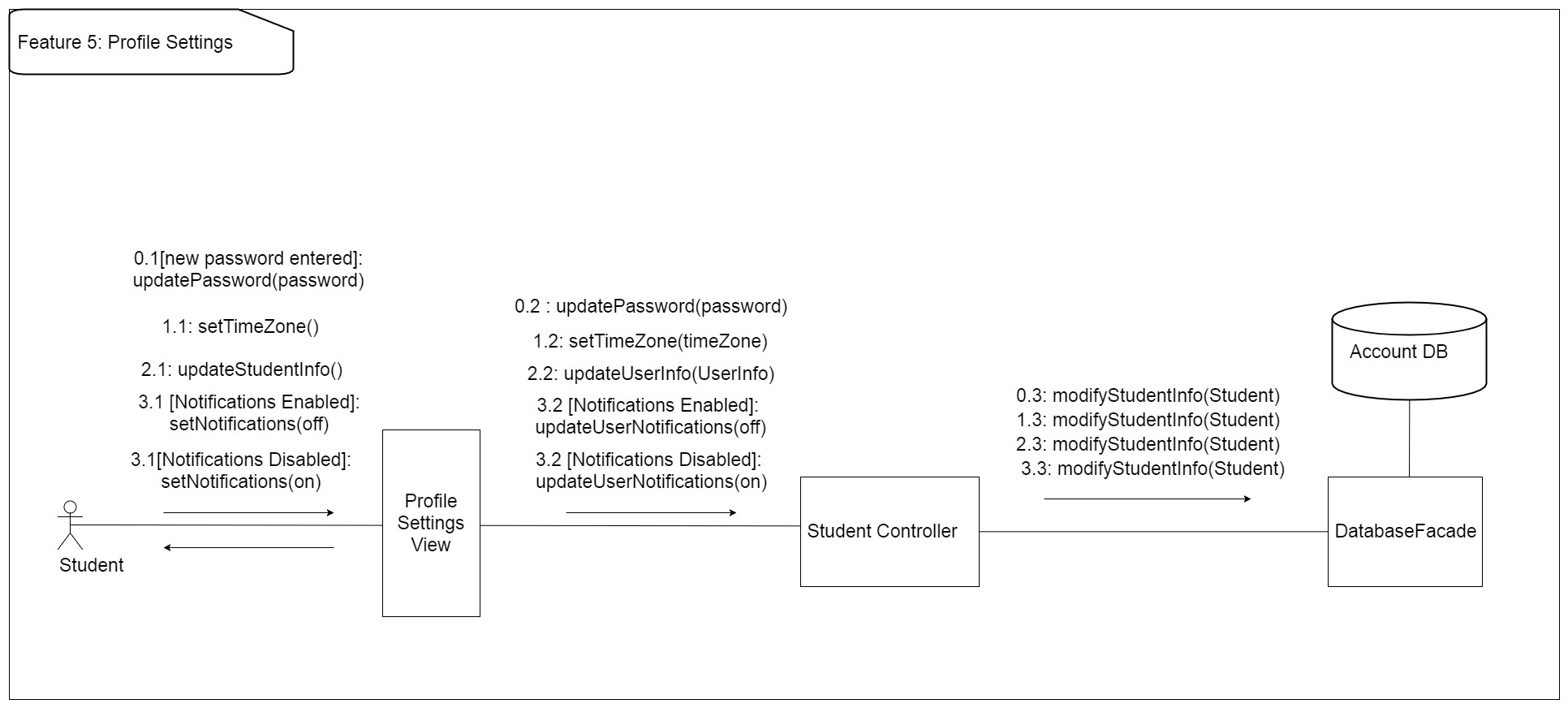
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/User%20Profile%20Settings%20UML%20Communication%20Diagram.jpg)

Figure 15 - User Profile Settings diagram

Design Concerns Addressed:

* Calendar managers should be able to change settings according to their preference.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

**3.2.5.0 User Profile Settings: Load Settings Page**

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.2.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.2.0.png) |
| Page Title: | Settings |
| Author: | Aaron Shore |
| Type: | Process. |
| Purpose: | Allow the user to change personal settings to their account and so that the notifications act as the user prefers. |
| Parent User Story: | none. |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Manager has an account.  2. User is logged-in. |
| Post-conditions/Product(s) Produced: | Settings page is up on the screen. |
| Links: | 3.2.2.0 |
| SRS Document | 3.2.57 |
| Description/Notes: | Password  1. Change Password  a. User enters the new desired password for their calendar log in into the "New" input field.  b. User confirms password by retyping into the "Confirmation" input field.  c. System verifies that the "New" input field and the "Confirmation" input field have the same text.  Notifications  1. Toggle Switch for Notifications  a. When notifications is switched on then a drop down will appear with all the different notifications the user can interact with. While switch is off then none of the notifications will be active nor will they show up on the screen to interact with.  2. Toggle Switch for Individual Notifications  a. Each notification can be turned on or off with the toggle switch next to each notification.  b. User will only receive notifications for those switched on. |

#### 3.2.5.1 User Profile Settings: Set Time Zone

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Set Time Zone |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Calendar Manager can choose a time zone so that the due dates and times are adjusted and correct for their current location. |
| Parent User Story: | 3.2.5.0 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | User is logged in. User is inside settings page. |
| Post-conditions/Product(s) Produced: | Settings page is open, and options are shown to the user. |
| Links: | None. |
| SRS Document | 3.2.15 |
| Description/Notes: | Time Zone option inside the settings menu can be selected and allows user to change to their current time zone. Thus, changing the due date times in sync with when they are due in MST.  1. Time Zone option is selected.  a. Shows the current selected time zone. b. The user can choose between all available time zones to reflect their current location. c. The selected time zone selected will reflect throughout the calendar by changing the due date times to adjust based on their selected time zone. |

#### 3.2.5.2 User Profile Settings: Change Profile Settings

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Change Profile Settings |
| Author: | Jefferson Santos |
| Type: | Process. |
| Purpose: | Allow the user to change and update personal information about themselves for their profile. |
| Parent User Story: | 3.2.5.2 |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in. 2.User is inside settings page. |
| Post-conditions/Product(s) Produced: | 1. If the Settings were validated, the message box with "Settings Saved" appears. 2. Settings page is refreshed, and settings changed are shown to the user. |
| Links: | None. |
| SRS Document | 3.2.57.1 |
| Description/Notes: | 1. Change profile option inside the settings menu can be selected.  2. Forms are showed with actual data.   Validation  1. Name input box will be filled out with the actual name.  2. Address input box will appear, with three lines to be filled out.  a. Address line 1, line 2 and line 3.  b. The zip code input box needs to follow the 5-digit numbers only.  3. Phone input box will appear and needs to be filled out with the following pattern xxx-xxx-xxxx to be validated.  4. E-mail input box will be filled out and validated with patterns of e-mail addresses, like [xxx@xxx.xxx](mailto:xxx@xxx.xxx).  5.School input box will be filled out with the complete name of the school and not with abbreviations e.g, BYU-I.  6. Button for images and the button to delete profile will be sent to the server to delete the profile or to upload the respective photo to his profile.   Observations  1. Any errors during the validation process, the respective input box will stay "red color" with the text in the side with the tip of data type must be filled out to be validated. |

#### 3.2.5.3 User Profile Settings: Change Notification Settings

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Change Notification Settings |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Allow the user to change their notifications so that they act the way the user prefers them. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in.  2. User goes to notification settings. |
| Post-conditions/Product(s) Produced: | 1. User's notification methods have been updated.  2. User is returned to their last viewed calendar. |
| Links: | 3.2.5.0 |
| SRS Document | 3.2.20 |
| Description/Notes: | Enable/disable and change settings for notifications  1. Notifications will be enabled by default.  2. The notification toggle will be found in the parent settings page.  3. Warnings  a. If notifications or notification permissions are blocked by the operating system, a prompt will notify the user of this situation.  b. The notification will have the option to redirect the user to the permissions page of their operating system or browser to correct the error.  c. The notification will have the option to never allow the permission and never remind the user of the error. 4. Configurable settings  a. How often they receive notifications. b. What email address or text messaging number to receive notifications at.  c. What they want to be notified about. |

### 3.2.6 Import Assignments from I-Learn

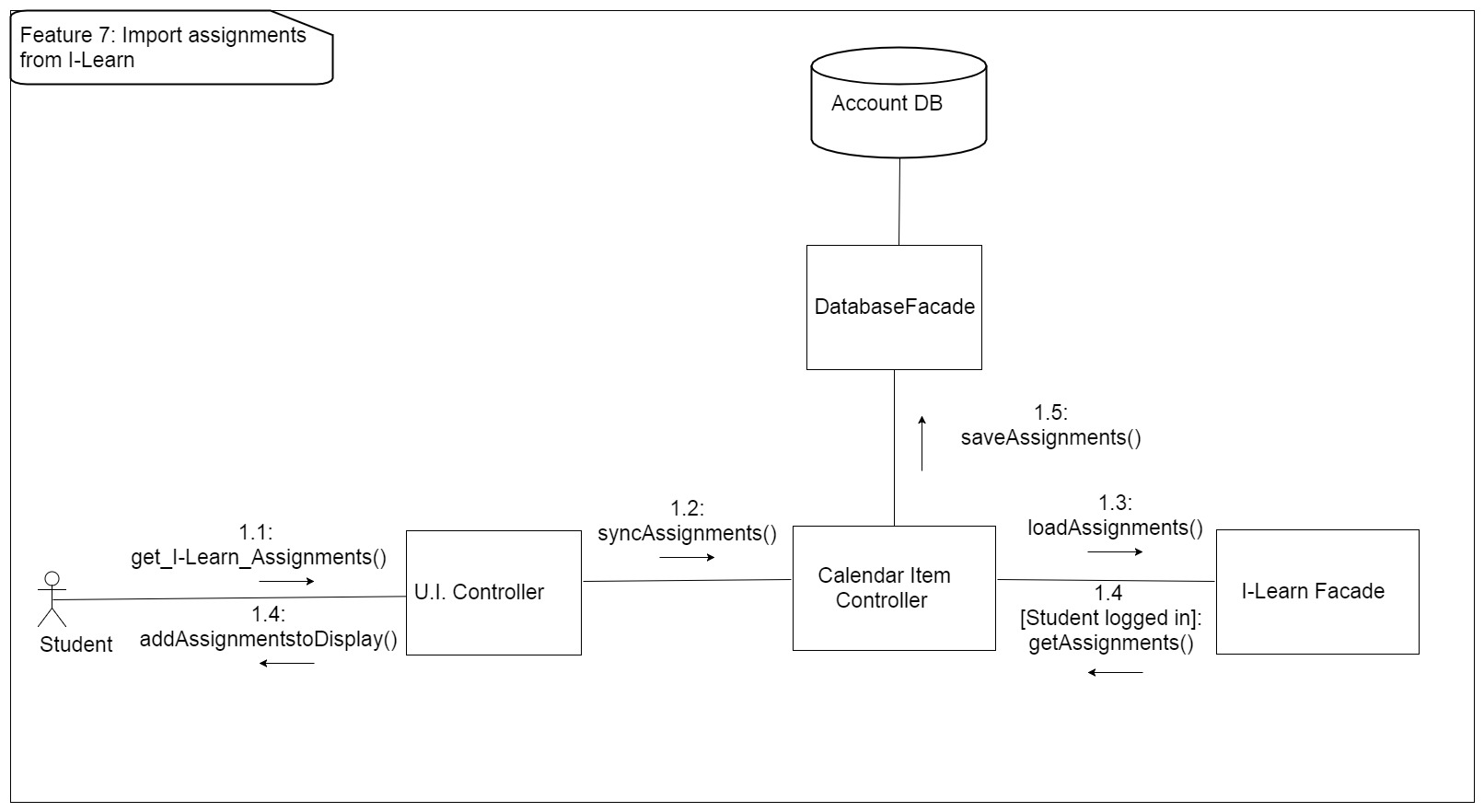
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/import%20from%20I-learn%20UML%20communication%20diagram.jpg)

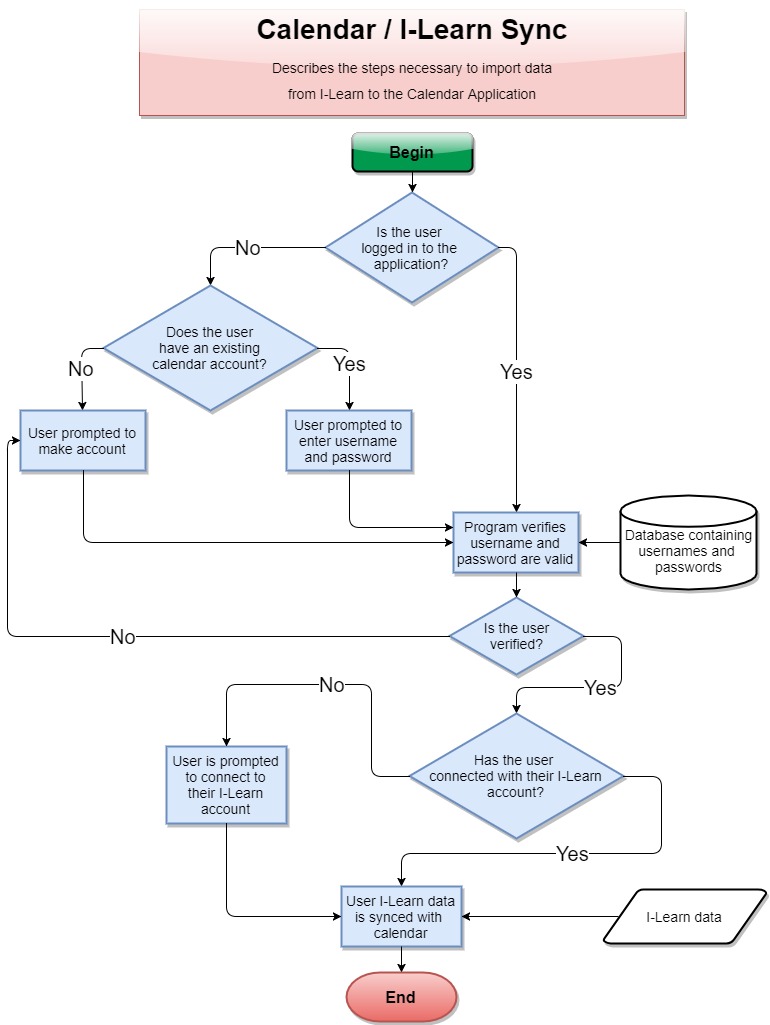
Figure 16 - Import Assignments from I-Learn Feature Diagram

Design Concerns Addressed:

* Calendar managers should be able to import assignments from I-Learn.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.6.0 Import Assignments from I-Learn: Manual Sync

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.6.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.6.0.png)  Figure 17 - Import Assignments from I-Learn: Manual Sync Mockup |
| Page Title: | Import Assignments from I-Learn |
| Author: | Joseph Keene |
| Type: | Process. |
| Purpose: | Allow the user to import assignments from I-Learn to see their due dates and descriptions of the assignments on the calendar. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged in on our Calendar App. |
| Post-conditions/Product(s) Produced: | 1. User is logged onto our Calendar App.  2. User is logged into their BYU-I account.  3. User has their assignments imported from I-Learn.  4. User can now access those assignments on our Calendar App. |
| Links: | None. |
| SRS Document | 3.2.6 |
| Description/Notes: | Importing Assignments from I-Learn  1. User clicks on sync button. 2. User is prompted to enter their BYU-I Credentials a. If user is already connected to their I-Learn account, this prompt is skipped. b. Once user has clicked "Submit", we check to see if the information is correct. c. If correct, they will be logged onto their BYU-I Account. If not, they will be prompted to re-enter their information. 3. Their Assignments on the calendar app will be synced with their assignments from I-Learn. a. A pop-up box will display to the user saying, "Please wait while your assignments are synced." 4. The user will be able to access their assignments now from our calendar application. |

[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/sync.jpg)

This image details the steps necessary to import I-Learn data into the calendar application.

#### 3.2.6.1 Import Assignments from I-Learn: Automatic Sync

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: | [https://github.com/MCLifeLeader/CS364/raw/master/SDD/resources/3.2.6.0.png](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/3.2.6.0.png)  Figure 18 - Import Assignments from I-Learn: Automatic Sync Mockup |
| Page Title: | Import Assignments from I-Learn Automatically |
| Author: | Joseph Keene |
| Type: | Process. |
| Purpose: | To have the assignments in I-Learn transfer over to the Calendar automatically to see their due dates and description of the assignments their in the Calendar. |
| Parent User Story: |  |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. User is logged into our Calendar Application. |
| Post-conditions/Product(s) Produced: | 1. User is logged into their BYU-I account.  2. User has the assignments synced automatically from I-Learn.  3. User is able to access their assignments on our Calendar Application. |
| Links: | None. |
| SRS Document | 3.2.6 |
| Description/Notes: | 1. User is prompted to log onto their BYU-I account when logging into the calendar application for the first time.  a. Once the user clicks the "Submit" button, we check to see if their information is correct. b. If correct, the User is logged onto their account. If not, we prompt them to re-enter their information. 2. User's assignments are synced automatically.  a. A pop-up box will display that says "Importing your Assignments from I-Learn. Please wait...". b. The pop-up box will disappear when we are done syncing their assignments. 3. The user is now able to access their assignments from our Calendar Application. |

### 3.2.7 Log-In Use case

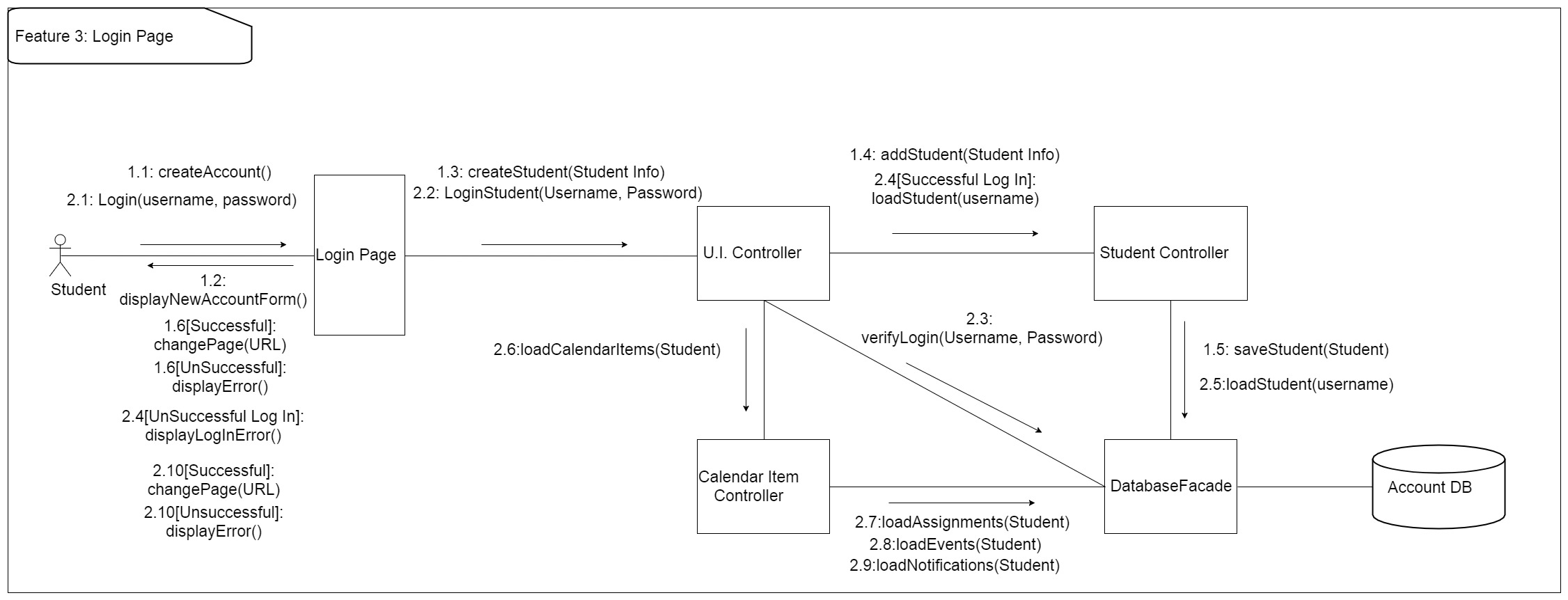
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/UML_Communication/Login%20(Non-SSO)%20UML%20Communication%20Diagram.jpg)

Figure 19 - Log-In Use Case Diagram

Design Concerns Addressed:

* Calendar managers should have their data be secure and only accessible by them.
* Developers and Testers should know of the expected inputs and outputs.
* Developers should implement error handling and data validation in their code.
* Testers should understand the processes associated to test all aspects of the product.
* Technical support should know how the interactions between the user and the calendar application take place, including both the correct and incorrect usages.

#### 3.2.7.0 Log-In Use Case: Create Account

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Create account |
| Author: | Brian Robertson |
| Type: | Process. |
| Purpose: | To add user to the data base for them to be able to use the Calendar Application. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured. |
| Post-conditions/Product(s) Produced: | The user will be sent an email for account verification. |
| Links: | None. |
| SRS Document | 3.2.32, 3.2.36 |
| Description/Notes: | 1. The user will be asked to input desired user name. 2. The user will be asked to input email address. 3. The user will be asked to input desired password. 4. The user will be instructed to click the link in an email they will receive to activate their calendar application account. 5. Following account creation, and activation, the user can then log onto the calendar app. |

#### 3.2.7.1 Log-in Use Cases: Validation

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Log-in Validation |
| Author: | Aaron Shore |
| Type: | Process. |
| Purpose: | To ensure that the correct user is logging into their corresponding account. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured.  2. User has made an account for the Calendar application. |
| Post-conditions/Product(s) Produced: | 1. If the user-name does not match any of the accounts created, the system will prompt the user to try again.  2. If the password does not match what the system has stored for the current user-name, then the system will prompt the user to try again.  3. When both the user-name and password match an account the system has stored then the user will be logged in. |
| Links: | None. |
| SRS Document | 3.2.36 |
| Description/Notes: | Validation  1. User-name input field  a. User-name input box will have auto focus by default.  b. As user types in their user-name, the auto-complete will detect previously used user-names for faster log in.  c. When login button is clicked system will verify to see if the user-name matches an account.  2. Password input field  a. Input field can be accessed by clicking or using the "tab" button on keyboard.  b. When login button is pushed it will verify that the password matches a user-name, which in full matches an account that has been created.  3. Login button  a. Button will be active by default  b. When button is clicked both the user-name and password input fields will be checked to match an account. c. While waiting for validation a loading spinner will appear.  d. IF user-name or password is not valid then user will be prompted to try again. e. When user-name and password match an account, the user will be logged in and their content will be loaded. |

#### 3.2.7.2 Log-Out Use Case

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Log-Out |
| Author: | Robert Nelson |
| Type: | Process. |
| Purpose: | Allow the user to sign out and close the application. |
| Parent User Story: | N/A |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar site is running, and user is logged in. |
| Post-conditions/Product(s) Produced: | The system logs the user out of the system. The user is taken to the log-in page. |
| Links: | None. |
| SRS Document | 3.2.32.2 |
| Description/Notes: | Primary path:  1. The user clicks on the "Sign Out" button.  2. The system will log the user out of the session. |

#### 3.2.7.3 Log-in Use Cases: Forgot Password

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Forgot Password. |
| Author: | Rex Nesbit |
| Type: | Process. |
| Purpose: | Help the user get back into their account if they have forgotten their password. |
| Parent User Story: | 3.2.7.1 Log-in Use Cases: Validation. |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured.  2. User has made an account for the Calendar application. |
| Post-conditions/Product(s) Produced: | 1. User is able to change their password to something they will remember, and to login to the system. |
| Links: | None. |
| SRS Document | 3.2.32 |
| Description/Notes: | Validation  1. User-name input field  a. User-name input box will have auto focus by default.  b. As user types in their user-name, the auto-complete will detect previously used user-names for faster log in.  c. When login button is clicked system will verify to see if the user-name matches an account.  2. Password input field  a. Input field can be accessed by clicking or using the "tab" button on keyboard.  b. When login button is pushed it will verify that the password matches a user-name, which in full matches an account that has been created.  3. Login button  a. Button will be active by default  b. When button is clicked both the user-name and password input fields will be checked to match an account. c. While waiting for validation a loading spinner will appear  d. If user-name or password is not valid then user will be prompted to try again 4. Forgot password button  a. When user selects the “forgot password” button, they will be asked to confirm the email address to have the reset password link sent to.  b. The user will then check their email and follow the link to set a new password. |

#### 3.2.7.4 Log-in Use Cases: Forgot Account

| **Data** | **Description** |
| --- | --- |
| Screenshot/Mockup: |  |
| Page Title: | Log-in Page |
| Author: | Robert Nelson |
| Type: | Process. |
| Purpose: | Help the user log back in to their account if they have forgotten their Username. |
| Parent User Story: | 3.2.7.1 Log-in Use Cases: Validation. |
| Stakeholders: | Calendar Manager, Developers, Testers, Technical Support |
| Pre-conditions/Product(s) Required: | 1. Calendar Portal has been brought up and configured.  2. User has made an account for the Calendar application with a registered e-mail address. |
| Post-conditions/Product(s) Produced: | The user will receive an e-mail with their username. |
| Links: | 1. Prepared Statements <https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html> |
| SRS Document | 3.2.32 |
| Description/Notes: | 1. Under the username field there will be a link that says, "Forgot Username."  2. The user will click the link. 3. The system will prompt the user for the email address associated with the account. 4. The system will accept input securely by using prepared statements.  5. The system will retrieve the username associated with the e-mail and send the username to the provided e-mail. 6. If the e-mail is not found, then the system will display an error message stating "The e-mail address is not registered to an account." |

#### 

# 4.0 Design Overview

The Design Overview will display a high-level representation of the Student Calendar Application's system design. The System Architecture section 4.1 presents a UML diagram to demonstrate the relationships between the software classes and how they will be integrated. This will provide the reader and user of the document a reference for the overall design. Further details of each design component is provided in section 4.2.

## 4.1 System Architecture

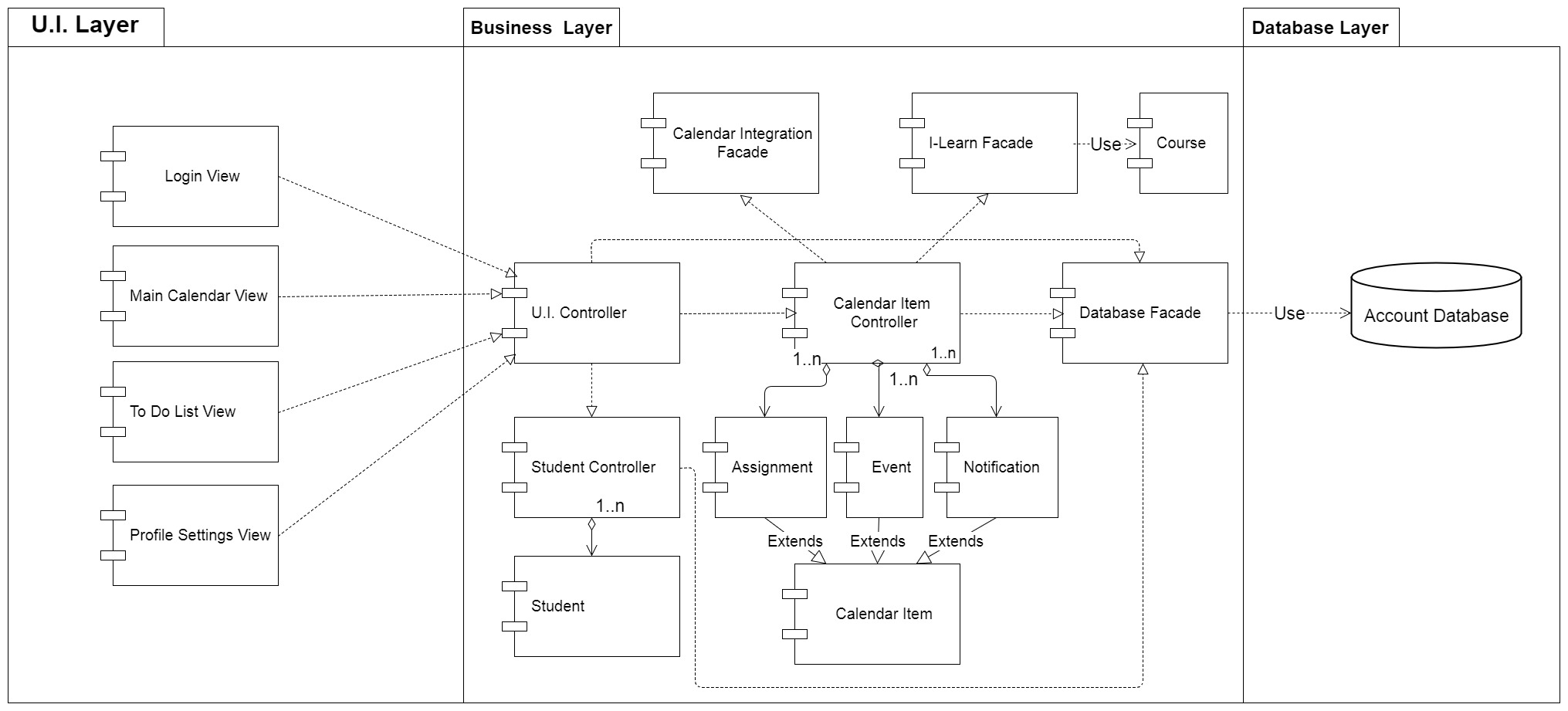
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/System%20Architecture%20UML.jpg)

Figure 20 - System Architecture Diagram

### 4.1.0 Architecture Description

The architecture design for the Student Calendar application is centered around the MVC(Model, View, Controller) design pattern which is a popular pattern for internet browser based applications. This pattern improves the system scalability, testability, and maintainability. Using the MVC pattern separates the visual front-end implementation from the back end development. This will allow developers to make modifications to the business and database layers without requiring the U.I. layer to be updated and vice versa.

The model classes contain all the data fields necessary for the insertion, modification, and deletion of student and calendar information. The model classes do not perform any functions that make changes to the system. Their sole purpose is to hold student and calendar data in order to make it easier for the system to internally store and pass around data. These model classes are accessed by controller classes.

The Controller classes contain all the methods that are necessary to work with the data in the model classes. These Controller classes also interface with the Database Facade class, allowing data to be retrieved from and stored into the system database. The I-Learn Facade class works with the Calendar Item Controller class to retrieve assignments from BYU-I's I-Learn service and store them in their appropriate location within the Student Calendar. The Calendar Integration Facade class allows the Student Calendar Application to interface with external API's in order to import/export calendar events from social media sites as well as other calendar applications.

The View classes are responsible for displaying information to and gathering input from the user. They send signals to and receive data from the U.I. controller class. The U.I. Controller class serves as a bridge between the U.I. layer and the Business layer. It receives signals from the View classes and makes the necessary function calls to update the system. It then sends any updated data back to the view classes so that changes may be displayed to the user.

The UML class diagram above serves as a pattern viewpoint demonstrating how each of the class entities is connected to one another. A further detailed description of each of these class entities, their attributes, and methods is provided in the section directly below.

## 

## 4.2.0 UML Class Descriptions

### 4.2.1 View Classes

The view classes perform all of the functions related to visually displaying the system to the user as well as allowing the user to interface with the system. They send signals to and receive data from the U.I. controller class.

### 4.2.x To-Do List View

|  |
| --- |
| **Class Name: ToDo List View** |
| This class represents the todo list view of the app. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| taskList | The list of I-Learn tasks that are to be displayed | |
| eventList | The list of user created events that are to be displayed. | |
| **Methods(operations)** |  |  |
| markAssignmentCompleted | **Method Description** |  |
| This method is to be called when the user marks an assignment completed (see 3.4.4.1) and takes the assignment object to be marked completed as a parameter |  |
| **Method Pseudo Code** |  |
| assignment.setCompleted(true) if assignment in taskList: taskList.remove(assignment) else if assignment in eventList: eventList.remove(assignment) |  |
| markAssignmentIncomplete | **Method Description** |  |
| This method is to be called when the user marks an assignment as incomplete and takes the assignment object to be marked completed as a parameter. |  |
| **Method Pseudo Code** |  |
| assignment.setCompleted(false) if isTask(assignment): taskList.add(assignment) if isEvent(assignment): eventList.add(assignment) |  |
| printTodoList | **Method Description** |  |
| This method is called when the print button is pressed from the todo list view, and it invokes the system procedures for printing a page |  |
| **Method Pseudo Code** |  |
| uiController.printTodoList() |  |
| updateTodoList() | **Method Description** |  |
| This method is called when the user clicks the todo list button again, refreshing the list of tasks |  |
| **Method Pseudo Code** |  |
| delete(eventList) delete(taskList) eventList = uiController.getEventList() taskList = uiController.getEventList() |  |

### 4.2.x Calendar Page Main View

|  |
| --- |
| **Class Name: Main Calendar Page View** |
| This class represents the main calendar view of the app. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| taskList | The list of I-Learn tasks that are to be displayed | |
| eventList | The list of user created events that are to be displayed. | |
| **Methods(operations)** |  |  |
| displayWeek(date) | **Method Description** |  |
| This method is called when the user toggles the display week view. It will display according to the currently opened calendar view (i.e. Monthly View, Day View). If Month View is active, Week View will list days Sunday through Saturday and number them in correspondence to the first week of the Month View. If Day View is selected, the method will find the corresponding week in Month View and will again display the week with a numbering consistent with the found week from Month View. This week will be passed to another function to organize the week with any matching event or assignment items, and create the elements necessary to display them in the application. The method “createCalendarWeek()” will also make style changes to a specific day if it matches the current date parameter. |  |
| **Method Pseudo Code** |  |
| weekArray <- (“Sun”, “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”) timeArray <- (“1am”, “2am”, “3am”… “11pm”)  weekArrayNumbering <- (“”) IF calendarDisplayElement == monthView weekArrayNumbering <- monthView week one numbering  ELSE IF calendarDisplayElement == dayView FOR all the days in monthView IF dayView numbering is found in a week of monthView weekArrayNumbering <- this week of monthView numbering ELSE return; calendarBody <- createCalendarWeek(weekArray, weekArrayNumbering, timeArray, getEvents(), getAssignments(), date) SET calendarDisplayElement <- calendarBody |  |
| displayMonth(date) | **Method Description** |  |
| This function toggles to a monthly view from any other view (weekly or daily) |  |
| **Method Pseudo Code** |  |
| monthNames <- ["January", "February", ... ,"December"] monthDays <- [1, 2, ..., 31] weekDays <- ["Mon", "Tue", ... , "Sun"] currentMonth <- getCurrentMonth(date) yearType <- isLeapYear(date) monthView <- drawCalendar(monthNames, monthDays, weekDays, currentMonth, yearType) SET calendarDisplayElement <- monthView |  |
| displayDay(date) | **Method Description** |  |
| This method sets up the daily view |  |
| **Method Pseudo Code** |  |
| dayNames <- (“Sun”, “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”) timeArray <- (“1am”, “2am”, “3am”… “11pm”) dayOfTheWeek <- getDayOfTheWeek(date, dayNames) dayView <- createCalendarDay(dayNames, dayOfTheWeek, timeArray,  getEvents(), getAssignments(), date)  SET calendarDisplayElement <- dayView |  |
| createEvent(eventDate, eventDescription) | **Method Description** |  |
| This method creates a new user created event (like other calendar tasks, but not from I-Learn; created by the user instead). The user will be prompted with a field that allows them to select a date/time and give the event a description. |  |
| **Method Pseudo Code** |  |
| API POST -> with eventDate and eventDescription in the body. IF RESPONSE.statusCode == 200 Dismiss modal and show success message. ELSE Show Error message |  |
| printCalendar() | **Method Description** |  |
| This method is called when the print button is pressed from the main calendar page view, and it invokes the system procedures for printing a page. Will open the default print dialog for printing a page in the web browser |  |
| **Method Pseudo Code** |  |
| uIController.print\_todo\_list() |  |
| followLink(externalURL) | **Method Description** |  |
| This method is called when a user clicks an external link |  |
| **Method Pseudo Code** |  |
| open externalURL in new tab |  |
| deleteEvent(event) | **Method Description** |  |
| This method is called when the user chooses to delete an event. |  |
| **Method Pseudo Code** |  |
| eventList.remove(event) |  |
| accessSettings() | **Method Description** |  |
| This function displays the settings page of the application. |  |
| **Method Pseudo Code** |  |
| document.backgroundColor = settings.darkColor userSettings <- loadUserSettings() |  |
| accessNotifications() | **Method Description** |  |
| This function displays the notifications received by the user. |  |
| **Method Pseudo Code** |  |
| document.backgroundColor = notifications.darkColor IF browser does not support notifications PRINT this browser does not support notifications IF notifications permissions are granted showNotifications() ELSE  requestNotificationPermission() |  |

### 4.2.x Login Page View

|  |
| --- |
| **Class Name: Login Page View** |
| The purpose of this class is to provide the user with a means or a gateway to access every functional aspect of the calendar application and draw from the feature sets contained within. It represents the user's identification and password as well as the methods needed to authenticate and create a new and existing accounts. It interfaces with the UI Controller class for the purpose of sending and verifying usernames and passwords. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| username | The identification used by the user to access the application. | |
| password | The secret string of characters used in conjunction with a username to gain access to the application. | |
| **Methods(operations)** |  |  |
| login(username, password) | **Method Description** |  |
| This method authenticates the user and logs them in. |  |
| **Method Pseudo Code** |  |
| uIController.LoginStudent(username, password); |  |
| createAccount() | **Method Description** |  |
| This method is to be called when the user attempts to create a new account with a new and unique username and new accompanying password. |  |
| **Method Pseudo Code** |  |
| createAccountButton.click() displayNewAccountForm() submitButton.click() SUBMIT newAccountForm Student Info <- input values from newAccountForm createStudent(Student Info) studentController.addstudent(Student Info) databaseFacade.saveStudent(Student) IF "successful" return to Login Page and alert user to successful account creation ELSE IF "unsuccessful" displayNewAccountForm() and alert user to error |  |
| displayNewAccountForm() | **Method Description** |  |
| This method is to be called when a user has opted to create a new account. It is responsible for providing a user with a form containing fields for input such as a new username, new password, etc. |  |
| **Method Pseudo Code** |  |
| url <- URL of page containing New Account Form window.location(url) |  |

### 4.2.2 Controller Classes

The Controller classes contain all the methods that are necessary to work with the data in the model classes. These Controller classes also include the facade classes which allow developers to more easily interact with interfaces such as the system database and external APIs.

#### 4.2.2.1 Calendar Item Controller

|  |
| --- |
| **Class Name: Calendar Item Controller** |
| Brief Description: The Calendar Item Controller (C.I. Controller) class is responsible for retrieving event and assignment data from the system and sending it to the user display. The C.I. Controller creates Assignments by accessing I-Learn through the I-Learn Facade class and stores the data in assignment class objects. The C.I. Controller stores calendar items into the system database by using the Database Facade class for future access. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Assignment List | The List of I-learn Assignments for the student | |
| Event List | The list of all calendar events that the user has created. | |
| Notification List | The list of all notifications that the user has created. | |
| I-Learn Facade | An Instance of the I-Learn Facade class that is used to pull data from the BYU-I I-Learn system. | |
| Database Facade | An Instance of the Database Facade class that is used to store and retrieve data from the account database. | |
| Student | The student class object. Holds the data for the student using the system. | |
| **Methods(operations)** |  |  |
| addEvent | **Method Description** |  |
| Creates a new Event calendar item to be added to the calendar. The user enters the event name, description, and sets how often the event will occur on the calendar. The weekly occurrence determines how often the event will reoccur for the long term(weekly, monthly, quarterly, etc.) and the daily occurrence determines which days of the week the event will occur on. This daily occurrence is passed in as a String array of days and is converted into an integer. This is called when the user selects the "Create Event" button. A new event is created with the passed parameters, stored in the database, and is then returned to the U.I. Controller class to display to the user. See use case 3.4.1.2 for details. |  |
| **Method Pseudo Code** |  |
| addEvent(eventName, eventDescription, eventWeeklyOccurrence, eventDailyOccurrence[]): eventOccurrenceNumber = convert Day Array into number(eventOccurrence[]) newEvent = create Event(eventName, eventDescription, eventFrequency, eventOccurrenceNumber) DatabaseFacade.addEvent(newEvent) return newEvent |  |
| loadAssignments | **Method Description** |  |
| This method loads assignments from I-Learn through using the I-Learn Facade class. The assignments are pulled from I-learn, stored in an array of assignment objects, and return to the C.I. Controller. The assignments are then stored into the database through the database facade class. Finally the assignments are sent to the U.I. Controller to be added to the calendar display. This method is called whenever the user chooses to import assignments from I-Learn. See use case 3.4.7.0 for details. |  |
| **Method Pseudo Code** |  |
| loadAssignments(User): I-LearnFacade.Login(User.username, User.password) assignments = I-LearnFacade.getAssignments() DatabaseFacade.addAssignments(assignments) return assignments |  |
| addNotification | **Method Description** |  |
| This method adds a notification to the system for the purpose of reminding the user of a calendar item at a later time. This method is called when the user creates a new notification for one of the items on their calendar. The calendar item object is passed into the method, a notification object is created and is linked to the object. The notfication is stored into the database. If the method is successful then a boolean is returned as True to indicate success to the U.I. If the method fails "False" is returned and the U.I. displays an error message. See use case 3.4.2 for details of when this function is used. |  |
| **Method Pseudo Code** |  |
| addNotification(CalendarItem): newNotification = create Notification(CalendarItem) DatabaseFacade.addNotification(newNotification) if successful: return true else return false |  |

#### 4.2.2.2 I-Learn Facade

|  |
| --- |
| **Class Name: I-Learn** |
| Brief Description: The I-Learn class will pull the user's information, assignments and courses from I-Learn using the Desire To Learn API. This class will interact with the Calendar Item Controller class to provide assignments (4.2.2.1). |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| credentials | The user's session credentials passed from I-Learn after authentication | |
| userId | The user's unique id that is used to retrieve student information | |
| **Methods(operations)** |  |  |
| login(username, password) | **Method Description** |  |
| Authenticate with I-Learn |  |
| **Method Pseudo Code** |  |
| credentials = post('<https://secure.byui.edu/cas/login?entityId=https://byui.brightspace.com/shibboleth-sp&service=https://shib.byui.edu/idp/Authn/Cas>', auth=(username, password)) userId = get('[https://byui.brightspace.com/d2l/api/lp/1.9/users/whoami')['id']](https://byui.brightspace.com/d2l/api/lp/1.9/users/whoami')%5b'id'%5d) |  |
| getCourse() | **Method Description** |  |
| This method uses the D2L API to get the users course list. It then filters to make sure the course is active |  |
| **Method Pseudo Code** |  |
| courseList = get('<https://byui.brightspace.com/d2l/api/lp/1.9/enrollments/myenrollments/?sortBy=-StartDate'>) FOR course in courseList['Items']    IF course['Access']['EndDate'] > currentDate and course['OrgUnit']['Type']['Id'] == 3       tempCourse = new Course       tempCourse.setName(course['OrgUnit']['Name'])       tempCourse.setId(course['OrgUnit']['Id'])       tempCourse.setStartDate(course['Access']['StartDate'])       tempCourse.setEndDate(course['Access']['EndDate'])       courses INSERT tempCourse RETURN courses |  |
| getAssignments() | **Method Description** |  |
| This method loops through each class and using the D2L API, pulls each assignment for that class. Each assignment is added to an assignment object and returned. |  |
| **Method Pseudo Code** |  |
| FOR course in courses    courseAssignments = get('<https://byui.brightspace.com/d2l/api/le/1.18/content/myItems/?orgUnitIdsCSV=>' + course.getId())    FOR assignment in courseAssignments       tempAssignment = new Assignment       tempAssignment.DueDate = assignment.DueDate       tempAssignment.courseName = course.getName()       tempAssignment.assignmentDescription = assignment.ItemName       IF assignment.DueDate != NULL          tempAssignment.assignmentCompleted = TRUE       ELSE          tempAssignment.assignmentCompleted = FALSE       tempAssignment.assignmentURL = assignment.href       assignments[course.getId()] INSERT tempAssignment RETURN assignments |  |
| getAssignmentURL(assignment) | **Method Description** |  |
| Returns the url of the passed in assignment |  |
| **Method Pseudo Code** |  |
| RETURN assignment.getAssignmentURL() |  |

#### 4.2.2.3 Student Controller

|  |
| --- |
| **Class Name: Student Controller** |
| Brief Description: The Student Controller class contains all the necessary methods for creating, modifying and deleting Student class objects. It interfaces with the Database Facade class (4.2.2.5) to save a student's information into the system database. It also loads a student's information from the Database Facade class. The Student Controller class is used by the U.I. Controller class (4.2.2.4) to pass student data between the system and the user. The Student Controller interfaces with the Log-In Use Case (3.2.7.0). It also interfaces with the Profile Settings use case (3.2.5.0 & 3.2.5.2) to set additional user settings. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Current Student | The student model class object that the Controller class is currently working with. | |
| **Methods(operations)** |  |  |
| addStudent | **Method Description** |  |
| Creates a new Student model object using input provided by the User. This is called when a new user registers with the system for the first time. |  |
| **Method Pseudo Code** |  |
| addStudent(name, username, password): newStudent = create Student(name, username, password) DatabaseFacade.saveStudent(newStudent) |  |
| loadStudent(username) | **Method Description** |  |
| Retrieves a student by username from the Database Facade and returns that student object |  |
| **Method Pseudo Code** |  |
| tempStudent = DatabaseFacade.loadStudent(username) RETURN tempStudent |  |
| updateUserNotifications(notificationToggle) | **Method Description** |  |
| Toggles notifications on or off for the student. |  |
| **Method Pseudo Code** |  |
| currentStudent.notifications = notificationToggle DatabaseFacade.modifyStudentInfo(currentStudent) |  |
| verifyEmail(Email) | **Method Description** |  |
| Verifies that the email address is associated with an account |  |
| **Method Pseudo Code** |  |
| IF currentStudent.email == Email    RETURN true ELSE    RETURN false |  |
| setTimeZone(timeZone) | **Method Description** |  |
| Sets the student's time zone |  |
|  |
|
| updatePassword(password) | **Method Description** |
| Updates the student's password in the database |
|
|
| updateUserInfo(UserInfo) | **Method Description** |
| Updates the current student with new user information |

#### 4.2.2.4 UI Controller

|  |
| --- |
| **Class Name: UI Controller** |
| Brief Description: The UI controller class is responsible for controlling everything the users can interact with. When a user interacts with anything in the app a request is sent to the UI controller class which will organize the information and send it to the appropriate class to be manipulated. If anything needs to be sent back for the user to view the UI controller class will receive the information and send it back to the UI layer to be displayed to the user. This class communicates with the database facade class to verify user credentials, the I-Learn facade class to retrieve assignment information, the student controller class to retrieve information about the current user, and the calendar item controller class to retrieve and update information about the user’s calendar. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| eventId | The current event being viewed by the user. | |
| searchQuery | The search information that was entered by the user into the search bar. | |
| startDate | The date and time that an assignment becomes available. | |
| endDate | The date and time that an assignment is due. | |
| class | The school class that the user’s assignment was assigned from. | |
| **Methods(operations)** |  |  |
| openCalendar | **Method Description** |  |
| Allows the application to have a functioning calendar. When the Calendar button is selected in the menu open the calendar view. The default view shows the current month. |  |
| **Method Pseudo Code** |  |
| getEvents(filters) Return databaseQuery(filters) |  |
| viewEvent | **Method Description** |  |
| Allows the user to have and view events. When an event is clicked on in the Calendar a page is brought up that gives the event details. |  |
| **Method Pseudo Code** |  |
| viewEvent(eventId) return event FROM eventTable WHERE eventId = eventId |  |
| viewSelection | **Method Description** |  |
| Lets the user view all of their events in a specific date range. After a range is selected by the user the event information for the events included in that range are displayed back to the user. |  |
| **Method Pseudo Code** |  |
| viewSelection(startDate, endDate, class) If class has no data return events FROM eventTable WHERE startDate = startDate AND endDate = endDate If only one class return events FROM eventTable WHERE startDate = startDate AND endDate = endDate AND class = class else return for each class in class events FROM eventTable WHERE startDate = startDate AND endDate = endDate AND class = class |  |
| createEvent | **Method Description** |  |
| Allows the user to create their own unique events. When the plus shaped button is selected on the bottom right of the calendar view it opens a form with different fields to be filled out by the user. |  |
| **Method Pseudo Code** |  |
| createEvent(eventInfo) New Event = CalendarItemController.addEvent(EventInfo) return New Event |  |
| searchCalendar | **Method Description** |  |
| This method lets the user search their entire calendar for events and assignments. When the search button is selected the search query that is filled into the search bar is compared to the data stored in the database for that user. All matching events and assignments are displayed on the screen. |  |
| **Method Pseudo Code** |  |
| searchCalendar(searchQuery) return events FROM eventTable WHERE eventName LIKE searchQuery OR eventDescription LIKE searchQuery |  |
| createAccount | **Method Description** |  |
| Lets the user create an account so they can save all of their personal information and come back to it later. When the user clicks create account the I-Learn login credentials are passed to the I-LearnFacade class which verifies if they are correct. |  |
| **Method Pseudo Code** |  |
| createAccount(username, password) { If (verifyAccount(username, password) Display “account created” else Display “invalid credentials” |  |
| login | **Method Description** |  |
| Allows the user to log back into their account and view all of their personal assignments and calendar. When the user clicks login the login credentials are passed to the databaseFacade class which verifies if they are correct. |  |
| **Method Pseudo Code** |  |
| login(username, password) { If (verifyLogin(username, password) Display “account created” else Display “invalid credentials” |  |
| printToDoList | **Method Description** |  |
| Allows the user to print out their To-Do list. When the user clicks the print button at the bottom of the To-Do list page the To-Do list is sent to the printerFacade class which handles the print job. |  |
| **Method Pseudo Code** |  |
| printToDoList(toDoList[]) { print(toDoList[]) } |  |
| printCalendar | **Method Description** |  |
| Allow the user to print out their calendar. When the user clicks the print button at the top of the calendar page the calendar is sent to the printerFacade class which handles the print job. |  |
| **Method Pseudo Code** |  |
| printCalendar(Calendar calendar) { print(calendar) } |  |
| createNotification | **Method Description** |  |
| This method is necessary to send the user notifications about their upcoming due dates and events. The calendarController class will check periodically for upcoming due dates. When an assignment’s due date is close it will send the information to the createNotification function in the UI controller class which will generate a push notification. |  |
| **Method Pseudo Code** |  |
| createNotification(Assignment assignment) { pushNotification(“Assignment: “ assignment.name “is due in 2 days.”) } |  |
| addToDoAssignment | **Method Description** |  |
| Allows the user to create their own entries to be added to the To-Do list. After the user clicks “Add To-Do Item” they are presented with a form to submit a custom entry in their To-Do list. Hitting submit will call this function which relays the information to the databaseFacade class where it will be stored in the database. |  |
| **Method Pseudo Code** |  |
| addToDoAssignment(toDo customToDo) { INSERT INTO toDoList VALUES(customToDo) } |  |
| removeToDoAssignment | **Method Description** |  |
| Removes a user specified item from the To-Do list. After the user clicks “Remove To-Do Item” next to a To-Do list entry this function relays the information to the databaseFacade class where it will be removed from the database. |  |
| **Method Pseudo Code** |  |
| removeToDoAssignment(toDo toDo) { REMOVE FROM toDoList VALUES(toDo) } |  |

#### 4.2.2.5 Database Facade

|  |
| --- |
| **Class Name: Database Facade** |
| Description: The Database Facade class is responsible for storing and retrieving data from the system database. The database facade class interfaces with each of the controller classes, allowing them to easily save and load data from the database. It's use in the overall scheme of the project is to allow the saving of user data to a given device. The abstraction of the database in this way allows for an easier implementation of its functions by other classes. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Connection | A connection that links the database facade to the database | |
| Database URL | A string containing the address to the database. | |
| Student | The student model class object that the Controller class is currently working with. | |
| **Methods(operations)** |  |  |
| saveEvent(Event) | **Method Description** |  |
| This method inserts a new event that a user has created into the system database. It returns true if the event was successfully saved and false if there was an error. |  |
| **Method Pseudo Code** |  |
| saveEvent(Event): Open connection to database using Connection Insert event into Event table in database Close Connection if successful: return true else: return false |  |
| SaveAssignments(Assignments[]) | **Method Description** |  |
| This method inserts a list of assignments retrieved from I-Learn into the system database. It returns true if the assignments were successfully saved and false if there was an error. |  |
| **Method Pseudo Code** |  |
| SaveAssignments(Assignments[]): Open connection to database using Connection for each assignment in the list of Assignments:    Insert assignment into assignment table in database Close Connection if successful: return true else: return false |  |
| saveNotification(Notification) | **Method Description** |  |
| This method inserts a new notification that the user has created into the system database.It returns true if the notification was successfully saved and false if there was an error. |  |
| **Method Pseudo Code** |  |
| saveNotification(Notification): Open connection to database using Connection Insert Notification into notification table in database Close Connection if successful: return true else: return false |  |
| saveStudent(Student) | **Method Description** |  |
| This method inserts a new Student into the database. This method is called whenever a user creates a new account, or the user has modified some of their account information. It returns true if the Student was successfully saved and false if there was an error. |  |
| **Method Pseudo Code** |  |
| saveStudent(Student): Open connection to database using Connection Insert Student into Student table in database Close Connection if successful: return true else: return false |  |
| verifyLogin(username, password) | **Method Description** |  |
| This method checks the database for a username and verifies that the password that is typed in by the user matches with the password that is contained in the database. If the username and password found in the database match what was typed in by the user the method returns true. If the username is not found in the database or the passwords do not match, the method returns false. |  |
| **Method Pseudo Code** |  |
| verifyLogin(username, password): Open connection to database using Connection Student = lookup username in Student table in database and return matching student Close Connection if username is not found:    return false  if Student.password is equal to password:    return true else: return false |  |

#### 4.2.2.6 Calendar Integration Facade

|  |
| --- |
| **Class Name: Calendar Integration Facade** |
| Brief Description: The Calendar Integration Facade class (3.4.2) is responsible for interacting with external calendar APIs or iCalendar files. One aspect of the interaction is importing events in an external calendar into the Student Calendar (3.2.4.0). Another aspect of the interaction is to export calendar events from the Student Calendar to an external calendar (3.2.4.0). |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Google Calendar API | An interface with Google Calendar. | |
| Apple Calendar API | An interface with Apple Calendar. | |
| Outlook Calendar API | An interface with Outlook Calendar. | |
| iCalendar | Holds iCalendar file information. | |
| **Methods(operations)** |  |  |
| getImportedCalendarItems() | **Method Description** |  |
| This method takes returns the imported calendar items. |  |
|  |
|
| exportCalendarItems(Event) | **Method Description** |
| This method exports an event into the desired calendar integration. |

### 4.2.3 Model Classes

The model classes contain all the data fields necessary for the insertion, modification, and deletion of student and calendar information. They do not perform any functions other than sending and updating their information through getters and setters.

#### 4.2.3.1 Calendar Item

|  |
| --- |
| **Class Name: Calendar Item** |
| Brief Description: The Calendar Item class holds the common information that is shared by any object that can be added and displayed on the calendar. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| name | The name of the individual item that is displayed on the calendar | |
| Start Time | The starting time of the Calendar Item. The value includes the hour and minute. This affects where the item is placed onto the calendar. | |
| Start Date | The starting day of the Calendar Item. This value includes the day, month, and year in the following format 'DD-MON-YYYY'. This affects where the item is placed onto the calendar. | |
| End Time | The Ending time of the Calendar Item. The value includes the hour and minute. This affects how much space the item takes on the calendar display. | |
| End Date | The Ending day of the Calendar Item. This value includes the day, month, and year in the following format 'DD-MON-YYYY'. This determines the last occurrence of a reoccurring event. | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |

#### 4.2.3.2 Assignment

|  |
| --- |
| **Class Name: Assignment** |
| Brief Description: The Assignment class is a type of calendar item. An assignment object contains all information related to a student's individual school assignment. When data is collected from I-Learn, each individual assignment is stored in an assignment instance and is then added into the Calendar's collection of visible items. This class includes all attributes that are found in the Calendar Item class. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Due Date | The time when the assignment is due on I-Learn. Includes the date and time. This affects where the item is placed onto the calendar. | |
| Course Name | The name of the course which the assignment belongs to. | |
| Assignment Description | The description of the individual assignment. | |
| Assignment Type | The type of the assignment. Used to set the priority level for the assignment. | |
| Assignment Completed | A boolean value that shows if an assignment has been completed | |
| Assignment URL | URL that points to the assignment | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |

#### 4.2.3.3 Event

|  |
| --- |
| **Class Name: Event** |
| Brief Description: The Event class is a type of calendar item. Unlike an assignment object, an event object is created by the user rather than I-Learn and does not relate to a school assignment. An event object contains all information relating to a future event that the user needs to add to their calendar. Most importantly an event object can be reoccurring in the calendar. This class includes all attributes that are found in the Calendar Item class. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| location | A brief description of the location for the event. | |
| event Description | A user generated description of the event. | |
| event frequency | a number representing how often the event is to be repeated. Frequencies include daily = 1 , weekly = 3, monthly = 7, quarterly = 11, semi-annually = 13, and annually = 17. | |
| event day occurrence | a number representing which days of the week the event is to be displayed on the calendar. Each day of the week is represented by a factor of 2(ex. Sunday = 1, Monday = 2, Tuesday = 4 etc.) each day checked by the user adds the corresponding day's value to get the final value, which determines the days that the event will be displayed. | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |
| Get day occurrence | **Method Description** |  |
| Returns a list of weekdays for which the event is to occur on the calendar. It does this by seperating each number value from the total value |  |
| **Method Pseudo Code** |  |
| getDayOccurrence(): dayArray = {sun, mon, tue, wed, thur, fri, sat} numberArray = {1, 2, 4, 8, 16, 32, 64} dayList = {} for each number in numberArray (starting at end){ if number >= event day occurrence: continue else: event day occurrence -= number add corresponding day to dayList } return dayList |  |

#### 4.2.3.4 Notification

|  |
| --- |
| **Class Name: Notification** |
| Brief Description: The Notification class is a type of calendar item. A notification object is essentially a reminder for the user and can be connected to any calendar item found within a user's calendar. A notification is created in one of two ways: The user creates a reminder for a calendar item and sets it for a specific time, or the user has set their calendar to notify them when a assignment or event's start time is approaching the current time. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| reminder time | The time for when the notification is to alert the user. Includes the date and time. | |
| calendar item | The item that the notification is connected to. | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |

#### 4.2.3.5 Student

|  |
| --- |
| **Class Name: Student** |
| Brief Description: The Student class holds all data related to a user and works with the student controller class to save new insertions and modifications of user data. The Student class interacts with the Student Controller (4.2.2.3) as a data structure to store student information. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| Name | The student's name | |
| Username | The student's username | |
| Password | The student's password | |
| Address | The student's address | |
| Zip Code | The student's zip code | |
| Phone Number | The student's phone number | |
| Email | The student's email | |
| School Name | The name of the student's school | |
| Notification | This value dictates whether notifications are on or off for the student | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |

#### 4.2.3.6 Course

|  |
| --- |
| **Class Name: Course** |
| Brief Description: The Course class stores the users Course pulled from I-Learn. This class primarily interacts with the I-Learn Facade (4.2.2.2) class as a data structure to store course information. |

|  |  |  |
| --- | --- | --- |
| **Attributes(Fields)** | **Attribute Description** | |
| id | The unique identifier for the course | |
| name | The name of the course | |
| startDate | The start date for the course | |
| endDate | The end date for the course | |
| **Methods(operations)** |  |  |
| Data getters | **Method Description** |  |
| This serves as a place holder for all “getter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| getValueofDataElement(): return dataElement |  |
| Data setters | **Method Description** |  |
| This serves as a place holder for all “setter” methods for the data elements that belong to this class. |  |
| **Method Pseudo Code** |  |
| setValueofDataElement(dataValue): dataElement = dataValue |  |

## 4.3 System Interfaces

### 4.3.1 User Interface

The user interface for the system will allow the user to interact with the Student Calendar app. The user will be able to add, remove, and modify any assignments and events on their calendar. The interface will include tabs at the top of the main calendar display to allow the user to switch between daily, weekly, and monthly views.

### 4.3.2 Software Interfaces

The software will need to interface with a database management system to retrieve data from and store data to a user account database. The connection will be a standard database connection.

## 4.4 Constraints and Assumptions

### 4.4.1 Constraints

#### 4.4.1.1 Web Browser Compatibility

Due to the large number of web browsers available and limited resources to ensure compatibility with every web browser, these browsers will be tested for compatibility: Google Chrome, Firefox, Chromium, Safari, and Microsoft Edge.

#### 4.4.1.2 Family Educational Rights and Privacy Act (FERPA)

The Student Calendar Integration Application will operate within FERPA, namely not disclosing educational records of any kind to parties other than the authorized user.

### 4.4.2 Assumptions

#### 4.4.2.1 University Authorization

It is assumed that the University will allow students to login to their university account through the Student Calendar Integration Application and continue to pull course, assignment, group, activity and grade information through the authorized account.

## 4.5 Error Handling

### 4.5.1 Database Facade Class Commit Errors

All commits to the database will be wrapped in a transaction. If any commit to the database fails during a procedure called in the database facade class, then the entire transaction will be rolled back, and an error message will be sent to the U.I. Layer.

### 4.5.2 Invalid Data Errors

All data entered into the system by the user will be validated to determine if the data type entered matches with expected input (ex. a student name field should not contain any numbers). If the user attempts to enter invalid data into the system an exception will be thrown, and the user will be notified that the data is invalid.

### 4.5.3 I-Learn Importing Errors

In the event that a user's credentials do not match up to their log in information for I-Learn, an error will be thrown, and the user will be prompted to reenter their credentials or cancel importing their assignments from I-Learn. If any error results in the I-Learn API as it attempts to import assignment information into the system, the entire import process will be aborted, and the user will be notified that an import error occurred.

### 4.5.4 UI Controller Class Errors

All data passing through the UI Controller will be validated before being sent to the UI layer. If the data appears to be corrupted a new request will be sent to the respective controller. If the same problem is retrieved a second time an error will be displayed to the user and sent to the administrative logs. If the information is missing or incomplete an error will be passed to the UI layer which will notify the user of the issue.

# 5.0 Data Design

## 5.1 Data Description

A Relational Database to hold the data and SQL to communicate with the database. It will provide the ability to handle the many relationships that are in the application. It will also be able to maintain data integrity while scaling with the anticipated user base.

## 5.2 Entity Relationship Diagram

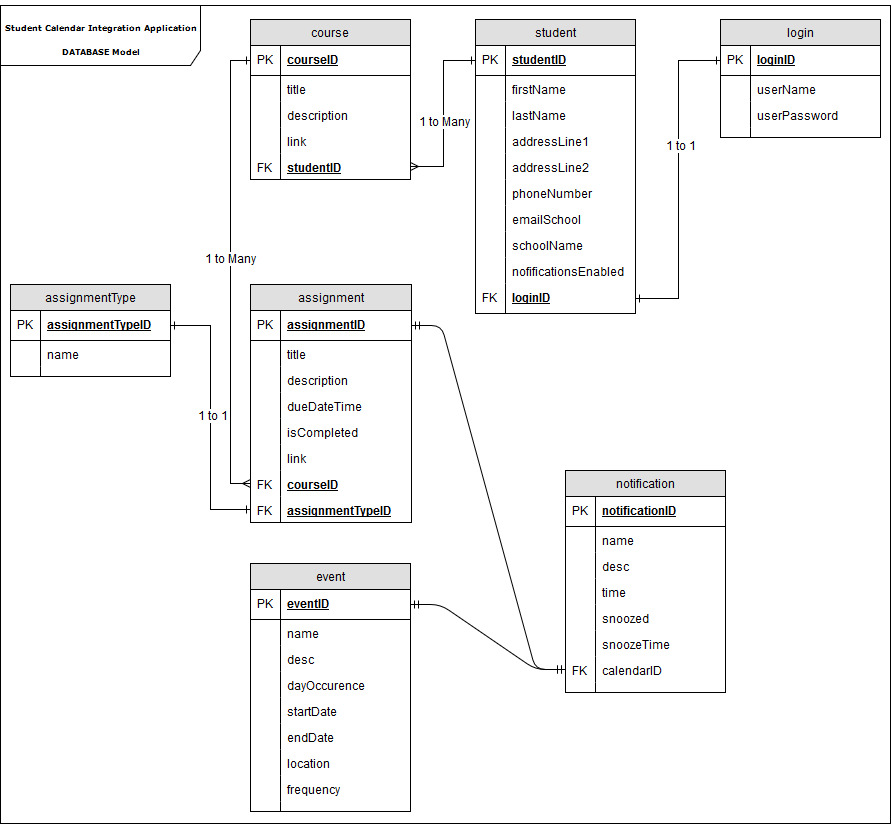
[](https://github.com/MCLifeLeader/CS364/blob/master/SDD/resources/FullDataModel.jpg)

Figure 21 - Entity Relationship Diagram

## 5.3 Data Dictionary

| **Table Name** | **Field** | **Type** | **Null** | **Default** |
| --- | --- | --- | --- | --- |
| Login | LoginId | int **PK Identity** | No | NA |
| Login | UserName | nvarchar(150) | No | NA |
| Login | UserPassword | nvarchar(150) | No | NA |
| Student | StudentId | int **PK Identity** | No | NA |
| Student | FirstName | nvarchar(35) | No | NA |
| Student | LastName | nvarchar(35) | No | NA |
| Student | AddressLine1 | nvarchar(50) | Yes | null |
| Student | AddressLine2 | nvarchar(50) | Yes | null |
| Student | PhoneNumber | int(9) | Yes | null |
| Student | EmailSchool | nvarchar(250) | No | NA |
| Student | SchoolName | nvarchar(150) | Yes | NA |
| Student | NotificationsEnabled | boolean | No | false |
| Student | LoginId | int FK Login | No | NA |
| Course | CourseId | int PK Identity | No | NA |
| Course | Title | nvarchar(150) | No | NA |
| Course | Description | nvarchar(300) | Yes | null |
| Course | Link | nvarchar(250) | Yes | null |
| Course | StudentId | int FK Student | No | NA |
| Assignment | AssignmentId | int PK Identity | No | NA |
| Assignment | Title | nvarchar(150) | No | NA |
| Assignment | Description | nvarchar(300) | Yes | null |
| Assignment | DueDateTime | datetime | Yes | null |
| Assignment | IsCompleted | boolean | No | false |
| Assignment | Link | nvarchar(250) | Yes | null |
| Assignment | CourseId | int FK Course | No | NA |
| Assignment | AssignmentTypeId | int FK AssignmentType | No | NA |
| AssignmentType | AssignmentTypeId | int PK Identity | No | NA |
| AssignmentType | Name | nvarchar(150) | No | NA |
| Notification | NotificationId | int PK Identity | No | NA |
| Notification | Name | nvarchar(150) | No | NA |
| Notification | Description | nvarchar(300) | Yes | null |
| Notification | Time | datetime | Yes | null |
| Notification | Snoozed | boolean | Yes | null |
| Notification | SnoozeTime | datetime | Yes | null |
| Event | EventId | int PK Identity | No | NA |
| Event | Name | nvarchar(150) | No | NA |
| Event | Description | nvarchar(300) | Yes | null |
| Event | DayOccurence | datetime | Yes | null |
| Event | StartDate | datetime | Yes | null |
| Event | EndDate | datetime | Yes | null |
| Event | Location | decimal(9,6) | Yes | null |
| Event | Frequency | int | Yes | null |