

▶ **BYU-IDAHO CS364**

**Student Calendar
Integration
Application**



**SOFTWARE
DESIGN
DOCUMENT**

VERSION 1.6
12/15/2017



/MCLifeLeader/CS364

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
Marcus Hedgecock
Jonah Green
Fernando Gomez
Michael Flindt

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.....	3
Project Manager.....	3
Team Leads.....	3
Team Members.....	3
Revision History.....	5
List of Figures.....	9
1.0 Introduction.....	12
1.1 Purpose.....	12
1.2 Document Target Audience.....	12
1.2.1 Developer.....	12
1.2.2 Tester.....	12
1.3 Scope.....	13
1.4 Context.....	13
1.5 References.....	15
2.0 Glossary.....	16
3.0 Use Cases.....	18
3.1 Identified Stakeholders and Design Concerns.....	19
3.1.1 Calendar Manager.....	19
3.1.2 Technical Support.....	20
3.2 Use Case Tables and Descriptions.....	21
3.2.1 Calendar Display.....	23

3.2.2 Calendar Event Notifications.....	47
3.2.3 To-Do List.....	61
3.2.4 Calendar Integrations	72
3.2.5 User Profile Settings.....	80
3.2.6 Import Assignments from I-Learn.....	91
3.2.7 Log-In Use case.....	100
4.0 Design Overview	114
4.1 System Architecture	115
4.1.0 Architecture Description.....	116
4.2.0 Class Descriptions.....	117
4.2.1 View Classes	117
4.2.2 Controller Classes.....	137
4.2.3 Model Classes.....	168
4.3 System Interfaces	180
4.3.1 User Interface.....	180
4.3.2 Software Interfaces.....	180
4.4 Constraints and Assumptions.....	181
4.4.1 Constraints.....	181
4.4.2 Assumptions.....	181
4.5 Error Handling	182
4.5.1 Database Facade Class Commit Errors.....	182
4.5.2 Invalid Data Errors.....	182

4.5.3 I-Learn Importing Errors	182
4.5.4 UI Controller Class Errors.....	182
5.0 Data Design.....	183
5.1 Data Description	183
5.2 Entity Relationship Diagram.....	184
5.3 Data Dictionary.....	185

List of Figures

Figure 1 - Projected User Base Chart	14
Figure 2 - Calendar Display UML Communication Diagram	23
Figure 3 - Calendar Home Screen Mockup	25
Figure 4 - Calendar Display: Checking the Schedule Mockup	28
Figure 5 - Calendar Event Creation Screen Mockup	30
Figure 6 - Calendar Display: Switching Views Mockup	32
Figure 7 - Calendar Display Screen: Creating/Deleting an Alert Mockup	35
Figure 8 - Assignment Access Screen Mockup	37
Figure 9 - Calendar Display: Show Assignment Details Mockup	40
Figure 10 - Holiday Display Screen Mockup	42
Figure 11 - Event Editing Screen Mockup	44
Figure 12 - Calendar Event Notifications UML Communication Diagram	47
Figure 13 - Notification Settings Screen Mockup	49
Figure 14 - Reminder Window Screen Mockup	52

Figure 15 - Calendar Event Notifications: View Missed Notifications Mockup	56
Figure 16 - Calendar Event Notifications: Set Notification Methods Mockup	59
Figure 17 – To-Do Assignment List UML Communication Diagram	61
Figure 18 – To-Do Assignment List Screen Mockup	63
Figure 19 - To-Do List: Mark Assignment Done Mockup	65
Figure 20 - To-Do List: Filter Mockup	69
Figure 21 - Calendar Integrations UML Communication Diagram	72
Figure 22.a - Import Calendar Screen Mockup	73
Figure 23.b - Import Calendar Screen Mockup	74
Figure 24 - Calendar Integrations Export Calendar Mockup	77
Figure 25 - User Profile Settings diagram	80
Figure 26 - Load Settings Page Mockup	82
Figure 27 - User Profile Settings: Set Time Zone Mockup	85
Figure 28 - User Profile Settings: Change Profile Settings Mockup	87
Figure 29 - Import Assignments from I-Learn Feature Diagram	91
Figure 30 - Import Assignments from I-Learn: Manual Sync Mockup	93

Figure 31 - Calendar / I-Learn Sync Chart	96
Figure 32 - Import Assignments from I-Learn: Automatic Sync Mockup	97
Figure 33 - Log-In Use Case Diagram	100
Figure 34 - Log-In Use Case: Create Account Mockup	101
Figure 35.a - Log-in Use Cases: Validation Mockup	103
Figure 36.b - Log-in Use Cases: Validation Mockup	104
Figure 37 - Log-Out Use Case Mockup	107
Figure 38 - Log-in Use Cases: Forgot Password Mockup	109
Figure 39 - Log-in Use Cases: Forgot Account Mockup	112
Figure 40 - System Architecture Diagram	115
Figure 41 - Entity Relationship Diagram	184

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 uses standards based the “IEEE Standards for Information Technology - Software Design Descriptions” [1]. The document utilizes requirements from the “Software Requirements Specification” [2] document and provides sufficient detail to implement the software product to stakeholder specifications. The documentation will describe the system architecture, the user interface, and the way that the system is to be implemented in order to meet design concerns.

1.2 Document Target Audience

1.2.1 Developer

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

1.2.1.1 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.

1.2.2 Tester

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

1.2.2.1 Design Concerns

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

1.3 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.

1.4 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. The application will be integrated with the OLP (Online Learning Platform) currently in use by the school, Brightspace. The application will be accessed via mobile or web interface. The purpose is to assist students with meeting deadlines to maximize learning outcomes.

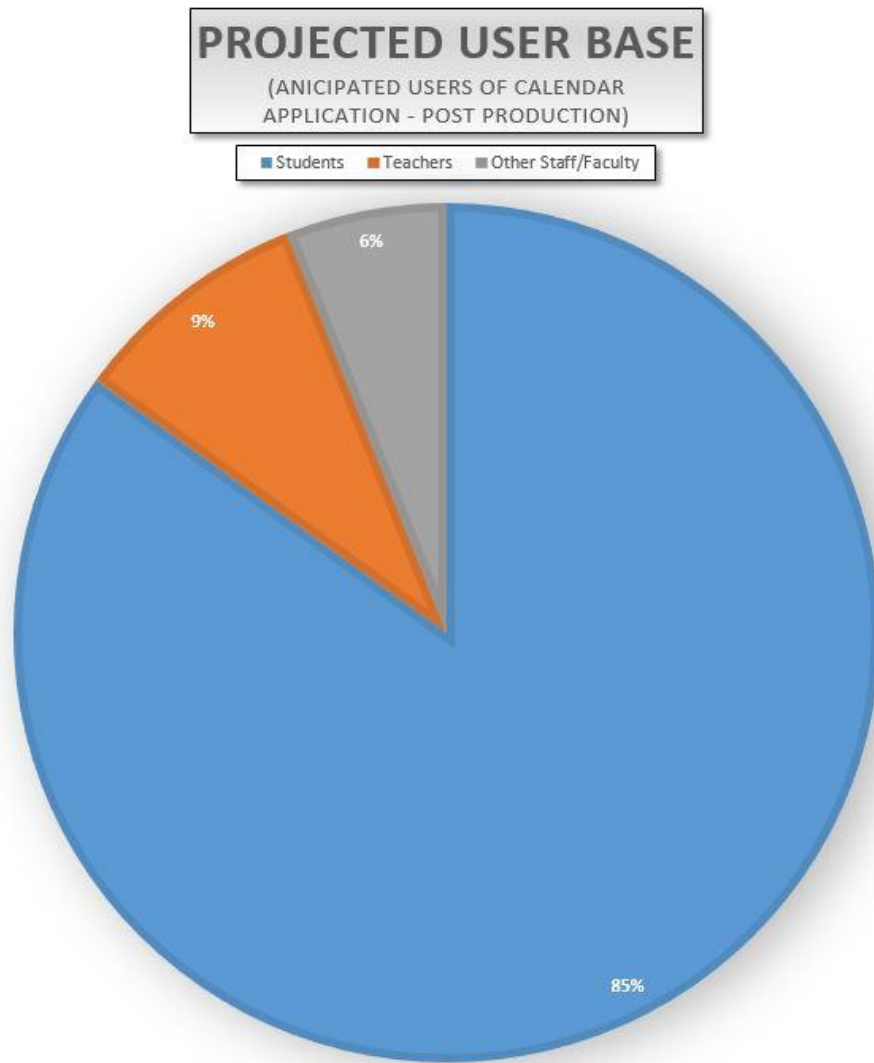


Figure 1 - Projected User Base Chart

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
- [3] Google Calendar API Reference. Last updated April 16, 2015. [Online] Available: <https://developers.google.com/google-apps/calendar/v3/reference/>
- [4] Apple Calendar EventKit Documentation. [Online] Available: <https://developer.apple.com/documentation/eventkit>
- [5] Outlook Calendar REST API Reference. [Online] Available: <https://msdn.microsoft.com/en-us/office/office365/api/calendar-rest-operations>
- [6] Brightspace API Reference. [Online] Available: <http://docs.valence.desire2learn.com/reference.html>

2.0 Glossary

Terms	Definitions
Actor	(In a UML context) Someone or something that interacts with a given system
API	<i>Application Programming Interface</i> ; 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	An attack on a system that involves "injecting" the attacker's own SQL code to perform unintended queries to a

Terms	Definitions
Injection	given database
UI	<i>User Interface</i> ; the space where interactions between humans and machines occur
UML	<i>Unified Modeling Language</i> ; 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	<i>User Experience</i> ; 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 section 3.1. In section 3.2, we provide several context viewpoints including UML communication diagrams to provide an overview of each viewpoint. We also provide 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.

Example: The function labeled "*0.1 addToDoAssignments()*" located in the To-Do Assignment List diagram (**Figure 17**), corresponds to use case 3.2.3.0 "Load List" while the function "*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 types of 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 user."

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 I-Learn.
- Calendar managers should have their data be secure and only accessible by them.

3.1.2 Technical Support

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

3.1.2.1 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 entities 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 3.2. 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 particular use case.

Data	Description
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

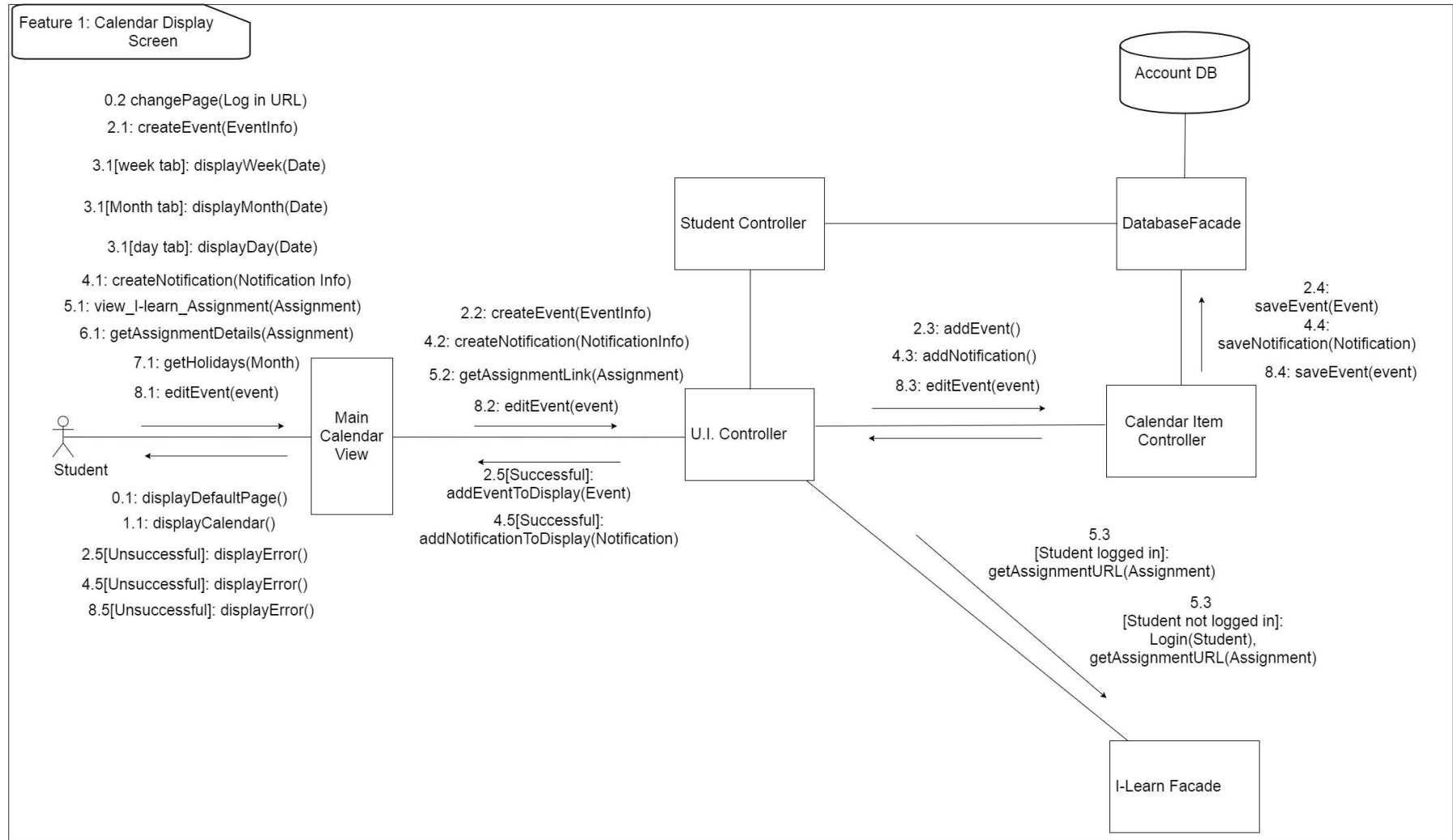
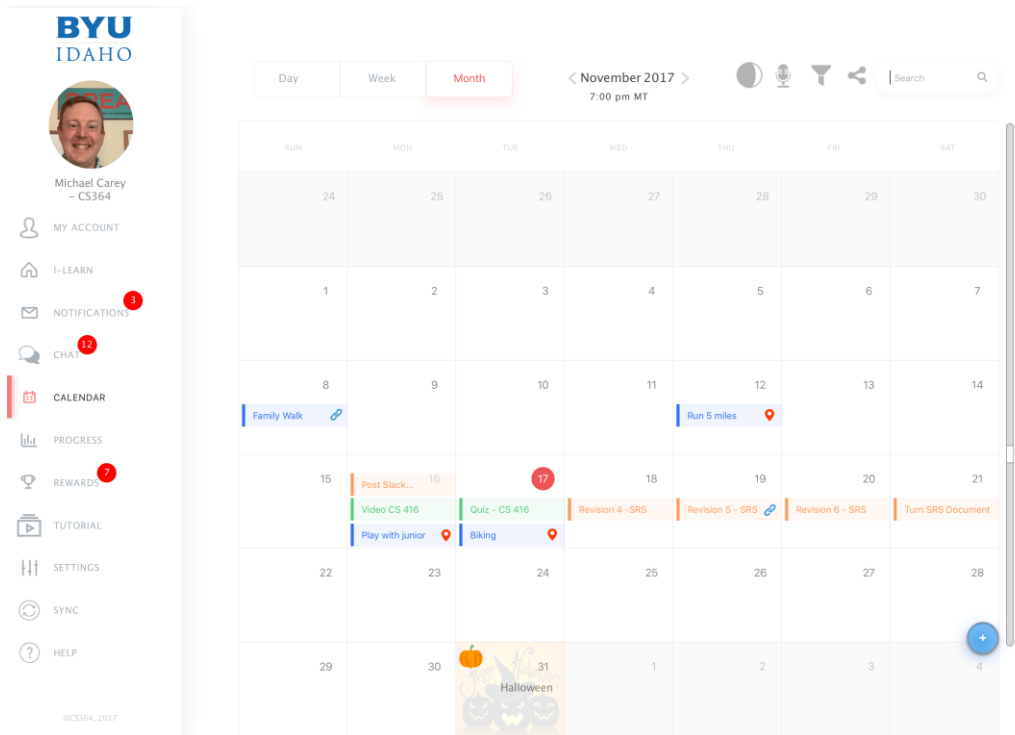


Figure 2 - Calendar Display UML Communication Diagram

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.

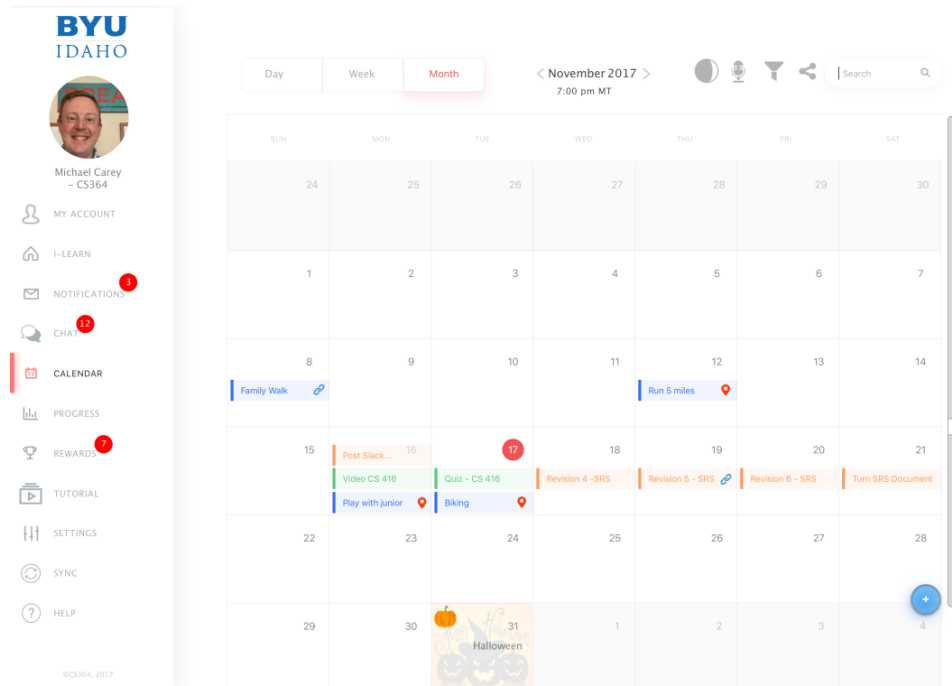
3.2.1.0 Calendar Display: Load Home Screen

Data	Description
<p>ScreenShot/Mockup:</p>	 <p>Figure 3 - Calendar Home Screen Mockup</p>
<p>Page Title:</p>	<p>Home Page</p>
<p>Parent User Story:</p>	<p>N/A</p>

Data	Description
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:	<ol style="list-style-type: none"> 1. The calendar user will navigate to the URL of the calendar website. 2. The website's home page will be loaded and displayed. <ol style="list-style-type: none"> 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.

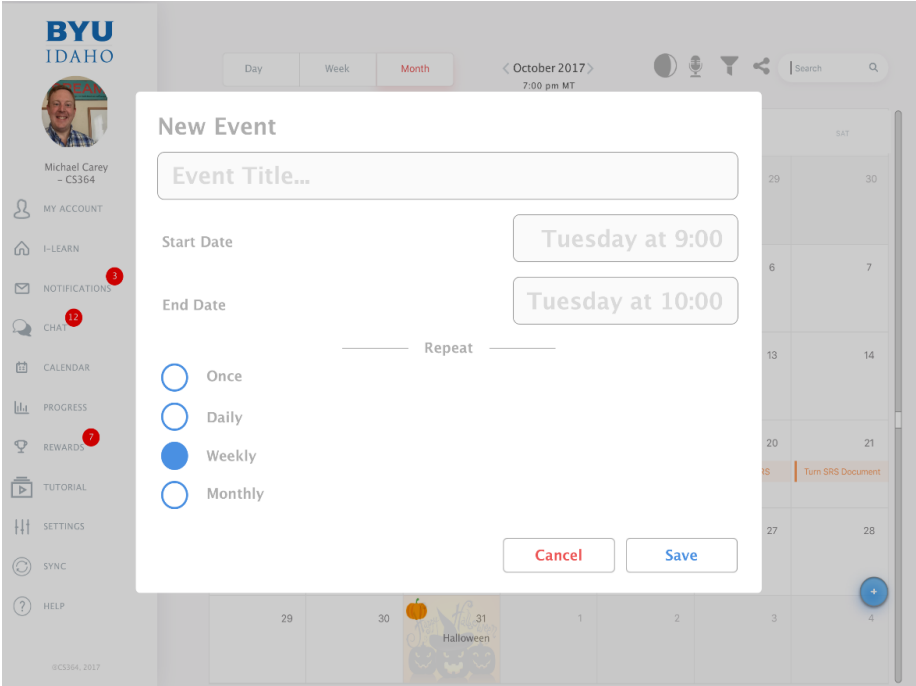
Data	Description
	4. Pages that users try to access but do not exist on the web site domain will show an error message and have a link back to the home page.

3.2.1.1 Calendar Display: Checking the Schedule

Data	Description
<p>ScreenShot/Mockup:</p>	 <p>The screenshot displays a mobile application interface for a user named Michael Carey (ID: CS364) at BYU Idaho. The main focus is a calendar view for November 2017, showing a weekly layout from Sunday to Saturday. The calendar includes several events: 'Family Walk' on Sunday, November 8th; 'Run 5 miles' on Thursday, November 12th; 'Post Stack...' on Friday, November 15th; 'Video CS 416' on Saturday, November 16th; 'Quiz - CS 416' on Sunday, November 17th; 'Revision 4 - SRS' on Monday, November 18th; 'Revision 5 - SRS' on Tuesday, November 19th; 'Revision 6 - SRS' on Wednesday, November 20th; 'Turn SRS Document' on Thursday, November 21st; 'Play with junior' on Friday, November 22nd; and 'Biking' on Saturday, November 23rd. A 'Halloween' event is also visible on Sunday, November 30th. The interface includes a navigation menu on the left with options like 'MY ACCOUNT', 'I-LEARN', 'NOTIFICATION', 'CHAT', 'CALENDAR', 'PROGRESS', 'REWARDS', 'TUTORIAL', 'SETTINGS', 'SYNC', and 'HELP'. The top of the calendar shows the current date and time (7:00 pm MT) and navigation controls for switching between Day, Week, and Month views.</p> <p>Figure 4 - Calendar Display: Checking the Schedule Mockup</p>
<p>Page Title:</p>	<p>Checking the Schedule</p>
<p>Parent User Story:</p>	<p>N/A</p>

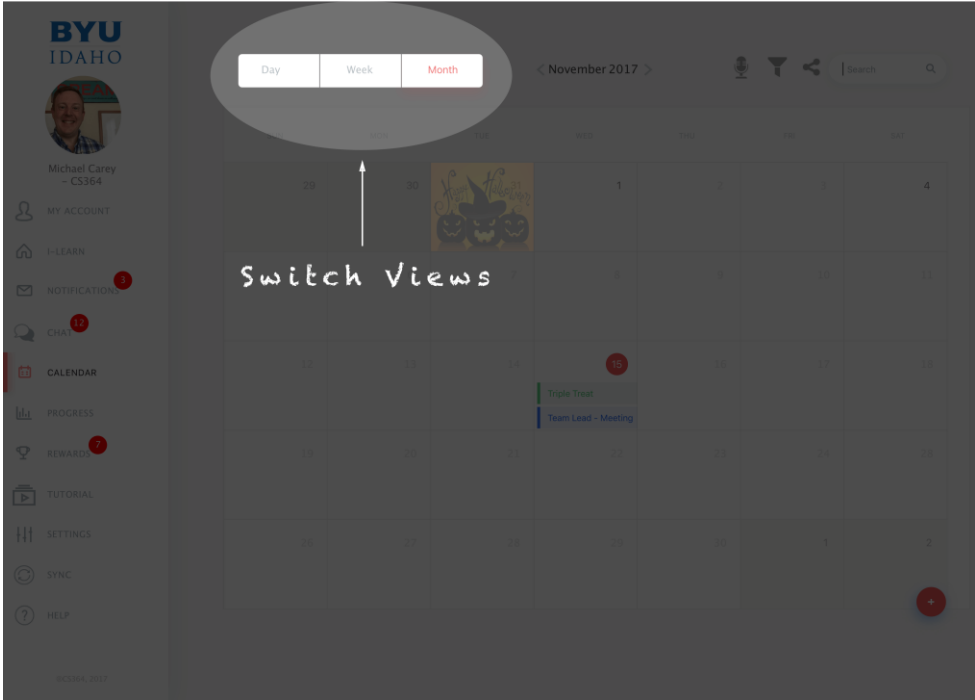
Data	Description
Actor(s)/Persona(s):	Calendar Users
Pre-conditions/Product(s) Required:	User is logged in.
Post-conditions/Product(s) Produced:	Calendar is up on display.
Links:	None
SRS Document	3.2.2
Description/Notes:	<ol style="list-style-type: none"> 1. When the user goes to the home page the calendar will be shown with the events the user has on their calendar. <ol style="list-style-type: none"> 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. 3. The assignment titles and due time will be displayed and resized to fit on the screen. <ol style="list-style-type: none"> a. If the assignment titles won't fit on a calendar day then the calendar will be resized to be larger. 4. The user can click the assignment to view its details.

3.2.1.2 Calendar Display: Creating an Event

Data	Description
<p>Screenshot/Mockup:</p>	 <p>Figure 5 - Calendar Event Creation Screen Mockup</p>
<p>Page Title:</p>	<p>Creating an Event</p>
<p>Author:</p>	<p>Jacob Keene</p>
<p>Type:</p>	<p>Process</p>

Data	Description
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.
Links:	None
SRS Document	3.2.1.6
Description/Notes:	<p>Creating an Event</p> <ol style="list-style-type: none"> 1. Create Event button is displayed on the Calendar App. <ol style="list-style-type: none"> a. User clicks Create Event button. b. User prompted to select day(s) to schedule event. c. User enters description of event. 2. Event is added and displayed on User's Calendar. <ol style="list-style-type: none"> a. User can interact with Event on Calendar.

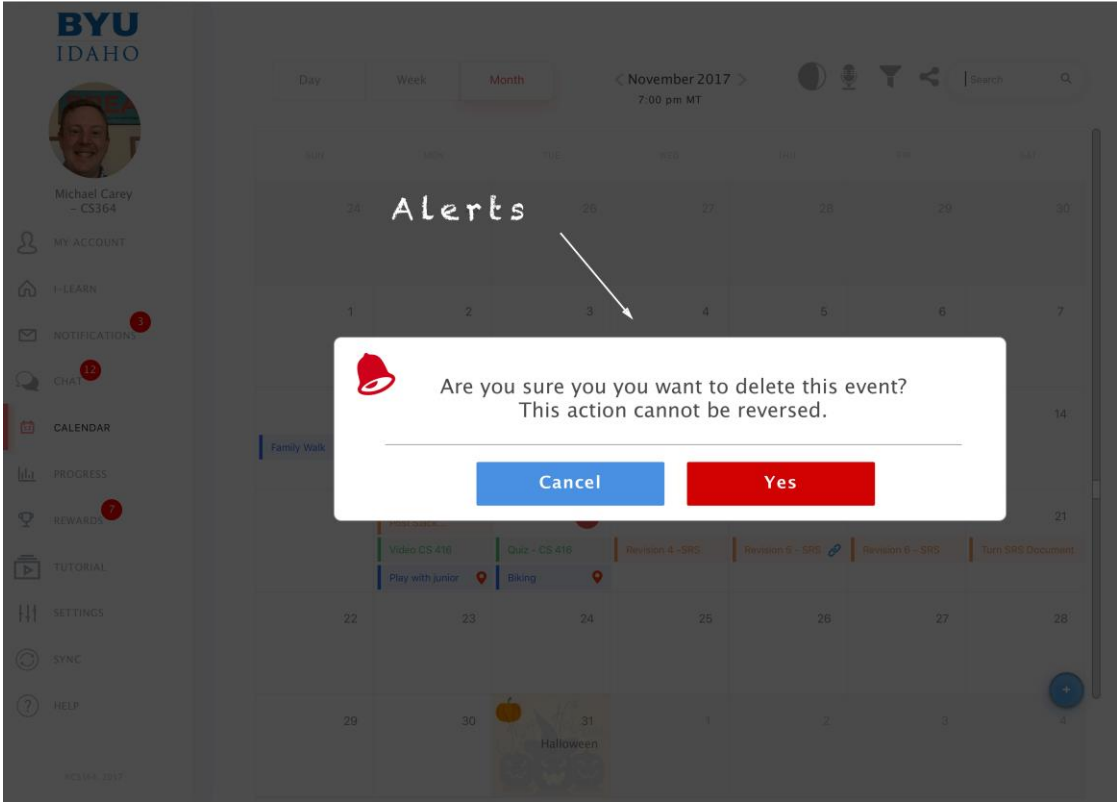
3.2.1.3 Calendar Display: Switching Views

Data	Description
<p>Screenshot/Mockup:</p>	 <p><i>Figure 6 - Calendar Display: Switching Views Mockup</i></p>
<p>Page Title:</p>	<p>Switching Views</p>
<p>Author:</p>	<p>Rex Nesbit</p>
<p>Type:</p>	<p>Process</p>

Data	Description
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.
Links:	None
SRS Document	3.2.53
Description/Notes:	<p>There will be 3 different tabs to switch between calendar views: Weekly, Monthly, and Daily.</p> <ol style="list-style-type: none"> 1. Weekly View Tab <ol style="list-style-type: none"> a. User can click the Weekly Tab if they are in monthly or daily view to view the Calendar for the current week. b. Events will be displayed as user has scheduled for the active week. c. School assignments will be displayed as synced with I-Learn for the active week. d. The next arrow button can be clicked to advance the active week forward one week. e. The previous arrow button can be clicked to go back one week from the active week.

Data	Description
	<p>f. Weekly view will be up by default when the display is loaded.</p> <p>2. Monthly View Tab</p> <ul style="list-style-type: none"> a. User can click the Monthly Tab to view the Calendar for the current month. b. Events will be displayed as user has scheduled for the active month. c. School assignments will be displayed as synced with I-Learn for the active month. d. The next arrow button can be clicked to advance the active month forward one month. e. The previous arrow button can be clicked to go back one month from the active month. <p>3. Daily View Tab</p> <ul style="list-style-type: none"> a. User can click the Daily Tab to view the Calendar for the current day. b. Events will be displayed as user has scheduled for the active day. c. School assignments will be displayed as synced with I-Learn for the active day. d. The next arrow button can be clicked to advance the active day forward one day. e. The previous arrow button can be clicked to go back one day from the active day.

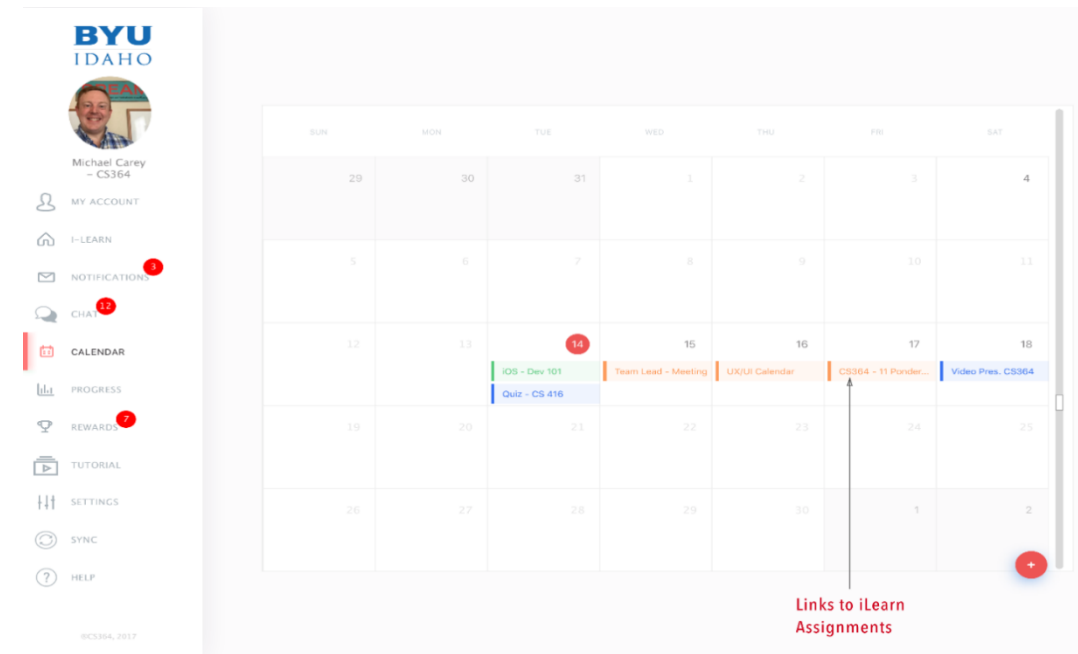
3.2.1.4 Calendar Display Screen: Creating/Deleting an Alert

Data	Description
<p>Screenshot/Mockup:</p>	 <p>The screenshot shows the BYU-Idaho mobile application interface. On the left is a navigation menu with options like 'MY ACCOUNT', 'I-LEARN', 'NOTIFICATIONS', 'CHAT', 'CALENDAR', 'PROGRESS', 'REWARDS', 'TUTORIAL', 'SETTINGS', 'SYNC', and 'HELP'. The main area displays a calendar for November 2017. A white confirmation dialog box is overlaid on the calendar, asking 'Are you sure you you want to delete this event? This action cannot be reversed.' with 'Cancel' and 'Yes' buttons. A white arrow points to the 'Alerts' text on the calendar grid.</p> <p><i>Figure 7 - Calendar Display Screen: Creating/Deleting an Alert Mockup</i></p>
<p>Page Title:</p>	<p>Creating/Deleting an Alert</p>

Data	Description
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:	<ol style="list-style-type: none"> 1. An application event requiring an alert occurs. 2. An error occurs that requires immediate user action or notification.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 1. The user is presented with the recommended course of action or the option to ignore.
Links:	None.
SRS Document	3.2.22
Description/Notes:	<ol style="list-style-type: none"> 1. Warnings <ol style="list-style-type: none"> a. An alert will appear when the user has performed or intends to perform an illegal or high-risk action, e.g.: creating an event without reminders or deleting a calendar event.

Data	Description
	2. Reminders a. Alerts will be used as a reminder in certain situations, e.g.: reminding the user that they have not synced their calendar in x days.

3.2.1.5 Calendar Display: Access Assignment

Data	Description
Screenshot/Mockup:	 <p>Figure 8 - Assignment Access Screen Mockup</p>

Data	Description
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:	<ol style="list-style-type: none"> 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:	<p>Accessing an Assignment.</p> <ol style="list-style-type: none"> 1. Assignments are displayed to the User on the Calendar as links.

Data	Description
	<ul style="list-style-type: none"> a. Users clicks on one of the links. b. The correct assignment is pulled up on screen for the user. <ul style="list-style-type: none"> 2. Assignment details button is shown. <ul style="list-style-type: none"> a. The user clicks assignments details button. b. The user is able to interact with different features for the event.

3.2.1.6 Calendar Display: Show Assignment Details

Data	Description

Data

Description

Screenshot/Mockup:

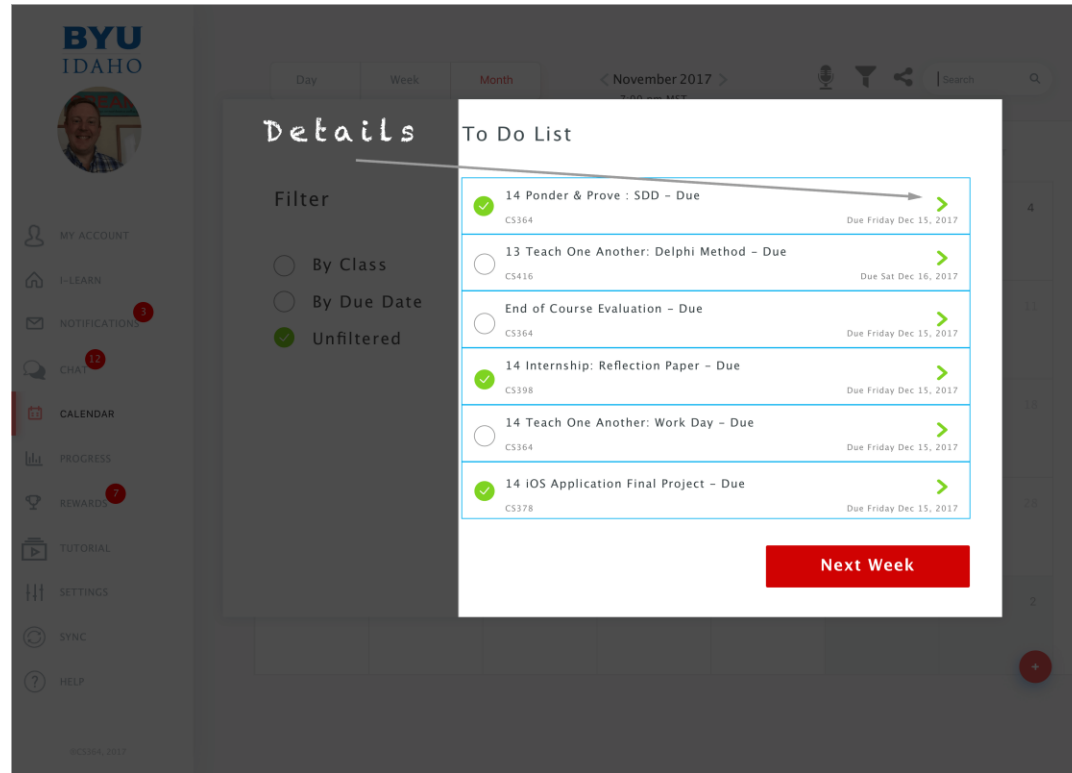


Figure 9 - Calendar Display: Show Assignment Details Mockup

Page Title:

Show Assignment Details

Author:

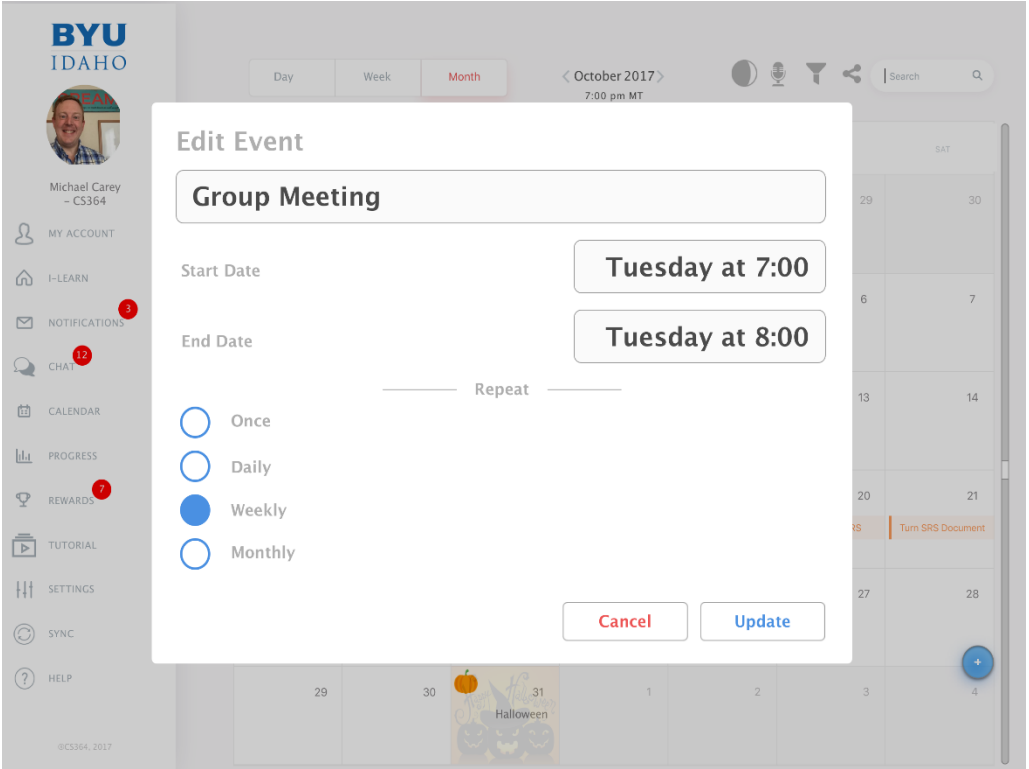
Joseph Keene

Type:

Process.

Data	Description
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. Assignments have been imported.
Post-conditions/Product(s) Produced:	Assignment details are shown and displayed correctly to the user.
Links:	None
SRS Document	3.2.6
Description/Notes:	<p>Show Assignment Details</p> <ol style="list-style-type: none"> 1. Show Details Button is displayed on the Calendar App by the Assignments. <ol style="list-style-type: none"> a. The user clicks the Details button. 2. Details are displayed to the User. <ol style="list-style-type: none"> a. An option to collapse the details is shown as well.

3.2.1.7 Calendar Display: Show Holidays

Data	Description
Screenshot/Mockup:	 <p>The screenshot shows a mobile application interface for a calendar. At the top, there's a navigation bar with 'BYU IDAHO' logo, a user profile for Michael Carey (CS364), and navigation icons. A sidebar menu on the left includes options like 'MY ACCOUNT', 'I-LEARN', 'NOTIFICATIONS', 'CHAT', 'CALENDAR', 'PROGRESS', 'REWARDS', 'TUTORIAL', 'SETTINGS', 'SYNC', and 'HELP'. The main area displays a calendar for October 2017. An 'Edit Event' dialog is open, showing the event title 'Group Meeting', start date 'Tuesday at 7:00', and end date 'Tuesday at 8:00'. The event is set to repeat 'Weekly'. There are 'Cancel' and 'Update' buttons at the bottom of the dialog. The background calendar shows a Halloween holiday display on October 31st.</p> <p>Figure 10 - Holiday Display Screen Mockup</p>
Page Title:	Show Holidays
Author:	Joseph Keene
Type:	Process

Data	Description
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

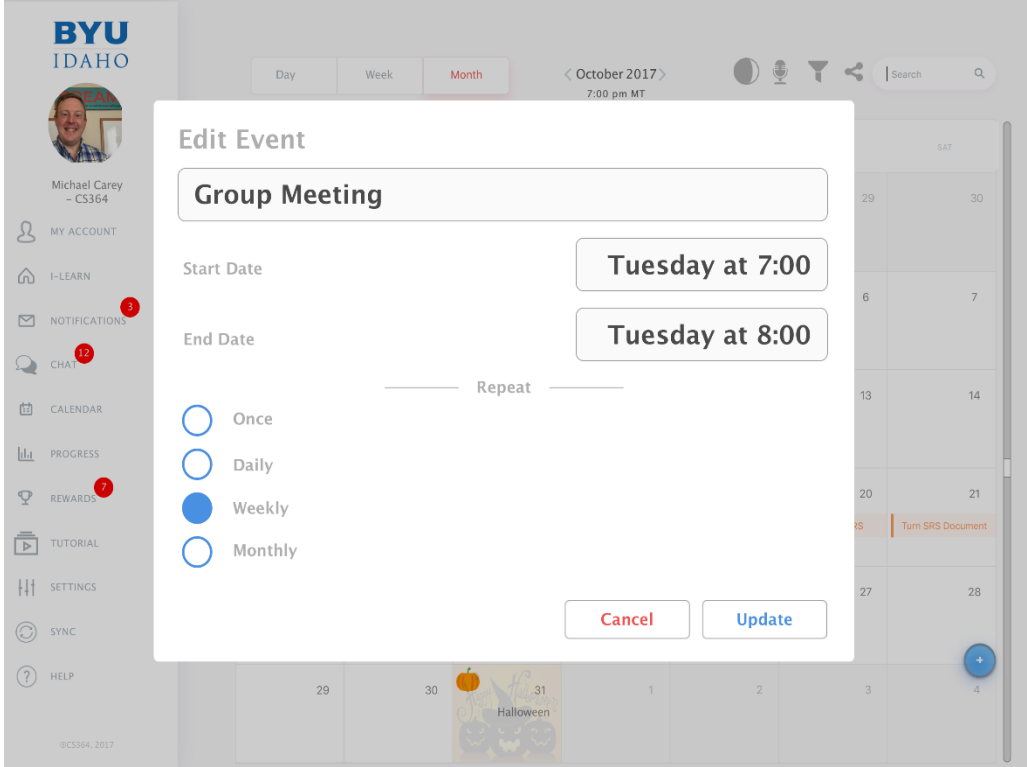
Data	Description
<p>Screenshot/Mockup:</p>	 <p>The screenshot displays a mobile application interface for editing a calendar event. At the top, the user's profile is shown as Michael Carey (CS364). The main area is a calendar for October 2017. A modal window titled 'Edit Event' is open, showing the event name 'Group Meeting', start date 'Tuesday at 7:00', and end date 'Tuesday at 8:00'. Below these are repeat options: 'Once', 'Daily', 'Weekly' (selected), and 'Monthly'. 'Cancel' and 'Update' buttons are at the bottom of the modal. The background calendar shows a 'Halloween' event on the 31st.</p>
<p>Page Title:</p>	<p>Edit Event</p>
<p>Author:</p>	<p>Brian Robertson</p>

Figure 11 - Event Editing Screen Mockup

Data	Description
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:	The event is updated and displays correctly on calendar.
Links:	None
SRS Document	3.2.16
Description/Notes:	<p>Editing an Event</p> <ol style="list-style-type: none"> 1. Edit Event button is displayed on the Calendar App. <ol style="list-style-type: none"> 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.

Data	Description
	a. User can interact with Event on Calendar.

3.2.2 Calendar Event Notifications

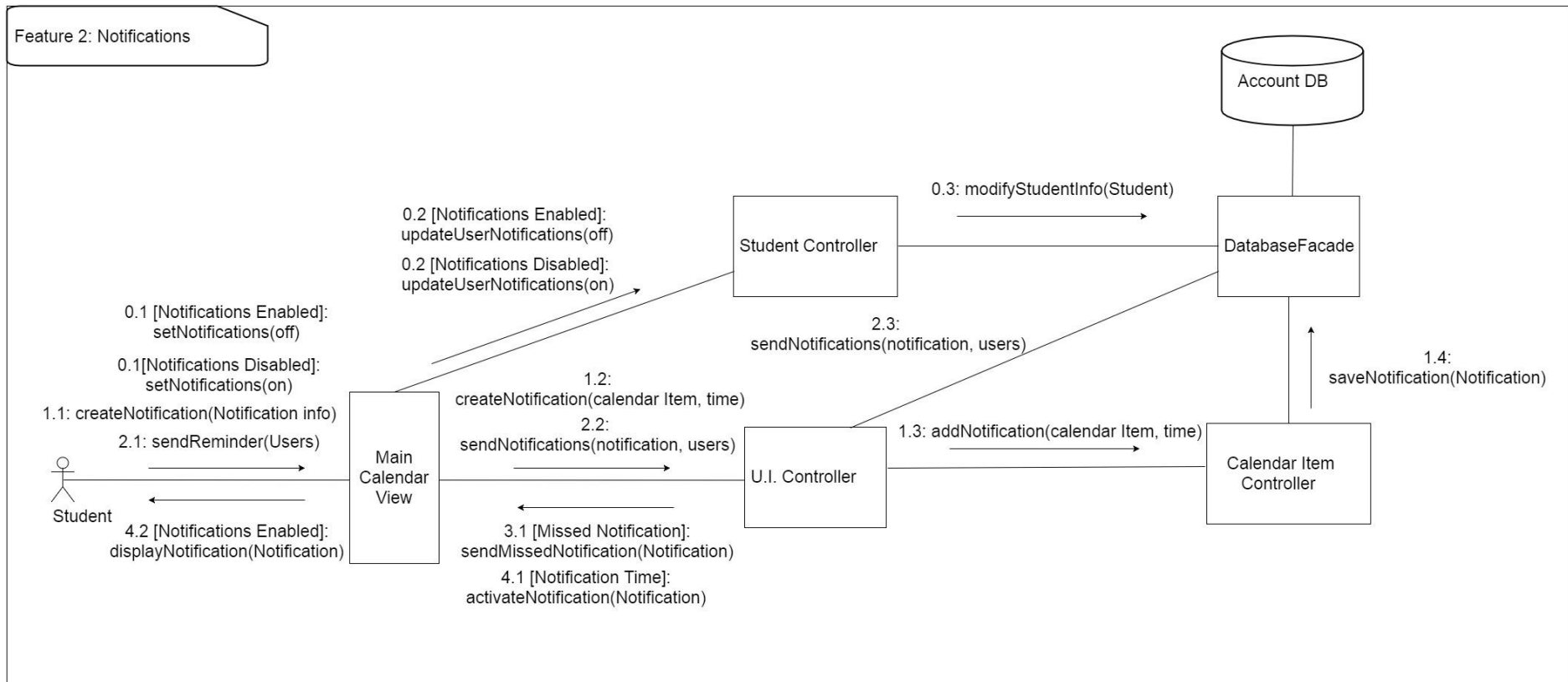


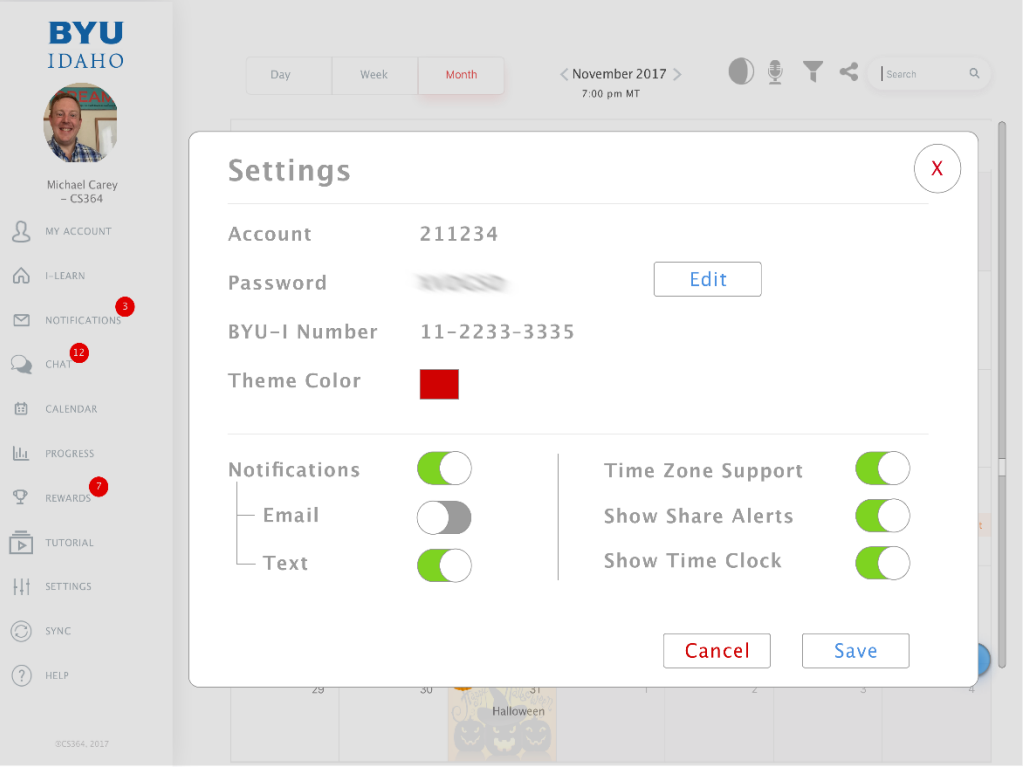
Figure 12 - Calendar Event Notifications UML Communication Diagram

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.

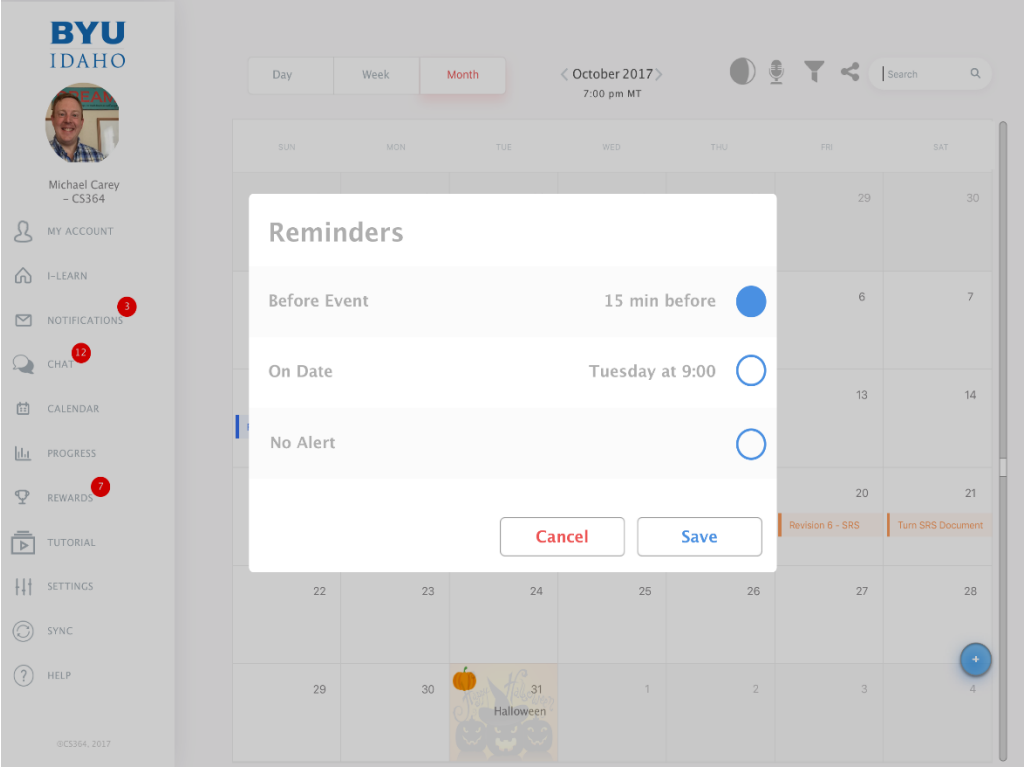
3.2.2.0 Calendar Event Notifications: Enable/Disable Notifications

Data	Description
<p>Screenshot/Mockup:</p>	 <p>Figure 13 - Notification Settings Screen Mockup</p>
<p>Page Title:</p>	<p>Enable/Disable Notifications</p>
<p>Author:</p>	<p>Zane West</p>

Data	Description
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. The 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	3.2.2.0
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

Data	Description
	<ul style="list-style-type: none">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
<p>Screenshot/Mockup:</p>	 <p>Figure 14 - Reminder Window Screen Mockup</p>
<p>Page Title:</p>	<p>Set Reminders</p>
<p>Author:</p>	<p>Zane West</p>

Data	Description
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. An event has been created.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 1. A reminder is prepared. 2. User is returned to their last viewed calendar.
Links:	None
SRS Document	3.2.2.2
Description/Notes:	<ol style="list-style-type: none"> 1. The user will have options as to how they would like to be reminded about their upcoming events. <ol style="list-style-type: none"> 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,

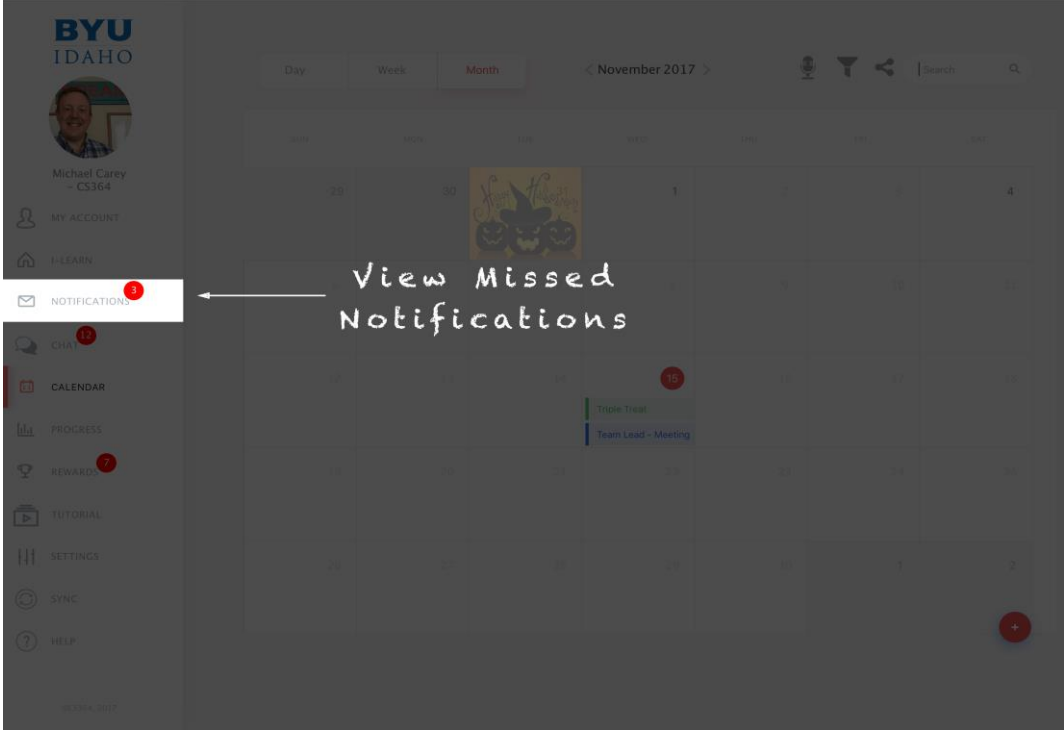
Data	Description
	e-mail or text message.

3.2.2.2 Calendar Event Notifications: Send Notification

Data	Description
Screenshot/Mockup:	N/A
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. An event has been created.

Data	Description
	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.

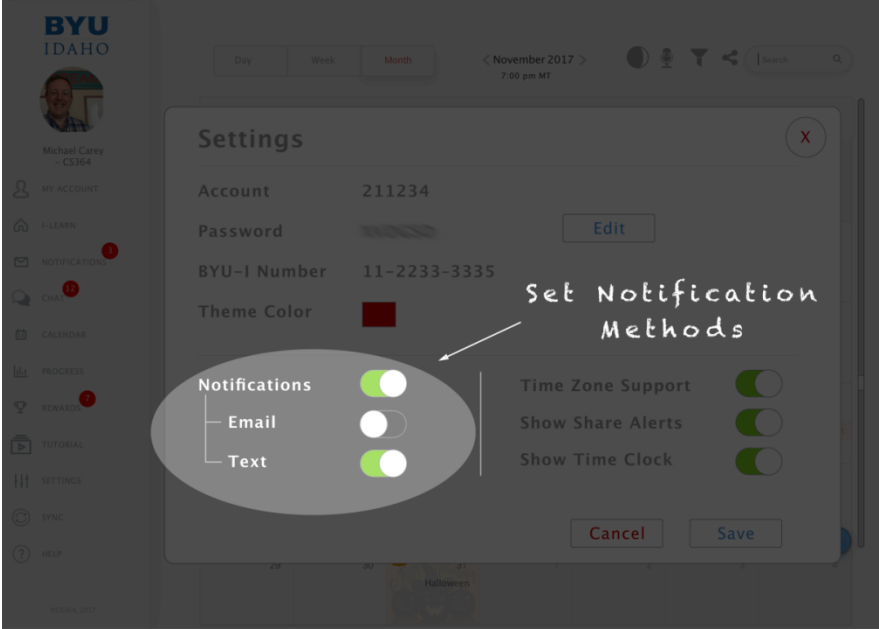
3.2.2.3 Calendar Event Notifications: View Missed Notifications

Data	Description
<p>Screenshot/Mockup:</p>	 <p><i>Figure 15 - Calendar Event Notifications: View Missed Notifications Mockup</i></p>
<p>Page Title:</p>	<p>View Missed Notifications</p>
<p>Author:</p>	<p>Jacob Keene</p>

Data	Description
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. A notification has been created. 3. Users have been sent a notification.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 1. A notification has been viewed. 2. User is returned to their last viewed calendar. 3. Missing notifications are empty.
Links:	None
Description/Notes:	<p>Viewing Missed Notifications</p> <ol style="list-style-type: none"> 1. Missed Notification <ol style="list-style-type: none"> a. A user has been sent a notification which hasn't been viewed yet. b. User is notified that he/she has a missed notification.

Data	Description
	c. User views missed notification, clearing the missed notifications. 2. Warning a. The user must view all missed notifications to clear all missed notifications.

3.2.2.4 Calendar Event Notifications: Set Notification Methods

Data	Description
<p>Screenshot/Mockup:</p>	 <p><i>Figure 16 - Calendar Event Notifications: Set Notification Methods Mockup</i></p>
<p>Page Title:</p>	<p>Set Notification Methods</p>
<p>Author:</p>	<p>Jacob Keene</p>
<p>Type:</p>	<p>Process</p>

Data	Description
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. User goes to notification settings.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 1. User's notification methods have been updated. 2. User is returned to their last viewed calendar.
Links:	None
SRS Document	3.2.20
Description/Notes:	<p>Set Notification Methods</p> <ol style="list-style-type: none"> 1. Notification Methods <ol style="list-style-type: none"> a. User is prompted to turn on or turn off notifications. b. User selects to turn on notifications and is returned to last viewed calendar. 2. Warning <ol style="list-style-type: none"> a. User selects to turn off notifications and is given a warning that they won't receive notifications when they disable this setting.

3.2.3 To-Do List

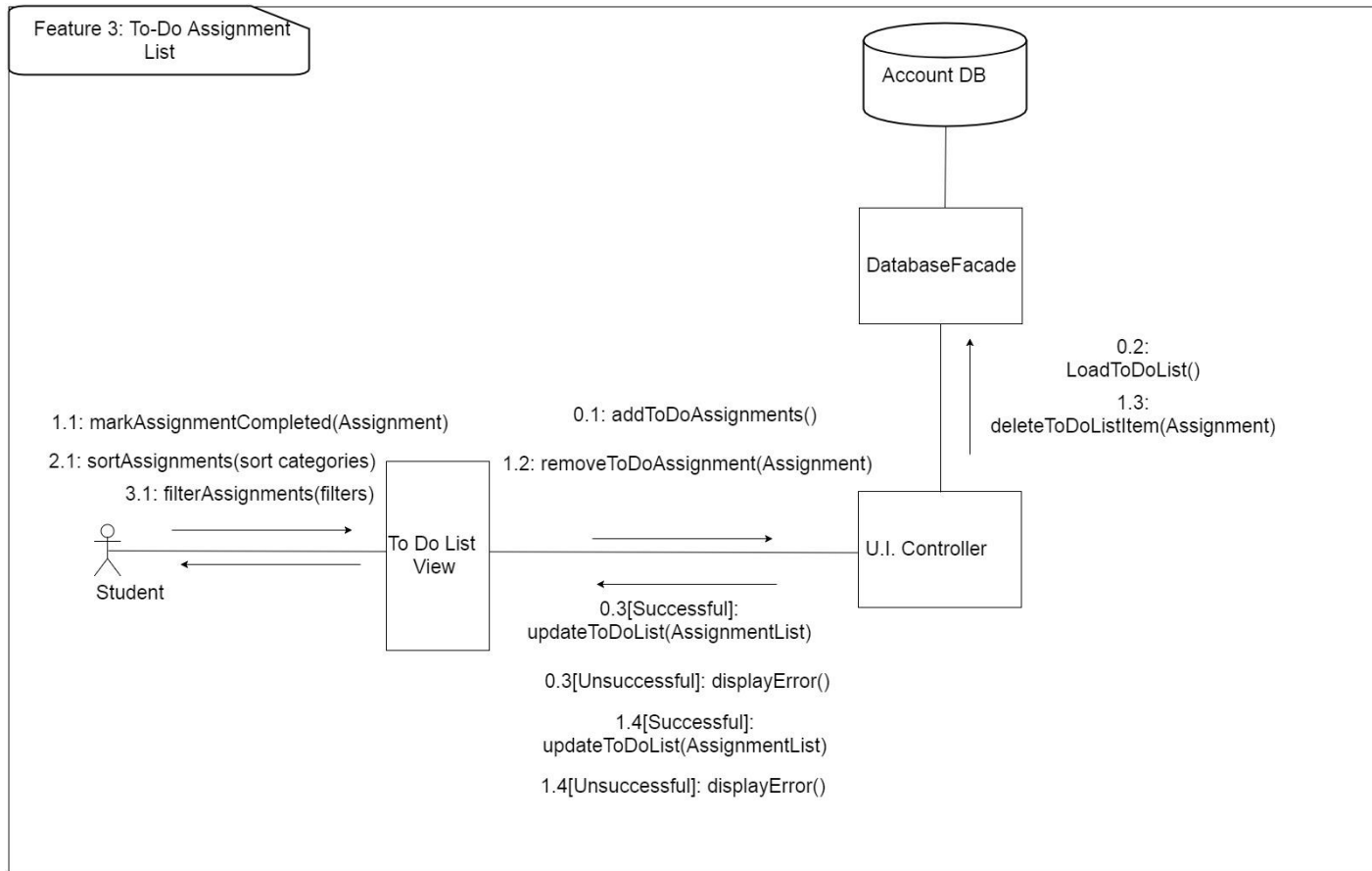


Figure 17 – To-Do Assignment List UML Communication Diagram

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

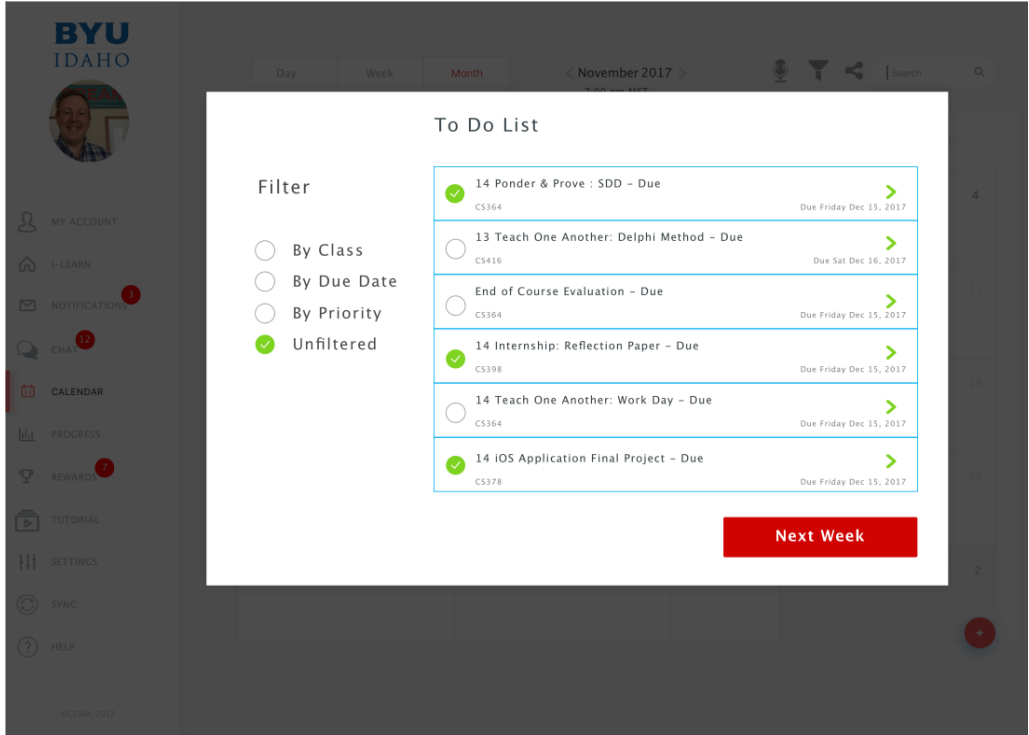
Data	Description																					
<p>Screenshot/Mockup:</p>	 <p>The screenshot shows the 'To Do List' screen in the BYU-Idaho mobile application. The interface includes a navigation menu on the left with options like 'MY ACCOUNT', 'I-LEARN', 'NOTIFICATIONS', 'CHAT', 'CALENDAR', 'PROGRESS', 'REWARD', 'TUTORIAL', 'SETTINGS', 'SYNC', and 'HELP'. The main content area displays a list of assignments with the following details:</p> <table border="1"> <thead> <tr> <th>Assignment</th> <th>Class</th> <th>Due Date</th> </tr> </thead> <tbody> <tr> <td>14 Ponder & Prove : SDD - Due</td> <td>CS364</td> <td>Due Friday Dec 15, 2017</td> </tr> <tr> <td>13 Teach One Another: Delphi Method - Due</td> <td>CS416</td> <td>Due Sat Dec 16, 2017</td> </tr> <tr> <td>End of Course Evaluation - Due</td> <td>CS364</td> <td>Due Friday Dec 15, 2017</td> </tr> <tr> <td>14 Internship: Reflection Paper - Due</td> <td>CS398</td> <td>Due Friday Dec 15, 2017</td> </tr> <tr> <td>14 Teach One Another: Work Day - Due</td> <td>CS364</td> <td>Due Friday Dec 15, 2017</td> </tr> <tr> <td>14 iOS Application Final Project - Due</td> <td>CS378</td> <td>Due Friday Dec 15, 2017</td> </tr> </tbody> </table> <p>Below the list is a red 'Next Week' button. The filter options are: By Class, By Due Date, By Priority, and Unfiltered (selected).</p>	Assignment	Class	Due Date	14 Ponder & Prove : SDD - Due	CS364	Due Friday Dec 15, 2017	13 Teach One Another: Delphi Method - Due	CS416	Due Sat Dec 16, 2017	End of Course Evaluation - Due	CS364	Due Friday Dec 15, 2017	14 Internship: Reflection Paper - Due	CS398	Due Friday Dec 15, 2017	14 Teach One Another: Work Day - Due	CS364	Due Friday Dec 15, 2017	14 iOS Application Final Project - Due	CS378	Due Friday Dec 15, 2017
Assignment	Class	Due Date																				
14 Ponder & Prove : SDD - Due	CS364	Due Friday Dec 15, 2017																				
13 Teach One Another: Delphi Method - Due	CS416	Due Sat Dec 16, 2017																				
End of Course Evaluation - Due	CS364	Due Friday Dec 15, 2017																				
14 Internship: Reflection Paper - Due	CS398	Due Friday Dec 15, 2017																				
14 Teach One Another: Work Day - Due	CS364	Due Friday Dec 15, 2017																				
14 iOS Application Final Project - Due	CS378	Due Friday Dec 15, 2017																				
<p>Page Title:</p>	<p>Load To-Do List</p>																					
<p>Author:</p>	<p>Rex Nesbit</p>																					

Figure 18 – To-Do Assignment List Screen Mockup

Data	Description
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:	<p>There will be a button to access and launch the To-Do List found near the views tabs.</p> <ol style="list-style-type: none"> 1. To-Do List launched will show events displayed to the user has scheduled for the active week. <ol style="list-style-type: none"> 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

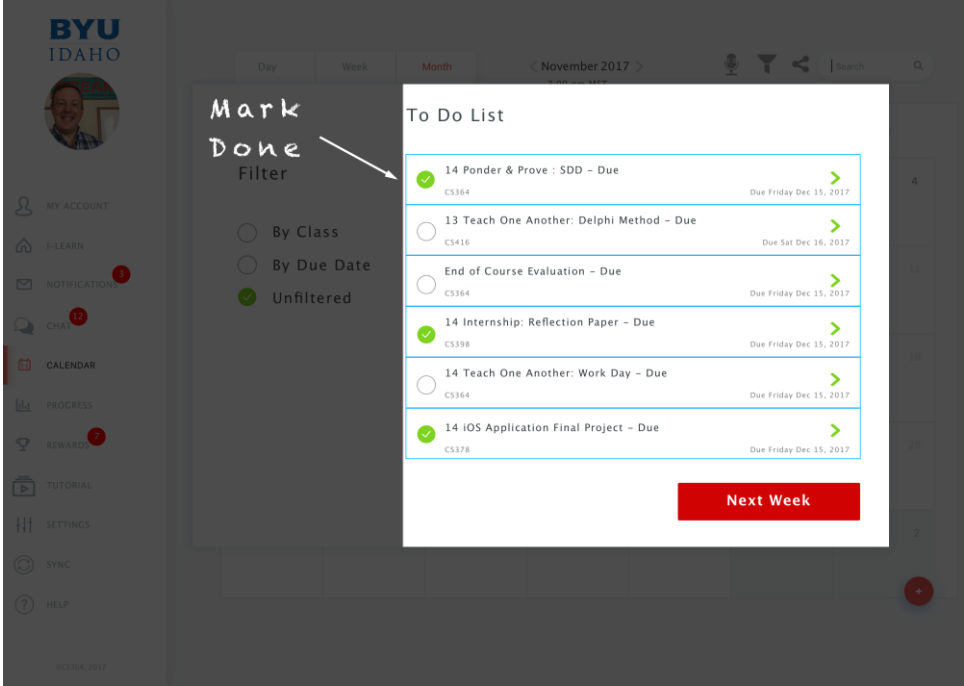
Data	Description
<p>Screenshot/Mockup:</p>	 <p>The screenshot shows a mobile application interface for BYU-Idaho. A 'To Do List' modal is open, displaying a list of assignments. The list includes items like '14 Ponder & Prove : SDD - Due', '13 Teach One Another: Delphi Method - Due', 'End of Course Evaluation - Due', '14 Internship: Reflection Paper - Due', '14 Teach One Another: Work Day - Due', and '14 iOS Application Final Project - Due'. Each item has a checkbox and a due date. A 'Mark Done Filter' is visible in the background, and a 'Next Week' button is at the bottom of the list.</p>
<p>Page Title:</p>	<p>Mark Assignment Done</p>
<p>Author:</p>	<p>Rex Nesbit</p>
<p>Type:</p>	<p>Process</p>

Figure 19 - To-Do List: Mark Assignment Done Mockup

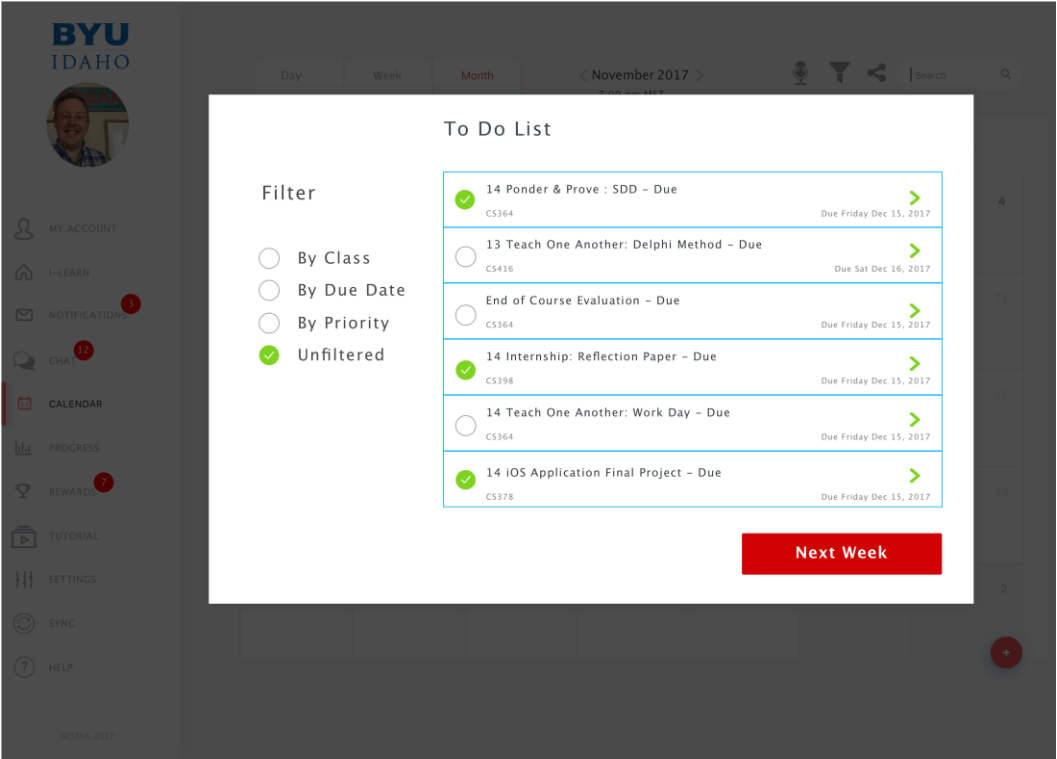
Data	Description
Purpose:	Allow the user to keep track of assignments done by checking them as completed.
Parent User Story:	3.2.3.0
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	User is logged in. User is on To-Do List.
Post-conditions/Product(s) Produced:	To-Do list is displayed with the user's incomplete events and assignments shown.
Links:	None
SRS Document	3.2.24
Description/Notes:	<ol style="list-style-type: none"> 1. To-Do List launched will show events displayed to the user has scheduled for the active week. <ol style="list-style-type: none"> 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. <ol style="list-style-type: none"> 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:	N/A
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:	<ol style="list-style-type: none">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.

Data	Description
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
<p>Screenshot/Mockup:</p>	 <p>Figure 20 - To-Do List: Filter Mockup</p>
<p>Page Title:</p>	<p>Filter</p>
<p>Author:</p>	<p>Zane West</p>

Data	Description
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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 1. Filter <ol style="list-style-type: none"> 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.

Data	Description
	<p>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.</p> <p>d. The calendar view updates to reflect only assignments meeting the filter criteria.</p> <p>2. Warning</p> <p>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.</p> <p>b. The filter shall reset with each Calendar Manager session to prevent the user from missing filtered assignments.</p>

3.2.4 Calendar Integrations

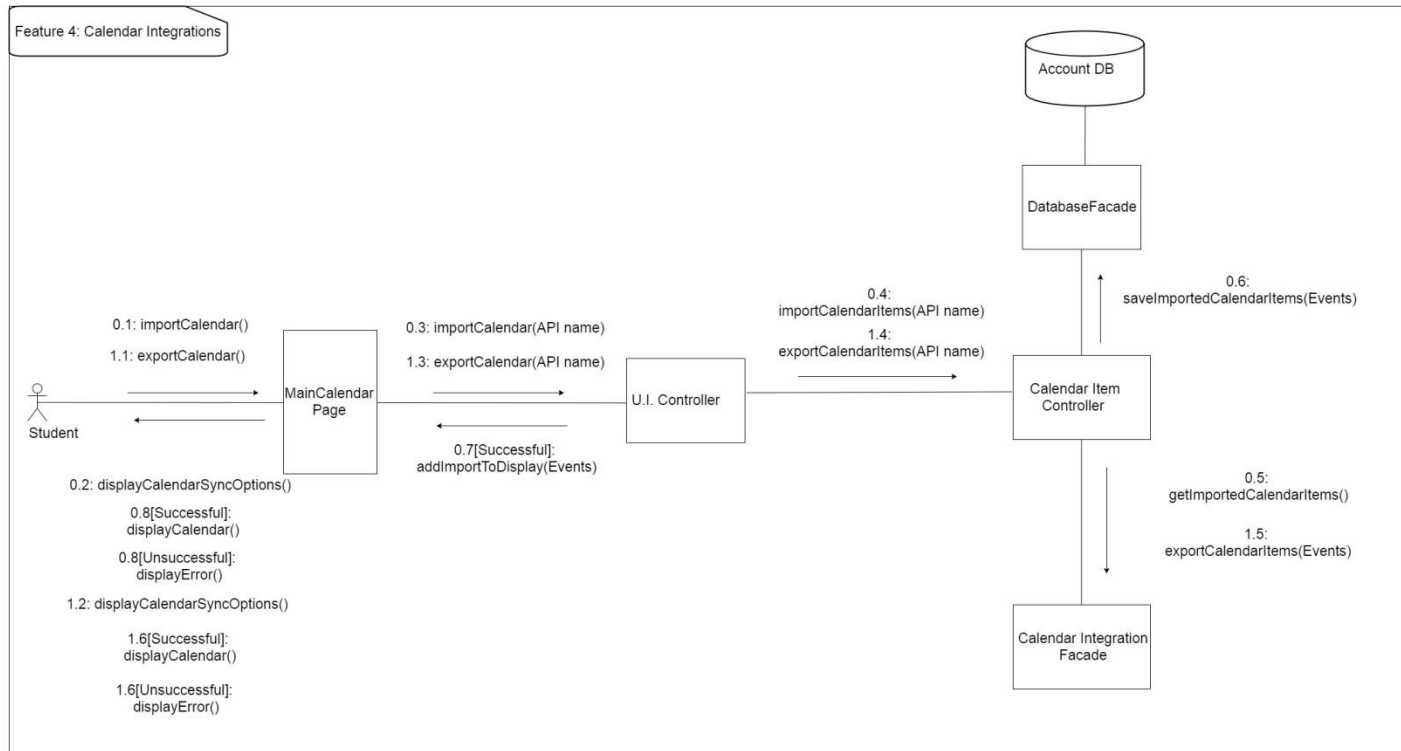
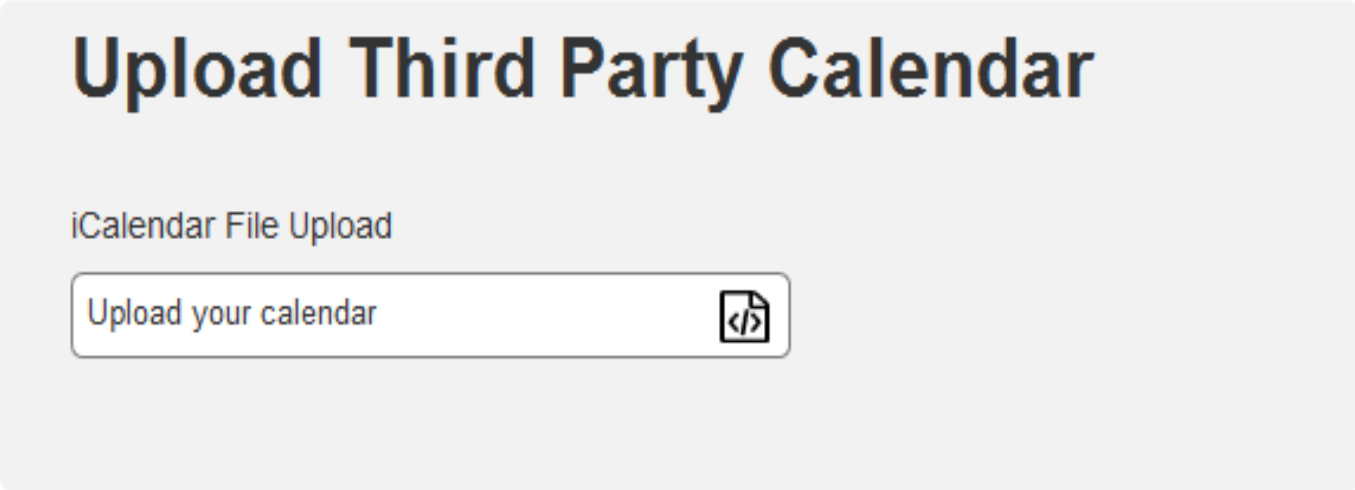


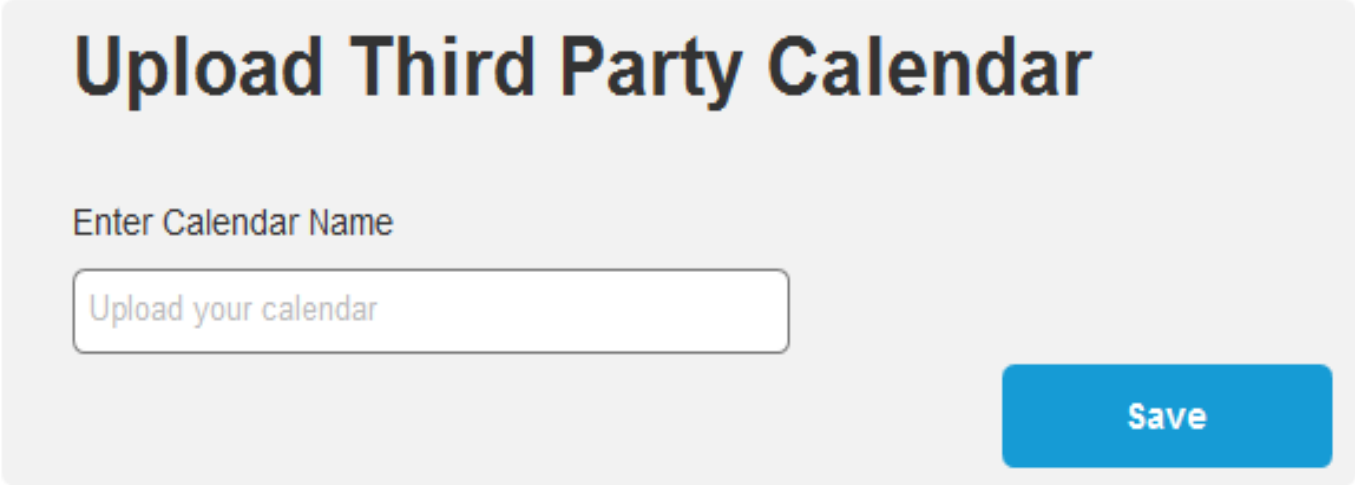
Figure 21 - Calendar Integrations UML Communication Diagram

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

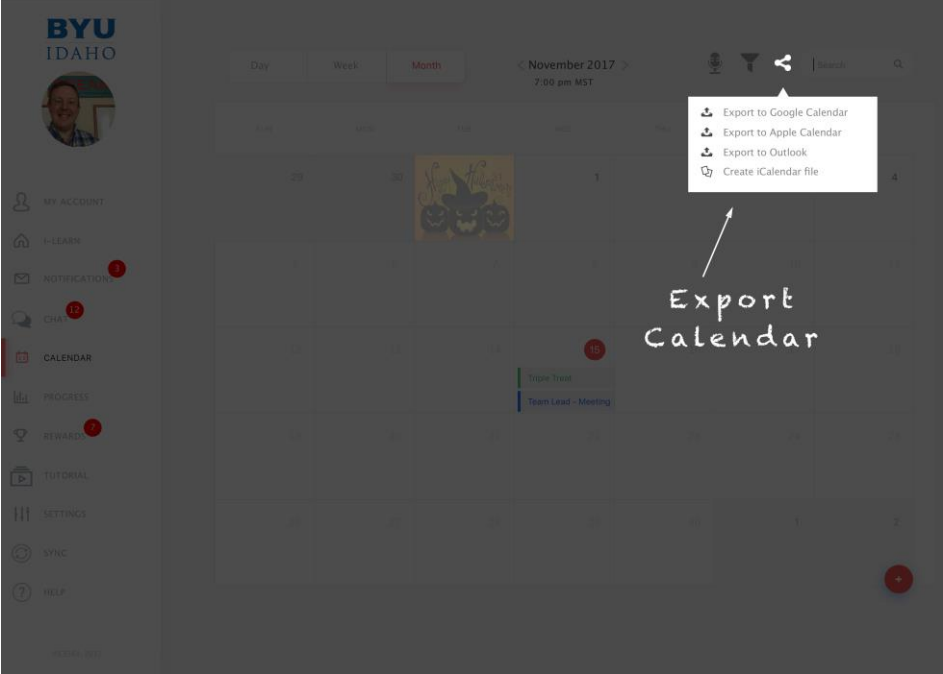
Data	Description
Screenshot/Mockup:	 <p>The screenshot shows a light gray background with the title "Upload Third Party Calendar" in large, bold, dark blue font. Below the title is the text "iCalendar File Upload" in a smaller, dark blue font. Underneath is a white rounded rectangular button with the text "Upload your calendar" in dark blue. To the right of the button is a small icon of a document with a code symbol (</>).</p> <p><i>Figure 22.a - Import Calendar Screen Mockup</i></p>

Data	Description
	 <p><i>Figure 23.b - Import Calendar Screen Mockup</i></p>
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

Data	Description
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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.
Links:	<ol style="list-style-type: none"> 1. Standard for iCalendar: https://tools.ietf.org/html/rfc5545
SRS Document	3.2.47-3.2.49
Description/Notes:	<ol style="list-style-type: none"> 1. Supported third party calendar applications include: <ol style="list-style-type: none"> a. Google Calendar b. Apple Calendar c. Outlook Calendar d. Any calendar that supports exporting an iCalendar file

Data	Description
	<p>2. Importing iCalendar</p> <ul style="list-style-type: none">a. The user will click import on the calendar page.b. The user will be prompted to select their iCalendar file that conforms to the IETF (Internet Engineering Task Force) standard.c. The user will select their exported iCalendar file.d. The import function will attempt to import the iCalendar.e. If successful, the events will be added to the user's calendar.f. If the file is corrupt or in an unexpected format, an error message will be displayed indicating such, and no events will be imported.

3.2.4.1 Calendar Integrations Export Calendar

Data	Description
<p>Screenshot/Mockup:</p>	 <p>The screenshot shows the BYU Idaho mobile application interface. At the top, there's a header with the BYU IDAHO logo and a navigation menu. The main content area displays a calendar for November 2017. A dropdown menu is open, showing four options: 'Export to Google Calendar', 'Export to Apple Calendar', 'Export to Outlook', and 'Create iCalendar file'. A white arrow points from the text 'Export Calendar' to the 'Export to Google Calendar' option.</p> <p><i>Figure 24 - Calendar Integrations Export Calendar Mockup</i></p>
<p>Page Title:</p>	<p>Export Calendar</p>
<p>Author:</p>	<p>Tyler Sorenson, Robert Nelson</p>
<p>Type:</p>	<p>Process</p>

Data	Description
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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 1. The user will download the calendar's events using the iCalendar format conforming to the IETF iCalendar standard.
Links:	<ol style="list-style-type: none"> 1. Standard for iCalendar: https://tools.ietf.org/html/rfc5545
SRS Document	3.2.47-3.2.49
Description/Notes:	<ol style="list-style-type: none"> 1. Supported third party calendar applications include: <ol style="list-style-type: none"> a. Google Calendar b. Apple Calendar c. Outlook Calendar d. Any calendar that supports importing an iCalendar file 2. Exporting iCalendar <ol style="list-style-type: none"> a. The user will click export on the calendar page.

Data	Description
	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

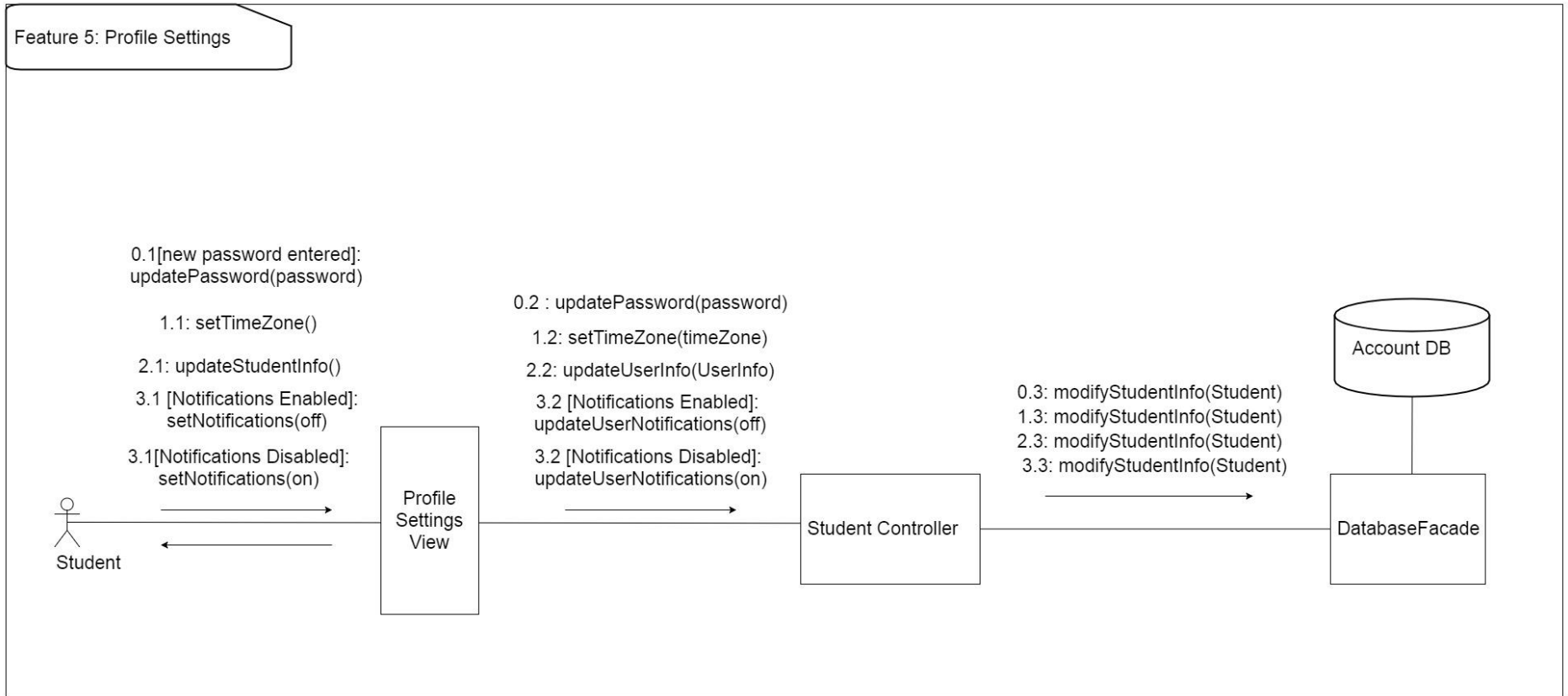


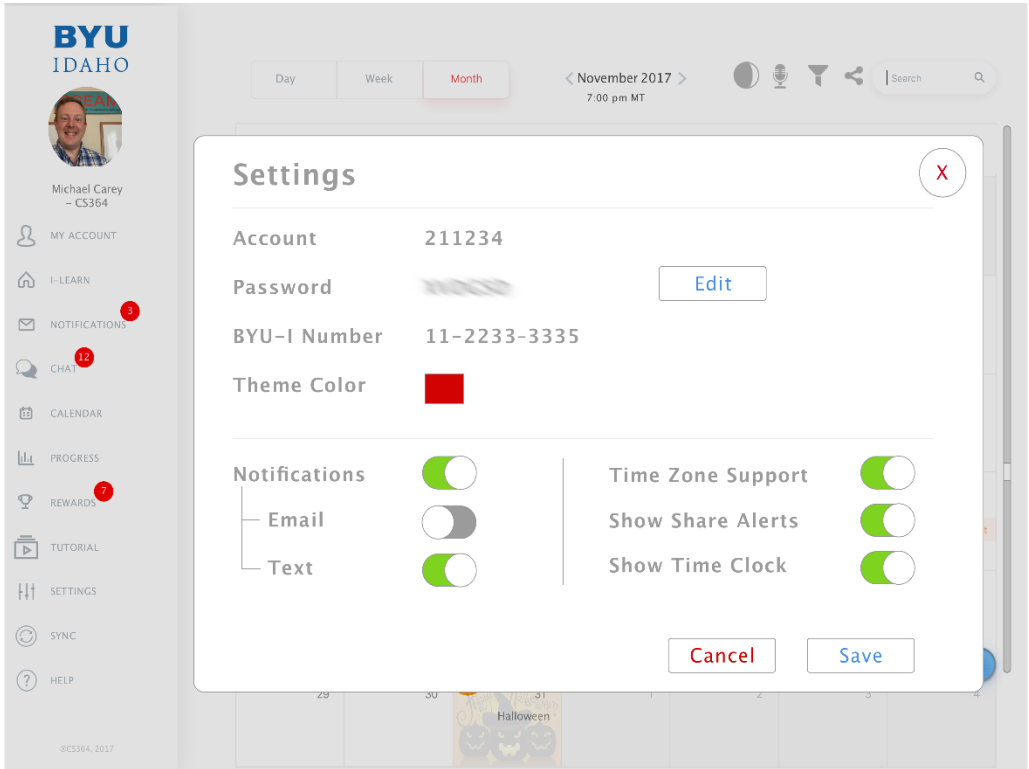
Figure 25 - User Profile Settings diagram

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
<p>Screenshot/Mockup:</p>	 <p>Figure 26 - Load Settings Page Mockup</p>
<p>Page Title:</p>	<p>Settings</p>
<p>Author:</p>	<p>Aaron Shore</p>
<p>Type:</p>	<p>Process</p>

Data	Description
Purpose:	Allow the user to change personal settings to their account and so that the notifications act as the user prefers.
Parent User Story:	N/A
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 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:	<p>Password</p> <ol style="list-style-type: none"> 1. Change Password <ol style="list-style-type: none"> 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. <p>Notifications</p> <ol style="list-style-type: none"> 1. Toggle Switch for Notifications <ol style="list-style-type: none"> a. When notifications is switched on then a drop down will appear with all the different notifications the user can interact with. <p>While switch is off then none of the notifications will be active nor will they show up on the</p>

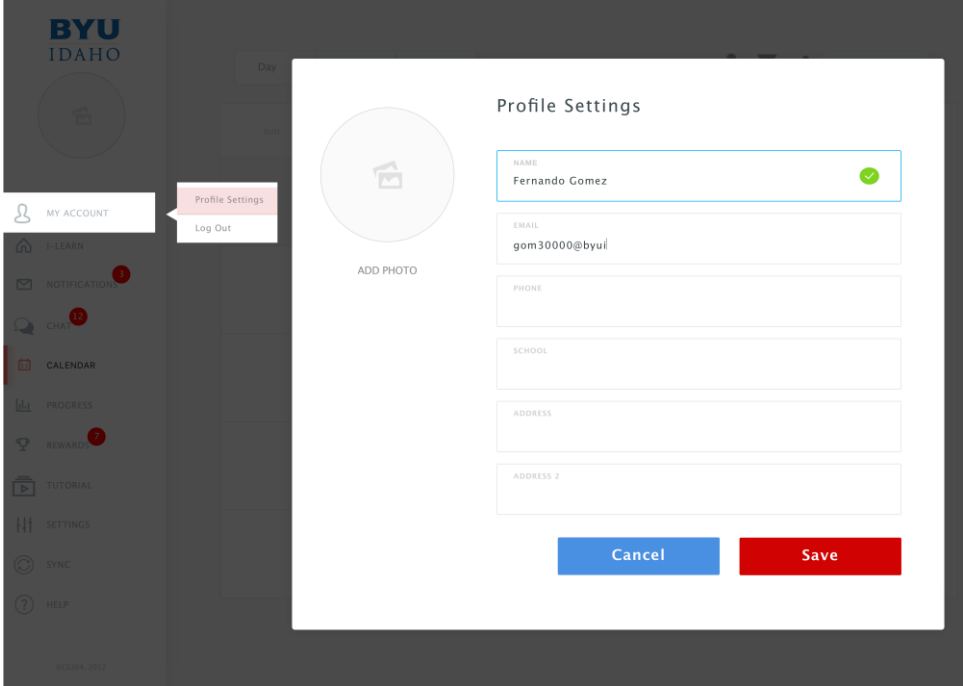
Data	Description
	screen to interact with. 2. Toggle Switch for Individual Notifications a. Each notification can be turned on or off with the toggle switch next to each notification. b. User will only receive notifications for those switched on.

3.2.5.1 User Profile Settings: Set Time Zone

Data	Description
<p>Screenshot/Mockup:</p>	 <p><i>Figure 27 - User Profile Settings: Set Time Zone Mockup</i></p>
Page Title:	Set Time Zone
Author:	Rex Nesbit
Type:	Process

Data	Description
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:	<p>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.</p> <ol style="list-style-type: none"> 1. Time Zone option is selected. <ol style="list-style-type: none"> 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
<p>Screenshot/Mockup:</p>	 <p><i>Figure 28 - User Profile Settings: Change Profile Settings Mockup</i></p>
Page Title:	Change Profile Settings
Author:	Jefferson Santos
Type:	Process

Data	Description
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:	<ol style="list-style-type: none"> 1. User is logged in. 2. User is inside settings page.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 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:	<p>Change profile option inside the settings menu can be selected. Forms are showed with actual data.</p> <p>Validation</p> <ol style="list-style-type: none"> 1. Name input box will be filled out with the actual name. 2. E-mail input box will be filled out and validated with patterns of e-mail addresses, like <u>xxx@xxx.xxx</u>. 3. Phone input box will appear and needs to be filled out with the following pattern xxx-xxx-xxxx to be validated.

Data	Description
	<p>4. School input box will be filled out with the complete name of the school and not with abbreviations e.g, BYU-I.</p> <p>5. Address input box will be filled out with the user's complete address</p> <p>a. The zip code input needs to be a 5-digit number.</p> <p>Observations</p> <p>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.</p>

3.2.5.3 User Profile Settings: Change Notification Settings

Data	Description
Screenshot/Mockup:	N/A
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

Data	Description
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 1. User is logged in. 2. User goes to notification settings.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 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:	<p>Enable/disable and change settings for notifications</p> <ol style="list-style-type: none"> 1. Notifications will be enabled by default. 2. The notification toggle will be found in the parent settings page. 3. Warnings <ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> 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

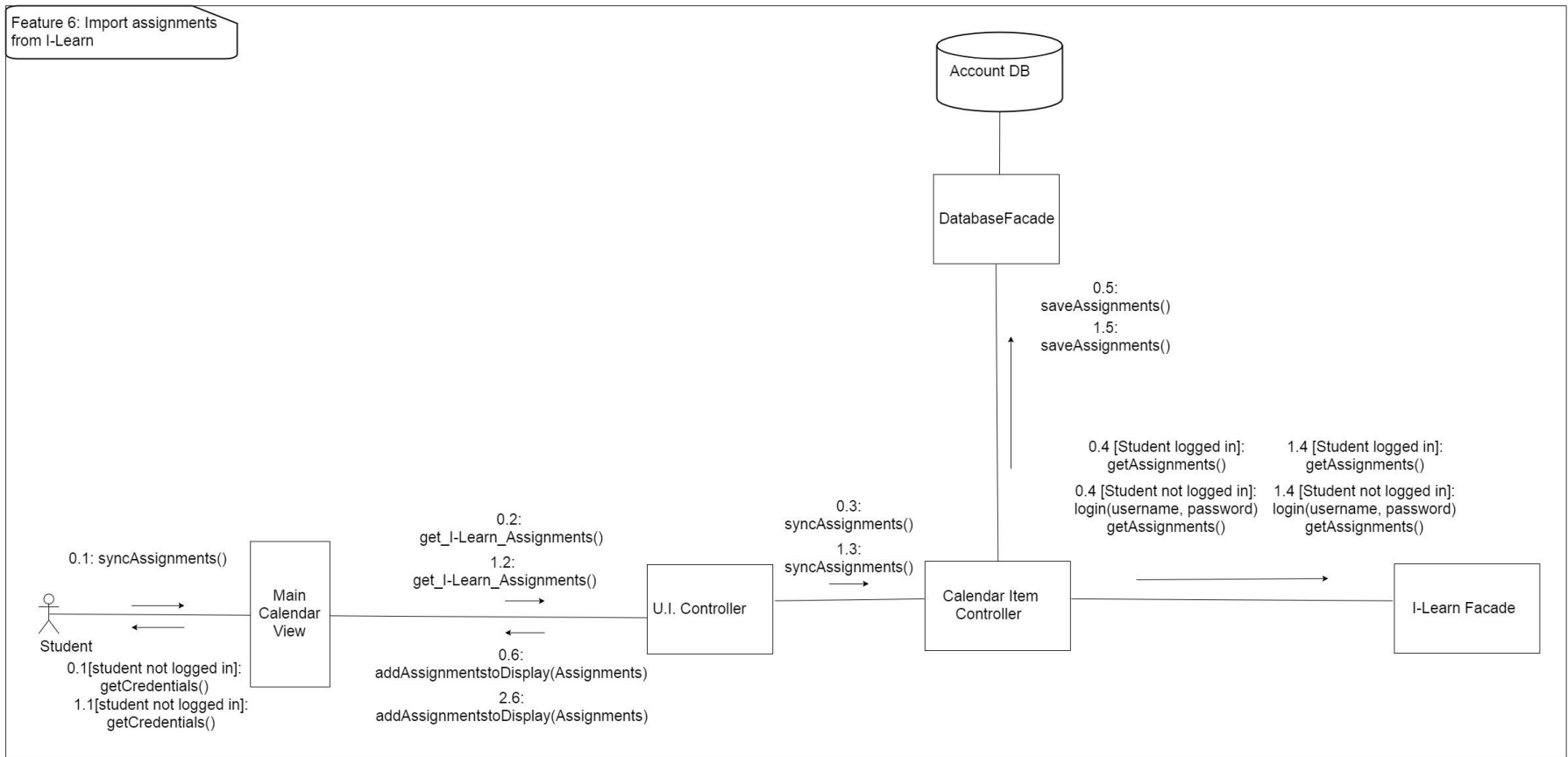
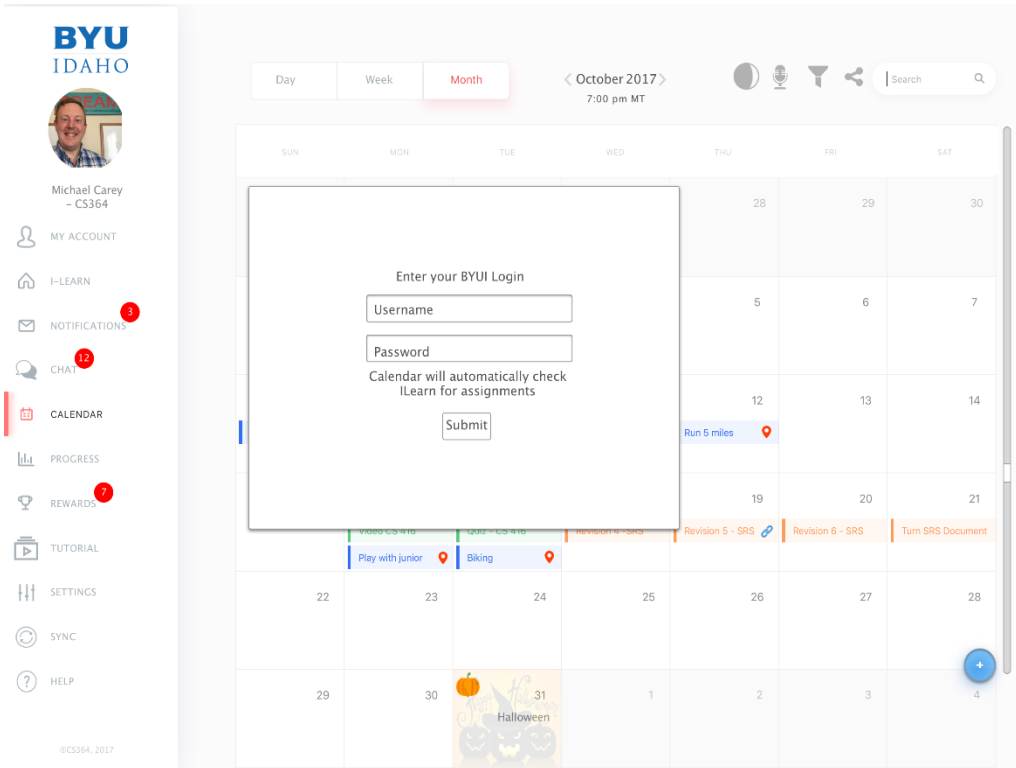


Figure 29 - Import Assignments from I-Learn Feature Diagram

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
<p>Screenshot/Mockup:</p>	 <p>Figure 30 - Import Assignments from I-Learn: Manual Sync Mockup</p>
<p>Page Title:</p>	<p>Import Assignments from I-Learn</p>

Data	Description
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:	<ol style="list-style-type: none"> 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

Data	Description
SRS Document	3.2.6
Description/Notes:	<p>Importing Assignments from I-Learn</p> <ol style="list-style-type: none"> 1. User clicks on sync button. 2. User is prompted to enter their BYU-I Credentials <ol style="list-style-type: none"> 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. <ol style="list-style-type: none"> 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. <p>The image below details the steps necessary to import I-Learn data into the calendar application.</p>

Data

Description

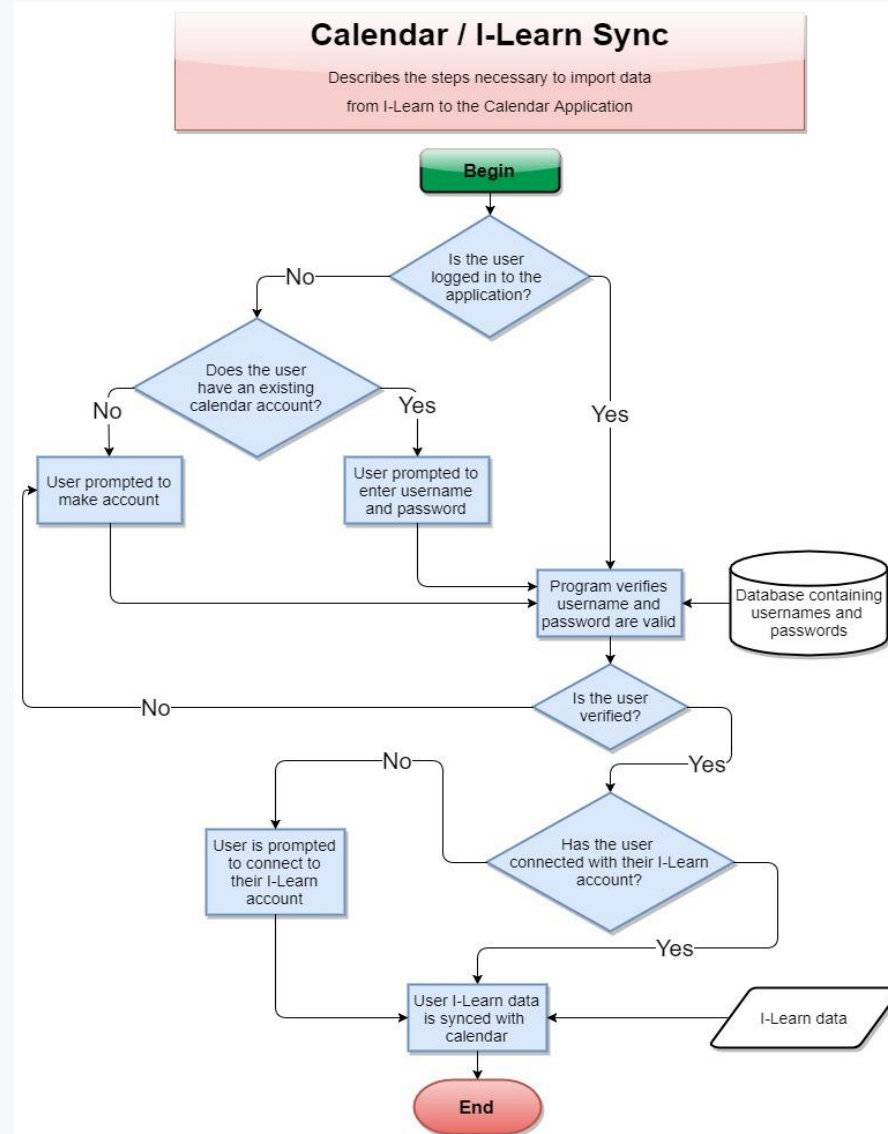
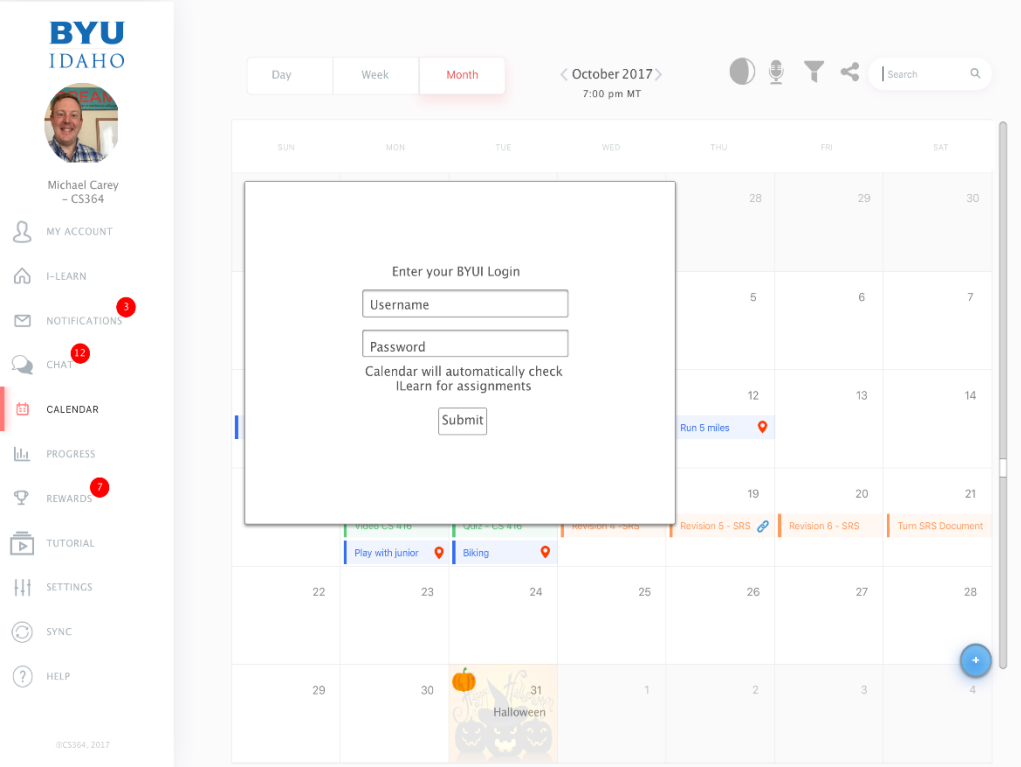


Figure 31 - Calendar / I-Learn Sync Chart

3.2.6.1 Import Assignments from I-Learn: Automatic Sync

Data	Description
<p>Screenshot/Mockup:</p>	 <p>The screenshot shows a user interface for a BYU IDAHO user named Michael Carey (CS364). On the left is a navigation menu with options: MY ACCOUNT, I-LEARN, NOTIFICATIONS (3), CHAT (12), CALENDAR, PROGRESS, REWARDS (7), TUTORIAL, SETTINGS, SYNC, and HELP. The main area displays a calendar for October 2017. A modal form titled 'Enter your BYUI Login' is overlaid on the calendar, containing fields for 'Username' and 'Password', and a 'Submit' button. Below the text 'Calendar will automatically check I-Learn for assignments'. The calendar shows dates from 28 to 4. Assignments are visible on several dates: 'Run 5 miles' on the 12th, 'Play with junior' on the 22nd, 'Biking' on the 23rd, and 'Halloween' on the 31st. Other assignments include 'Revision 4 - SRS', 'Revision 5 - SRS', 'Revision 6 - SRS', and 'Turn SRS Document'.</p> <p>Figure 32 - Import Assignments from I-Learn: Automatic Sync Mockup</p>
Page Title:	Import Assignments from I-Learn Automatically
Author:	Joseph Keene
Type:	Process

Data	Description
Purpose:	To have the assignments in I-Learn transfer over to the Calendar automatically to see their due dates and description of the assignments there in the Calendar.
Parent User Story:	N/A
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:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 1. User is prompted to log onto their BYU-I account when logging into the calendar application for the first time. <ol style="list-style-type: none"> 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. <ol style="list-style-type: none"> a. A pop-up box will display that says "Importing your Assignments from I-Learn. Please

Data	Description
	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

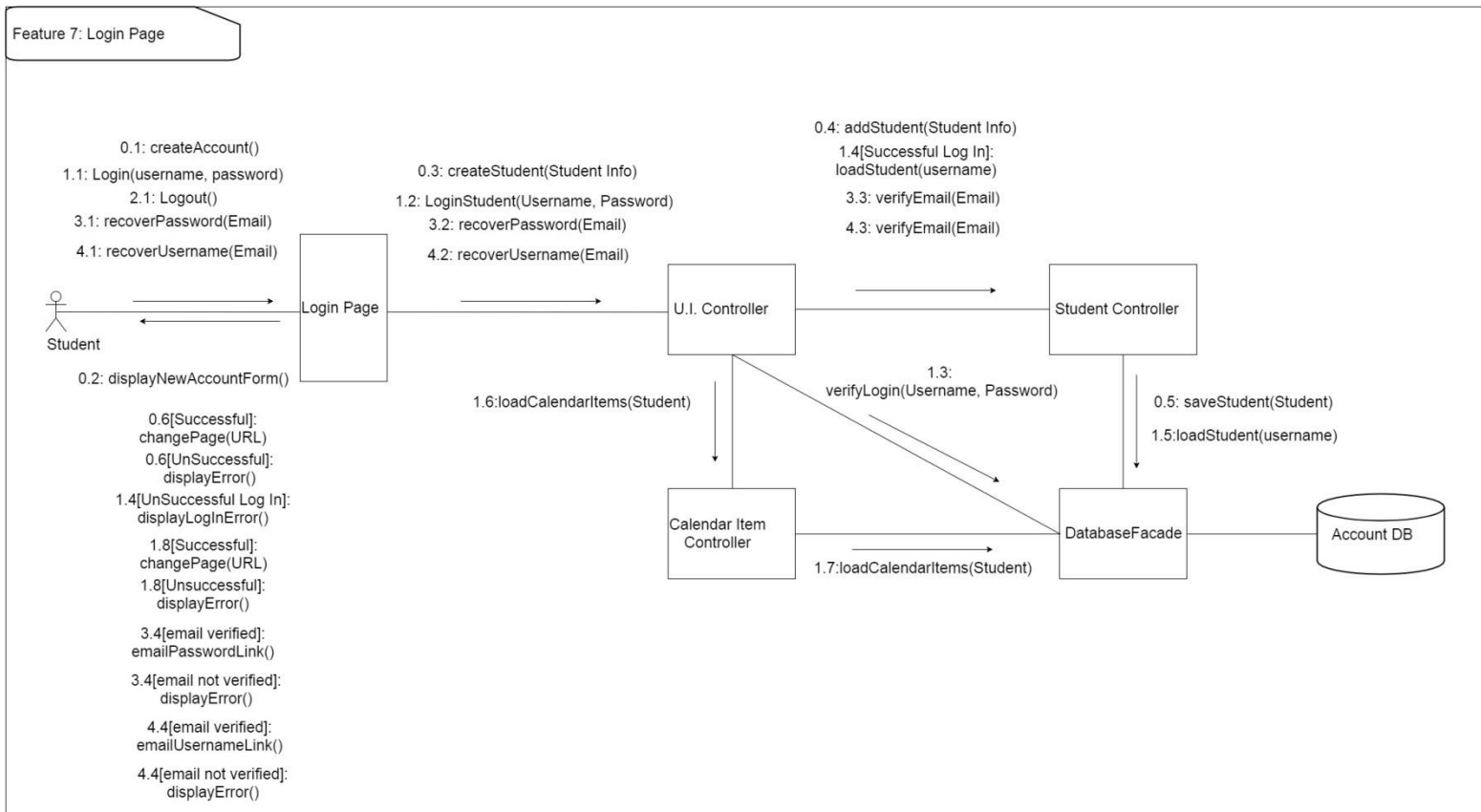


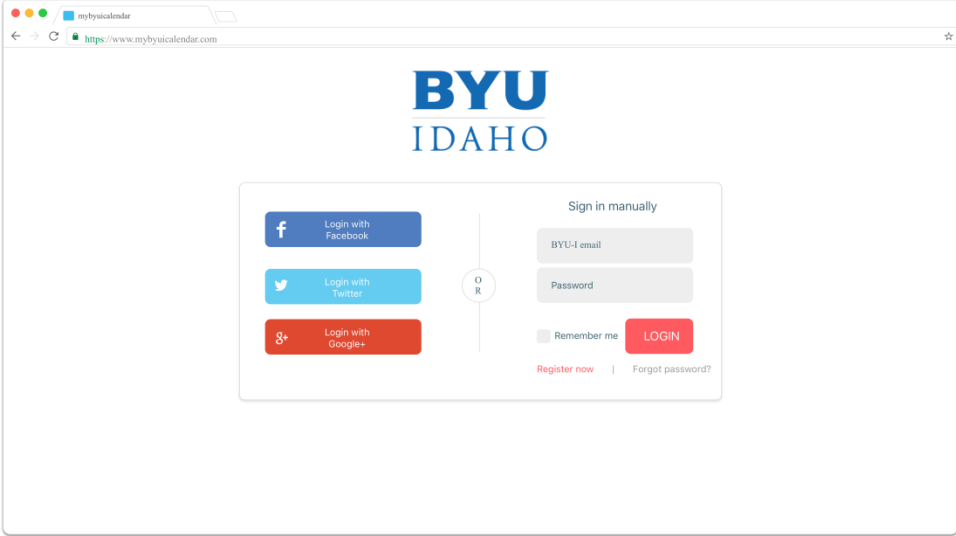
Figure 33 - Log-In Use Case Diagram

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.

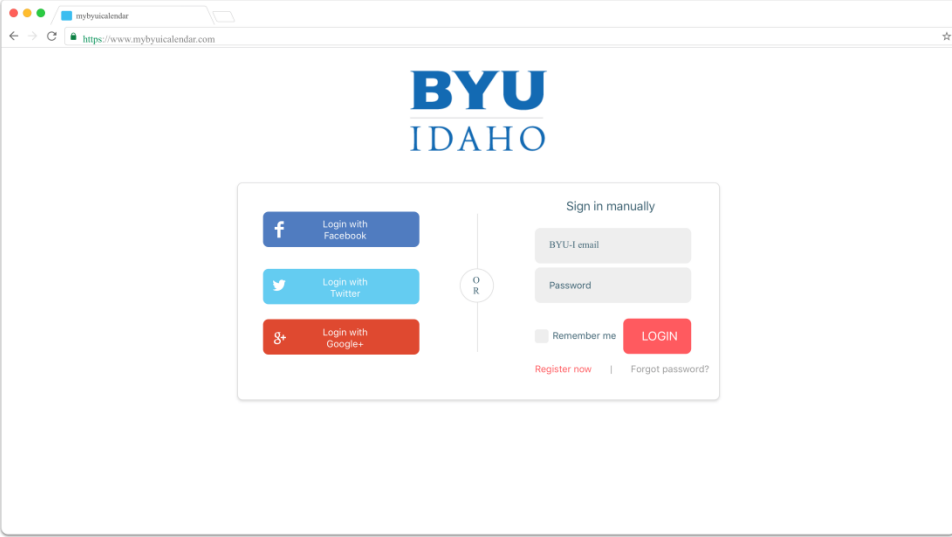
3.2.7.0 Log-In Use Case: Create Account

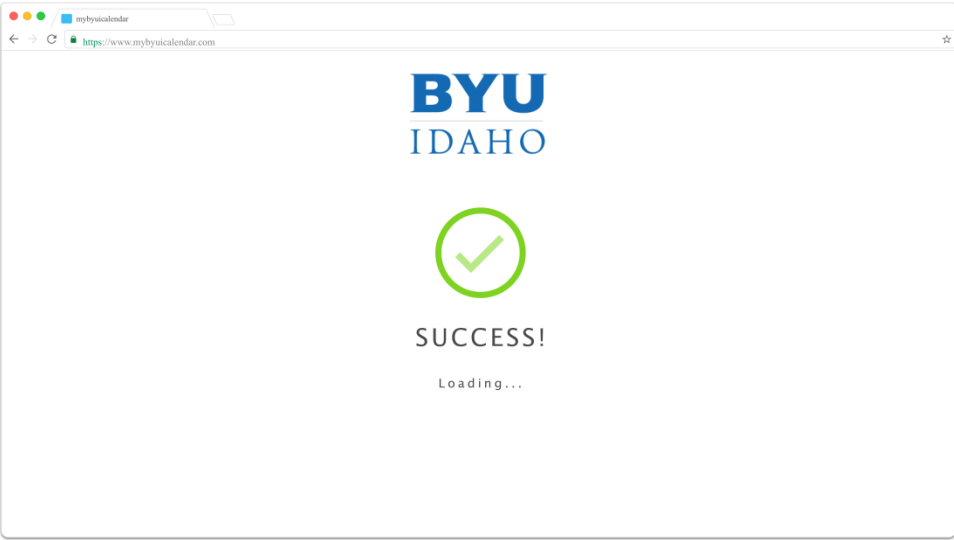
Data	Description
Screenshot/Mockup:	 <p><i>Figure 34 - Log-In Use Case: Create Account Mockup</i></p>
Page Title:	Create account
Author:	Brian Robertson

Data	Description
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:	<ol style="list-style-type: none"> 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.

Data	Description
	5. Following account creation, and activation, the user can then log onto the calendar app.

3.2.7.1 Log-in Use Cases: Validation

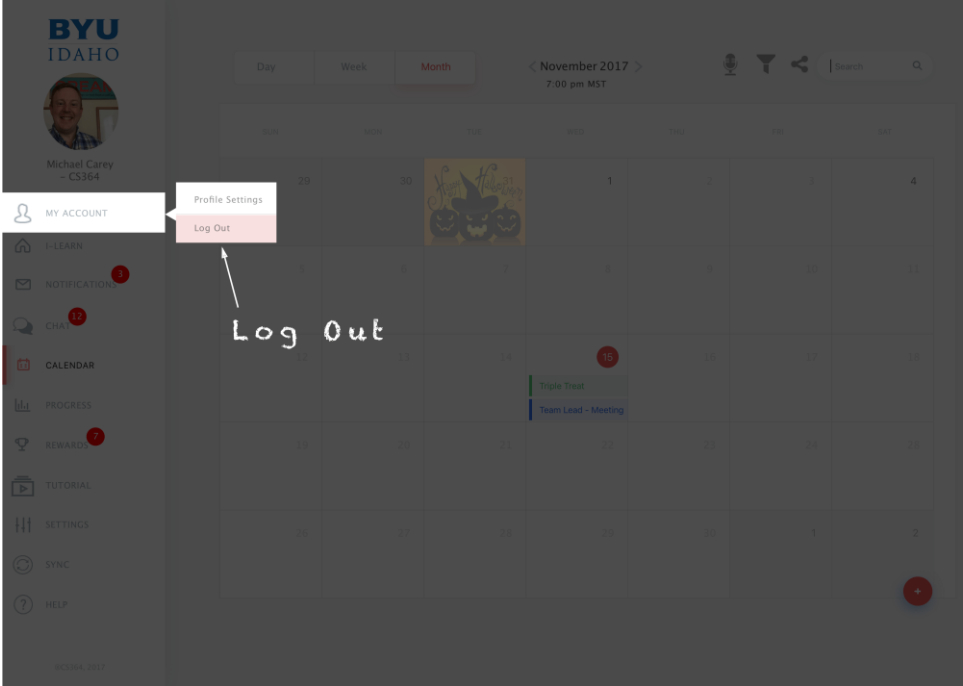
Data	Description
Screenshot/Mockup:	 <p><i>Figure 35.a - Log-in Use Cases: Validation Mockup</i></p>

Data	Description
	 <p><i>Figure 36.b - Log-in Use Cases: Validation Mockup</i></p>
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

Data	Description
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 1. Calendar Portal has been brought up and configured. 2. User has made an account for the Calendar application.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 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:	<p>Validation</p> <ol style="list-style-type: none"> 1. User-name input field <ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> a. Input field can be accessed by clicking or using the "tab" button on keyboard.

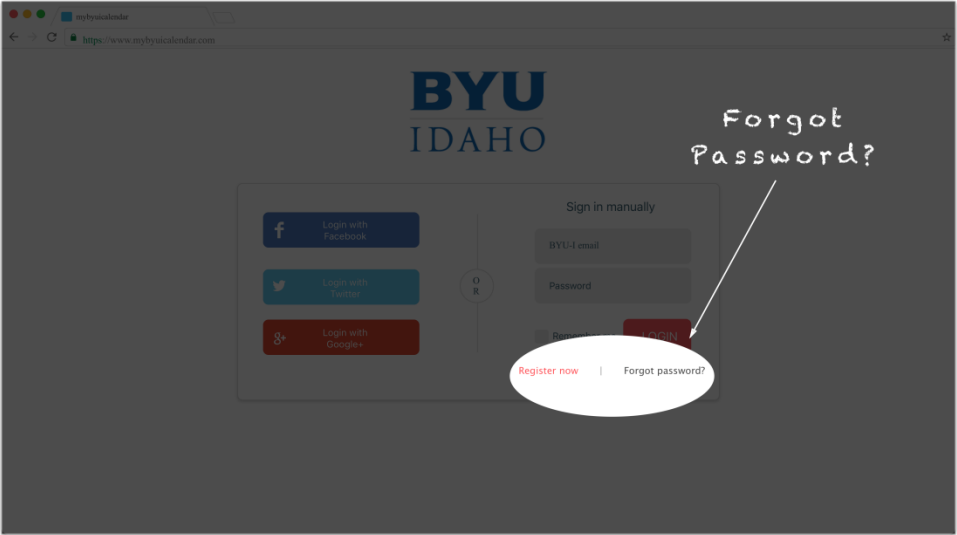
Data	Description
	<p>b. When login button is pushed it will verify that the password matches a user-name, which in full matches an account that has been created.</p> <p>3. Login button</p> <p>a. Button will be active by default</p> <p>b. When button is clicked both the user-name and password input fields will be checked to match an account.</p> <p>c. While waiting for validation a loading spinner will appear.</p> <p>d. IF user-name or password is not valid then user will be prompted to try again.</p> <p>e. When user-name and password match an account, the user will be logged in and their content will be loaded.</p>

3.2.7.2 Log-Out Use Case

Data	Description
<p>Screenshot/Mockup:</p>	 <p><i>Figure 37 - Log-Out Use Case Mockup</i></p>
<p>Page Title:</p>	<p>Log-Out</p>
<p>Author:</p>	<p>Robert Nelson</p>
<p>Type:</p>	<p>Process</p>

Data	Description
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:	 <p><i>Figure 38 - Log-in Use Cases: Forgot Password Mockup</i></p>
Page Title:	Forgot Password
Author:	Rex Nesbit
Type:	Process
Purpose:	Help the user get back into their account if they have forgotten their password.

Data	Description
Parent User Story:	3.2.7.1 Log-in Use Cases: Validation.
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 1. Calendar Portal has been brought up and configured. 2. User has made an account for the Calendar application.
Post-conditions/Product(s) Produced:	<ol style="list-style-type: none"> 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:	<p>Validation</p> <ol style="list-style-type: none"> 1. User-name input field <ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> 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.

Data	Description
	<ul style="list-style-type: none">3. Login button<ul style="list-style-type: none">a. Button will be active by defaultb. 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 appeard. If user-name or password is not valid then user will be prompted to try again4. Forgot password button<ul style="list-style-type: none">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:	 <p>The screenshot shows the login page for BYU-Idaho. It features the university's logo at the top. Below the logo, there are three social media login options: Facebook, Twitter, and Google+. To the right, there is a 'Sign in manually' section with input fields for 'BYU-Idaho email' and 'Password'. Below these fields, there are links for 'Remember me' and 'Forgot password?'. A white callout bubble with a black border points to the 'Forgot password?' link, containing the text 'Forgot username?'. The text 'Forgot password?' and 'Forgot username?' are also visible in the original image.</p> <p><i>Figure 39 - Log-in Use Cases: Forgot Account Mockup</i></p>
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.

Data	Description
Parent User Story:	3.2.7.1 Log-in Use Cases: Validation.
Stakeholders:	Calendar Manager, Developers, Testers, Technical Support
Pre-conditions/Product(s) Required:	<ol style="list-style-type: none"> 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:	<ol style="list-style-type: none"> 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 Integration 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 are provided in section 4.2.

4.1 System Architecture

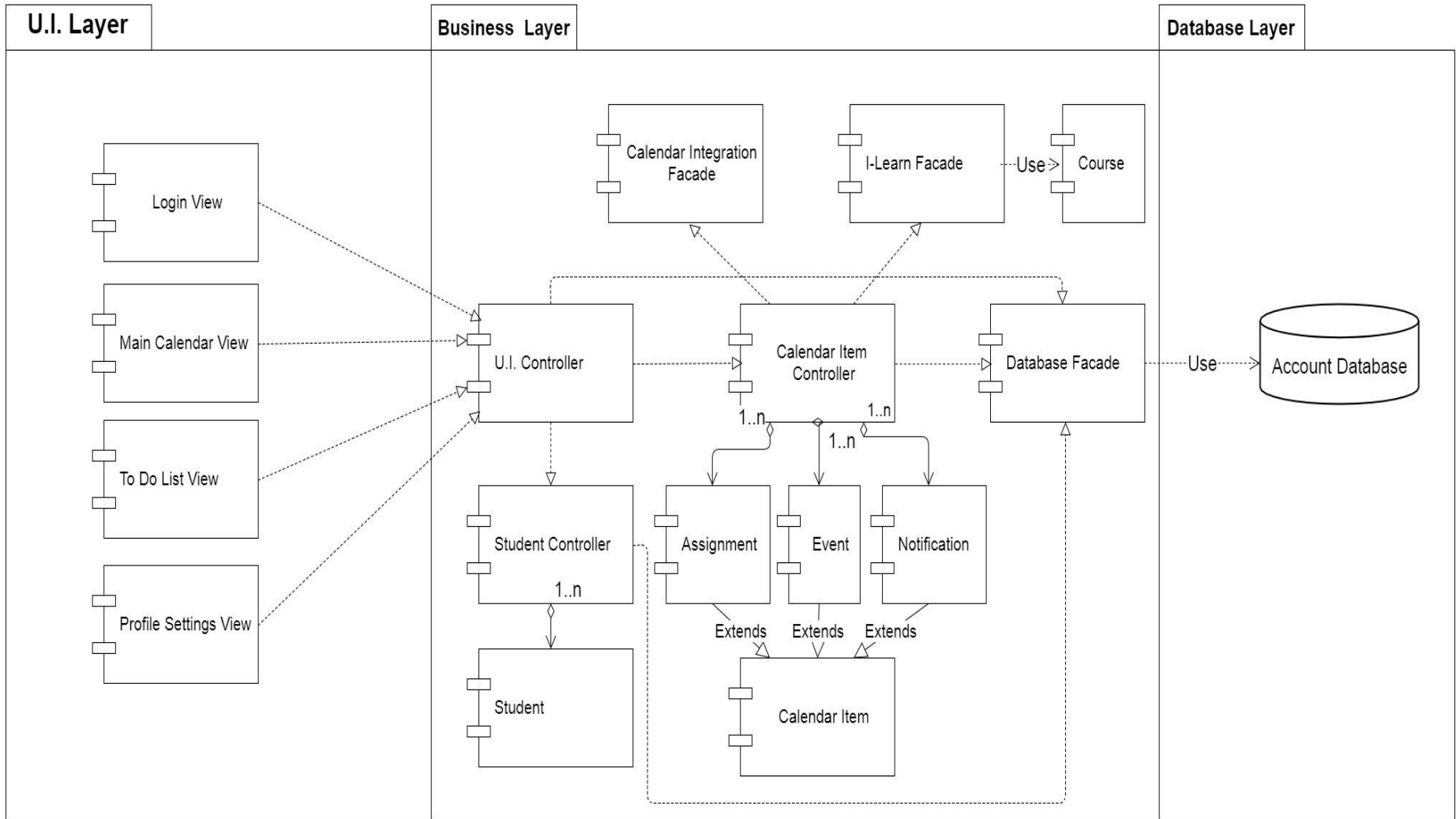


Figure 40 - System Architecture Diagram

4.1.0 Architecture Description

The architecture design for the Student Calendar Integration application is centered on 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 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 to import/export calendar events from 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 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.

4.2.1.1 To-Do List View

Class Name: To-Do List View	
Description: This class embodies the concept of a task list comprised of assignments the user needs to complete. It features the ability to delineate each assignment as complete or incomplete.	
The calendar application as a whole is designed to provide a student or teacher with tools to assist in the management of school schedules and tasks. As a result, this class provides a pertinent and necessary functionality that provides users with a means to monitor and manage one of the most crucial categories in their school focus, assignments.	
Attributes(Fields)	Attribute Description
Task List	The list of I-Learn tasks that are to be displayed.
Event List	The list of user created events that are to be displayed.
Methods(operations)	

markAssignmentCompleted	Method Description
	<p>This method is to be called when the user marks an assignment completed. It takes the assignment object to be marked as a parameter.</p> <p>Parameters: Assignment - The Assignment object that the user has marked as completed.</p> <p>Return: None</p>
	Method Pseudo Code
markAssignmentIncomplete	Method Description
	<p>This method is to be called when the user marks an assignment as incomplete. It takes the assignment object to be marked as a parameter.</p> <p>Parameters: Assignment - The Assignment object that the user has marked as incomplete.</p> <p>Return: None</p>
	Method Pseudo Code

	<pre>assignment.setCompleted(false) if isTask(assignment): taskList.add(assignment) if isEvent(assignment): eventList.add(assignment)</pre>
updateTodoList()	Method Description
	<p>This method is called when the user clicks the To-Do list button again, refreshing the list of tasks.</p> <p>Parameters: None</p> <p>Return: None</p>
	Method Pseudo Code
	<pre>delete(eventList) delete(taskList) eventList = uiController.getEventList() taskList = uiController.getEventList()</pre>
sortAssignments(sort Categories)	Method Description
	<p>This method allows the user to sort their To-Do list items by class, due date, or priority level. The user selects which category they would like to sort their To-Do list by. The categories are passed into the system and the to-do list items are sorted by either ascending or descending order according to the category.</p> <p>Parameters: Sort Categories - String list of categories that determines how the To-Do list is sorted.</p>

	Return: None
	Method Description
filterAssignments(filters)	<p>This method allows the user to filter their To-Do list assignments by course, due date, or priority level. The user selects the filters they want for their To-Do list and the assignments are filtered accordingly.</p> <p>Parameters: Filters - String list that define how the assignments are filtered.</p> <p>Return: None</p>

4.2.1.2 Main Calendar View

Class Name: Main Calendar Page View	
<p>Description: This class is responsible for many of the main functionality of the calendar. This class would be akin to a homepage of a website or the main page of a mobile application. It serves to help the user navigate through dates. It allows for the changing of views between month, week, and day. It populates dates with information that may include, but is not limited to, assignments, events, tasks, meetings, links, etc.</p> <p>It provides the user with a visual representation of a user's personal calendar, with options to create and edit events for specified dates. It also contains the methods necessary for the application's settings and notifications functionality. It is the core class of the application and the hub from which many of the basic calendar features connect.</p>	
Attributes(Fields)	Attribute Description

Task List	The list of I-Learn tasks that are to be displayed
Event List	The list of user created events that are to be displayed.
Holidays	A collection of holiday dates that will be used to display holidays in their proper locations.
Methods(operations)	
displayWeek(date)	Method Description
	<p>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.</p> <p>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 week found 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.</p> <p>Parameters: Date - The current date that is being displayed on the Calendar. Determines the numbering to be displayed in the weekly view.</p> <p>Return: None</p>
	Method Pseudo Code
	<pre>weekArray <- ("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")</pre>

	<pre> 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 </pre>
displayMonth(date)	Method Description
	<p>This function toggles to a monthly view from any other view (weekly or daily).</p> <p>Parameters: Date - The current date that is being displayed on the Calendar.</p> <p>Return: None</p>
	Method Pseudo Code
<pre> 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 </pre>	

displayDay(date)	Method Description
	<p>This method sets up the daily view.</p> <p>Parameters: Date - The current day date that is to be displayed on the Calendar.</p> <p>Return: None</p>
	Method Pseudo Code
createEvent(eventDate, eventDescription)	<pre> 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 </pre>
	Method Description
	<p>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.</p> <p>Parameters:</p> <p>event Date - The date that the event is being created for.</p> <p>event Description - The information entered by the user that will be used to create the event.</p> <p>Return: None</p>
Method Pseudo Code	
<p>API POST -> with eventDate and eventDescription in the body.</p> <p>IF RESPONSE.statusCode == 200</p>	

	Dismiss modal and show success message. ELSE Show Error message
followLink(externalURL)	Method Description
	This method is called when a user clicks an external link. Parameters: ExternalURL - The URL link address that the user has clicked on. Return: None
	Method Pseudo Code
	open externalURL in new tab
viewILearnAssignment(Assignment)	Method Description
	This method opens the page in I-learn that corresponds to the assignment that is clicked on by the user in the student calendar app. The assignment that is clicked on by the user is passed as a parameter. The method gets the assignment URL and opens a new tab to display the assignment to the user. Parameters: Assignment - The assignment object that the user has clicked on. This is used to get the correct I-Learn URL link. Return: None
	Method Pseudo Code
get Assignment Details(Assignment)	Method Description
	This method gets the description of the assignment and displays it to the user. The assignment that the user clicks on is passed in as a parameter and the system gets the data for that assignment.

	<p>Parameters: Assignment - The assignment object that the user has clicked on. This is used to get the correct assignment details.</p> <p>Return: None</p>
getHolidays(Month)	Method Description
	<p>This method searches the collection of holidays for any holidays that occur during the month that is passed in as a parameter. All corresponding holidays for that month are returned and displayed on the calendar.</p> <p>Parameters: Month - the current month being displayed on the calendar. All corresponding holidays for the month are displayed.</p> <p>Return: None</p>
editEvent(Event)	Method Description
	<p>This method updates the information for a calendar event based on the user input. The Event object is passed in as a parameter and the system updates the new event information into the database.</p> <p>Parameters: Event - The Event object that is being edited by the user.</p> <p>Return: None</p>
displayError()	Method Description
	<p>This method displays an error message to the user. The message is determined by the type of error that took place in the system.</p>

	<p>Parameters: None</p> <p>Return: None</p>
	Method Description
sendMissedNotification(Notification)	<p>This method is called when a notification is activated but the user is currently not logged in to their student calendar application. It sends a message to the user based on their personal notification settings.</p> <p>Parameters: Notification - the notification object that has been activated.</p> <p>Return: None</p>
	Method Description
activateNotification(Notification)	<p>This method is called when a notification is set to be sent to the user. The notification object is passed as a parameter which is then added to the users list of notifications and the calendar shows the user that a new notification has activated.</p> <p>Parameters:</p> <p>Notification - the notification object that has been activated.</p> <p>Return: None</p>
	Method Description
sendReminder(notification, users)	<p>This method is called when a user wants to share a notification with other users. A notification is passed in as well as a list of users to be notified.</p> <p>Parameters:</p> <p>Notification - the notification object that the user wants to share with other users.</p>

	<p>Users - the list of users that have been selected to receive the notification.</p> <p>Return: None</p>
	Method Description
displayCalendarSyncOptions()	<p>This method is called when a user chooses to import or export their calendar information. A window is displayed allowing the user to select which external calendar they would like to sync with. The options include all external calendars as defined in section 3.2.4.1</p> <p>Parameters: None</p> <p>Return: None</p>
	Method Description
importCalendar()	<p>This method is called when a user chooses to import calendar information from an external calendar. The method calls on the UI Controller class to import calendar events. The imported events are saved to the database and then the main display is updated to show the imported items.</p> <p>Parameters: None</p> <p>Return: None</p>
	Method Description
exportCalendar()	<p>This method is called when a user chooses to export their Student Calendar information into an external calendar. The method calls on the UI Controller class to export calendar events. If the process is successful it displays a success message to the user. If there is an error an error message is displayed.</p>

	<p>Parameters: None</p> <p>Return: None</p>
syncAssignments()	Method Description
	<p>This method is called when a user chooses to sync their calendar information with their assignments on the I-learn website. The system pulls the assignment information from the I-learn website, saves them to the database, and updates the calendar display to show new assignments that may have been added.</p> <p>Parameters: None</p> <p>Return: None</p>
deleteEvent(event)	Method Description
	<p>This method is called when the user chooses to delete an event.</p> <p>Parameters: Event - the event object that is clicked on by the user to be deleted.</p> <p>Return: None</p>
	Method Pseudo Code
	<pre>eventList.remove(event)</pre>
accessSettings()	Method Description
	<p>This function displays the settings page of the application.</p> <p>Parameters: None</p> <p>Return: None</p>

	Method Pseudo Code
	<pre>document.backgroundColor = settings.darkColor userSettings <- loadUserSettings()</pre>
	Method Description
	<p>This function displays the notifications received by the user.</p> <p>Parameters: None</p> <p>Return: None</p>
accessNotifications()	Method Pseudo Code
	<pre>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()</pre>

4.2.1.3 Login Page View

Class Name: Login Page View

Description: 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. This class holds the user's identification and password as well as the methods needed to authenticate and create a new and existing account. This class 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)	Method Description
login(username, password)	<p>This method authenticates the user and logs them in. If the user successfully logs in the main calendar page is displayed. If either the username or password is incorrect the user will be notified and prompted to reenter their credentials.</p> <p>Parameters: Username - The username entered by the user attempting to log in.</p> <p>Password - The password entered by the user attempting to log in.</p>

	Return: None
	Method Pseudo Code
	ulController.LoginStudent(username, password);
	Method Description
	<p>This method is to be called when the user attempts to create a new account. This method opens a new account form window for the user to enter their new account information. The information is then sent to the database, verified, and the new account is created. If the information is invalid an error message is sent to the user.</p> <p>Parameters: None</p> <p>Return: None</p>
createAccount()	Method Pseudo Code
	<pre> 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 </pre>

	<p>ELSE IF "unsuccessful" displayNewAccountForm() and alert user to error</p>
displayNewAccountForm()	Method Description
	<p>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.</p> <p>Parameters: None</p> <p>Return: None</p>
	Method Pseudo Code
	<pre>url <- URL of page containing New Account Form window.location(url)</pre>
recoverPassword(email)	Method Description
	<p>This method allows the user to recover their password in the event that they have forgotten their login password. The system prompts the user to enter their email which is passed into the system. If the email is verified to belong to the user an email is sent with a password recovery link. If the email is not verified the user is notified and prompted to reenter an email address.</p> <p>Parameters: email - The email address entered by the user that will be verified by the system and used to send a recovery link.</p> <p>Return: None</p>

recoverUsername(email)	Method Description
	<p>This method allows the user to recover their username in the event that they have forgotten their login information. The system prompts the user to enter their email which is passed into the system. If the email is verified to belong to the user an email is sent with a username recovery link. If the email is not verified the user is notified and prompted to reenter an email address.</p> <p>Parameters: email - The email address entered by the user that will be verified by the system and used to send a recovery link.</p> <p>Return: None</p>
changePage(URL)	Method Description
	<p>This method is called when a user successfully logs in to their account. The URL for the main calendar page is passed into the method and the system takes the user to the main calendar display screen.</p> <p>Parameters: URL - The URL address that determines which web page is displayed to the user.</p> <p>Return: None</p>
displayLoginError()	Method Description
	<p>This method is called when a user enters the wrong credentials for their account. The user is notified that their credentials were incorrect, and the user is prompted to reenter their username and password.</p> <p>Parameters: None</p> <p>Return: None</p>

emailPasswordLink(email)	Method Description
	<p>This method is called when a user is attempting to recover their password. If the user is able to verify their email address, then the method sends them a password recovery link which will allow them to change their password.</p> <p>Parameters: email - The email address entered by the user that will be used to send a recovery link.</p> <p>Return: None</p>
emailUsernameLink(email)	Method Description
	<p>This method is called when a user is attempting to recover their Username. If the user is able to verify their email address, then the method sends them an email notifying them of their username.</p> <p>Parameters: email - The email address entered by the user that will be used to send a recovery link.</p> <p>Return: None</p>

4.2.1.4 Profile Settings View

Class Name: Profile Settings View	
<p>Description: This class allows the user to change settings about their profile on the profile settings page. It contains all information pertaining to how the profile settings will be displayed to the user and provides the user with an interactive text fields for entering their account information.</p>	
Attributes(Fields)	Attribute Description

Text Fields	The fields through which the user enters all of their profile information.
Time Zones	A list of time zones which the user may choose from to change their time zone settings.
Methods(operations)	
updatePassword(new password)	Method Description
	<p>This method allows the user to update the password for their account. The method passes in the new password and updates the student's information in the database.</p> <p>Parameters: New Password - The password that the user has submitted as their new password.</p> <p>Return: None</p>
setTimeZone(Time Zone)	Method Description
	<p>This method allows the user to update their time zone for their account. The user selects from the list of time zones. The time zone is then passed into the system and the student's information is updated in the database.</p> <p>Parameters: Time Zone - The time zone indicating which time zone the user wants their account set to.</p> <p>Return: None</p>
updateStudentInfo(Text Fields)	Method Description
	<p>This method allows the user to update their personal contact information for their account. The user enters their updated information into the text fields on the profile settings page. These text fields are then passed into the system and the student's information is updated in the database.</p> <p>Parameters: Text Fields - The information pulled from the profile settings page text fields. Used to update the student's information.</p>

	Return: None
	Method Description
setNotification(toggle)	<p>This method allows the user to enable or disable their notifications. The page presents a toggle switch for the user to press on or off. This boolean value is passed into the system and the notifications are either enabled or disabled depending on the state of the toggle switch.</p> <p>Parameters: Toggle - This boolean value that determines if the notifications are either enabled or disabled depending on the state of the toggle switch.</p> <p>Return: None</p>

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

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 (see [4.2.3.1 for definition of Calendar Item Class](#)) 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 (see 4.2.3.2 for definition of Assignment Class).
Event List	The list of all calendar events that the user has created (see 4.2.3.3 for definition of Event Class).
Notification List	The list of all notifications that the user has created (see 4.2.3.4 for definition of Notification Class).

I-Learn Facade	An Instance of the I-Learn Facade class that is used to pull data from the BYU-I I-Learn system (see 4.2.2.2 for definition of I-Learn Facade).
Database Facade	An Instance of the Database Facade class that is used to store and retrieve data from the account database (see 4.2.2.2 for definition of Database Façade Class).
Student	The student class object. Holds the data for the student using the system (see 4.2.3.5 for definition of Student Class).
Methods(operations)	
	Method Description
addEvent	<p>This method 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.</p> <p>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. 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.</p> <p>Parameters: Event Info- Information submitted by the user including the event name, description, weekly occurrence, and the daily occurrence. This information is used to create a new event object. Return: The Event object that is created.</p>

	<p style="text-align: center;">Method Pseudo Code</p> <pre> addEvent(Event Info): eventOccurrenceNumber = convert Day Array into number(eventOccurrence[]) newEvent = create Event(eventName, eventDescription, eventFrequency, eventOccurrenceNumber) DatabaseFacade.addEvent(newEvent) return newEvent </pre>
loadAssignments()	<p style="text-align: center;">Method Description</p> <p>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.</p> <p>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.</p> <p>Parameters: None</p> <p>Return: The list of assignments pulled from I-learn</p>
	<p style="text-align: center;">Method Pseudo Code</p>
	<pre> loadAssignments(User): I-LearnFacade.Login(User.username, User.password) assignments = I-LearnFacade.getAssignments() DatabaseFacade.addAssignments(assignments) return assignments </pre>

addNotification	Method Description
	<p>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 notification 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.</p> <p>Parameters: Calendar Item - The Calendar Item object which the new notification will become associated with.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>
	Method Pseudo Code
editEvent(eventInfo)	<pre> addNotification(CalendarItem): newNotification = create Notification(CalendarItem) DatabaseFacade.addNotification(newNotification) if successful: return true else return false </pre>
	Method Description
<p>This method allows a user to modify the information for an event that already exists on their calendar. The method takes in the new event information provided by the user and saves those</p>	

	<p>changes into the system database.</p> <p>Parameters: Event Info - Information submitted by the user including the event name, description, weekly occurrence, and the daily occurrence. This information is used to create a new event object.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>
<p>importCalendarItems(API name)</p>	<p style="text-align: center;">Method Description</p> <p>This method allows a user to import external calendar information into the system. The method takes in the name of an API (Google, Apple, etc.) which is used to determine which API call is needed in the Calendar Integration Facade class.</p> <p>After the events are imported they are saved into the database and added to the display. If there is an error during the import process the procedure is aborted and the user is notified through an error message.</p> <p>Parameters: API name - A string identifier of which API the method will be using. It determines which API call is needed in the Calendar Integration Facade class.</p> <p>Return: The imported calendar Items.</p>
<p>exportCalendarItems(API name)</p>	<p style="text-align: center;">Method Description</p> <p>This method allows a user to export events from the Student Calendar system to an external calendar. The method takes in the name of an API (Google, Apple, etc.) which is used to determine which API call is needed in the Calendar Integration Facade class. After the events are exported the user is notified. If an error occurs the procedure is aborted, and the user is notified through an error message.</p>

	<p>Parameters: API name - A string identifier of which API the method will be using. It determines which API call is needed in the Calendar Integration Facade class.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>
syncAssignments()	Method Description
	<p>This method allows the user to synchronize the assignments that are displayed on their Student Calendar with the assignments that exist on their I-learn account. The method pulls a list of the user's assignments from I-learn, updates the list that exists in the system database, and adds new assignments to the display. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: None</p> <p>Return: The list of assignments pulled from I-Learn</p>
loadCalendarItems(Student)	Method Description
	<p>This method loads the user's calendar item's when the user first logs in to the system. The method passes in the student class object containing the user data. This is used by the database to retrieve all the calendar items associated with that student. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: Student - the student class object containing the user data. This is used by the database to retrieve all the calendar items associated with that student.</p> <p>Return: collection of Calendar Items associated with that student.</p>

4.2.2.2 I-Learn Facade

Class Name: I-Learn Facade

Description: The I-Learn Façade class will pull the user's information, assignments and courses from I-Learn using the Desire To Learn API([see reference \[6\] in section 1.5](#)). This class will interact with the Calendar Item Controller class ([see 4.2.2.1](#)) to provide assignments ([see 4.2.3.2 for definition of Assignment Class](#)).

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)	Method Description
login(username, password)	Authenticate with I-Learn, this is used by the Calendar Item Controller to first authenticate with I-Learn. Parameters: username - A string of the username the student uses to login to I-Learn password - A string of the password the student uses to login to I-Learn

	Return: None
	Method Pseudo Code
	<pre> 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'] </pre>
getCourse()	Method Description
	<p>This method uses the D2L API to get the users course list. It then filters to make sure the course is active. These courses are stored in the Course class.</p> <p>Parameters: None</p> <p>Return: Course object with all courses retrieved from I-Learn</p>
	Method Pseudo Code
	<pre> 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']) </pre>

	<pre>tempCourse.setStartDate(course['Access']['StartDate']) tempCourse.setEndDate(course['Access']['EndDate']) courses INSERT tempCourse RETURN courses</pre>
getAssignments()	Method Description
	<p>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.</p> <p>Parameters: None Return: Assignment object with all assignments from each course</p>
	Method Pseudo Code
<pre>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</pre>	

	<pre>tempAssignment.assignmentCompleted = FALSE tempAssignment.assignmentURL = assignment.href assignments[course.getId()] INSERT tempAssignment RETURN assignments</pre>
	Method Description
getAssignmentURL(assignment)	<p>Finds the URL link for the corresponding I-Learn Assignment for the user to open on their browser.</p> <p>Parameters: Assignment – The assignment selected by the user. This will be used to determine which URL link will be returned.</p> <p>Return: A string with the URL of the assignment</p>

4.2.2.3 Student Controller

Class Name: Student Controller	
<p>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 (see 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 (see 4.2.2.4) to pass student data between the system and the user. The Student Controller interfaces with the Log-In Use Case (see 3.2.7.0). It also interfaces with the Profile Settings use case (see 3.2.5.0 & 3.2.5.2) to set additional user settings.</p>	
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. Parameters: Name - A string with the name of the student Username - A string with the username of the student Password - A string with the password of the student Return: None
	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

	<p>Parameters: Username - A string with the username of the student</p> <p>Return: A student object with the desired student to retrieve</p>
	Method Pseudo Code
	<pre>tempStudent = DatabaseFacade.loadStudent(username) RETURN tempStudent</pre>
updateUserNotifications(notificationToggle)	<p style="text-align: center;">Method Description</p> <p>Toggles notifications on or off for the student.</p> <p>Parameters: notificationToggle - A boolean with on or off state for notifications</p> <p>Return: None</p> <p style="text-align: center;">Method Pseudo Code</p> <pre>currentStudent.notifications = notificationToggle DatabaseFacade.modifyStudentInfo(currentStudent)</pre>
verifyEmail(Email)	Method Description

	<p>Verifies that the email address is associated with an account</p> <p>Parameters: Email - A string with the email of the student</p> <p>Return: A Boolean if the email matches an existing student account</p> <p style="text-align: center;">Method Pseudo Code</p> <pre>IF currentStudent.email == Email RETURN true ELSE RETURN false</pre>
setTimeZone(timeZone)	<p style="text-align: center;">Method Description</p> <p>Sets the student's time zone</p> <p>Parameters: timeZone - A string representation of the timezone</p>
updatePassword(password)	<p style="text-align: center;">Method Description</p> <p>Updates the student's password in the database</p> <p>Parameters:</p>

	password - A string with the student's password
	Method Description
updateUserInfo(UserInfo)	<p>Updates the current student with new user information</p> <p>Parameters: UserInfo - A Student object with student information</p>

4.2.2.4 UI Controller

Class Name: UI Controller	
<p>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.</p> <p>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 (see 4.2.2.5) to verify user credentials, the I-Learn facade class (see 4.2.2.2) to retrieve assignment information, the student controller class (see 4.2.2.3) to retrieve information about the current user, and the calendar item controller class to retrieve and update information about the user's calendar.</p>	
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.
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.
Class	The university course that the user's assignment was assigned from.
Calendar View	The web app page that displays the user's calendar.
I-Learn Credentials	The unique username and password that the user has already established with their university to login to the I-Learn application.
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. Parameters: Filters - the calendar filters previously selected by the user.
	Method Pseudo Code

	<p>getEvents(filters) Return databaseQuery(filters)</p>
viewEvent	Method Description
	<p>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. Parameters: eventId - the ID for the event that was selected by the user. Return: An event from the event class to be displayed</p>
	Method Pseudo Code
	<p>viewEvent(eventId) return event FROM eventTable WHERE eventId = eventId</p>
viewSelection	Method Description
	<p>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. Parameters: startDate - the date the user specifies to start listing events. endDate - the date the user specifies to stop listing events. class - the course the student is currently viewing. Return:</p>

	<p>Each event that falls within the specified range.</p> <p style="text-align: center;">Method Pseudo Code</p> <pre> 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 </pre>
CreateEvent	<p style="text-align: center;">Method Description</p> <p>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. Parameters: eventId - the ID for the event that was selected by the user.</p> <p style="text-align: center;">Method Pseudo Code</p> <pre> createEvent(eventId) New Event = CalendarItemController.addEvent(EventId) return New Event </pre>

searchCalendar	Method Description
	<p>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.</p> <p>Parameters: searchQuery - the text that was entered into the search bar.</p> <p>Return: Each event that matches the search query.</p>
	Method Pseudo Code
	<pre>searchCalendar(searchQuery) return events FROM eventTable WHERE eventName LIKE searchQuery OR eventDescription LIKE searchQuery</pre>
createAccount	Method Description
	<p>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.</p> <p>Parameters: username - the user's username. password - the user's password.</p> <p>Return: A boolean that is true if creating account was successful.</p>

	Method Pseudo Code
	<pre>createAccount(username, password) { If (verifyAccount(username, password) return true</pre>
login	Method Description
	<p>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.</p> <p>Parameters: username - the user's username. password - the user's password.</p> <p>Return: A boolean that is true if creating account was successful.</p>
	Method Pseudo Code
	<pre>login(username, password) { If (verifyLogin(username, password) return true</pre>
recoverPassword	Method Description
	<p>Allows the user to recover a forgotten password. The request is sent to the database facade class which sends the user an email to the email address provided allowing them to retrieve their</p>

	<p>information.</p> <p>Parameters: email - the user's email.</p> <p>Return: A boolean that is true if recovery was successful.</p>
recoverUsername	Method Description
	<p>Allows the user to recover a username. The request is sent to the database facade class which sends the user an email to the email address provided allowing them to retrieve their information.</p> <p>Parameters: email - the user's email.</p> <p>Return: A boolean that is true if recovery was successful.</p>
createNotification	Method Description
	<p>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.</p> <p>Parameters: assignment - an object of type Assignment that holds all of the information about the assignment that is due.</p>
	Method Pseudo Code
	<pre>createNotification(Assignment assignment) { pushNotification("Assignment: " + assignment.name + "is due</pre>

	in 2 days.") }
addToDoAssignment	Method Description
	<p>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.</p> <p>Parameters: customToDo - a toDo object that holds all of the information about a to-do list entry.</p> <p>Return: A boolean that is true if adding the to-do list item was successful.</p>
	Method Pseudo Code
	<pre>addToDoAssignment(toDo customToDo) { if (INSERT INTO toDoList VALUES(customToDo)) return true }</pre>
removeToDoAssignment	Method Description
	<p>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.</p> <p>Parameters: toDo - a toDo object that holds all of the information about a to-do list entry.</p> <p>Return:</p>

	A boolean that is true if adding the to-do list item was successful.
	Method Pseudo Code
	<pre>removeToDoAssignment(toDo toDo) { if REMOVE FROM toDoList VALUES(toDo) return true }</pre>
get_Ilearn_Assignments	Method Description
	Allows the user to import their assignments from I-Learn. Sends a request to the I-Learn facade class which retrieves the assignments and sends them to the Database Facade class to be stored.
sendNotification	Method Description
	<p>This method allows the user to share a notification they have received with other users. To share the user clicks the share button when a notification is received and then specifies which users to share with. The UI Controller class sends this information to the other users and displays a confirmation message to the user.</p> <p>Parameters: notification - the notification that will be sent to other user's. users - an array holding every user's ID that will receive the notification.</p>
	Method Pseudo Code
	<pre>sendNotification(notification, users) { for each user {</pre>

	<code>notification.send(user[i]) } }</code>
	Method Description
importCalendar	<p>Enables the user to import a calendar from one of the supported calendar APIs. In the calendar settings the user selects import calendar and specifies which calendar they would like to import. The UI controller class sends this information to the Calendar Item Controller which makes the request to the Calendar Integration Facade. The user is then notified if the integration was successful or not.</p> <p>Parameters: apiName - the name of the API that hosts the specified calendar.</p> <p>Return: A boolean that is true if adding the calendar was successful.</p>
	Method Description
exportCalendar	<p>Enables the user to export their calendar to one of the supported calendar APIs. In the calendar settings the user selects export calendar and specifies which calendar api they would like to export to. The UI controller class sends this information to the Calendar Item Controller which makes the request to the Calendar Integration Facade. The user is then notified if the export was successful or not.</p> <p>Parameters: apiName - the name of the API that will receive the user's calendar.</p> <p>Return: A boolean that is true if exporting the calendar was successful.</p>

4.2.2.5 Database Facade

Class Name: Database Façade

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.

The Database Façade's use in the overall scheme of the project is to allow the saving of user data to the system database. 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)	Method Description
saveEvent(Event)	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.

	<p>Boolean that is true if the procedure was successful, false if an error occurred.</p>
	<p style="text-align: center;">Method Pseudo Code</p>
	<pre>saveEvent(Event): Open connection to database using Connection Insert event into Event table in database Close Connection if successful: return true else: return false</pre>
<p>SaveAssignments(Assignments[])</p>	<p style="text-align: center;">Method Description</p>
	<p>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.</p>
	<p style="text-align: center;">Method Pseudo Code</p>
	<pre>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</pre>

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
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

	<p>Insert Student into Student table in database Close Connection if successful: return true else: return false</p>
<p>verifyLogin(username, password)</p>	<p>Method Description</p>
	<p>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.</p> <p>Parameters: Username - the user's username. Password - the user's password.</p> <p>Return: A Boolean that is true if the login was successful.</p>
	<p>Method Pseudo Code</p>
	<p>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</p>

	<p>if Student.password is equal to password: return true else: return false</p>
	Method Description
loadCalendarItems(Student)	<p>This method loads the user's calendar item's when the user first logs in to the system. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: Student - the student class object containing the user data. This is used by the database to retrieve all of the calendar items associated with that student.</p> <p>Return: collection of Calendar Items associated with that student.</p>
	Method Description
loadToDoList()	<p>This method loads the user's assignments from the database to be included in the to-do List. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: none</p> <p>Return: List of Assignments to be added to the To-Do list.</p>
	Method Description
deleteToDoListItem(Assignment)	<p>This method removes a selected assignment from the database. If an error occurs the procedure is aborted, and the user is notified through an error message.</p>

	<p>Parameters: Assignment - the assignment object that has been marked by the Student for deletion from the Calendar.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>
saveImportedCalendarItems(Events)	<p style="text-align: center;">Method Description</p> <p>This method stores all imported information received from an external calendar. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: Events - Calendar Events imported from the external Calendar.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>
modifyStudentInfo(Student)	<p style="text-align: center;">Method Description</p> <p>This method allows a user to modify their personal account information. The passed information is saved into the database. If an error occurs the procedure is aborted, and the user is notified through an error message.</p> <p>Parameters: Student - The Student object whose information is being updated through the profile settings.</p> <p>Return: Boolean that is true if the procedure was successful, false if an error occurred.</p>

4.2.2.6 Calendar Integration Facade

Class Name: Calendar Integration Façade	
<p>Description: The Calendar Integration Facade class 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 (3.2.4.1).</p>	
Attributes(Fields)	Attribute Description
Google Calendar API	An interface with Google Calendar. Version 3 of the API will be implemented (see reference [3] in section 1.5)
Apple Calendar API	An interface with Apple Calendar. This will be the EventKit Framework which allows the access of Apple Calendar events (see reference [4] in section 1.5). The Minimum version available as of 2017 will be implemented.
Outlook Calendar API	An interface with Outlook Calendar. The Minimum version available as of 2017 will be implemented (see reference [5] in section 1.5).
iCalendar	Holds iCalendar file information. Used for importing and exporting data to external calendars.
Methods(operations)	Method Description
getImportedCalendarItems()	<p>This method imports information from an external calendar using the API that the user has selected. The method returns the imported calendar items.</p> <p>Parameters: None</p> <p>Return: A list of Calendar Item objects with the items that are imported</p>

	Method Description
exportCalendarItems(Event)	This method exports an event into the desired calendar integration. Parameters: Event - A Calendar Item

4.2.3 Model Classes

The model classes contain all of 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	
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

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

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

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 an 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

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	
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. Used to find the correct course in I-Learn.
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 use a standard database connection technology.

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. The browsers will be tested using their respective minimum versions that are available as of 2017.

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

The *Student Calendar Integration Application* will store student, course, event, and notification information in a relational database. A relational database is a good candidate for managing complex relationships between entities and for maintaining integrity while reading and writing items to the database. The relational model depicted in the ERD (section 5.2) is designed to scale with the anticipated student user base. The Data Dictionary (section 5.3) provides detail on table, fieldname, data type, and nullability.

5.2 Entity Relationship Diagram

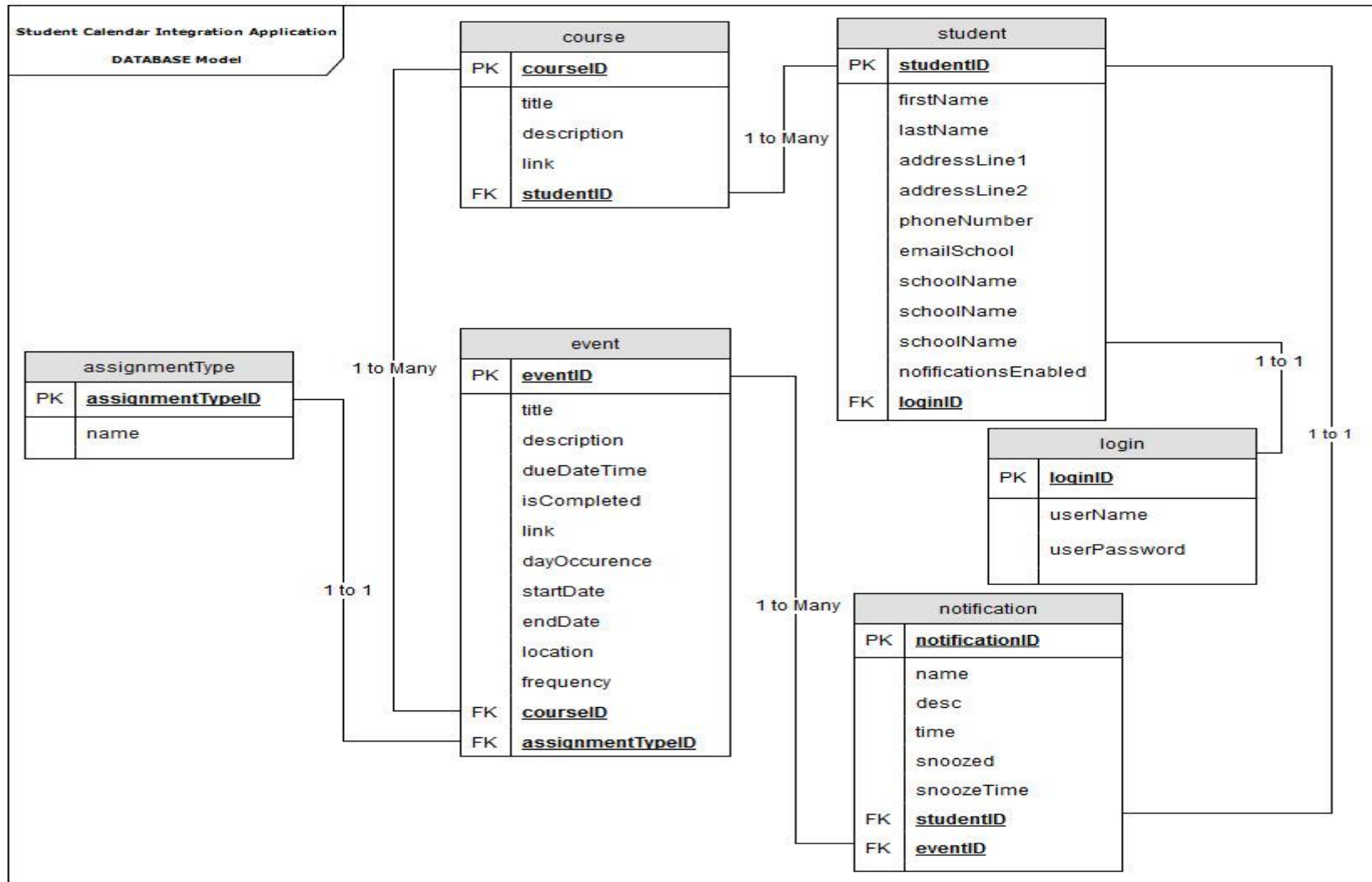


Figure 41 - Entity Relationship Diagram

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

Table Name	Field	Type	Null	Default
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

Table Name	Field	Type	Null	Default
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