Software Requirements Specification (SRS)
Enhanced Blackboard Calendar

Team: Project Team 6
Authors: Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura
Customer: Blackboard Inc.
Instructor: Dr. James Daly

Table of Contents

1. Introduction

Blackboard is a learning management system used to coordinate assignments between students and instructors at UMass Lowell and other universities. The product's main functionality is targeted to college students who need a centralized place to organize their schedule around classes and other events.

Blackboard's calendar component is quite lackluster in features. Currently, Blackboard only acts as a simple snapshot of class work that is due, but there is a lot of room for improvement. The event modifications are very limited, and Blackboard's web component simply lacks advanced features to help the students manage their class work. With improvements, it potentially can act as a main scheduling tool for students.

In this document, we propose what our vision of an Enhanced Blackboard Calendar would be. We propose features that we as students would find useful if the current Blackboard calendar had them. The document covers what features will be added on top of its base functionality. The requirements for these functionalities, as well as some visual diagrams to act as guidelines for the development process of the new Enhanced Blackboard Calendar, will be described in detail in the following sections.

1.1 Purpose

This SRS provides an overall vision of the new Blackboard calendar for the developers. The document is to act as a guideline for the development process, which includes proposed features that improve the user's experience of Blackboard's calendar component. This document shows our client how we plan to implement these new features.

1.2 Scope

The Enhanced Blackboard Calendar is simply an upgraded calendar component of the Blackboard application. The improvements we propose would make the calendar application appear more complete and give Blackboard users a better experience with the application. The object of this improvement is to give the students a better way to use this calendar application to maintain their busy academic life. Although the prototypes will be a standalone web application, we envision our software to be a direct replacement for the current calendar component of the Blackboard application. The software should give its users better management of their schedule, with increased customization of their schedule and new features such as email or SMS notification for critical events. However, users cannot change official classwork assigned to them by course instructors.

1.3 Definitions, Acronyms, and Abbreviations

- **Events -** Basic units of a calendar. Described by a title and dates for when an event occurs.
- **Calendar** - A titled collection of events.

- **Class Calendar** - A calendar loaded from Blackboard's course information for the user's courses**.**
- **Class Events -** Events belonging to a class calendar. Class events vary from assignments, quizzes, etc.
- **Main Calendar Display**- The frontend display with which the user interacts. Communicates with the backend to create, read, update, and delete data.
- **Dates** - Gregorian calendar dates, which include a day, month, year, and time.
- **Notification -** A reminder for an event that a user requests to receive via SMS or email.

1.4 Organization

The document will cover descriptions for basic and additional features of the Enhanced Blackboard Calendar. Assumptions for this updated component of Blackboard are stated. Such statements include an overall vision of how the new component would behave and interact with other systems. Assumptions on intended users and how they would integrate this new component into their daily lives are made. High-level designs of how the new component should behave are also described in this document.

These high-level designs include, but are not limited to: requirements; a use case diagram; descriptions for each use case; a class diagram; descriptions for each class; sequence diagrams of particular use cases; and an overall high-level state diagram of the system. These high-level diagrams describe our planned structures of the new calendar component. Lastly, we describe the prototype that will be iterated upon for the final product and how to run said prototype.

2. Overall Description

This section covers what the Enhanced Blackboard Calendar is, and how users will interact with it. The **Product Perspective** section details how the user will interact with the product, and what the user can do with the application. The specific features and improvements of the product compared to the current Blackboard calendar are detailed in the **Product Function** section. The target demographic for the product and prerequisite knowledge necessary to use it is specified within the constraints of the product. The dependencies required to deliver the service we envisioned are also discussed in the **Dependencies** section. The scope of the project and the apportioning of requirements are also described with respect to the limited time available to dedicate to the project.

2.1. Product Perspective

The Enhanced Blackboard Calendar aims to be an upgraded version of the current calendar page on Blackboard. Compared to the current Blackboard calendar, the main advantages of our project is a more straightforward user interface and the ability to notify users, alongside other upgrades.

The product is designed to be a component of Blackboard. Ideally, the Enhanced Blackboard Calendar would replace the existing calendar within Blackboard, with access to information about the student's classwork. The interface students use to interact with the product is a web app primarily consisting of a main calendar, which displays events from various calendars. This portion of the app is called the Main Calendar Display. Students will be able to add calendars, customize their name and appearance, and have the ability to add events occurring at a specified time. Each event has their own optional notification settings, which can notify the students via email or text message ahead of time.

Any computer capable of running a reasonably modern Internet browser is able to access the Enhanced Blackboard Calendar. The skills needed to navigate and use it are largely limited to moving and clicking a mouse and entering text through a keyboard. However, an email address and/or a cell phone is necessary to receive notifications, which requires an Internet connection or cell phone signal. An Internet connection is necessary to download the page itself and to modify calendar elements.

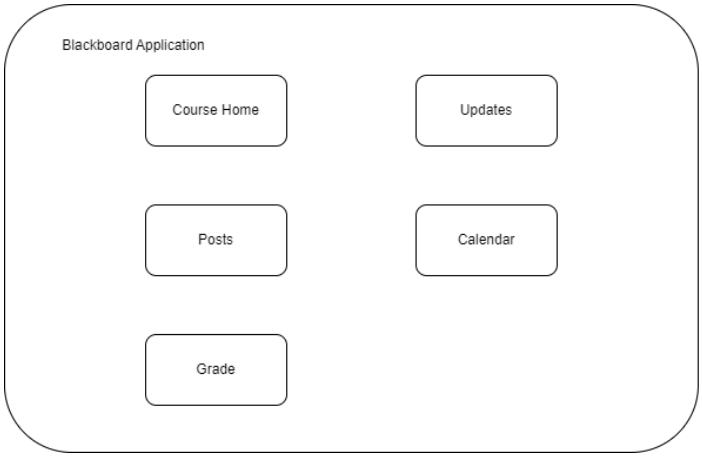

Figure 1: Pictorial Representation of Bigger System

2.2. Product Functions

The Enhanced Blackboard Calendar is a web app with a calendar that has the ability to store events that can notify the user. The user interacts with a main calendar through the Main Calendar Display, and can create new calendars, events, and notifications through it. The Enhanced Blackboard Calendar provides users with a comprehensive, customizable calendar that is integrated with their enrolled classes.

The main display will show all calendars with their associated events. Users can select certain calendars to be visible, and export or print the selected calendars. Users can create and edit calendars, events for each calendar, and notifications for each event. Each calendar has a unique title and color which distinguishes it from other calendars, in

addition to having a group of events. Each event has a unique title compared to other events in the same calendar and has an optional notification. Notifications can notify the user by a text message and/or email.

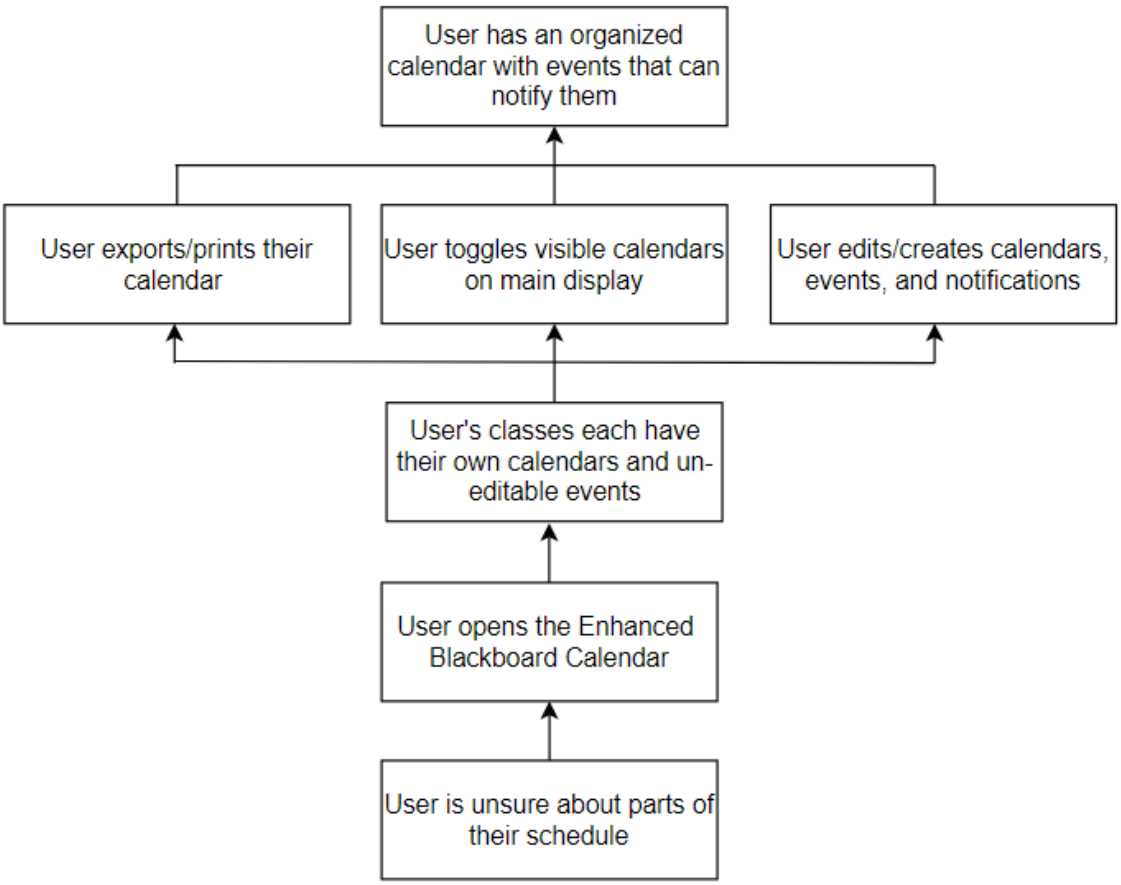

Figure 2: High-level Goals

2.3. User Characteristics

The expected users of this program will be college students, ranging anywhere from freshmen to graduates. Their majors can span any that UMass Lowell provides, and can be participating in a vast number of different courses. Realistically, it is expected that every user on average will be taking four classes per semester, generally pertaining to their major of choice.

The users' skills regarding the use of the calendar software are expected to vary drastically between students, as some may have extensive experience with similar calendar programs, and others may have never used one before. All students are expected

to have some level of basic website navigation skills and understanding of the English language in order to properly use this program.

In order for a user to have the best experience while using this software, it is ideal that the user has some degree of planning and coordination skills to best utilize the calendar's features. Additionally, it is assumed that this program will be run on a laptop or desktop computer with a web browser and a stable Internet connection.

2.4. Constraints

One constraint this project has is the Internet connection. As it currently stands, our program does not support continuous synchronization of data, and only saves changes as they are applied. In the event that a user loses their Internet connection while creating a calendar, event, or notification, their progress will be lost, and they will need to retry from the beginning once their connection is restored.

Another constraint is that the user must have certain devices to utilize some of the program's features. Firstly, for the user to receive text notifications, the user must have a smartphone with a carrier service. Secondly, to print a copy of their calendar, the user must have a printer with sufficient ink and paper. Without either of these devices, the user will be prevented from using large features of the program.

2.5. Assumptions and Dependencies

We expect users to access the calendar through a web browser. This means the system will be platform-independent. We expect users to access our application with phones, tablets, personal computers, and laptops. As such, inputs can be through a mouse, keyboard, or touch screen controls.

Users must have access to the Internet. There are many browsers the user could use, but we mainly expect users to access the application through a popular and well-supported browser like Google Chrome or Safari. We are not accounting for many different browsers such as Microsoft Edge, Firefox, etc. Since the application is accessed through a web browser, any operating system can be used.

In terms of user interactions, we expect users will use this application to have an overall view of the classworks assigned to them by their professors. We expect users will organize their schedule by adding events to calendars.

A fully integrated Enhanced Blackboard Calendar application would require course information for the student from the main Blackboard application.

2.6. Apportioning of Requirements

There are some features that are likely to be implemented in later versions, which we determined would not be possible for our prototype for various reasons. Firstly, we discussed the possibility of exporting calendars in the form of a standard iCalendar file (.ics extension), which we determined to be far too complex for our prototype given our time constraints. We also thought about allowing the user to export a calendar to print it

out. This feature was not implemented, since it is also too complex given our time constraints.

Another large feature that we considered was a mobile version of the program, which would drastically alter the UI and general behavior to better fit a small touchscreen. We discussed this feature for a considerable amount of time, but we have reached the conclusion that the effort it would require would draw from the effort required to add more important features to the program.

Additionally, the SMS text messages are not implemented at all. Conceptually, it would be easy to implement- there are many SMS APIs available to developers for such purposes. However, these APIs are not free, and we are broke college students who didn't want to spend money if we didn't have to.

Finally, as the project is a prototype and not a full integration of the application into Blackboard, we will not be using live data from real student Blackboard courses.

3. Specific Requirements

1. Events will have a title, a date on which they occur, an optional duration, and an optional description.

- **1.1.** The user can create personal events.
  - **1.1.1.** The user must enter the date at which the event occurs.
  - **1.1.2.** The user may specify a duration for the personal event.
  - **1.1.3.** The user may specify if the event repeats.
- **1.2.** The user can delete personal events they have created.
- **1.3.** The user can edit the attributes of existing personal events.

2. Calendars will have a title, and a color.

- **2.1.** A calendar’s title must be unique.
- **2.2.** Calendars contain events.
  - **2.2.1.** The event’s title must be unique in a calendar.
- **2.3.** A personal calendar will be automatically created for the user.
  - **2.3.1.** Events created by the user are assigned to this calendar by default.
- **2.4.** The user can create calendars.
  - **2.4.1.** The user may add new events to calendars they have created.
- **2.5.** The user may delete calendars they have created.
  - **2.5.1.** Upon deletion, the calendar’s events will also be deleted.
    - **2.5.1.1.** The user may transfer events from the calendar before it is deleted.

2.6. The user can change the attributes of existing calendars.

2.7. The user can transfer events from one calendar to another.

2.7.1. The event's attributes do not get altered by the transfer process.

2.7.1.1. If the event's title exactly matches the title of another event in the new calendar to which it will be moved, one of the two event titles must be altered so that every title within the calendar is unique.

2.8. Class calendars will be automatically created for each class the user is taking.

2.8.1. Class events are created in the class calendar for each assignment the class has.

2.8.1.1. Class events have additional submission, completion, and grade attributes.

2.8.2. The user cannot change any attributes of a class calendar, or its contained class events, except for the class calendar’s color.

2.8.3. The user cannot delete any class calendar.

3. The user can set notifications that are sent to the user to remind them of any upcoming event.

3.1. The user can specify whether they are delivered via email, text message (SMS), or both.

3.2. Notifications can be enabled or disabled based on chosen calendars or individual events.

3.3. The user can change the amount of time prior to the event or the assignment they will be notified.

3.4. The user can create repeating reminder notifications.

3.4.1. The user will specify the amount of time between repeating reminders.

3.5. Notifications stop being sent once the event has completed.

4. The user can export calendars.

4.1. The user can save a file or print a copy of the currently displayed calendars.

4.1.1. The user can select certain days or months to export.

4.1.2. The user can specify and filter which events to include.

5. The main calendar display consists of the visual components that the user will be able to view.

5.1. The main calendar display shows every day of the current calendar month by default.

5.2. The main calendar display shows the name of the current month.

5.3. The user can choose between three display modes- daily, weekly, and monthly.

5.3.1. Daily mode displays an hourly view of the currently selected day.

5.3.2. Weekly mode displays the days of the currently selected week.

5.3.3. Monthly mode provides a simultaneous view of every day of the current calendar month.

5.4. The user can toggle which calendars are displayed.

5.5. The main calendar display presents events at the time at which they occur.

5.5.1. Events may also display additional information about themselves.

5.5.2. Class events marked as submitted are displayed differently from unsubmitted class events.

5.5.3. Graded assignments display their grades in the calendar.

5.5.3.1. The user can show or hide displayed grades of graded assignments.

5.5.4. The user can filter the displayed events by one or more criteria.

5.5.4.1. The user can hide finished class events upon completion.

5.5.5. The main calendar display will automatically update and refresh based on changes made to events and assignments, including creation and deletion.

5.5.6. The user can navigate between different dates on the calendar.

5.5.7. The main calendar display will dynamically change its display proportions based on the current resolution.

4. Modeling Requirements

The following section deals with the modeling requirements for the Enhanced Blackboard Calendar. The diagrams and descriptions contained herein describe what the software is supposed to do, and how it is supposed to do it. This section deals with the architecture of the software. Therefore, this section does not include implementation details, but it does include the key features of every correct implementation of the

Enhanced Blackboard Calendar system. The diagrams follow the conventions of Unified Modeling Language (UML).

4.1. Use Case Diagram and Descriptions

Our use case diagram below shows the various goals that the user can achieve through use of our Enhanced Blackboard Calendar application. In the diagram below, the box consists of the Enhanced Blackboard Calendar application. The stick figure represents the user actor, who is the end-user that interacts with the system through various use cases. The ovals inside the box are the use cases, and the text inside each oval indicates that use case's title. The information contained therein is indicative of what the user can achieve through their use of the Enhanced Blackboard Calendar. For example, the user will be able to create events that will be managed by calendars that the user can also create. The user can create notifications, update calendar/event/notification information, and export selected calendars for use external to the application. An arrow from the user to a use case indicates that the user can achieve that goal through some amount of user interaction with the system. Use cases can also include or extend other use cases. When a use case is included by another use case, the included use case will always occur in addition to the original use case occurring. For example, Change Date Mode includes Show Main Calendar Display, since changing the viewing mode will always affect the view shown in the Main Calendar Display. This is denoted with a dashed arrow between use cases, and the text "<<includes>>". The arrow points from the use case to the use case being included. If a use case extends another use case, then it is a special case of the original use case. For example, Toggle Email/Text Notification extends Change Notification, since it is a special case of changing a notification's information. When you change a notification's information, it is not always the case that you will toggle email or text notifications. This relationship is indicated by a dashed arrow, and the text "<<extends>"". The arrow points from the extended use case to the original use case.

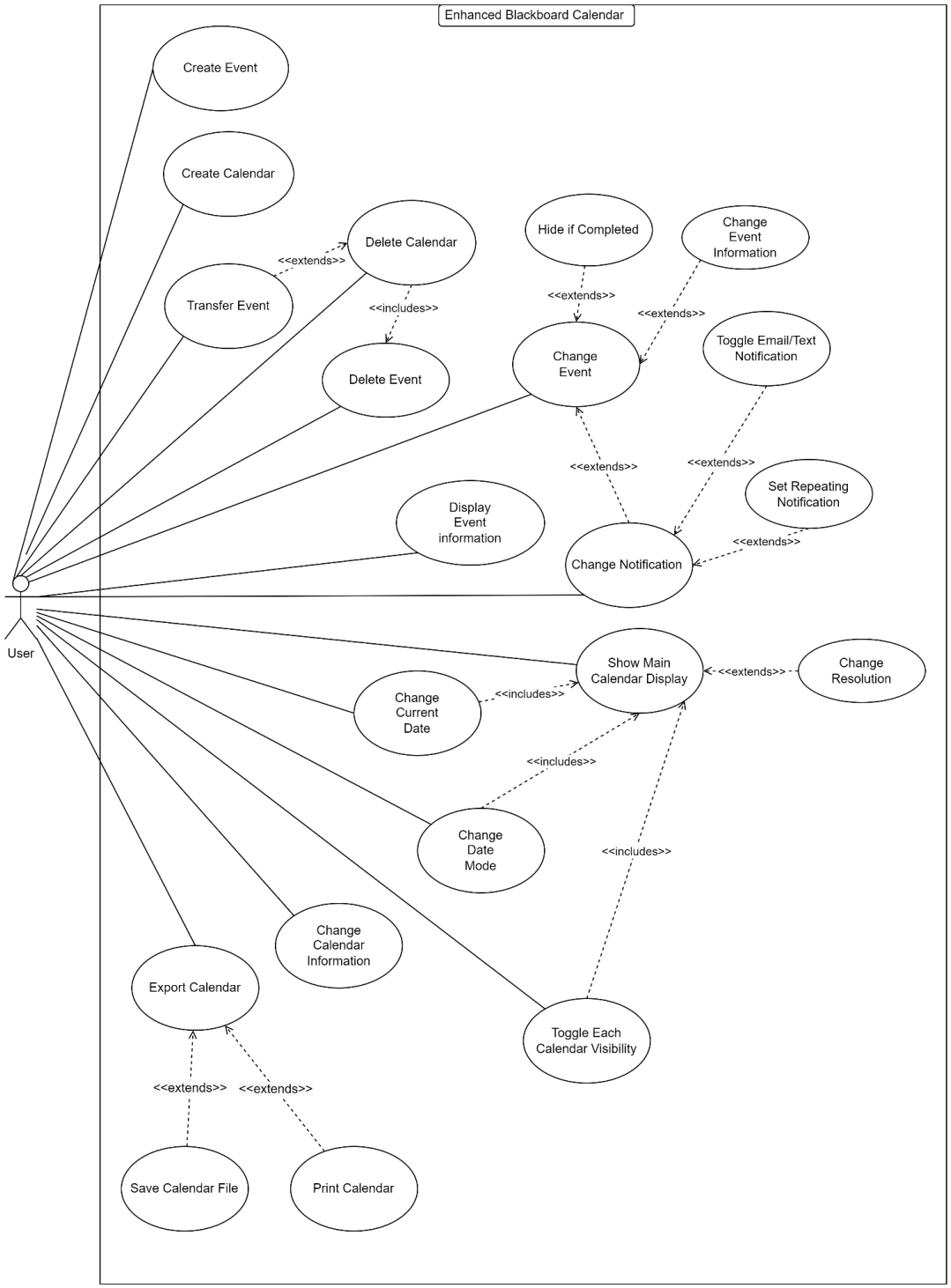

Figure 3: Use Case Diagram for the Enhanced Blackboard Calendar

The use cases included in the use case diagram are explained in more detail in the following descriptions.

Use Case Name:	Create Event
Actors:	User
Description:	The user creates a new event. The user will specify the event's attributes; namely, a title, a date, an optional duration (specifying beginning and ending times for the event), and an optional description. The user will also add the event to an already existing personal calendar. By default, the calendar chosen will be the automatically created personal calendar. The event will be saved internally by the system for future use.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 1, 2.3, and 2.4
Uses cases:	Used whenever the user has a new event they wish to track in the calendar application.

Use Case Name:	Create Calendar
Actors:	User
Description:	The user creates a new personal calendar into which they may add new events or transfer other personal events from another personal calendar. The user will specify the calendar's title and color, which will both be displayed on the main calendar display. The calendar will be saved by the system for future use.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 2
Uses cases:	Used primarily when the user wishes to track a collection of events. The user could create a personal calendar to group together similar events.

Use Case Name:	Transfer Event
Actors:	User
Description:	The user can transfer an already existing event from one personal calendar to another personal calendar. The event's attributes will not be altered in the transfer process. This use case presupposes the existence of at least two personal calendars that the user has created, one as the source calendar and the other as the destination calendar.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 2.5.1.1 and 2.7
Uses cases:	Used when the user wishes to change the personal calendar that tracks the event to be transferred.

Use Case Name:	Delete Calendar
Actors:	User
Description:	The user deletes a personal calendar that they have previously created. If the calendar contains any events, the user will be prompted for each one to either move the event to another calendar, or for the event to be completely deleted.
Type:	Primary
Includes:	Delete Event
Extends:	N/A
Cross-refs:	Requirements 2.5 and 2.8.3
Uses cases:	Used to delete a personal calendar that the user no longer wishes to have.

Use Case Name:	Delete Event
Actors:	User
Description:	The user deletes any personal event that they have created. Class events cannot be deleted. A personal event is deleted when the user deletes a personal calendar and does not transfer the event to another calendar.
Type:	Primary

Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 1.2
Uses cases:	Used to delete a personal event that the user no longer wishes to have. Also used when a personal calendar is deleted, and the user does not wish to transfer that calendar's events to another personal calendar.

Use Case Name:	Display Event Information
Actors:	User
Description:	When the user selects an event, the system will display the event's attributes to the user. The user will be able to see the event's title, date, duration, description, and the calendar of which it is a part.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 1 and 5.5.1
Uses cases:	Used when the user wishes to see more information about a particular event.

Use Case Name:	Change Event
Actors:	User
Description:	The user can change the status of an event. There are two special cases- changing the event's attributes, and hiding the event if the event is completed.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 1.3, 3, 5.5.2, 5.5.3, and 5.5.4.1
Uses cases:	Used when the user wants to alter the state or attributes of an event.

Use Case Name:	Hide if Completed
Actors:	N/A
Description:	If a completable event has been completed, the completed event can be specified to not be shown by the main calendar display. Completable events are events that have already happened, and submittable events (i.e. assignments for a class).
Type:	Primary
Includes:	N/A
Extends:	Change Event
Cross-refs:	Requirements 5.5.2 and 5.5.4.1
Uses cases:	Used when the user wants to hide completed events. The user may wish to hide events that have already occurred, or to hide class assignments that have already been submitted.

Use Case Name:	Change Event Information
Actors:	N/A
Description:	All of a personal event's attributes can be altered. The event's title, date, duration, and description can be edited. The user cannot edit the attributes of a class's event.
Type:	Primary
Includes:	N/A
Extends:	Change Event
Cross-refs:	Requirements 1.3 and 2.8.2
Uses cases:	Used when the user wishes to alter some or all of an event's attributes. The user can specify which fields to change, and what their new values will be, to better reflect the user's purpose for the event.

Use Case Name:	Change Notification
Actors:	User
Description:	The user can specify if an event or all of a particular calendar's events will have a corresponding notification. The user will determine the

	notification's method of delivery, when the notification is to be sent, and if the notification is to be repeated until the event completes.
Type:	Primary
Includes:	N/A
Extends:	Change Event
Cross-refs:	Requirement 3
Uses cases:	The user may wish to receive an email or text message reminding them of an upcoming event. Additionally, they may wish to receive a repeating reminder for the upcoming event- for example, they could be emailed every week reminding them to study for an exam in four weeks' time.

Use Case Name:	Toggle Email/Text Notification
Actors:	N/A
Description:	The user can specify the notification's method of delivery. The user will set the notification to one of the following scenarios: 1) only email, 2) only text message, 3) both email and text message, or 4) no notification at all. The setting chosen will be used to determine the notification's behavior.
Type:	Primary
Includes:	N/A
Extends:	Change Notification
Cross-refs:	Requirement 3.1
Uses cases:	Used when the user wants to configure how a notification is sent. Also used to disable a notification altogether.

Use Case Name:	Set Repeating Notification
Actors:	N/A
Description:	The user can set a notification to be repeated. The user will specify how often the notification should be repeated until the event completes.
Type:	Primary
Includes:	N/A
Extends:	Change Notification

Cross-refs:	Requirement 3.4
Uses cases:	Used when the user wants to have a notification that repeats for a given event.

Use Case Name:	Show Main Calendar Display
Actors:	User
Description:	The user displays the main calendar display simply by running the application. The main calendar display shows all of the calendars and events that are selected to be shown, in the display mode that the user wishes them to be shown.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 5
Uses cases:	Used throughout the entire application to show the interface for the various calendars, events, and information.

Use Case Name:	Change Resolution
Actors:	N/A
Description:	The resolution of the main calendar display will be dynamically and automatically resized to coincide with the resolution of the current browser window that has the application opened.
Type:	Secondary
Includes:	N/A
Extends:	Show Main Calendar Display
Cross-refs:	Requirement 5.5.7
Uses cases:	Used when the window showing the main calendar display is resized in some way. The main calendar display will automatically resize to appear correctly within the newly resized window.

Use Case Name:	Change Current Date

Actors:	User
Description:	The user can change the date that is chosen to be displayed in daily, weekly, or monthly display mode in the main calendar display.
Type:	Secondary
Includes:	Show Main Calendar Display
Extends:	N/A
Cross-refs:	Requirement 5.5.6
Uses cases:	Used when the user decides to view a different day than the one currency selected.

Use Case Name:	Change Date Mode
Actors:	User
Description:	The user can change whether the main calendar display will be shown in monthly, weekly, or daily mode. Monthly mode shows the entire currently selected month. Weekly mode shows the currently selected week, Sunday to Saturday. Daily mode shows the currently selected day.
Type:	Secondary
Includes:	Show Main Calendar Display
Extends:	N/A
Cross-refs:	Requirement 5.3
Uses cases:	Used when the user wishes to show a different view of the timescope they currently have selected. Used to view either the entire calendar month, the week, or just the selected day.

Use Case Name:	Toggle Each Calendar Visibility
Actors:	User
Description:	The user can specify which of their calendars they want to have shown on the main calendar display. Both personal and class calendars can be toggled on and off. When a calendar is toggled off, it and its constituent events will automatically not be shown in the main calendar display.
Type:	Secondary
Includes:	Show Main Calendar Display

Extends:	N/A
Cross-refs:	Requirement 5.4
Uses cases:	Used when the user no longer wishes to view one or more calendars in the main calendar display of the application. Also used to when the user wishes to once again view a calendar in the main calendar display of the application.

Use Case Name:	Change Calendar Information
Actors:	User
Description:	The user can edit a calendar's attributes. For a personal calendar, the user can edit the title and the color. For a class calendar, the user can only edit the color.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirements 2.6 and 2.8.2
Uses cases:	Used when the user wants to edit what a personal calendar is named. Also used when the user wants to change the color of any calendar, affecting how the calendar and its events get shown in the main calendar display.

Use Case Name:	Export Calendar
Actors:	User
Description:	The user can export a calendar to be printed or saved to a file. What gets exported will be filtered based on which calendars, events, and dates are selected. Only those selected items will be included in the file or printed copy that gets exported.
Type:	Primary
Includes:	N/A
Extends:	N/A
Cross-refs:	Requirement 4
Uses cases:	Used when the user wishes to export some or all of their events. Can be used to transfer to another calendar application, or to print.

Use Case Name:	Save Calendar File
Actors:	N/A
Description:	The user can choose to save selected events, calendars, and dates to a file. The user will specify which filetype they want the file to be saved as.
Type:	Primary
Includes:	N/A
Extends:	Export Calendar
Cross-refs:	Requirement 4.1
Uses cases:	Used when the user wishes to save a file copy of their calendar. The file can then be used to transfer into another calendar application, or to send to another computer system.

Use Case Name:	Print Calendar
Actors:	N/A
Description:	The user can print the calendar events and dates they wish to export. This will send a request to an available printer to print out the exported document.
Type:	Primary
Includes:	N/A
Extends:	Export Calendar
Cross-refs:	Requirement 4.1
Uses cases:	Used when the user wishes to print out a physical copy of their calendar.

4.2. Class Diagram and Descriptions

The class diagram below in Figure 4 contains every class in the program. The classes are structured in a layered and partitioned architecture. The topmost layer can communicate to the manager classes in the layer immediately below it, which manage their respective objects they are in control of in the next layer beneath them at the lowermost layer. Each class is described by a box with their title, containing the attributes and variables they will possess, followed by the functions they will use, with their respective return types. Each line represents a relationship between each class. The

dashed arrows represent temporary relationships, the solid black arrows represent basic usage relationships, the solid black diamonds represent composition, and the open diamonds represent aggregation.

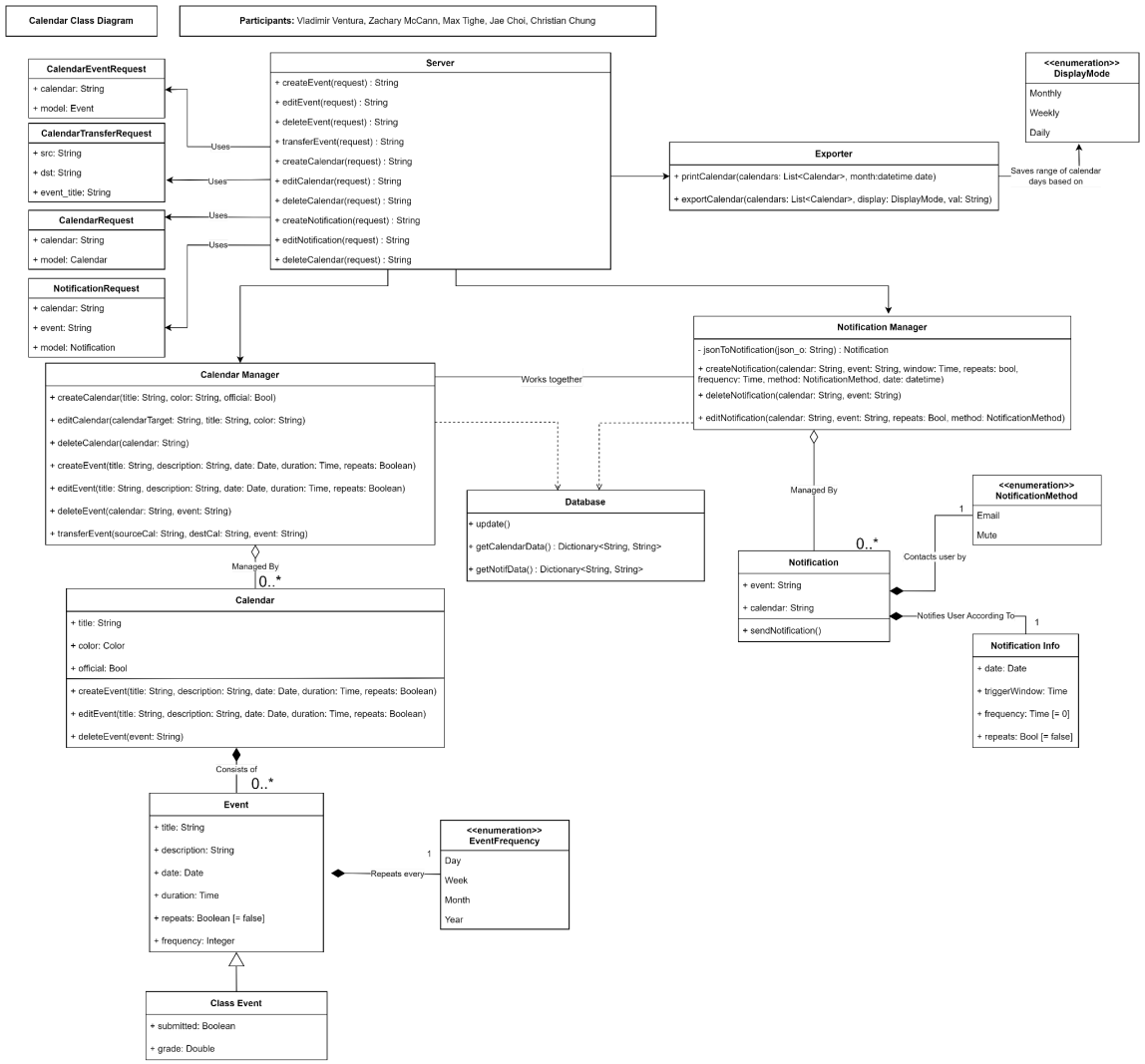

Figure 4: Class Diagram of Calendar Program

The classes contained in the class diagram are explained in further detail in the following class descriptions.

Element Name	**Description**

Server	Acts as the controller that interacts with the backend services of the program.
+ createEvent(request) : String	Sends a request to the Calendar Manager to create an event.
+ editEvent(request) : String	Sends a request to the Calendar Manager to edit an event.
+ deleteEvent(request) : String	Sends a request to the Calendar Manager to delete an event.
+ transferEvent(request) : String	Sends a request to the Calendar Manager to transfer an event between calendars.
+ createCalendar(request) : String	Sends a request to the Calendar Manager to create a calendar.
+ editCalendar(request) : String	Sends a request to the Calendar Manager to edit a calendar.
+ deleteCalendar(request) : String	Sends a request to the Calendar Manager to delete a calendar.
+ createNotification(request) : String	Sends a request to the Notification Manager to create a notification.
+ editNotification(request) : String	Sends a request to the Notification Manager to edit a notification.
+ deleteNotification(request) : String	Sends a request to the Notification Manager to delete a notification.
Relationships	The server controls both Manager classes for Calendar and Notification, as well as the Exporter class, in order to carry out the user's actions.

Element Name	**Description**
CalendarEventRequest	Acts as storage to pass data to other

	classes for event objects.
+ calendar: String	The calendar’s title.
+ model : CalendarEventModel	The container of the Event’s data.
Relationships	Holds data for the Server to read.

Element Name	**Description**
CalendarTransferRequest	Acts as storage to transfer events between calendar classes.
+ src: String	The source calendar’s title.
+ dst: String	The destination calendar’s title.
+ event_title: String	The title of the target event.
Relationships	Holds data for the Server to read.

Element Name	**Description**
CalendarRequest	This class stores the data for an instance of a calendar.
+ calendar: String	The calendar’s title.
+ model: CalendarModel	The container of the calendar’s data.
Relationships	Holds data for the Server to read.

Element Name	**Description**
NotificationRequest	This class stores the data for an instance of a notification.

+ calendar: String	The calendar's title
+ event: String	The event's title.
+ model: Notification	The container of the Notification's data.
Relationships	Holds data for the Server to read.

Element Name	**Description**
Exporter	Used for exporting calendar information, either to a file or or a printer. The interface is through the Server.
+ printCalendar(calendars: List<Calendar>)	Used to print the calendar display. Takes an iterable list of all the Calendars to be included in the printed copy.
+ exportCalendar(calendars: List<Calendar>)	Used to save calendars to a file. Takes an iterable list of all the Calendars to be included in the file.
Relationships	Communicates with the Server class to get the request to export, as well as the information of what Calendars to include.

Element Name	**Description**
Calendar Manager	This class is responsible for every calendar in the program, and contains functionality to create, edit, and delete them and their contents.
+ createCalendar(title: String, color: String, official, Bool)	Function for creating a calendar. Writes to the database upon completing the creation.
+ editCalendar(calendarTarget: String, title: String, color: String)	Function for editing an existing calendar, which is targeted by using the unique title

	as an ID. Writes any changes to the database upon completing the function.
+ deleteCalendar(calendar: String)	Function for deleting a calendar, identified through a String variable. Erases target calendar's data from database.
+ createEvent(title: String, description: String, date: Date, duration: Time, repeats: Boolean)	Function for creating an event, which in turn calls the target calendar's createEvent() function, passing the given values to it.
+ editEvent(title: String, description: String, date: Date, duration: Time, repeats: Boolean)	Function for editing an event, which calls the target calendar's editEvent() function, passing the given values to it.
+ deleteEvent(calendar: String, event: String)	Function to delete an event using the passed String variable as an ID for the target event in a calendar.
+ transferEvent(sourceCal: String, destCal: String, event: String)	Function for transferring an event from a source calendar to a target calendar.
Relationships	The Calendar Manager class manages an arbitrary number of Calendar objects, and is controlled by the Server class.

Element Name	**Description**
Calendar	Represents a calendar to be displayed on the main calendar display.
+ title: String	The name of the calendar.
+ color: String	The color to be used to display all events contained within the calendar.
+ official: Bool	A Boolean value to determine if the calendar is an official class calendar or an unofficial personal calendar.
+ createEvent(title: String, description:	Function to create an event given the

String, date: Date, duration: Time, repeats: Boolean)	information about the event.
+ editEvent(title: String, description: String, date: Date, duration: Time, repeats: Boolean)	Function to edit an event given the ID of an event and certain fields
+ deleteEvent(event: String)	Function to delete an event given the ID of an event. Will also delete the attached notification.
Relationships	Each calendar is managed by the Calendar Manager and also owns events.

Element Name	**Description**
Event	An event belongs to a calendar. An event can be anything from quiz, personal events, etc.
+title: String	The unique title to the event
+description: String	Describes the event in detail.
+date: Date	Gregorian date information for the event. Has day, month, year, and time.
+duration: Time	The duration for the event. Can be 0
+repeats: Boolean [=false]	Whether this event is a repeating one. Default is false
<<enumeration>> EventFrequency	Contains information for repeats, non-zero should repeat is enabled.
Relationships	An event object belongs to a calendar object and is controlled by the calendar object.

Element Name	**Description**
Class Event	Child of an Event - has extended member

	variables for the option of displaying the completed class event.
+ submitted: Boolean	True if the user has submitted the corresponding assignment (event) through the Blackboard application, false otherwise.
+ grade: Double	Each assignment will have a grade once the user submits it and the instructor grades it.
Relationship	Only contained by the Calendar object with official: Bool value of true.

Element Name	**Description**
Notification Manager	Responsible for every notification in the program, and can create, edit, and delete them.
+ createNotification(calendar: String, event: String, window: Time, repeats: bool, frequency: Time, method: NotificationMethod)	Function used to create a notification for a given event. Takes the event's date, as well as a time for when the notification should occur within, and the method for which the user will be notified.
+ deleteNotification(calendar: String, event: String)	Function to delete a notification for a given event in a calendar.
+ editNotification(calendar: String, event: String, repeats: Bool, method: NotificationMethod)	Function to edit a given notification associated with an event in a given calendar.
Relationships	Manages an arbitrary number of notifications, and is controlled by the Server class.

Element Name	**Description**
Notification	The Notification class is managed by the

	Notification Manager class. This class contains notification information for the event it's matched with.
+ event: String	The unique event name of the event.
+ calendar: String	The unique calendar name for the calendar.
+ sendNotification()	Called by the manager to send notification for this Notification object.
<<enumeration>> NotificationMethod	Enum class to describe what notification is to be sent out for this object.
Relationships	This object is managed by the Notification Manager and can only be accessed by the manager object. The manager can activate the Notification object when required. The Notification object also contains a Notification Info object.

Element Name	**Description**
Notification Info	Notification information for the Notification class. This includes the time when the user should be notified, how often, etc.
+ date: Date	Date information for the notification. Includes day, month, year, hour.
+ triggerWindow: Time	How long after the last notification before sending another one.
+ frequency: Time [=0]	How often it should send notification
+ repeats: Bool [=false]	Notifications have the ability to repeatedly notify the user if this variable is set to true. The time in between each notification is determined by the frequency variable.

Relationship	A Notification Info object contains the necessary time and date information to notify the user. This is a data object belonging to a Notification and is a necessary component.

Element Name	**Description**
Database	A class used for reading from and writing to an internal database for storing calendar, event, and notification information.
+ update()	Used to write calendar and event information to the internal database.
+ getCalendarData() : Dictionary<String, String>	Used to read the information of all the calendars.
+ getNotifData() : Dictionary<String, String>	Used to read the information of all the notifications.
Relationships	Communicates with the Calendar Manager and Notification Manager classes to read and write Event, Calendar, and Notification object data into and out of the internal database.

4.3. Sequence Diagrams

These sequence diagrams represent two major functions of the program. Each rectangle is a class within the program. The stick figure represents the user. Each arrow represents a message between classes, solid arrows represent synchronous messages, and dashed arrows represent asynchronous messages. The vertical dashed lines represent the classes' lifelines. The rectangles over each lifeline represents the time at which the classes are active during the sequence. A large "X" represents the end of an object's life.

Sequence 1: Creating a Personal Calendar

The program allows for the creation of personal calendars. As shown in Figure 5, the user begins by clicking on a “Create Calendar” button in the UI, which triggers the calendar display to handle the event, and a prompt will be displayed for the user to enter the title and color of the new calendar. If the user’s input is valid, and the user hasn’t canceled the calendar creation, the input will then be used by the Calendar Manager to create a new Calendar object. Once the Calendar object has successfully been created, the new Calendar object will be written to the database, before returning and refreshing the display to show the new calendar.

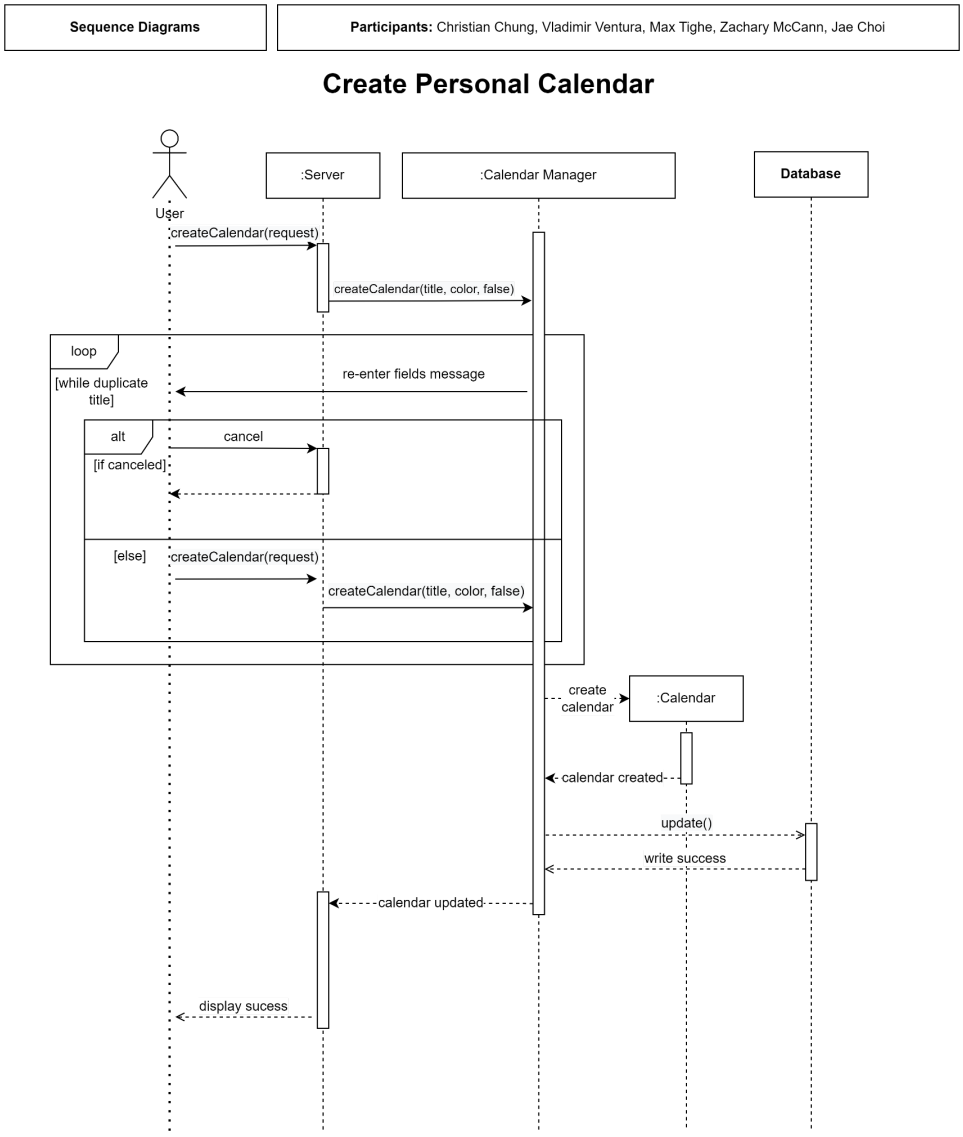

Figure 5: Personal Calendar Creation Sequence Diagram

Sequence 2: Transferring an Event

The program allows for events to be transferred between calendars. When a user wishes to transfer an event, the user selects the option to do so, which triggers the Calendar Display class to display a prompt for the user to enter the destination calendar. If the title is the same as an existing event in the destination calendar, the user is prompted to change the title, or cancel. Once the user has entered a new unique title, the event is created in the destination calendar, and the original calendar is then deleted from the source calendar. This change is written to the database immediately after each action. The transfer then returns and refreshes the display. The sequence diagram is shown below in Figure 6.

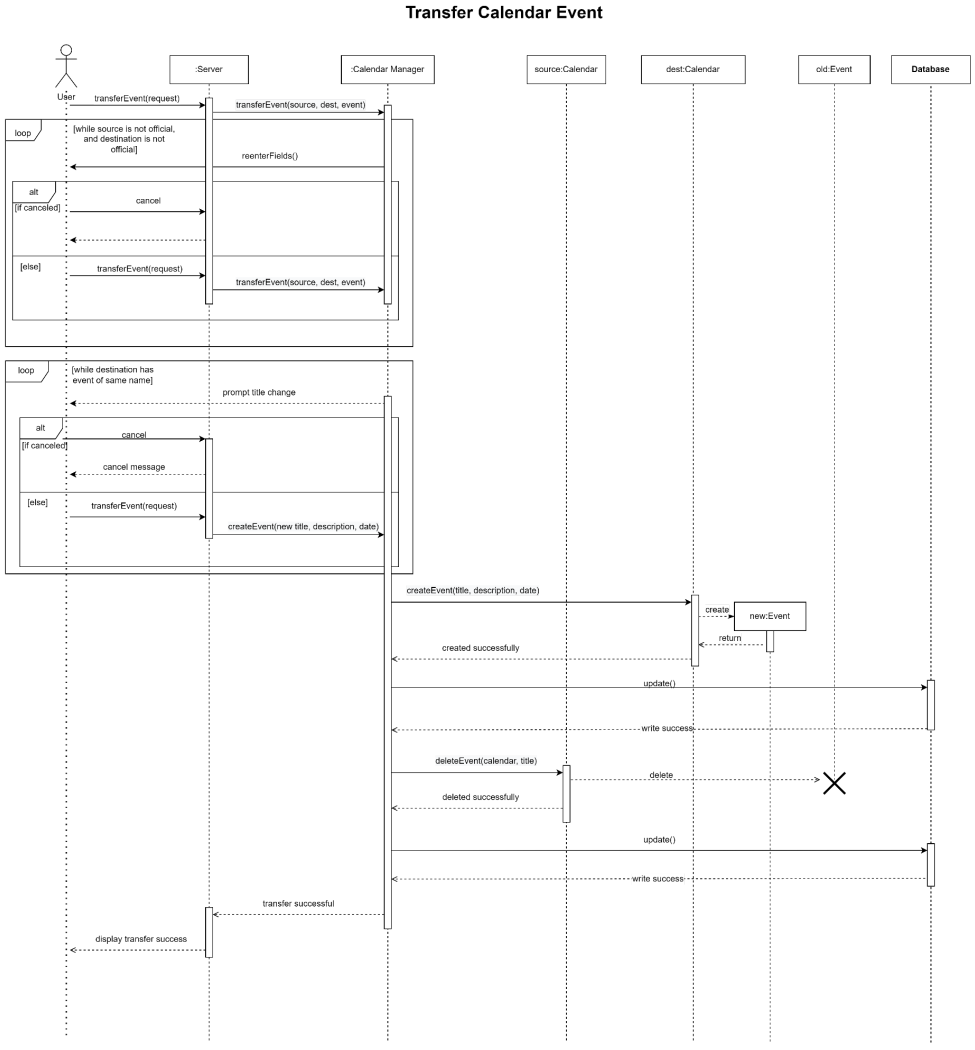

Figure 6: Transfer Calendar Event Sequence Diagram

4.4. State Diagram

The state diagram describes all of the program's states, as well as the conditions that prompt the program to transition states as it runs. The directional arrows denote the direction of the transition between states, and the labels describe the conditions under which the transitions take place. Each state may contain a "do" attribute that indicates what the program will repeatedly do while in the state, or "enter" and "exit" attributes, which indicate what the program will perform while entering or leaving a state respectively. The State Diagram is shown below in Figure 7.

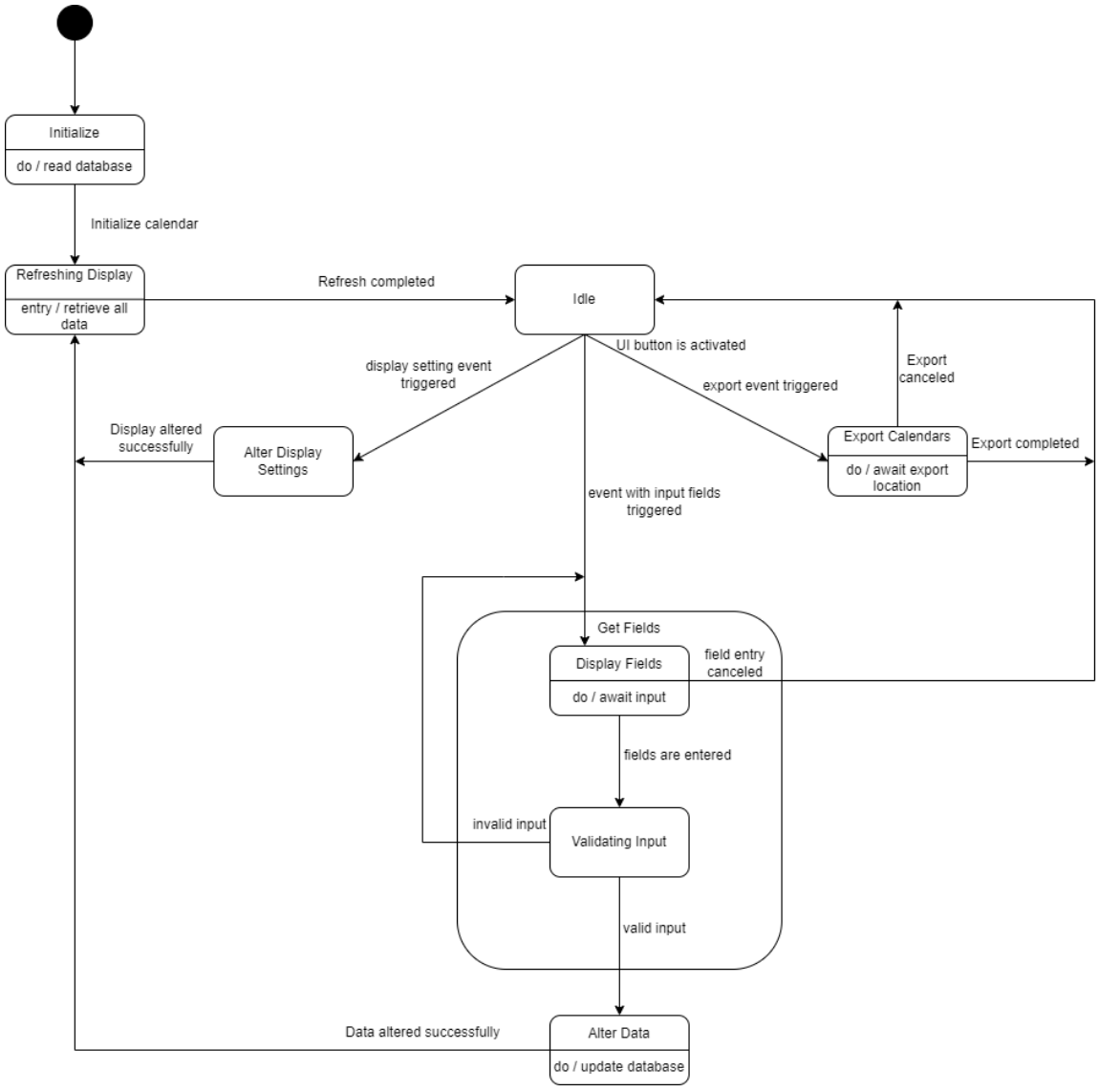

Figure 7: Calendar Program State Diagram

5. Prototype

The prototype for our project, in essence, is a page where the calendar, a sidebar, and a header bar are shown. The calendar contains several days, and each day contains a mock event. The sidebar contains stubs for creating a calendar, a small calendar preview by day, a list of the currently displayed calendars, and action buttons to create events, notifications, and export the current calendar. Lastly, the header bar contains the UML logo, a stub "calendar mode" dropdown menu, and a user avatar. The functionality of these stubs is, for the time being, displaying a modal menu where there is a small message of what should be there, or an alert that something should happen at that point; a good example of these two behaviors are when clicking the user avatar, and clicking a "Create" button.

5.1. How to Run the Prototype

The prototype is hosted on GitHub and can also be run locally. The site is hosted at https://vladventura.github.io/software-engineering-1-project/. These are the tools needed to run the prototype locally:

- NodeJS (version 18 recommended)
- Python 3
- Git (to clone the repo)
- PyPDF2

First, the project Github repo must be cloned from this URL: https://github.com/vladventura/software-engineering-1-project. Then, inside of the "frontend" directory of the repo, the command "npm install" must be executed first to install all of the dependencies for the project. When that command finishes, to finally run the frontend of the project, "npm start" must be executed to serve the webpage onto the local host. For the backend, inside of the "backend" directory of the repo, "python -r requirements.txt" must be run. As an optional step before installing the backend requirements, there's also the option of creating a virtual environment. When the installation is done, we have to move into the "src" directory, to finally run "uvicorn server:app" to run the server.

5.2. Prototype Images

First, the user is shown an entry page with their calendar and events for each day, as shown in Figure 8 below.

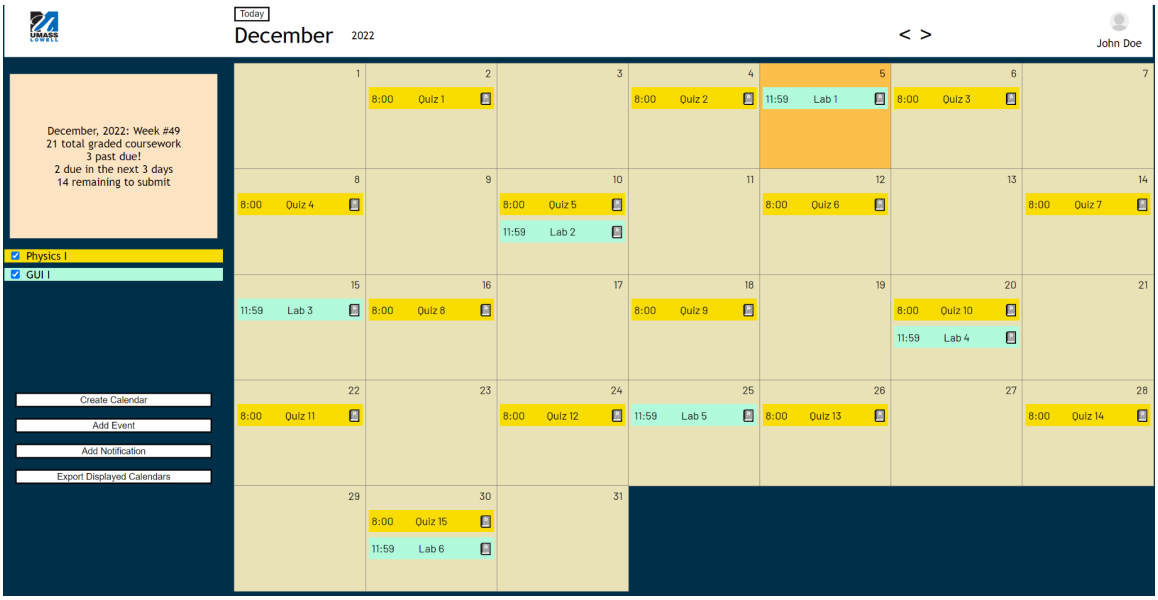

Figure 8: The entry page of the Main Calendar Display

From there, the user can hide a calendar's events like in Figure 9,

Figure 9: Hiding a calendar's events in the Main Calendar Display

Add a sample calendar, as shown in Figure 10,

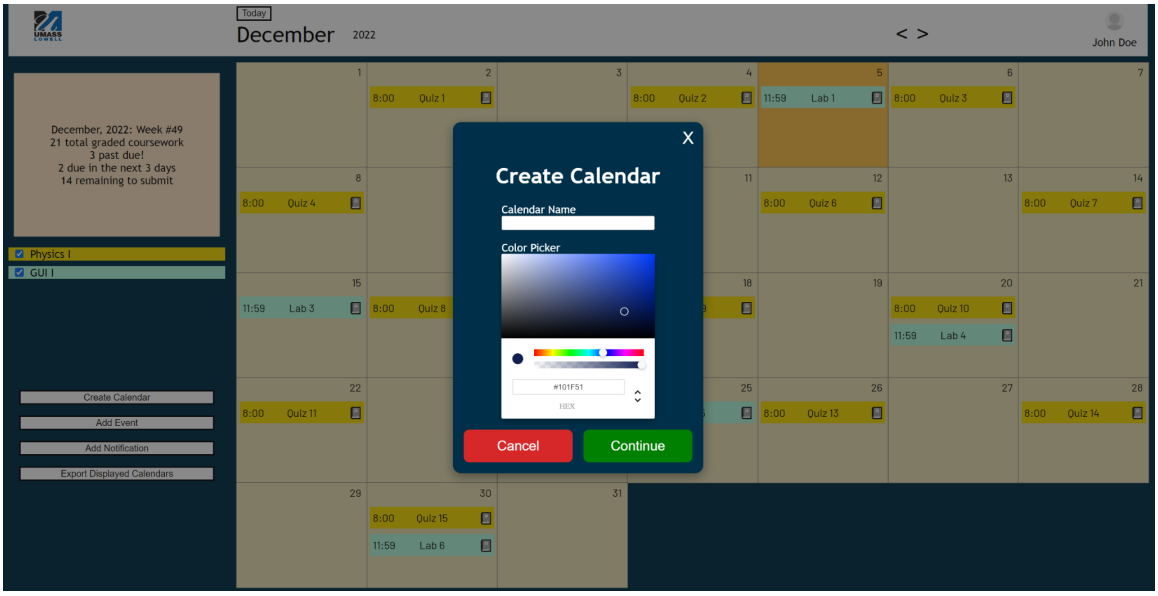

Figure 10: Creating a personal calendar

Add an event like in Figure 11,

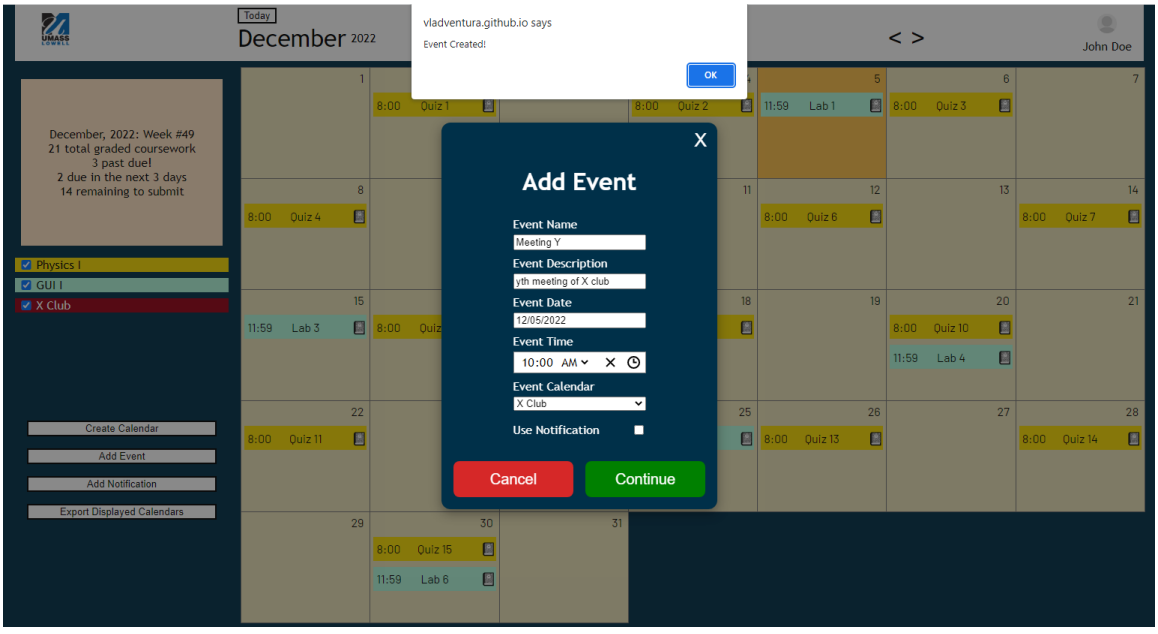

Figure 11: Adding an event to the new personal calendar

Create a notification, as shown in Figure 12,

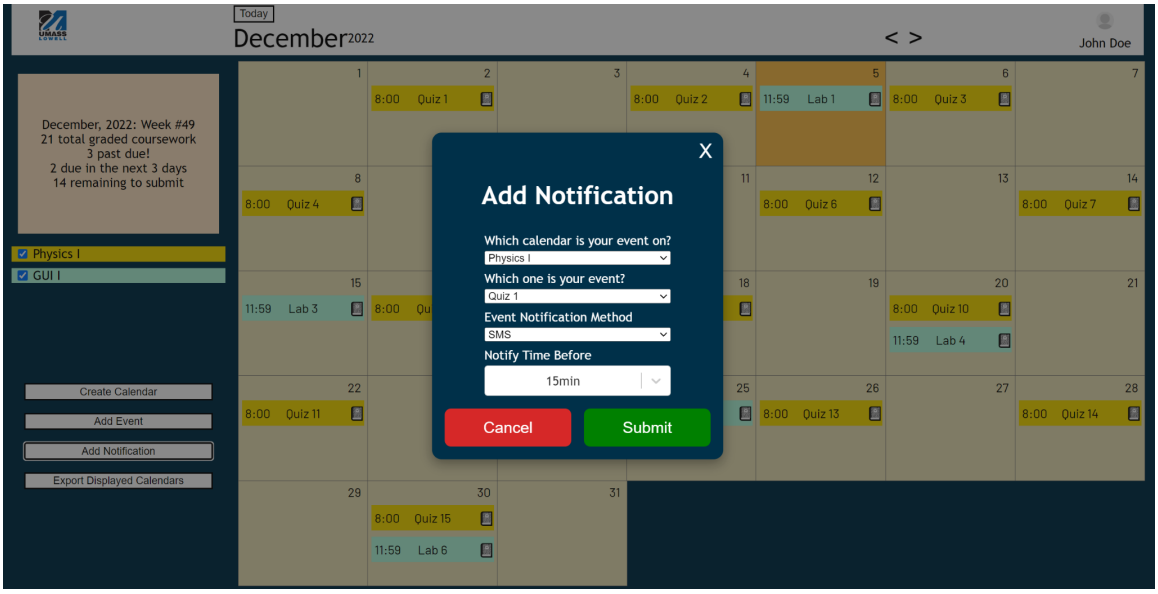

Figure 12: Adding a notification to the calendar

Edit the notification of an event in Figure 13,

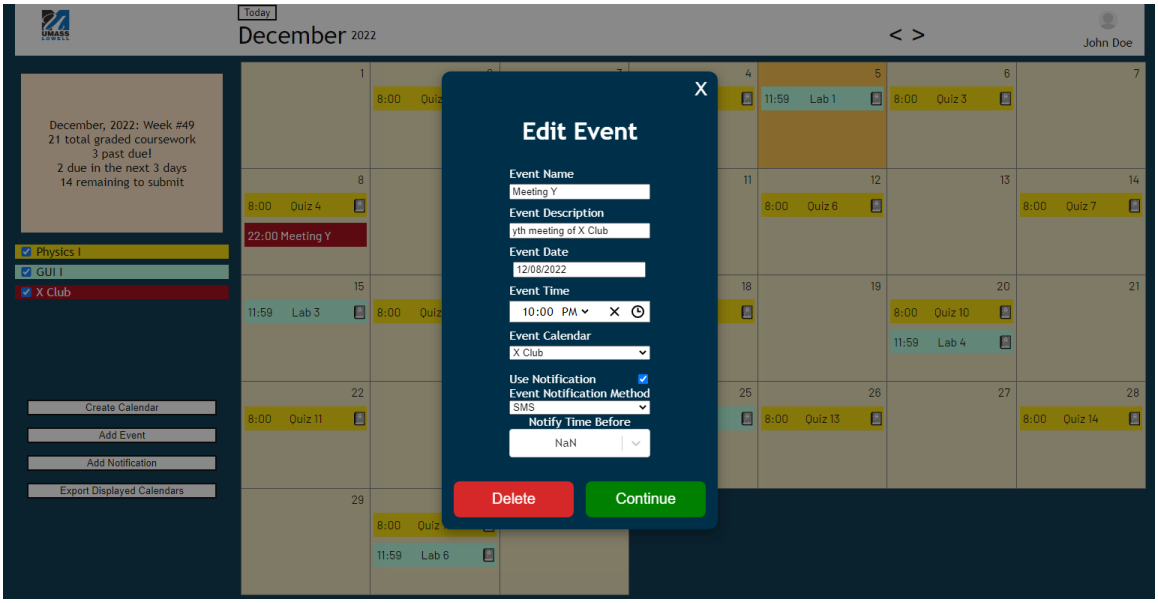

Figure 13: Editing an event with a notification

Print the calendar currently displayed like in Figure 14,

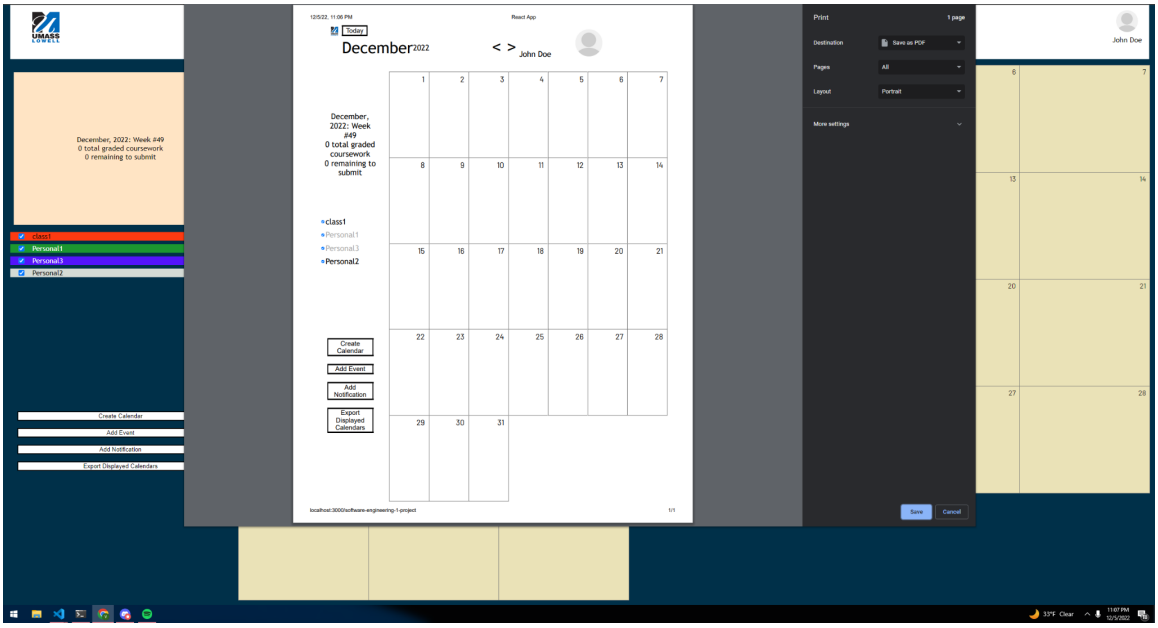

Figure 14: Printing the currently displayed calendar

6. References

[1] V. Ventura et al. Main group website for Group #6. *Enhanced Blackboard Calendar*: https://vladventura.github.io/se1-webpage/, Nov 2022

[2] Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura, "Blackboard Calendar Project Requirements" https://docs.google.com/document/d/17MUa23V_Jvdn1jz2za4Wdc0Tii_7_GZGTKUXiVb-OH4/edit?usp=sharing, Nov 2022

[3] Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura, "Use Case Diagram," https://drive.google.com/file/d/1rm8hyk4_v1KgvE3hGX1ROHxewDMP1P7p/view?usp=sharing, Nov 2022

[4] Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura, "Calendar Class Diagram," https://drive.google.com/file/d/1c7t0ybA6Jt-fW3Xy8A1iX-Fg9iphpVwe/view?usp=sharing, Nov 2022

[5] Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura, "Sequence Diagrams," https://drive.google.com/file/d/1hj-IUlgqp4Q0y3dDqDxOX6JnPsEZ7saQ/view?usp=sharing, Nov 2022

[6] Jae Choi, Christian Chung, Zachary McCann, Max Tighe, Vladimir Ventura, "State Diagram," https://drive.google.com/file/d/1QwtRFuVadErb396etX9GnLCwvuUFaRkX/view?usp=sharing, Nov 2022

7. Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.