Chapter 1: Product Scope and Purpose

for

FT Capstone Project

Prepared by Michael S. Heinzman

Florida Institute of Technology

1. Purpose

The aim of this product is to create a relationship between a student's knowledge of time and their actual ability to achieve tasks in a certain time. It is to produce an awareness inside the student's mind that he/she may not manage time as well as they thought. This product will help them realize the amount of time they think they need to achieve a goal can be wildly misguided by their own minds' lack of time management skills. This product's purpose is to make aware of the difference between the student's actual time taken and estimated time taken to achieve an event in their life. Why is this product needed? It has been shown that students with a more accurate perception of time will have a higher correlation of achieving better grades and performing better in tasks given to them [3.1]. Having better time management skills will also provide the student with lower stress levels, which in turn can also produce higher functioning results in everyday life [3.2]. Cognitive reactions have shown that a positive association to time management strategies indicates an improvement in problem solving ability, which can come in handy in everyday life outside of school and inside school [3.1]. While this product isn't designed to master time management, it is a small tool that can help bring awareness to a student that they may have misconceptions about their abilities, which can lead them to working on their time management skills in the future. As the saying goes, knowing is the first step to solving a problem.

2. Scope

The product will be a scheduling and time management app that college students will use to test their perception of time and the actual time it takes to complete a task or event in their life. The product will allow students to submit events in their life and the time estimated to complete said event. After the event is completed, the product will also ask how long the event took. By seeing the comparison of what the student estimated and the actual time it took to complete a task, a student can gain an understanding of their time management skills.

3. References

3.1 Misra, Ranjita & Mckean, Michelle. (2000). College students' academic stress and its relation to their anxiety, time management, and leisure satisfaction. American journal of health studies. 16. 41-51.

3.2 Macan, T. H., Shahani, C., Dipboye, R. L., & Phillips, A. P. (1990). College students' time management: Correlations with academic performance and stress. *Journal of Educational Psychology, 82*(4), 760–768.

3.3 Kaushar, Mehnaz. "Study of Impact of Time Management on Academic Performance of College Students: Semantic Scholar." *Undefined*, 1 Jan. 1970, https://www.semanticscholar.org/paper/Study-of-Impact-of-Time-Management-on-Academic-of-Kaushar/7303267fe8557e57212beff3ef2ed5eba41415bc#citing-papers.

Chapter 2: Software Requirements Specification

for

FT Capstone Project

Version 1.0 approved

Prepared by Michael S. Heinzman

Florida Institute of Technology

10/13/2022

Table of Contents

1. Introduction

1.1 Purpose

The aim of this product is to create a relationship between a student's knowledge of time and their actual ability to achieve tasks in a certain time. It is to produce an awareness inside the student's mind that he/she may not manage time as well as they thought. This product will help them realize the amount of time they think they need to achieve a goal can be wildly misguided by their own minds' lack of time management skills. This product's purpose is to make aware of the difference between the students actual time taken and estimated time taken to achieve an event in their life. Why is this product needed? It has been shown that students with a more accurate perception of time will have a higher correlation of achieving better grades and performing better in tasks given to them [1.4.1]. Having better time management skills will also provide the student with lower stress levels, which in turn can also produce higher functioning results in everyday life [1.4.2]. Cognitive reactions have shown that a positive association to time management strategies indicates an improvement in problem solving ability, which can come in handy in everyday life outside of school and inside school [1.4.1]. While this product isn't designed to master time management, it is a small tool that can help bring awareness to a student that they may have misconceptions about their abilities, which can lead them to working on their time management skills in the future. As the saying goes, knowing is the first step to solving a problem.

1.2 Intended Audience and Reading Suggestions

This document is intended for the Florida Tech Software Engineering department to read as evidence that I completed this part of my Capstone Project and the person who is using this document to design, develop, and test the project's system (Michael Heinzman).

1.3 Product Scope

The product will be a scheduling and time management app that college students will use to test their perception of time and the actual time it takes to complete a task or event in their life. The product will allow students to submit events in their life and the time estimated to complete said event. After the event is completed, the product will also ask how long the event actually took. By seeing the comparison of what the student estimated and the actual time it took to complete a task, a student can gain an understanding of their time management skills.

1.4 References

Misra, Ranjita & Mckean, Michelle. (2000). College students'academic stress and its relation to their anxiety, time management, and leisure satisfaction. American journal of health studies. 16. 41-51.

Macan, T. H., Shahani, C., Dipboye, R. L., & Phillips, A. P. (1990). College students' time management: Correlations with academic performance and stress. *Journal of Educational Psychology, 82*(4), 760–768.

Kaushar, Mehnaz. "Study of Impact of Time Management on Academic Performance of College Students: Semantic Scholar." *Undefined*, 1 Jan. 1970, https://www.semanticscholar.org/paper/Study-of-Impact-of-Time-Management-on-Academic-of-Kaushar/7303267fe8557e57212beff3ef2ed5eba41415bc#citing-papers.

2. Overall Description

2.1 Product Perspective

This product is a mobile application that will be existing in IOS and Android devices. This is a new and self-contained product. It has two actors, the Database / Server, and the User.

2.1.1 System Environment

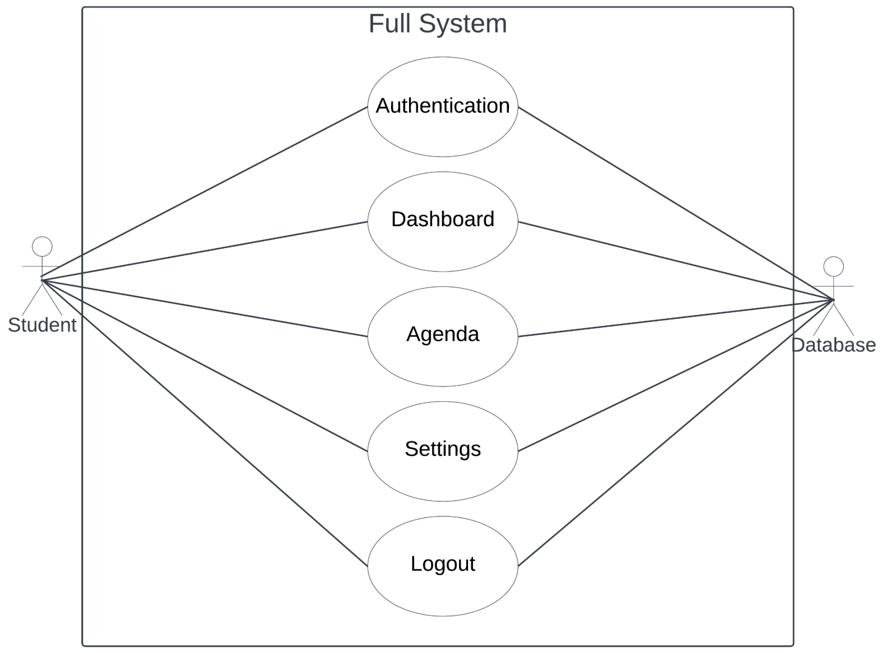
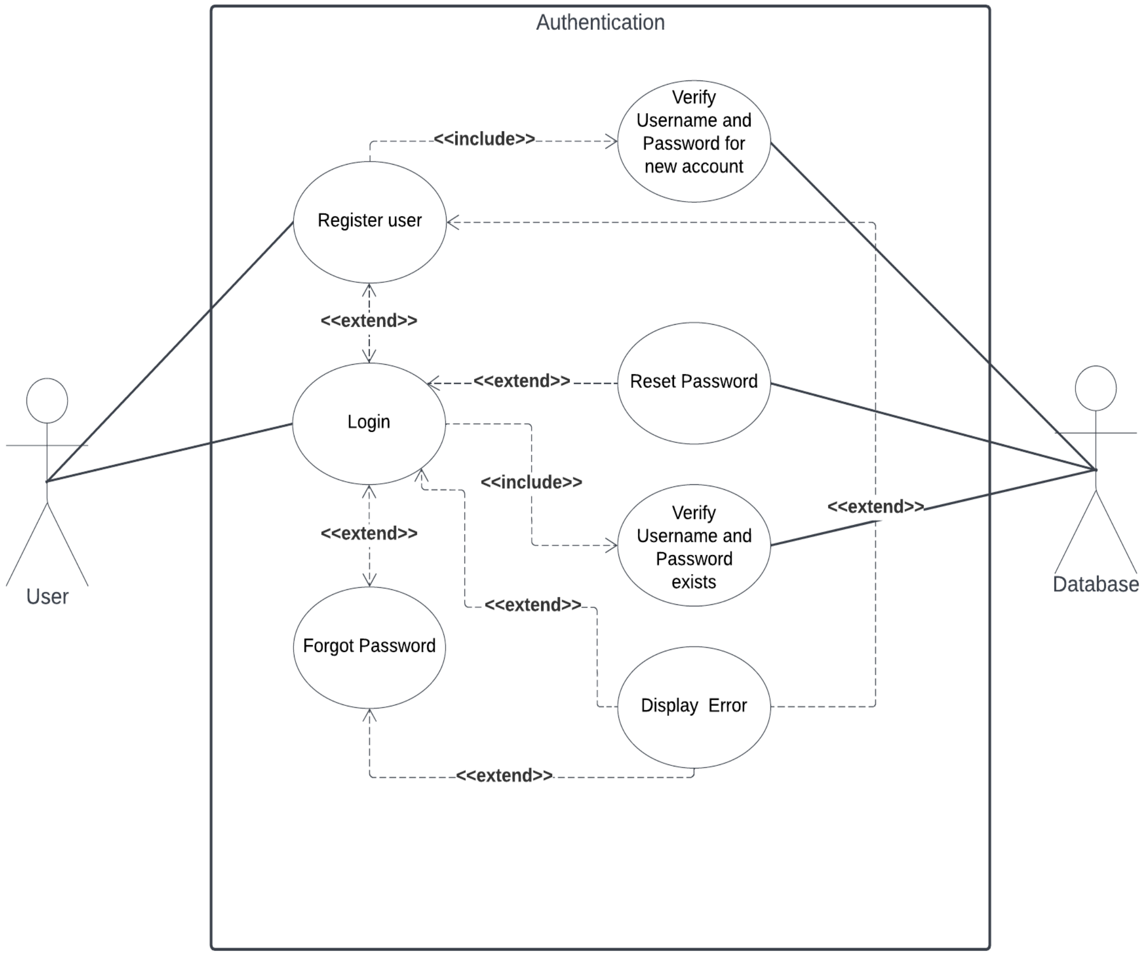


Figure 1 System Environment

2.2 Product Functions

2.2.1 Authenticate User

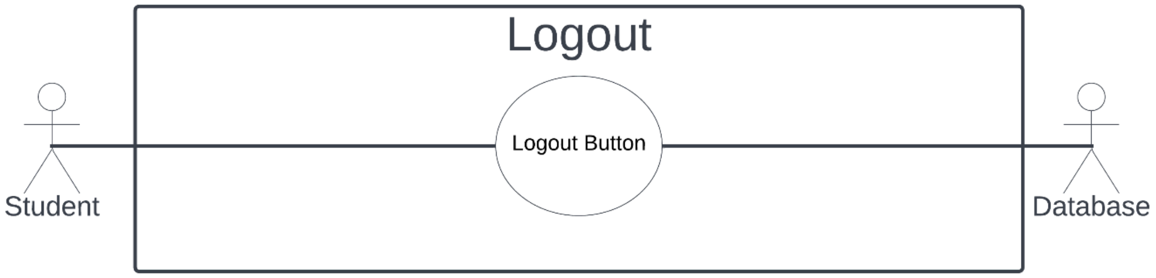
Authentication Use Case



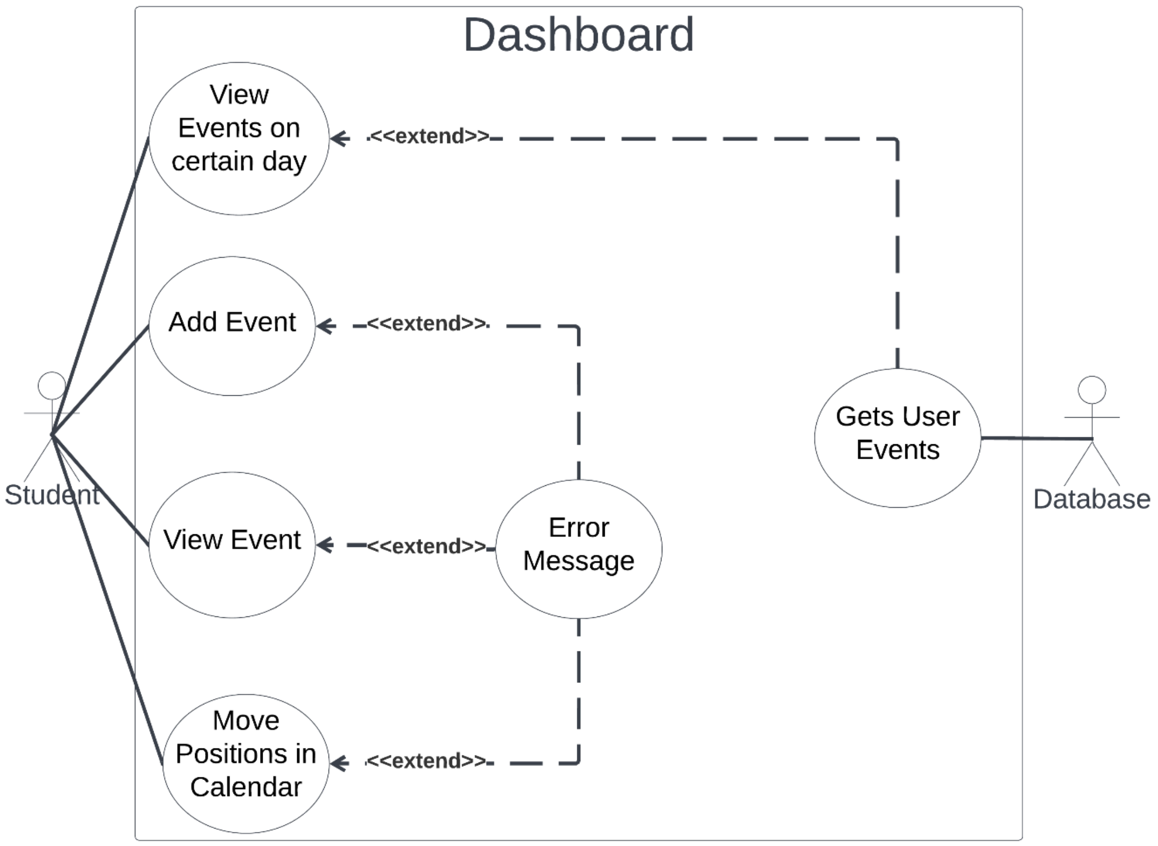
Use Case Name	Authentication
Actors	User, Database
Flow of Events	1. The user is directed to the login page. 2. The user can enter their username or email, and password. 3. The user can press the login button to login to their account. 4. If the login button is pressed then the username or email, and password will be verified by the server. 5. The user will be shown an error message if the information is incorrect. 6. The user can press the "Forgot Password?" button to reset their password. 7. The user can press the register button to create a new account.

	8. The user can switch between the forgotten password, register and login pages.
Entry Conditions	1. Users must be logged out when in the application.

2.2.2 Logout

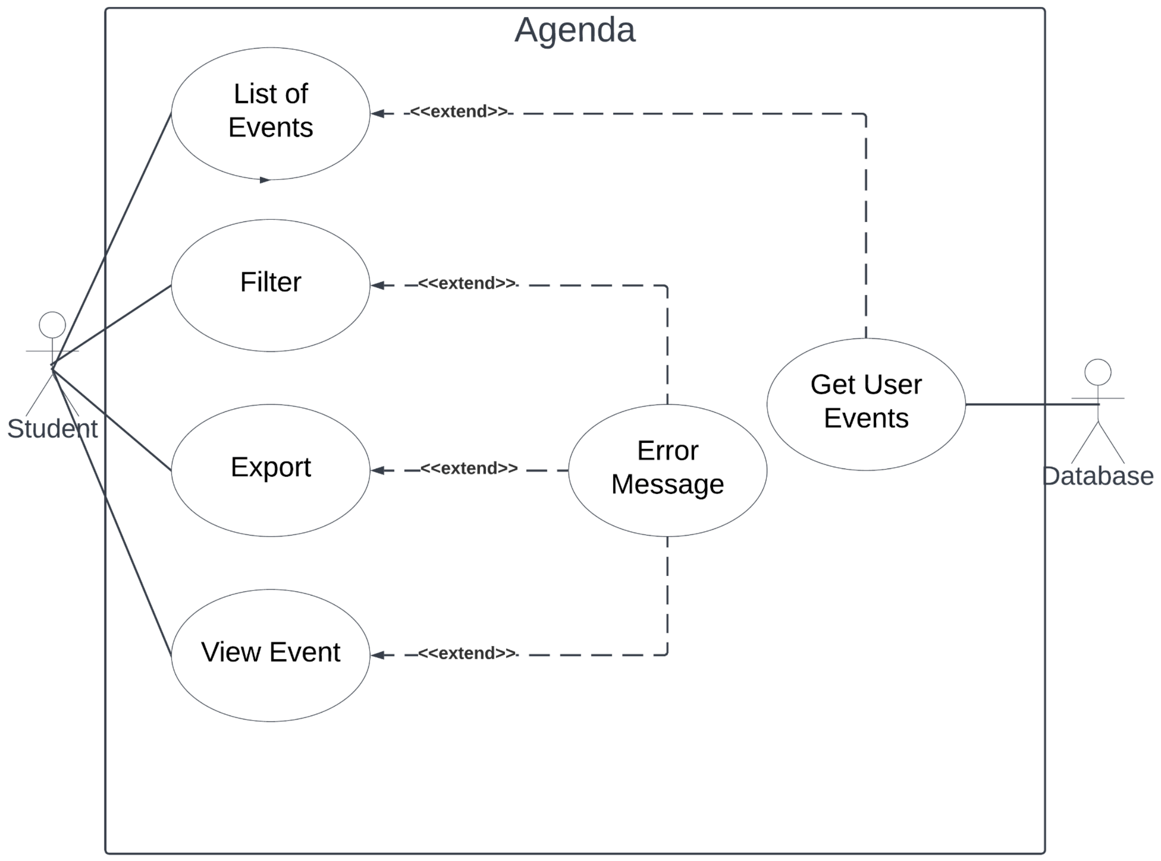
Logout Use Case	
	
Use Case Name	Logout
Actors	User, Database
Flow of Events	1. The user can logout of their account by pressing a button. 2. Once logged out, the user will be navigated back to the login page.
Entry Conditions	2. Users must be logged in.

2.2.3 Dashboard

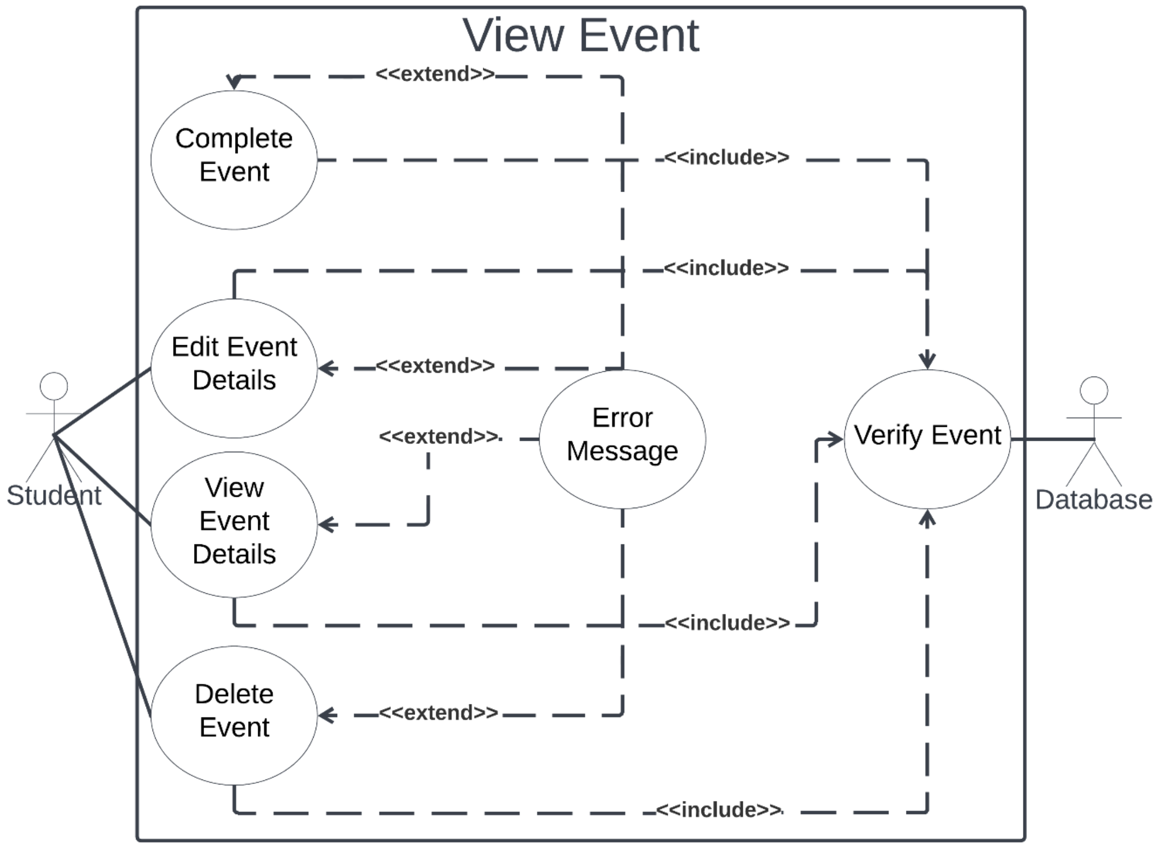


Use Case Name	Dashboard
Actors	User, Database
Flow of Events	1. User is sent to the dashboard page via login or navigation. 2. The database will receive a message to send user events when the user lands on the dashboard page. 3. Users can move positions in the calendar to view events for different days. 4. Users can choose to view an event or add a new event. 5. Users can navigate to another page. 6. The user will receive an error message if any user requests fail.
Entry Conditions	1. User is logged into the application and navigates to the dashboard.

2.2.4 Agenda

Agenda Use Case	
 Agenda List of Events Filter Export View Event Error Message Get User Events <<extend>> <<extend>> <<extend>> <<extend>> Student Database	
Use Case Name	Agenda
Actors	User, Database
Flow of Events	1. User is sent to the agenda page. 2. User chooses a filter action button. 3. User is sent to a filter selection page to filter out the events listed on the agenda page. 4. User saves filters and goes back to the agenda page with events filtered. 5. User clicks export to save a pdf of filtered or non filtered events. 6. Users can click an event to go to the event view page.
Entry Conditions	1. User is logged into the application and navigates to the Agenda page.

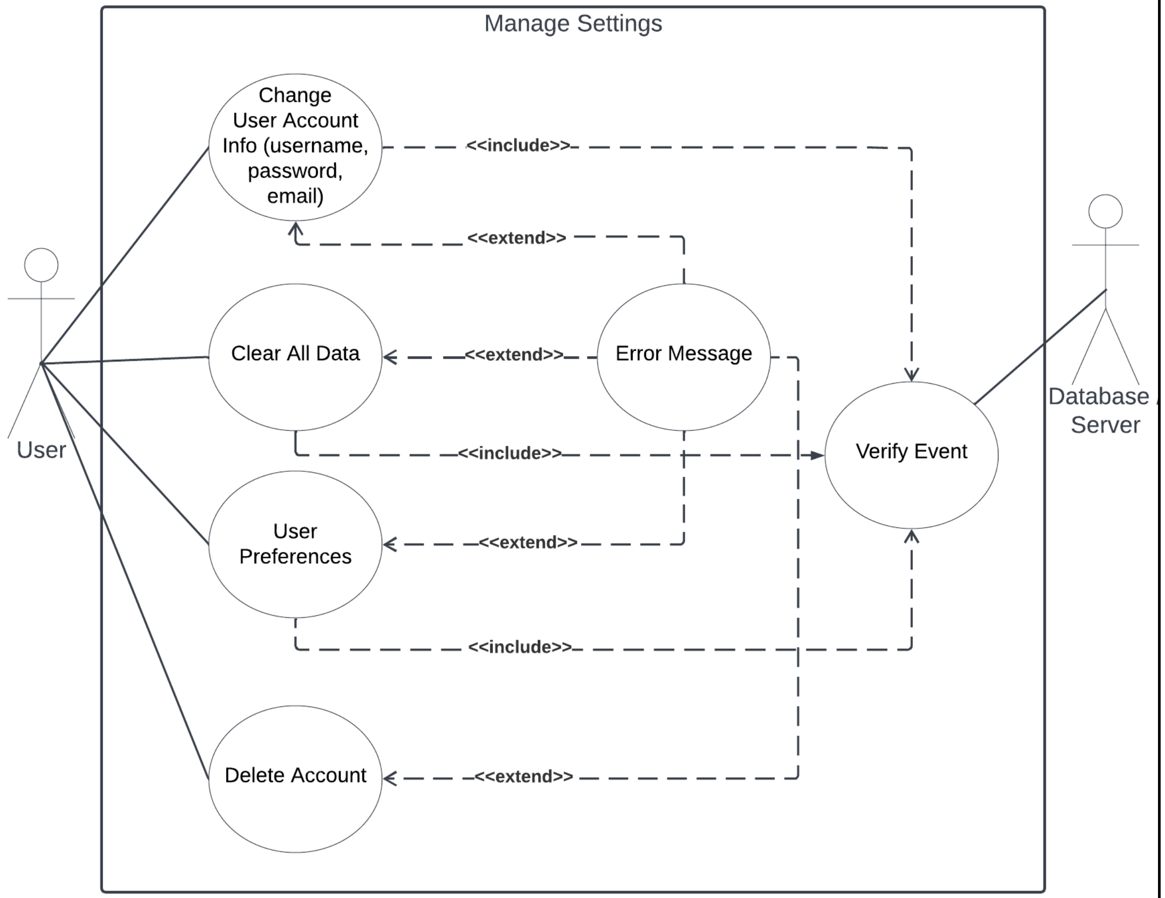
2.2.5 View Event

<table>
<tr><th colspan="2">View Event Use Case</th></tr>
<tr><td colspan="2">
</td></tr>
<tr><td><b>Use Case Name</b></td><td>View Event</td></tr>
<tr><td><b>Actors</b></td><td>User, Database</td></tr>
<tr><td><b>Flow of Events</b></td><td>1. Users can choose to complete the event.
2. If chosen, the user will enter how long it took to complete the task and select the finish action.
3. Users will be directed back to the dashboard page when an event is completed.
4. Users can also select the edit details button.
5. The user will be presented with inputs to change the current events details if edit details are pressed.
6. The user can save the event details edited and will be sent back to the View event details page.
7. The user can also choose to delete an event.</td></tr>
</table>

	8. If chosen the event will be removed from the database and the user will be sent back to the dashboard.
Entry Conditions	1. User clicks to view an event and is sent to the View Event page.

2.2.6 Manage Settings

Manage Settings Use Case



Use Case Name	Manage Settings
Actors	User, Database
Flow of Events	1. The user can edit their information on the settings page, such as their username, password, and email. 2. The user can edit their preferences for each page. 3. The user can choose to delete their account. 4. The user can clear all data.

Entry Conditions	3. Users must be logged in.

2.3 User Classes and Characteristics

The user is expected to be Internet literate and be able to read and understand how a calendar works. The main feature of this product is the editing, reading, and writing of a mobile app calendar.

The user is expected to be mobile app literate and to be able to use buttons, text fields, and similar tools.

The detailed look of these pages is discussed in section 3.2 below.

2.4 Assumptions and Dependencies

The product will use Firebase as its main database and authentication service. Firebase has some constraints when using the free version which limits the amount of data calls a developer can place. It also requires the use of Firebase components in the application which change frequently.

The product will use the React Calendar component which is updated by a third party.

3. System Features

3.1 User Authentication

3.1.1 Login

Use Case Name	Login
XRef	**Section 2.2.1, Authenticate User**
Trigger	The user enters the application or is in the application while logged out and is directed to the login page.
Precondition	The user enters the application. The user is logged out.
Basic Path	1. The login page will be the first page that the users see in the mobile application. 2. The login page should provide two text fields.

	3. The first text field should be for entering a username or email. 4. The second text field should be for entering the user's password. 5. The login page should have a submit button that can be pressed by the user. 6. The submit button when pressed should verify the users text field information. 7. If either of the text fields are left blank, it will result in an error that must be reported to the user when the submit button is pressed. 8. If both fields are filled in but there is no record of the username or email, or the password is incorrect, that must also be reported to the user. 9. Users can click the register command button to switch pages to the signup page. 10. Users can click the Forgot Password command button to switch to the reset password page.
Alternative Paths	None
Postcondition	The user is successfully logged in.
Exception Paths	The user can abandon the action at any time and stay logged out.
Other	None

3.1.2 Signup

Use Case Name	Signup
XRef	**Section 2.2.1, Authenticate User**
Trigger	The user clicks the Register command button on the login page.
Precondition	The user accesses the Signup page.
Basic Path	1. The Signup page should be displayed when the user clicks on the Signup command in the login page.

	2. The page should have four text fields. 3. The first text field should be the user's username. 4. The second text field should be the user's email. 5. The third text field should be the user's password. 6. The fourth text field should ask the user to confirm the user's password. 7. The page will have a submit button that will initiate the verification of the information in the text fields. 8. If any text field is blank, then the user will be prompted with an error message and will need to fill in the text field with information. 9. If the username or email exists in the database already that must be reported as an error. 10. If the password does not match the confirmed password or vice versa that must be reported as an error. 11. The page will have a login command button. 12. The user should be able to press the login command button to switch to the login page.
Alternative Paths	None
Postcondition	
Exception Paths	The user can abandon the action at any time and stay not registered.
Other	None

3.1.3 Reset / Forgot Password

Use Case Name	Reset / Forgot Password
XRef	**Section 2.2.1, Authenticate User**
Trigger	The user clicks the Forgot Password command button on the login page.
Precondition	The user accesses the Forgot Password page.
Basic Path	1. The page should have one text field. 2. The text field should ask for the user's email address. 3. The page will have a reset password command button. 4. If the reset password command button is pressed the user's email will be checked if it exists.

	5. If the user's email exists, then the user will be sent an email to reset the user's password. 6. If the user's email does not exist, the user will be notified with an error. 7. The page will have a back to login page command button. 8. The back to login page command button will switch pages to the login page if pressed.
Alternative Paths	None
Postcondition	The user's password is reset.
Exception Paths	The user can abandon the action at any time and their password will remain unchanged.
Other	None

3.2 Settings

3.2.1 Change User Account Information

Use Case Name	Change User Account Information
XRef	**Section 2.2.6, Manage Settings**
Trigger	The user is on the settings page and selects a username, password, or email to change.
Precondition	The user is logged in and on the application. The user is on the Settings page. The user clicks the change username, password, or email buttons.
Basic Path	1. A user will be presented with three text fields showing the username, password, and email. 2. The text fields should be in the order of Username, Email, Password. 3. The user should be able to change the information in all three text fields. 4. The user must be asked to confirm the change by typing in their password before submitting the information.

Alternative Paths	1. The user decided not to change their user information.
Postcondition	1. The user's information was changed.
Exception Paths	1. The server fails to change the user's information. 2. The password the user entered was not correct. 3. User is shown an error message.
Other	None

3.2.2 Clear All Data

Use Case Name	Change User Account Information
XRef	**Section 2.2.6, Manage Settings**
Trigger	The user is on the settings page and presses the clear all data buttons.
Precondition	The user is logged in and on the application. The user is on the Settings page. The user clicks the clear all data buttons.
Basic Path	1. A user should be able to press a button labeled "Clear All Data". 2. The user should be asked to confirm the action by entering their password. 3. The user should be able to cancel the request by hitting a cancel button at any point after clicking the "Clear All Data" button.
Alternative Paths	1. The user decided not to clear their data.
Postcondition	1. The user's information was cleared or deleted.
Exception Paths	1. The server fails to delete the user's data. 2. The user receives an error message.
Other	None

3.2.3 Delete Account

Use Case Name	Delete Account

XRef	**Section 2.2.6, Manage Settings**
Trigger	The user clicks the delete account button.
Precondition	The user is logged in and on the application. The user is on the Settings page. The user clicks the delete account button.
Basic Path	1. A user should be asked to confirm the deletion of their account by entering their password. 2. A user should be able to cancel the deletion of their account by hitting a cancel button during confirmation. 3. A user should be able to hit a submit button after entering their password. 4. Once the submit button is pressed, the user's password will be verified. 5. If valid, the user's account will be deleted. 6. If a user's account is deleted, the user should be sent to the login page. 7. If not valid, the user should be given an error message. 8. A user should be able to retry the deletion of their account as many times as they want.
Alternative Paths	1. The user chooses not to delete their account.
Postcondition	1. The user's account is deleted.
Exception Paths	1. The server fails to delete the user's account. 2. The user is sent an error message. 3. The user enters the wrong password. 4. The user is asked to re enter their password.
Other	None

3.3 Logout

Use Case Name	Logout
XRef	**Section 2.2.2, Logout**
Trigger	The user clicks the logout command button.
Precondition	The user is logged in and on the application.

Basic Path	1. The authentication service receives a logout request. 2. The user is logged out of their current account. 3. The user switches pages to the login page.
Alternative Paths	None
Postcondition	1. The user is logged out.
Exception Paths	1. Error with the authentication service and the login request gets denied. 2. User stays logged in and tries again.
Other	None

3.4 Dashboard

3.4.1 Change Dates in Dashboard

Use Case Name	Dashboard
XRef	**Section 2.2.3, Dashboard**
Trigger	The user is on the dashboard page.
Precondition	The user is logged in and on the application. The user is on the dashboard page.
Basic Path	1. A user will be shown a list of dates at the top of the page. 2. A user will be able to select a date from the list. 3. The dates of the current week will be shown. 4. The user will be able to navigate between the weeks in a year by clicking a forward and back button. 5. If the back button is clicked the previous week of the current week will be shown. 6. If the user clicks the forward button the next week of the current week will be shown. 7. Once a date is selected, the date will be changed to that date. 8. The user will see the events of that current date on the dashboard page.
Alternative Paths	1. The user decided not to change the calendar view.
Postcondition	1. Date is changed.

Exception Paths	None
Other	None

3.4.2 Event in Dashboard

Use Case Name	Dashboard
XRef	**Section 2.2.3, Dashboard**
Trigger	The user is on the dashboard page.
Precondition	1. The user is logged in and on the application. 2. The user is on the dashboard page.
Basic Path	1. A user will be shown a list of events in order of time for the selected date. 2. An events title must be shown to the user on the dashboard page. 3. When an event is clicked, the user must be directed to the event view page.
Alternative Paths	The user decided not to change the calendar view.
Postcondition	1. Event is clicked and the user is navigated to the event view page. 2. Event isn't clicked and the user stays on the dashboard and is shown the events for that day.
Exception Paths	1. Events fail to load from the database. 2. If events fail to load, the user sees no events on the dashboard and a list of times for the day remains.
Other	None

3.5 Event View

3.5.1 Edit Event in Event View

Use Case Name	Edit Event in Event View
XRef	**Section 2.2.5, View Event**
Trigger	The user is on the calendar page and clicks an event and clicks the edit button.

Precondition	The user is logged in and on the application. The user is on the Calendar Page. The user clicks an event. Edit event button is clicked.
Basic Path	1. A user must be able to change the title of the event. 2. A user must be able to change the description of the event. 3. A user must be able to change the type of event. 4. A user must be able to change the subject of the event. 5. A user must be able to change the date and time the event will start. 6. A user must be able to change how often the event will happen. 7. A user must be able to enter the amount of time the event is estimated to take. 8. A user must be able to click a submit button to submit the information given and save the edit of the event.
Alternative Paths	1. The user should be able to cancel the editing of an event by hitting the cancel button.
Postcondition	1. An event is changed.
Exception Paths	1. Error with the creation of the event with the server and user's event was not changed. 1. Users are asked to try again or cancel the event.
Other	None

3.5.2 Edit Event in Event View

Use Case Name	Edit Event in Event View
XRef	**Section 2.2.5, View Event**
Trigger	The user is on the event view page and clicks the delete button.
Precondition	The user is logged in and on the application. The user is on the Event View Page of an Event. The Delete event button is clicked.

Basic Path	1. A user must be asked to confirm the deletion of the event. 2. If yes, the event will be deleted. 3. If not, the event will not be deleted.
Alternative Paths	1. None
Postcondition	1. An event is deleted.
Exception Paths	1. Error with the deletion of the event with the server and user's event was not deleted. 2. Users are asked to try again or cancel the event.
Other	None

3.5.3 Add Event in Event View

Use Case Name	Add Event to Calendar
XRef	**Section 2.2.5, View Event**
Trigger	The user is on the calendar page and clicks the add event button.
Precondition	The user is logged in and on the application. The user is on the Dashboard page. Add event button is clicked.
Basic Path	1. A user must be able to enter the type of event. 2. A user must be able to enter the title of the event. 3. A user must be able to enter a description of the event. 4. A user must be presented with previous subjects created by the user. 5. A user must be able to choose a previous subject. 6. A user must be able to create a new subject. 7. A user must be able to enter the subject of the event. 8. A user must be presented with a list of types based on the subject chosen. 9. A user must be able to create a new type of event. 10. A user must be able to enter the type of the event. 11. A user must be able to enter the date and time the event will start. 12. A user must be able to enter the date and time the event will end.

	13. A user must be able to enter how often the event will happen. 14. A user must be able to enter the amount of time estimated for the event to be completed. 15. A user must be able to click a submit button to submit the information given and create the event.
Alternative Paths	1. The user must be able to cancel the creation of an event by hitting the cancel button.
Postcondition	1. A new event is created.
Exception Paths	2. Error with the creation of the event with the server and user's event was not created. 3. Users are asked to try again or cancel the event.
Other	None

3.6 Agenda

3.6.1 Filter Event List

Use Case Name	Filter events in Agenda List
XRef	**Section 2.2.4, Agenda**
Trigger	The user is on the calendar page and clicks an event and clicks the edit button.
Precondition	The user is logged in and on the application. The user is on the Agenda page. User clicks the filter action button.
Basic Path	1. Users must be able to select subjects to view events with that subject. 2. Users must be able to deselect subjects selected. 3. Users will be shown a list of subjects to select. 4. Users must be able to go back to the agenda list page. 5. Users must be able to save the changes made to filter. 6. If no subjects are selected, all events will be shown.
Alternative Paths	1. Error occurs saving filter. 2. User goes back to the agenda page.

Postcondition	1. Filter is saved and only events with the certain subject are shown.
Exception Paths	1. User clicks to go to another page. 2. User exits the application.
Other	None

3.6.2 Filter Event List

Use Case Name	Filter events in Agenda List
XRef	**Section 2.2.4, Agenda**
Trigger	The user is on the calendar page and clicks an event and clicks the edit button.
Precondition	The user is logged in and on the application. The user is on the Agenda page. User clicks the export action button.
Basic Path	1. Users must be able to press an export button. 2. If the export button is clicked, the user will be given an option to save a PDF file of events. 3. The user must be able to save the PDF file wherever they would like in their phone. 4. The events on the PDF file must be in order of dates and times.
Alternative Paths	1. User presses to cancel to not save pdf export of events.
Postcondition	1. Export file is saved and the user is returned to the Agenda page.
Exception Paths	1. User clicks to go to another page. 2. User exits the application.
Other	None

3.6.3 Event List

Use Case Name	Event List in Agenda page.
XRef	**Section 2.2.4, Agenda**

Trigger	The user is on the Agenda page.
Precondition	The user is logged in and on the application. The user is on the Agenda page.
Basic Path	1. Users must be able to see events in order of date and time. 2. Users must be able to see the day events are happening. 3. Users must be able to see the title of each event. 4. Users must be able to click an event.
Alternative Paths	None
Postcondition	1. Users are able to see events in an ordered fashion and know which days they happen in.
Exception Paths	1. Events do not load from the database. 2. User is given an error.
Other	None

4. External Interface Requirements

4.1 Software Interfaces

This product will be using the following software interfaces: React Native, Firebase, Expo CLI, React Native Calendars, React Dropdown Picker, React Color Picker. React Native is an open-source UI software framework created by Meta Platforms, Inc which is partnered with Facebook. It allows for the development of IOS and Android applications on one code base. Firebase is a set of hosting services for any type of application. I will be using Firebase for the Authentication and Database parts of my application. The database service that Firebase offers is Firestore. Firebase will handle all data that will be taken by the user and store it in Firestore, unless that data is a user's email and password. The email and password of the user will be stored using Firebase's authentication services which handle all security for the login information. The rest of the software interfaces that will be used are React Native Components that will be imported into the application to provide better functionality than what I can code up.

4.1.1 Logical Structure of the Data

4.1.1.1 User Data Entity

Name	**User**		
Data Item	**Type**	**Description**	**Comment**
Email Address	Text	Internet address	
Password	Text	Password For Account	
Events	List of Event	A list of the Event Data Entity created by the user.	**XRef** Section **4.1.1.2** For Event Data Entity
Subjects	List of Subject	A list of the Subject Data Entity created by the user.	**XRef** Section **4.1.1.6** For Subject Data Entity

4.1.1.2 Event Data Entity

Name	**Event**		
Data Item	**Type**	**Description**	**Comment**
ID	Text	Id given by Firebase when the event is added. Personal Identifier.	
Title	Text	Title of Event.	
Description	Text	Explanation of what the event is.	
Subject	Text	The event subject.	Separate from the Subject Data Entity. It stores the id representing a subject but not the data in the Subject Data Entity.
Type	Text	The type of event based on the given subject.	
Recurring	Data Entity	Contains information based on when the event should recur.	**XRef** Section **4.1.1.6** For Subject Data Entity
Color	Text	Color of the event. What color should be displayed when viewing the event.	

Dates	Data Entity	Contains the start and end dates of an event.	**XRef** Section **4.1.1.3** For Dates Data Entity
Times	Data Entity	Contains the time the event was expected to take and the time it took.	**XRef** Section **4.1.1.4** For Times Data Entity
Alarm	Data Entity	Contains whether an alarm should be active and the time it should be active.	**XRef** Section **4.1.1.5** For Alarm Data Entity
Completed	Boolean	A value that describes whether an event was completed or not.	

4.1.1.3 **Dates Data Entity**

Name	**Dates**		
Data Item	**Type**	**Description**	**Comment**
Start	Text	The time and day the date starts.	
End	Text	The time and day the date ends.	The difference between the start and end dates is how long the event is expected to take.

4.1.1.4 **Times Data Entity**

Name	**Times**		
Data Item	**Type**	**Description**	**Comment**
Time Expected to Spend	Number	The time the user expects to spend on the Event.	
Time Actually Took	Number	The time the user spent on the event.	

4.1.1.5 **Alarm Data Entity**

Name	**Alarm**

Data Item	Type	Description	Comment
On	Boolean	Whether or not the user decides there should be an alarm. If false, the alarm doesn't happen.	
Time	Number	The time and date the alarm should happen.	

4.1.1.6 **Subject Data Entity**

Name	**Subject**		
Data Item	**Type**	**Description**	**Comment**
ID	Text	Personal Identifier. A unique value that is used to retrieve the subject. This will be the subject's name.	Duplicate subjects are not possible.
Name	Text	The subject's name.	The way Firestore is set up, I have two values holding the name and id.
Types	List of Type	A list of Types of events that the subject can have.	For example, a Subject "Class" can have types of events that are "Assignment", "Quiz", "Test", etc... **XRef** Section **4.1.1.7** For Type Data Entity

4.1.1.7 **Type Data Entity**

Name	**Types**		
Data Item	**Type**	**Description**	**Comment**
ID	Text	Personal Identifier. A unique value that is used to retrieve the type inside a subject. This will be the name of the type.	Duplicate types in a subject are not possible.

Average Estimated Time Took	Number	The average time estimated to take for an event to be finished of a certain type in a subject.	
Average Time Took	Number	The average time taken for an event to be finished of a certain type in a subject.	

4.2 Communications Interfaces

The only communication this application will provide is an email when the user requests an email to be sent to reset their password. The email will be handled by Firebase's authentication service which will be notified through the app when a button is pressed.

5. Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Dashboard

Use Case Name	Dashboard
XRef	**Section 2.2.3, Dashboard**
Loading Events	1. The events must be retrieved from the database and presented to the user in no more than 3 seconds. 2. The events must be placed at the correct times. 3. The events must have the correct information.
Loading this week's dates at the top of page for selection	1. This week's dates must be presented first at the top of the page for the user to select in less than 3 seconds. 2. The current date must be selected and displaying the correct events for that day. 3. The buttons to navigate between weeks must be able to go forward a week and backwards a week from the presented week. 4. The dates must only be marked when an event is in them.

Add Event Button	1. The add event button must navigate the user to the add event page when clicked. 2. The navigation to the add event page should take no longer than 3 seconds.
View Event	1. An event pressed must navigate the user to the View Event page. 2. The navigation of the user to the view event page should take no longer than 3 seconds.

5.1.2 View Event

Use Case Name	View Event
XRef	**Section 2.2.4, View Event**
Showing Event Details	1. The correct data for the event pressed on the dashboard page must be displayed to the user.
Edit Event	1. The user must be navigated to the edit event page in less than 3 seconds when the edit button is pressed.
Complete Event	1. When the complete event button is pressed, the user must be asked to enter a time the event took. 2. They must not complete until the user enters a time.
Delete Event	1. When a user clicks the delete button, the user must be asked if they truly want to delete the event. 2. The deletion of an event should take no more than 3 times to delete if an error occurs.

5.1.3 Authentication

Use Case Name	Authentication
XRef	**Section 2.2.1, Authentication**
Resetting Password	1. The user should receive the reset password email when the submit button is pressed no more than 5 minutes after the request.
Login	1. The user should be able to login with less than 3 or within 3 tries.

5.1.4 Settings

Use Case Name	Settings
XRef	**Section 2.2.5, Settings**
Clearing all Data	1. It should take no more than 10 seconds to delete all the data on the user.

Chapter 3: Design

for

FT Capstone Project

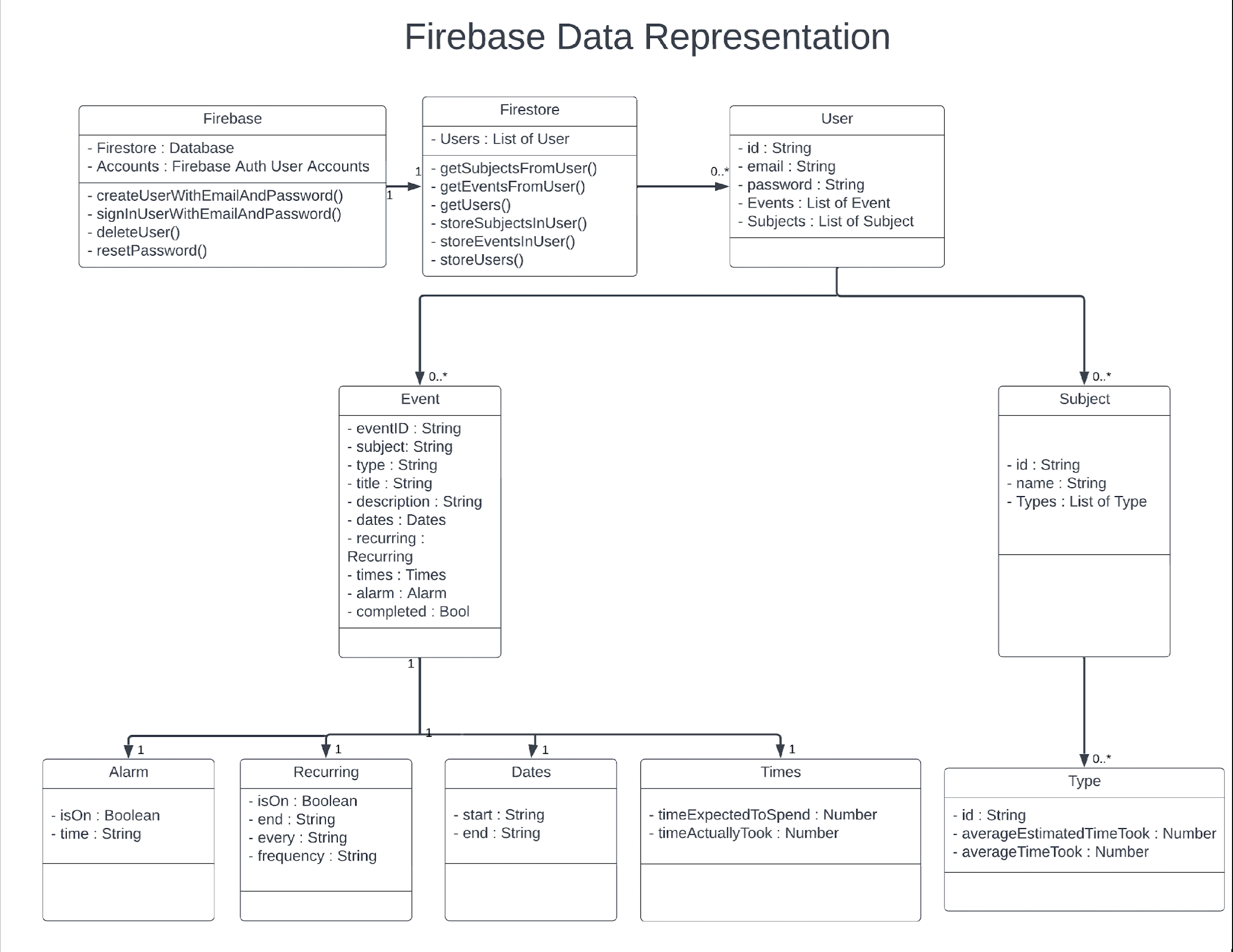
Prepared by Michael S. Heinzman

Florida Institute of Technology

1. Design

1.1 Firebase

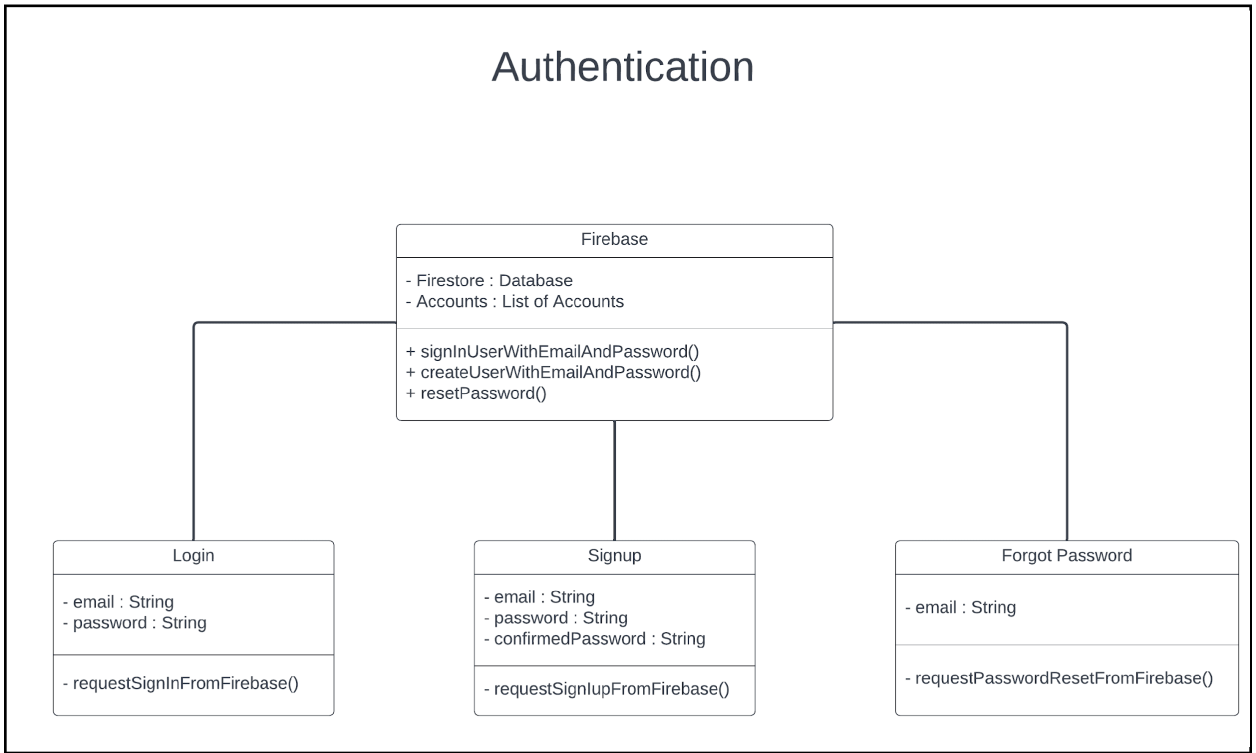
1.1.1 UML Class Diagram



Represents the firebase data architecture as a UML Class Diagram

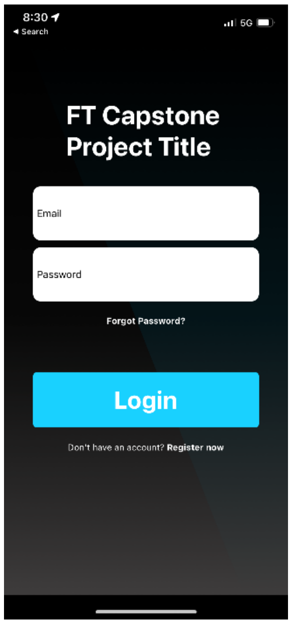
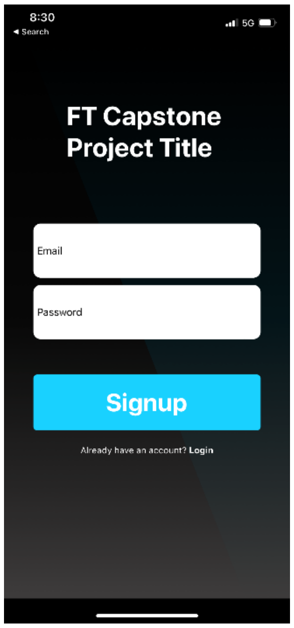
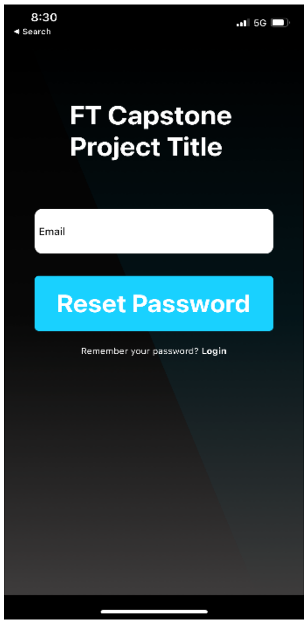
1.2 Authentication

1.2.1 UML Class Diagram


Authentication
Firebase
- Firestore : Database
- Accounts : List of Accounts
+ signInUserWithEmailAndPassword()
+ createUserWithEmailAndPassword()
+ resetPassword()
Login
- email : String
- password : String
- requestSignInFromFirebase()
Signup
- email : String
- password : String
- confirmedPassword : String
- requestSignlupFromFirebase()
Forgot Password
- email : String
- requestPasswordResetFromFirebase()

The Authentication data architecture as a UML Class Diagram. Since Firebase Architecture was included before this, I chose to admit the rest of the interactions. Assume they are implied.

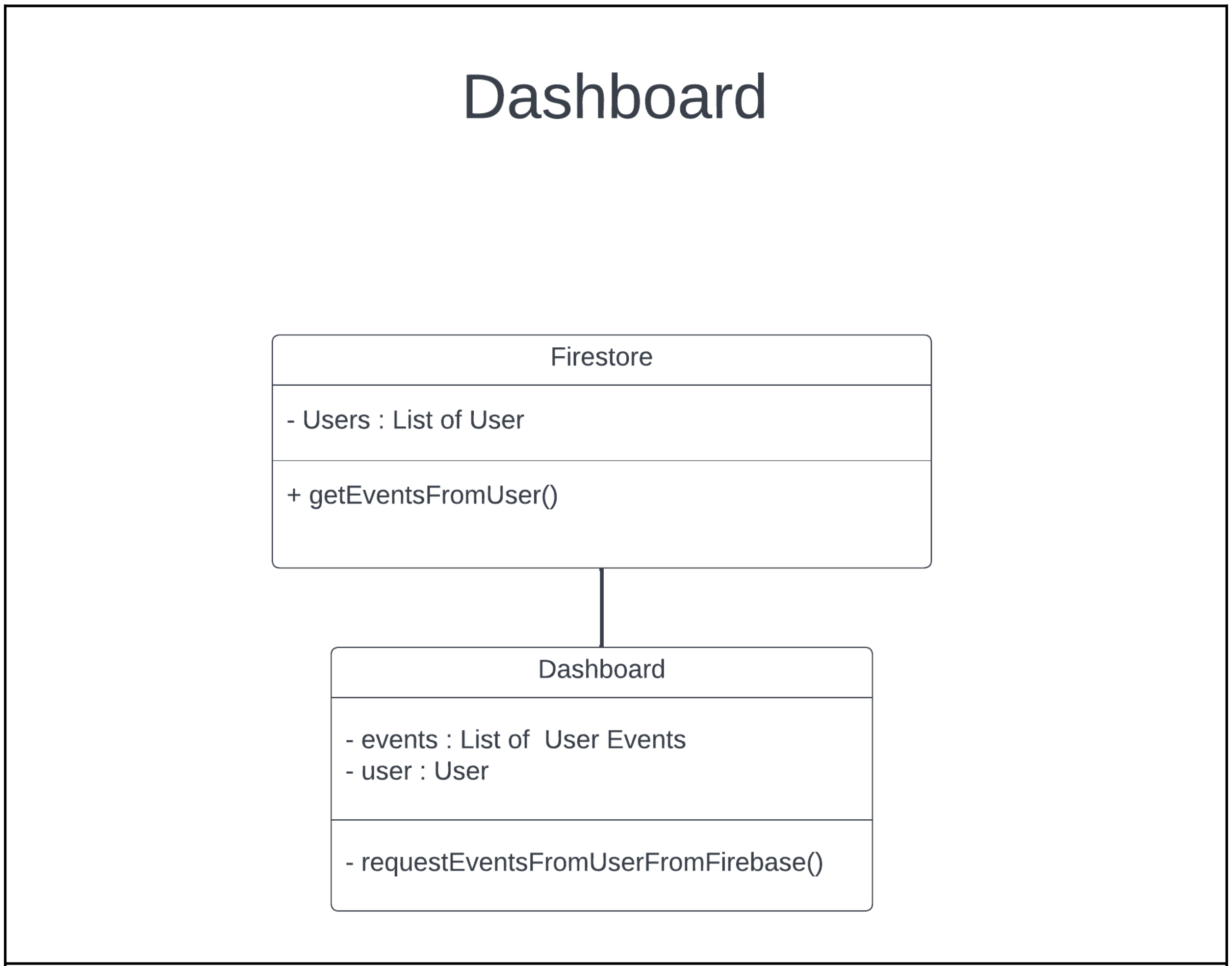
1.2.2 User Interface

Login Page	Signup Page	Forgot Password Page
 8:30 ◂ Search 5G FT Capstone Project Title Email Password Forgot Password? Login Don't have an account? Register now	 8:30 ◂ Search 5G FT Capstone Project Title Email Password Signup Already have an account? Login	 8:30 ◂ Search 5G FT Capstone Project Title Email Reset Password Remember your password? Login

The login page where the requestSignInFromFirebase () happens. The user types in their email and password on the inputs provided and when pressing login, the user is directed inside the app or receives an error message.	The Signup page where the requestSignupFromFirebase () happens. The user types in their email and password on the inputs provided and when pressing signup, the user is directed inside the app or receives an error message.	The Forgot Password page where requestPasswordFromFirebase () happens. User inputs in their email and presses the Reset Password button. If the request works, the user will be notified through email.

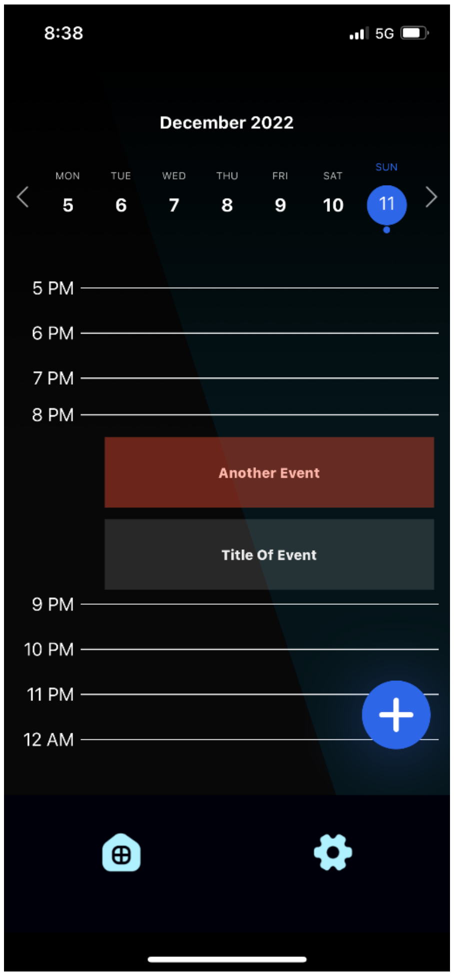
1.3 Dashboard

1.3.1 UML Class Diagram



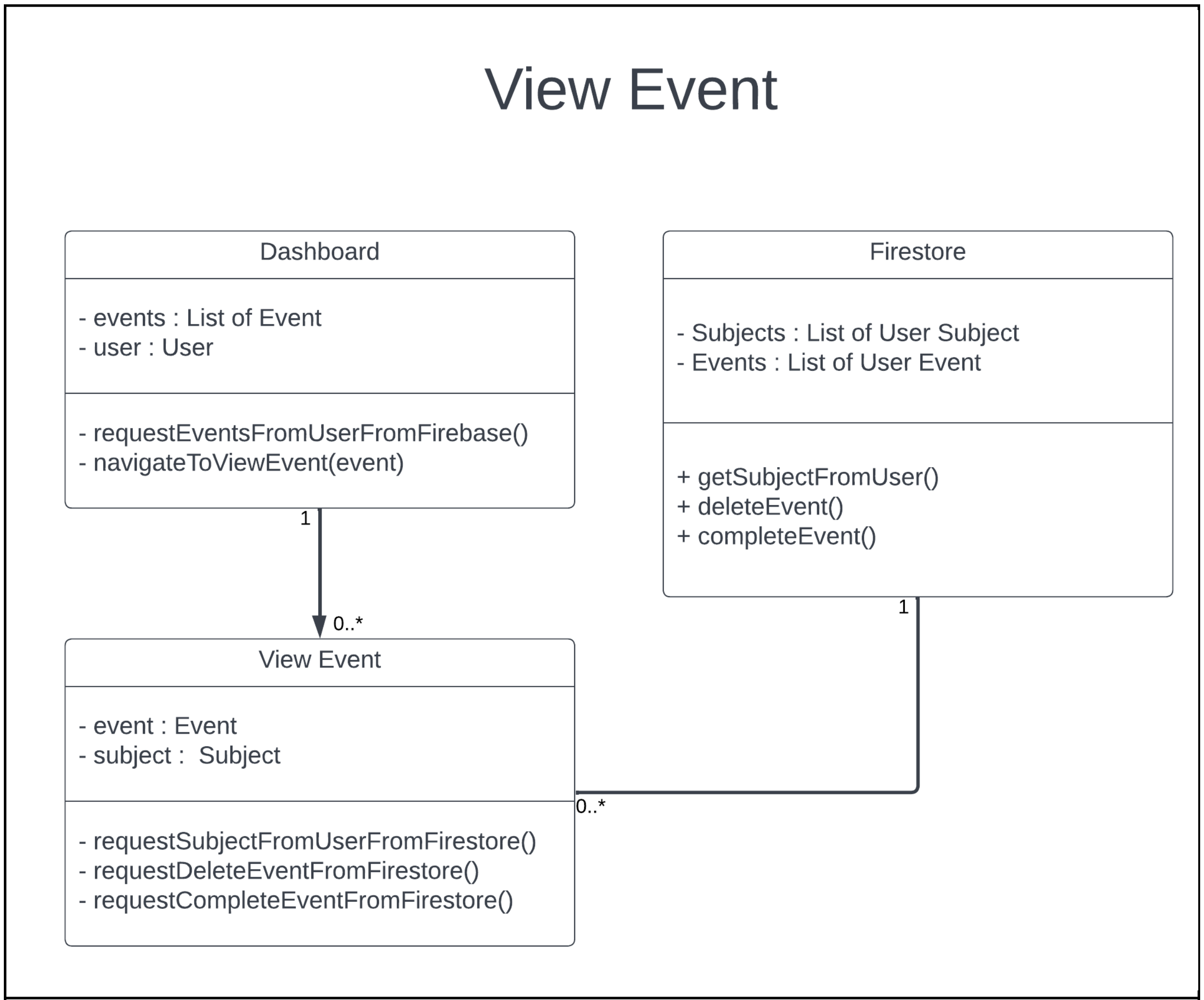
Dashboard data architecture as a UML Class Diagram. Requests events from Firestore from the user and stores them in a list to use.

1.3.2 User Interface

Dashboard User Interface
 8:38 5G December 2022 MON 5 TUE 6 WED 7 THU 8 FRI 9 SAT 10 SUN 11 5 PM 6 PM 7 PM 8 PM Another Event Title Of Event 9 PM 10 PM 11 PM 12 AM
The user interface of the dashboard page. The list of events is shown between the times they are supposed to happen with their respective titles. The dates for the week are represented at the top of the page where the user can choose to select or navigate to another week. The add event button is the plus at the bottom right of the page. This is where the requestEventsFromFirestore () happens, if no events are found then a list of times are shown.

1.4 View Event

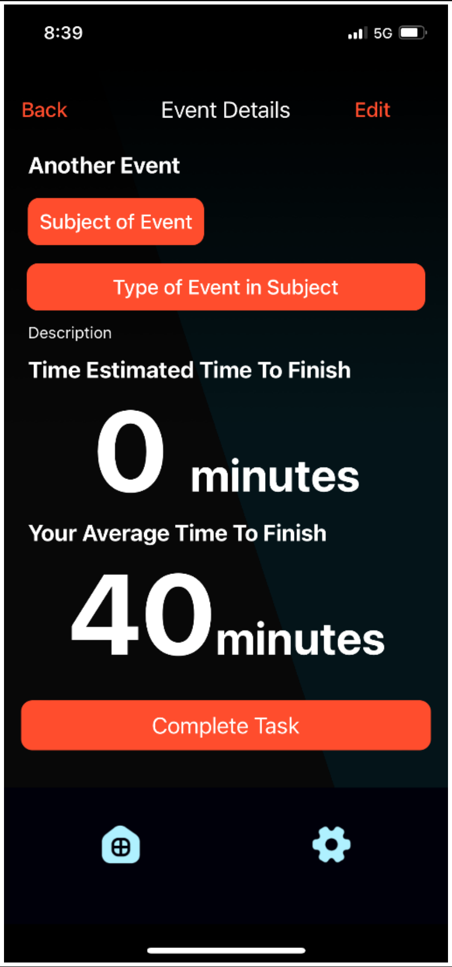
1.4.1 UML Class Diagram


View Event
Dashboard
- events : List of Event
- user : User
- requestEventsFromUserFromFirebase()
- navigateToViewEvent(event)
Firestore
- Subjects : List of User Subject
- Events : List of User Event
+ getSubjectFromUser()
+ deleteEvent()
+ completeEvent()
1
0..*
1
View Event
- event : Event
- subject : Subject
- requestSubjectFromUserFromFirestore()
- requestDeleteEventFromFirestore()
- requestCompleteEventFromFirestore()
0..*

View Event data architecture as a UML Class Diagram. Takes an event from the dashboard and gets the subject from that event to request the subject object from Firestore. Can request to delete, complete and subject from Firestore.

1.4.2 User Interface

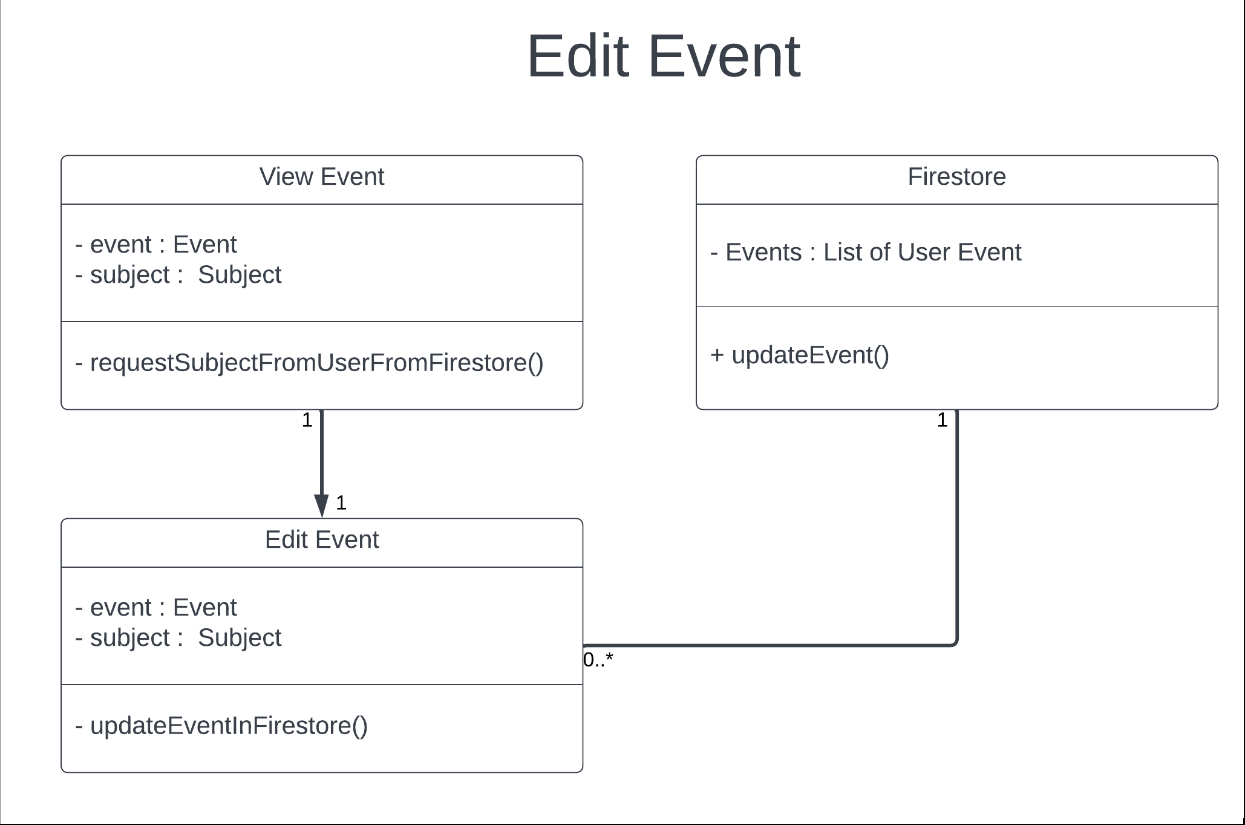
Event View User Interface



The event view user interface comes from clicking an event on the dashboard. The details of the event are sent to this page where you can see the data inside the event. The view event page uses the requestSubjectFromUserFromFirestore () method to receive the subject's data for the type of event. The average time to finish an event is shown which is the averageTimeTook variable displayed in the Firebase Data Architecture Type class in section 1.1.1. The requestCompleteEventFromFirestore () is a method that is called when a complete task is pressed, and the user inputs the actual time it took to complete the task. The deleteEventFromFirestore () is a method that is called when pressing the delete button which would be below the Complete Task button in the same format.

1.5 Edit Event

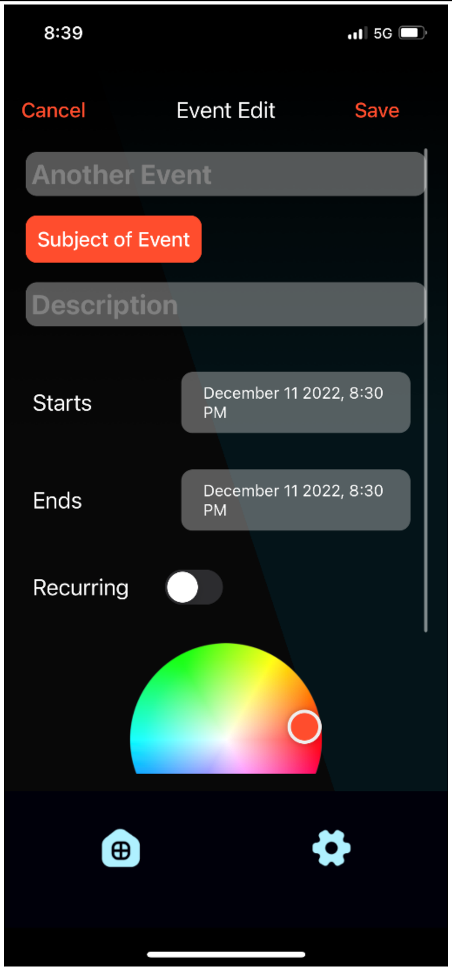
1.5.1 UML Class Diagram



The Edit Event data architecture as a UML Class Diagram. Takes the event from View Event and uses that data as reference when the user is editing. It can request to update that event in Firestore.

1.5.2 User Interface

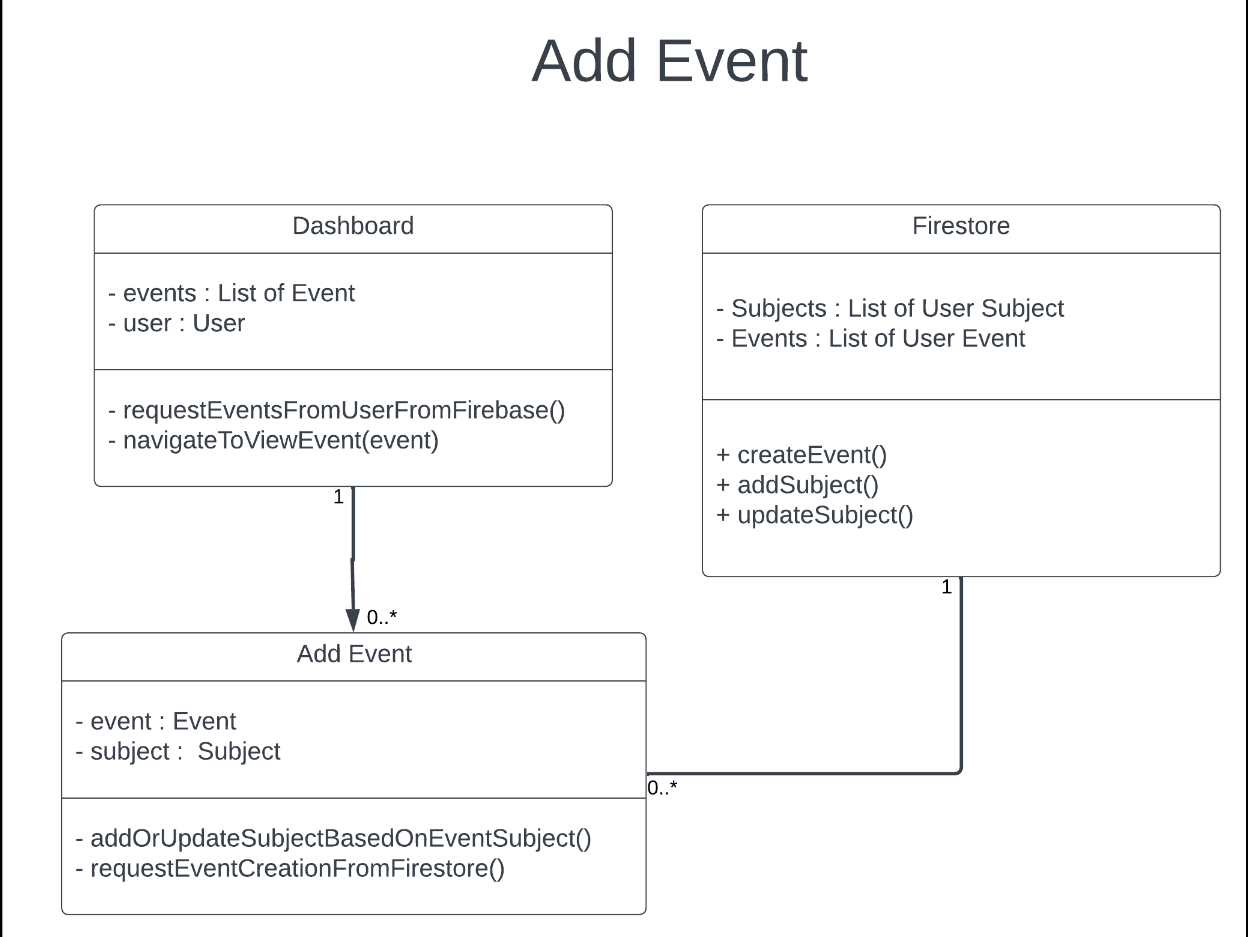
Edit Event User Interface



The Event Edit User Interface provides multiple inputs from the user so that all the information needed for the Event class in Firebase Architecture is fulfilled. When the recurring switch is pressed it will bring up an interface that provides a way to input the Recurring data for the Recurring Data Class. When the save button is pressed the updateEventInFirestore () is called and the event is updated in Firestore. If cancel is pressed, then the user is navigated back to the event view page.

1.6 Add Event

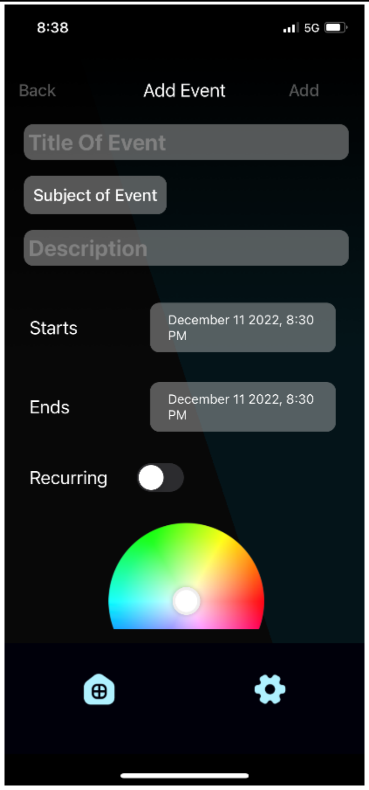
1.6.1 UML Class Diagram


Add Event
Dashboard
- events : List of Event
- user : User
- requestEventsFromUserFromFirebase()
- navigateToViewEvent(event)
Firestore
- Subjects : List of User Subject
- Events : List of User Event
+ createEvent()
+ addSubject()
+ updateSubject()
Add Event
- event : Event
- subject : Subject
- addOrUpdateSubjectBasedOnEventSubject()
- requestEventCreationFromFirestore()
1
0..*
1
0..*

Add Event data architecture as a UML Class Diagram. Does not receive an event from Dashboard. Creates an event template for the user to update. Requests creation of an event to Firestore. When an event is created, requests either the updating of a subject or the addition of a subject in Firestore.

1.6.2 User Interface

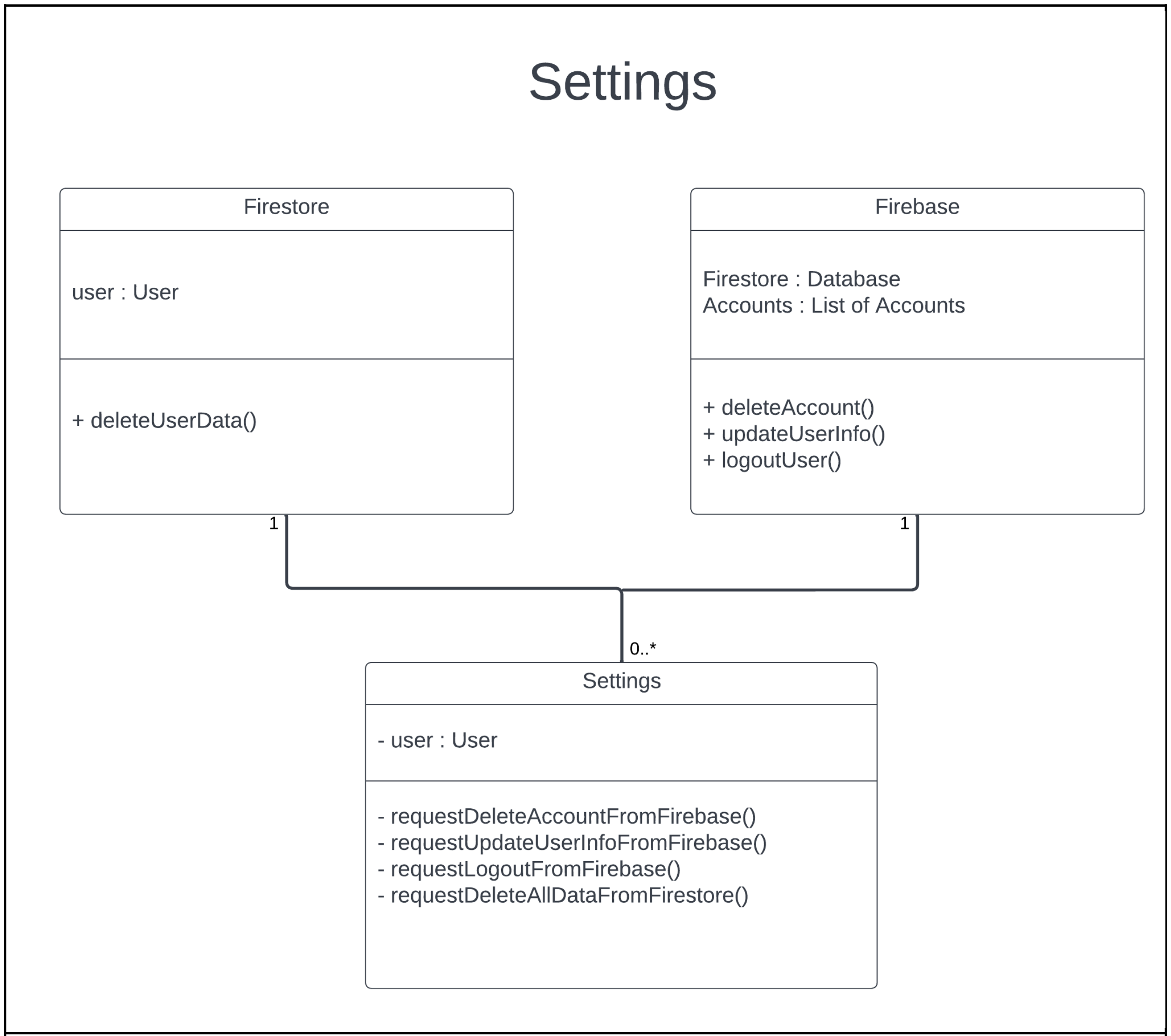
Add Event User Interface



The Add Event page is like the edit event page since both require inputting Event Data that fulfill the Event Class in Firebase Architecture UML Class Diagram. When the add button is pressed the requestEventCreationFromFirestore () method is called where Firestore adds the events data into the Events list. The addOrUpdateSubjectBasedOnEventSubject () method is called when the user completes the requestEventCreationFromFirestore () method.

1.7 Settings

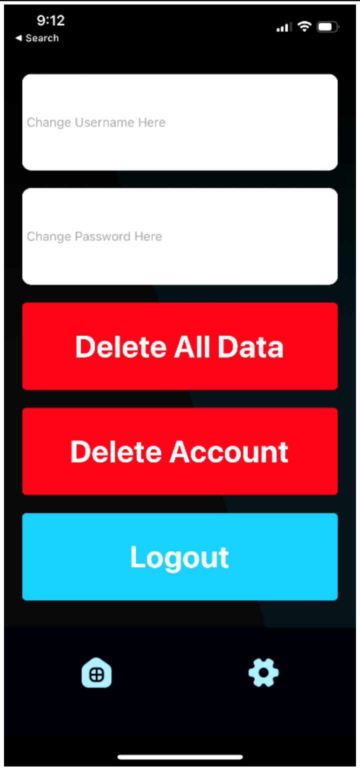
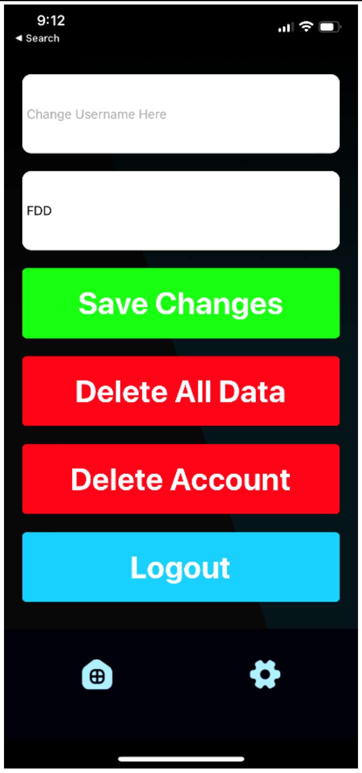
1.7.1 UML Class Diagram


Settings
Firestore
user : User
+ deleteUserData()
Firebase
Firestore : Database
Accounts : List of Accounts
+ deleteAccount()
+ updateUserInfo()
+ logoutUser()
1
1
0..*
Settings
- user : User
- requestDeleteAccountFromFirebase()
- requestUpdateUserInfoFromFirebase()
- requestLogoutFromFirebase()
- requestDeleteAllDataFromFirestore()

Settings data architecture as a UML Class Diagram. Provides a way to update user account information, logout the user, delete all data the user has produced or delete the user's account.

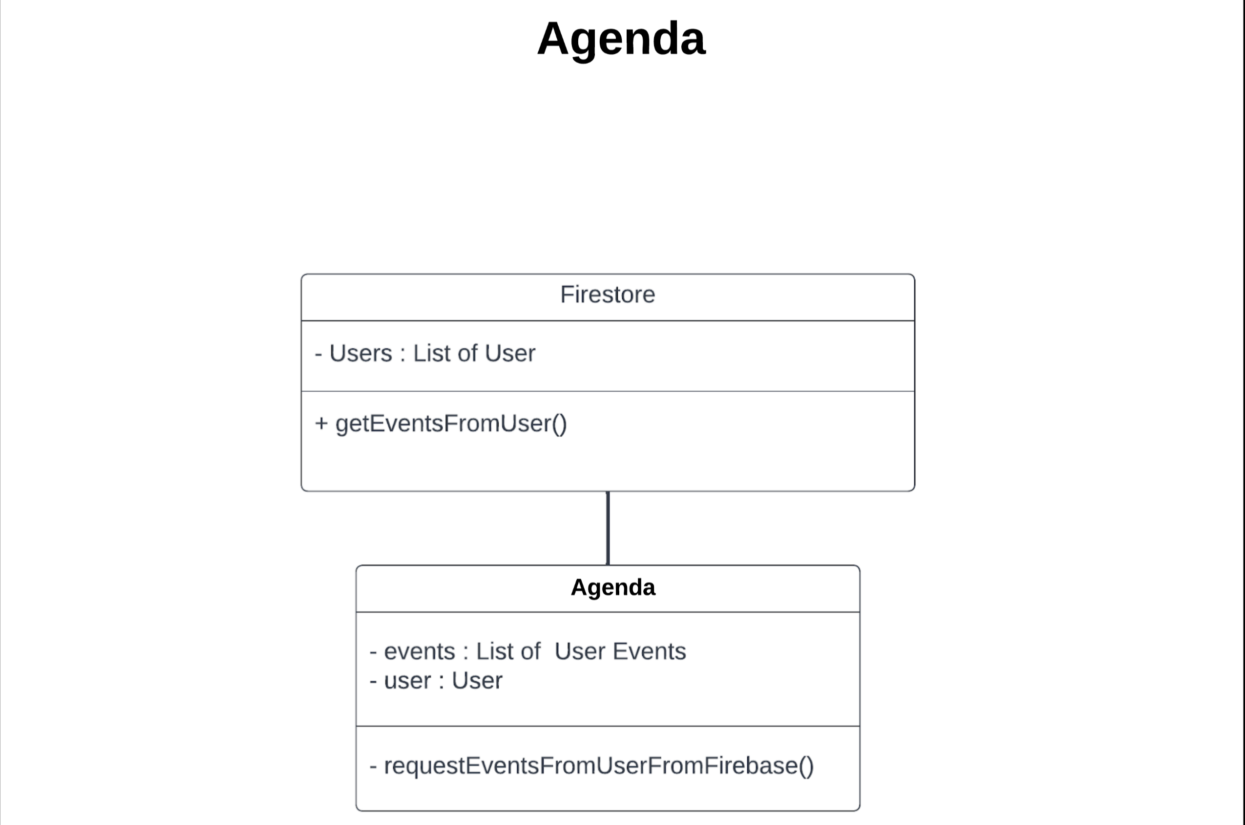
1.7.2 User Interface

Settings User Interface

 9:12 ◂ Search Change Username Here Change Password Here Delete All Data Delete Account Logout	 9:12 ◂ Search Change Username Here FDD Save Changes Delete All Data Delete Account Logout
The settings page has two inputs where the user can input a new email or password. When either a username or a password is inputted by the user, a Save Changes button will appear to call the requestUpdateUserInfoFromFirebase () method when pressed which updates the information in Firebase Authentication and in Firestore where the info is held. The delete all data will call the requestDeleteAllDataFromFirestore () method which will delete all the data of the user. The delete account button will call the requestDeleteAccountFromFirebase () method which will delete the user's account. Finally, the Logout button will call the requestLogoutFromFirebase () method which will logout the user from Firebase's Authentication service.	

1.8 Agenda

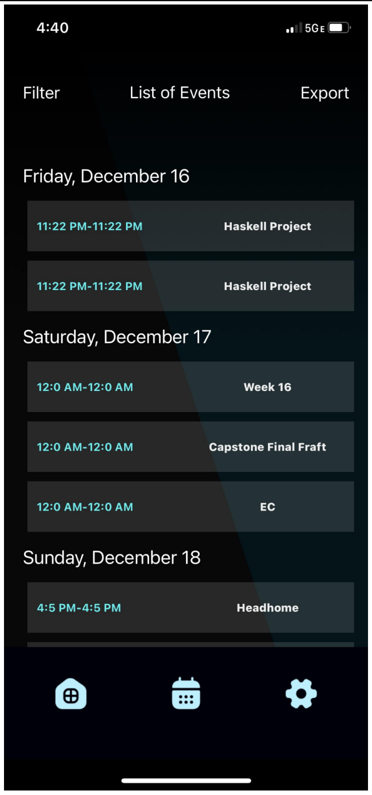
1.8.1 UML Class Diagram



Agenda Data architecture as a UML Class Diagram. Requests events from Firestore from the user and stores them in a list to use.

1.8.2 User Interface

Agenda User Interface



4:40

Filter　List of Events　Export

Friday, December 16

11:22 PM-11:22 PM　Haskell Project

11:22 PM-11:22 PM　Haskell Project

Saturday, December 17

12:0 AM-12:0 AM　Week 16

12:0 AM-12:0 AM　Capstone Final Fraft

12:0 AM-12:0 AM　EC

Sunday, December 18

4:5 PM-4:5 PM　Headhome

The Agenda page where a quick view of the upcoming events are shown. A filter and export button are shown in the top corners of the application. The export option utilizes the filtered events to export a PDF list of the events. The filter takes the incoming events from firestore and filters those events based on subjects chosen from the user. The list of events is ordered by date and time.

Chapter 4: Testing

for

FT Capstone Project

Prepared by Michael S. Heinzman

Florida Institute of Technology

Requirements Testing

Features I will be testing the requirements on.	
Single Responsibility	A requirement describes one thing, and one thing only.
Completeness	A requirement is fully laid out in one spot, and all the necessary information is present.
Consistency	A requirement does not contradict other requirements and fully complies with external documentation.
Atomicity	A requirement can't be divided into several more detailed requirements without losing completeness.
Traceability	A requirement fully or partially complies with business needs as stated by the stakeholders and is documented.
Relevance	A requirement hasn't become obsolete after time has passed.
Feasibility	A requirement can be realized within the given project.

Authentication Requirements

Login Requirements Testing

Passed or fail	Requirement	XREF (In requirements document)
Passed	The login page will be the first page that the users see in the mobile application.	3.1.1.1
Passed	The login page should provide two text fields.	3.1. 1.2
Passed	The first text field should be for entering a username or email.	3.1. 1.3
Passed	The second text field should be for entering the user's password.	3.1. 1.4
Passed	The login page should have a submit button that can be pressed by the user.	3.1. 1.5
Passed	The submit button when pressed should verify the users text field information.	3.1. 1.6
Fail Atomicity	If either of the text fields are left blank, it will result in an error that must be reported to the user when the submit button is pressed.	3.1. 1.7
Fail Atomicity	If both fields are filled in but there is no record of the username or email, or the	3.1. 1.8

	password is incorrect, that must also be reported to the user.	
Passed	Users can click the register command button to switch pages to the signup page.	3.1. 1.9
Passed	Users can click the Forgot Password command button to switch to the reset password page.	3.1.1.10

Comments:

3.1.7 Atomicity Fail: I believe that it can be separated into two statements. The first, if the text fields are left blank, it will result in an error. The second, if an error occurs then it must be reported to the user.

3.1.8 Atomicity Fail: I believe it can be separated into two statements. If there is no record of the username or email, then that must be reported to the user. If the password is incorrect, that must be reported to the user.

Signup Requirements Testing

Passed or fail	Requirement	XREF (In requirements document)
Passed	The Signup page should be displayed when the user clicks on the Signup command in the login page.	3.1.2.1
Passed	The page should have four text fields.	3.1.2.2
Passed	The first text field should be the user's username.	3.1.2.3
Passed	The second text field should be the user's email.	3.1.2.4
Passed	The third text field should be the user's password.	3.1.2.5
Passed	The fourth text field should ask the user to confirm the user's password.	3.1.2.6
Passed	The page will have a submit button that will initiate the verification of the information in the text fields.	3.1.2.7
Fail Atomicity	If any text field is blank, then the user will be prompted with an error message and will need to fill in the text field with information.	3.1.2.8
Passed	If the username or email exists in the database already that must be reported as an error.	3.1.2.9

Passed	If the password does not match the confirmed password or vice versa that must be reported as an error.	3.1.2.10
Passed	The page will have a login command button.	3.1.2.11
Passed	The user should be able to press the login command button to switch to the login page.	3.1.2.12

Comments:

3.1.2.8 Fail Atomicity: This can be separated into two statements. If a text field is blank, then the user will be prompted with an error message. If an error message occurs because of a text field, then the user will need to fill the input with correct information.

Forgot Password Requirements Testing

Passed or fail	Requirement	XREF (In requirements document)
Passed	The page should have one text field.	3.1.3.1
Passed	The text field should ask for the user's email address.	3.1.3.2
Passed	The page will have a reset password command button.	3.1.3.3
Passed	If the reset password command button is pressed the user's email will be checked if it exists.	3.1.3.4
Passed	If the user's email exists, then the user will be sent an email to reset the user's password.	3.1.3.5
Passed	If the user's email does not exist, the user will be notified with an error.	3.1.3.6
Passed	The page will have a back to login page command button.	3.1.3.7
Passed	The back to login page command button will switch pages to the login page if pressed.	3.1.3.8

Comments:

NA

Settings

Change User Account Information

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user will be presented with three text fields showing the username, password, and email.	3.2.1.1
Passed	The text fields should be in the order of Username, Email, Password.	3.2.1.2
Passed	The user should be able to change the information in all three text fields.	3.2.1.3
Passed	The user must be asked to confirm the change by typing in their password before submitting the information.	3.2.1.4

Comments:

NA

Clear All Data

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user should be able to press a button labeled “Clear All Data”.	3.2.2.1
Passed	The user should be asked to confirm the action by entering their password.	3.2.2.2
Passed	The user should be able to cancel the request by hitting a cancel button at any point after clicking the “Clear All Data” button.	3.2.2.3

Comments:

NA

Delete Account

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user should be asked to confirm the deletion of their account by entering their password.	3.2.3.1
Passed	A user should be able to cancel the deletion of their account by hitting a cancel button during confirmation.	3.2.3.2
Passed	A user should be able to hit a submit button after entering their password.	3.2.3.3
Passed	Once the submit button is pressed, the user’s password will be verified.	3.2.3.4
Passed	If valid, the user’s account will be deleted.	3.2.3.5
Passed	If a user’s account is deleted, the user should be sent to the login page.	3.2.3.6

Passed	If not valid, the user should be given an error message.	3.2.3.7
Passed	A user should be able to retry the deletion of their account as many times as they want.	3.2.3.8

Comments:

NA

Logout

Passed or fail	Requirement	XREF (In requirements document)
Passed	The authentication service receives a logout request.	3.3.1
Passed	The user is logged out of their current account.	3.3.2
Passed	The user switches pages to the login page.	3.3.3

Comments:

NA

Dashboard

Change Dates in Dashboard

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user will be shown a list of dates at the top of the page.	3.4.1.1
Passed	A user will be able to select a date from the list.	3.4.1.2
Passed	The dates of the current week will be shown.	3.4.1.3
Passed	The user will be able to navigate between the weeks in a year by clicking a forward and back button.	3.4.1.4
Passed	If the back button is clicked the previous week of the current week will be shown.	3.4.1.5
Passed	If the user clicks the forward button the next week of the current week will be shown.	3.4.1.6
Passed	Once a date is selected, the date will be changed to that date.	3.4.1.7
Passed	The user will see the events of that current date on the dashboard page.	3.4.1.8

Comments:

3.4.1.4 – 3.4.1.6 Similar: I believe these can be simplified to two statements. 3.4.1.5 and 3.4.1.6 are summarized by 3.4.1.4.

Event in Dashboard

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user will be shown a list of events in order of time for the selected date.	3.4.2.1
Passed	An events title must be shown to the user on the dashboard page.	3.4.2.2
Passed	When an event is clicked, the user must be directed to the event view page.	3.4.2.3

Comments:

NA

Event View

Edit Event in Event View

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user must be able to change the title of the event.	3.5.1.1
Passed	A user must be able to change the description of the event.	3.5.1.2
Passed	A user must be able to change the type of event.	3.5.1.3
Passed	A user must be able to change the subject of the event.	3.5.1.4
Passed	A user must be able to change the date and time the event will start.	3.5.1.5
Passed	A user must be able to change how often the event will happen.	3.5.1.6
Passed	A user must be able to enter the amount of time the event is estimated to take.	3.5.1.7
Fail Atomicity	A user must be able to click a submit button to submit the information given and save the edit of the event.	3.5.1.8

Comments:

3.5.1.8 Fail Atomicity: This requirement can be separated into two. First, A user must be able to click a submit button to submit the information given. Second, when information is submitted, the information of the event is saved. However, I don't think the second is needed that much.

Delete Event in Event View

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user must be asked to confirm the deletion of the event.	3.5.2.1
Fail	If yes, the event will be deleted.	3.5.2.2
Fail	If not, the event will not be deleted.	3.5.2.3

Comments:

3.5.2.2 Fail Completeness: The "If yes" part is reliant on the requirement 3.5.2.1.

3.5.2.3 Fail Completeness: The "If yes" part is reliant on the requirement 3.5.2.1.

Add Event in Event View

Passed or fail	Requirement	XREF (In requirements document)
Passed	A user must be able to enter the type of event.	3.5.3.1
Passed	A user must be able to enter the title of the event.	3.5.3.2
Passed	A user must be able to enter a description of the event.	3.5.3.3
Passed	A user must be presented with previous subjects created by the user.	3.5.3.4
Passed	A user must be able to choose a previous subject.	3.5.3.5
Passed	A user must be able to create a new subject.	3.5.3.6
Passed	A user must be able to enter the subject of the event.	3.5.3.7
Passed	A user must be presented with a list of types based on the subject chosen.	3.5.3.8
Passed	A user must be able to create a new type of event.	3.5.3.9
Passed	A user must be able to enter the type of the event.	3.5.3.10
Passed	A user must be able to enter the date and time the event will start.	3.5.3.11
Passed	A user must be able to enter the date and time the event will end.	3.5.3.12
Passed	A user must be able to enter how often the event will happen.	3.5.3.13
Passed	A user must be able to enter the amount of time estimated for the event to be completed.	3.5.3.14

Passed	A user must be able to click a submit button to submit the information given and create the event.	3.5.3.15

Comments:

NA

Performance Requirements

Dashboard

Passed or fail	Requirement	XREF (In requirements document)
Loading Events		
Passed	The events must be retrieved from the database and presented to the user in no more than 3 seconds.	5.1.1.1
Passed	The events must be placed at the correct times.	5.1.1.2
Passed	The events must have the correct information.	5.1.1.3
Loading this week's dates at the top of page for selection		
Passed	This week's dates must be presented first at the top of the page for the user to select in less than 3 seconds.	5.1.1.1
Passed	The current date must be selected and displaying the correct events for that day.	5.1.1.2
Fail Atomicity	The buttons to navigate between weeks must be able to go forward a week and backwards a week from the presented week.	5.1.1.3
Add Event Button		
Fail	The dates must only be marked when an event is in them.	5.1.1.1
Passed	The add event button must navigate the user to the add event page when clicked.	5.1.1.2
Passed	The navigation to the add event page should take no longer than 3 seconds.	5.1.1.3
View Event		
Passed	An event pressed must navigate the user to the View Event page.	5.1.1.1
Passed	The navigation of the user to the view event page should take no longer than 3 seconds.	5.1.1.2

Comments:

Loading this week’s dates at the top of the page for selection: 5.1.1.3 Fail Single Responsibility: I believe that it can be separated into two statements. Once saying it can navigate forward and one saying it can navigate backward.

Add Event Button: 5.1.1.1 Ambiguous: What does it mean to have an even be in them?

View Event

Passed or fail	Requirement	XREF (In requirements document)
Show Event Details		
Passed	The correct data for the event pressed on the dashboard page must be displayed to the user.	5.1.2.1
Edit Event		
Passed	The user must be navigated to the edit event page in less than 3 seconds when the edit button is pressed.	5.1.2.1
Complete Event		
Passed	When the complete event button is pressed, the user must be asked to enter a time the event took.	5.1.2.1
Passed	They must not complete until the user enters a time.	5.1.2.2
Delete Event		
Passed	When a user clicks the delete button, the user must be asked if they truly want to delete the event.	5.1.2.1
Passed	The deletion of an event should take no more than 3 times to delete if an error occurs.	5.1.2.2

Comments:

NA

Authentication

Passed or fail	Requirement	XREF (In requirements document)
Resetting Password		
Passed	The user should receive the reset password email when the submit button is pressed no more than 5 minutes after the request.	5.1.3.1
Login		
Passed	The user should be able to login with less than 3 or within 3 tries.	5.1.3.1

Comments:

NA

Settings

Passed or fail	Requirement	XREF (In requirements document)
Clearing All Data		
Passed	It should take no more than 10 seconds to delete all the data on the user.	5.1.4.1

Comments:

NA

Chapter 5: Conclusion

for

FT Capstone Project

Prepared by Michael S. Heinzman

Florida Institute of Technology

Conclusion

The situation involving students and managing their time is always going to be a problem transitioning from high school to college. This mobile application may not be the end all be all of the solution to students achieving time management skills, but it will provide an awareness to the lack of knowledge students have on the time it takes to finish their own tasks. Regardless of the purpose of this project which I believe I have achieved, I learned many key components involved with eliciting requirements, designing a user interface, designing a data plan, and testing a mobile application. In the beginning, I learned that requirements change and mold as you realize the limitations of what your team and you can achieve in the schedule. I had originally planned to build a much larger application with higher level of functionality, but through writing the requirements and implementation I learned what I can handle and how experience with building similar applications and specification of requirements can lead to a more accurate time frame and success in creation of a finished product. There were many times I had to backtrack and tell myself this isn't going to work this way and think of a different way that was easier for one person to achieve. Building the design was a similar process, I decided to build the design in figma where I learned what it takes to make a user interface that is simple and corresponds to accessibility guidelines. Combining the design I built in figma and the data was a challenge, I had to learn how to use UML class diagrams and conform them to a react project which is mainly components. It took many days to figure out the best way to implement Firebase into the project, since Firebase had recently been updated to the newest version where there was little information. Overall, I achieved a product that was usable and although limited than what I originally intended, achieved my goal of providing an application that students can use to see how their perception of their own time management stacks up against their actual time management skills. If I had to do anything differently I would have not rushed the actual programming as much as I did and do a little of each (design, data, programming, requirements) throughout the process instead of all at once. This was a great opportunity that allowed me to grow even more as a developer.