CatMap

Design Document

Group 2:
Austin M Sill, Chaker Baloch, Devlin Hamill,
Eli Holter, Krai Pongrapeeporn, Vandan Amin

Department of Computer Science, Central Washington University
CS481 - Capstone Project

Dr. Donald Davendra
January 31, 2025

Contents

1 Introduction

1.1 Purpose

The purpose of this project is to create a schedule-based campus navigational aid Android app for the Central Washington University (CWU) campus. This app can be used by students to plan out where they need to go to attend classes in their schedule, and can be used to navigate to any other major location on campus.

1.2 Scope

There are a wide range of things that can and cannot be incorporate into this application based on given requirements and time restrictions. To stay on task and avoid feature creep, the minimum requirements and known restrictions are listed below:

1.2.1 Things that MUST be in the app:

- An intractable graphical user interface (GUI)
- A map of the CWU campus
- User profile functionality
- Sign-in and sign-up pages
- The user's schedule
- A favorite locations page
- Map based scheduling
- App generated directions/pathing used by the user to find the buildings for the classes in their schedule
- Modification support such as added, editing, and deleting specific parts of the schedule.
- The daily schedule that displays the user's classes, buildings and times.

1.2.2 Things that must NOT be in the app:

- A map of the entire world will not be provided
- Directions to places outside the campus will not be used
- Addresses will not need to be taken into consideration.
- Google street view will be unnecessary.
- Buildings outside of the campus will not need to be visualized.
- Times outside Ellensburg's time zone will not be provided.

1.3 Glossary

- Event: a singular period of time that represents scheduled reoccurring classes or singular events
- Application Programming Interface (API): A set of functions and commands provided by a software provider that allows connection and interaction with an external software component.
- Database: This will serve as shorthand for Firebase Cloud Firestore database.
- Inactivity: when something is not being interacted with by the user.
- Class: internally, classes will be represented as reoccurring scheduled events.

2 Environment

The app will only be runnable and available on Android devices. It will not function on every version of Android OS, and to be allowed to published it on the play store, the required target API level must be above a specific version.

2.1 Minimum Software Platform

- Minimum Android OS version: Android 5.0
- Minimum Android API version: API level 21

2.2 Target Software Platform

- Target Android OS version: Android 14
- Target Android API version: API level 34

2.3 Minimum Hardware Requirements

- **Global Positioning Service (GPS) antenna:** The app will rely on the device's ability to connect to the GPS satellite network to get the users current position.
- **Wi-Fi and/or Cellular data antenna:** The app needs to be able to connect to the internet to access data in the Firestore database.
- **Touch Screen:** The user will interact with the app using the touch screen. using a mouse, keyboard, and monitor will not be a viable option as the reduction of mobility would defeat the purpose of the app.
- **Storage:** The app will store data about their schedule, preferences and location data. The app will require at least 500MB of free storage space to be installed and run correctly.
- **Memory:** In accordance with Google's Play Store requirements, devices must have at least 2 GB of memory to be allowed access to the play store. This requirement will extend to the minimum requirements to run the app
- **Processor:** Android stipulates that one or many specific application binary interfaces (ABI) must be implemented in a device for it to run Android OS. The app will also have this requirement, meaning only devices that implement the ARM v7a, ARM v8a, x86, x86-64 ABIs will be able to run the app.

3 Software Design

3.1 Philosophy

When designing the class structure and general architecture of the app, there was a focus on the principles of object-oriented programming (OOP) and avoided creating classes that already exist in the Java Development Kit (JDK) or Android Software Development Kit (SDK). While it may be fun, reinventing the wheel is a waste of time, and a lack of time is the biggest obstacle in the way of completing this software project.

The main OOP principles applied in the class structure of the software include inheritance and polymorphism.

- **Inheritance:** All event subtypes extend an abstract class 'Event' that contains fields that all events require. The 'Class' class extends the 'EventGroup' class as classes have the same functionality as event group but distinction fits the mental model better.
- **Polymorphism:** The 'Schedule' classes will store their respective events as 'Event' objects instead of treating each event type differently. This generalizes event functionality and allows for new event types if needed.

3.2 Software Dependencies

The app will use a handful of packages provided by the JDK and Android SDK. Class names are not very descriptive, so package descriptions will be provided.

- **Color:** Android class that stores byte values for red, green, blue and alpha of a color.
- **LocalDateTime:** Java class that stores information about a specific date and time.
- **LocalDate:** Java class that stores information about a specific date.

3.3 Software Architecture

3.3.1 Overview

The software architecture diagram shown in Figure 1 details the transfer of information between components of the app. It uses a layered and modular approach to make the system scalable, reliable, and provide a smooth user experience. The back-end is built on Firebase, offering robust functionality, while additional features like scheduling and navigation are seamlessly integrated using Google's Maps. By clearly separating responsibilities within its various layers, the architecture is easy to maintain and adaptable for future upgrades or new features.

3.3.2 Key Components

1. **Mobile Application:** The mobile app is the central component of the system, serving as the main platform for users to interact with the campus navigation features.
    - **UI Layer:** The user interface is the first point of interaction, featuring intuitive and responsive design. It captures user inputs and provides an engaging experience.
    - **Business Logic Layer:** This acts as the app's brain, managing user actions, validating inputs, and applying system rules through components like ViewModels and Controllers.
    - **Service Layer:** Handles communication between the app and external systems, such as Firebase or APIs, by managing requests and processing responses.
2. **Firebase:** Firebase forms the back-end infrastructure, delivering secure and reliable services:
    - **Authentication:** Provides secure login and user management using Google OAuth 2.0.

- **Firestore:** Stores user preferences and other personalized data in a structured and scalable database.
- **Storage:** Manages user files, such as images, ensuring easy access and secure storage.

3. **API Gateway:** The API Gateway acts as a mediator between the app and external APIs. It validates incoming requests, applies security protocols, and validates if responses are formatted correctly for the app.
4. **External APIs:** These extend the app's functionality with specialized features:
   - **Google Maps API:** Provides accurate navigation and geo-location services for campus mapping.

3.3.3 Data Flow

The system is designed for smooth and efficient data interaction:

- User inputs are collected through the UI Layer and sent to the Business Logic Layer for processing.
- The Service Layer communicates with Firebase to store or retrieve data as needed.
- External API data is accessed via the API Gateway.
- Notifications, such as reminders, are delivered to users through Firebase Cloud Messaging.

3.3.4 Scalability and Extensibility

The architecture is built to handle growing user demands and adapt to new requirements:

- Its modular design allows individual components to be developed, tested, and upgraded without affecting other parts of the system.
- The cloud-based Firebase back-end can scale dynamically to support increased traffic or data storage needs.
- Adding new external APIs or services is straightforward through the API Gateway, ensuring the system remains flexible and future-proof.

3.4 Software Architecture Diagram

The architecture is visually represented in Figure 1, which outlines the different layers, components, and their interactions.

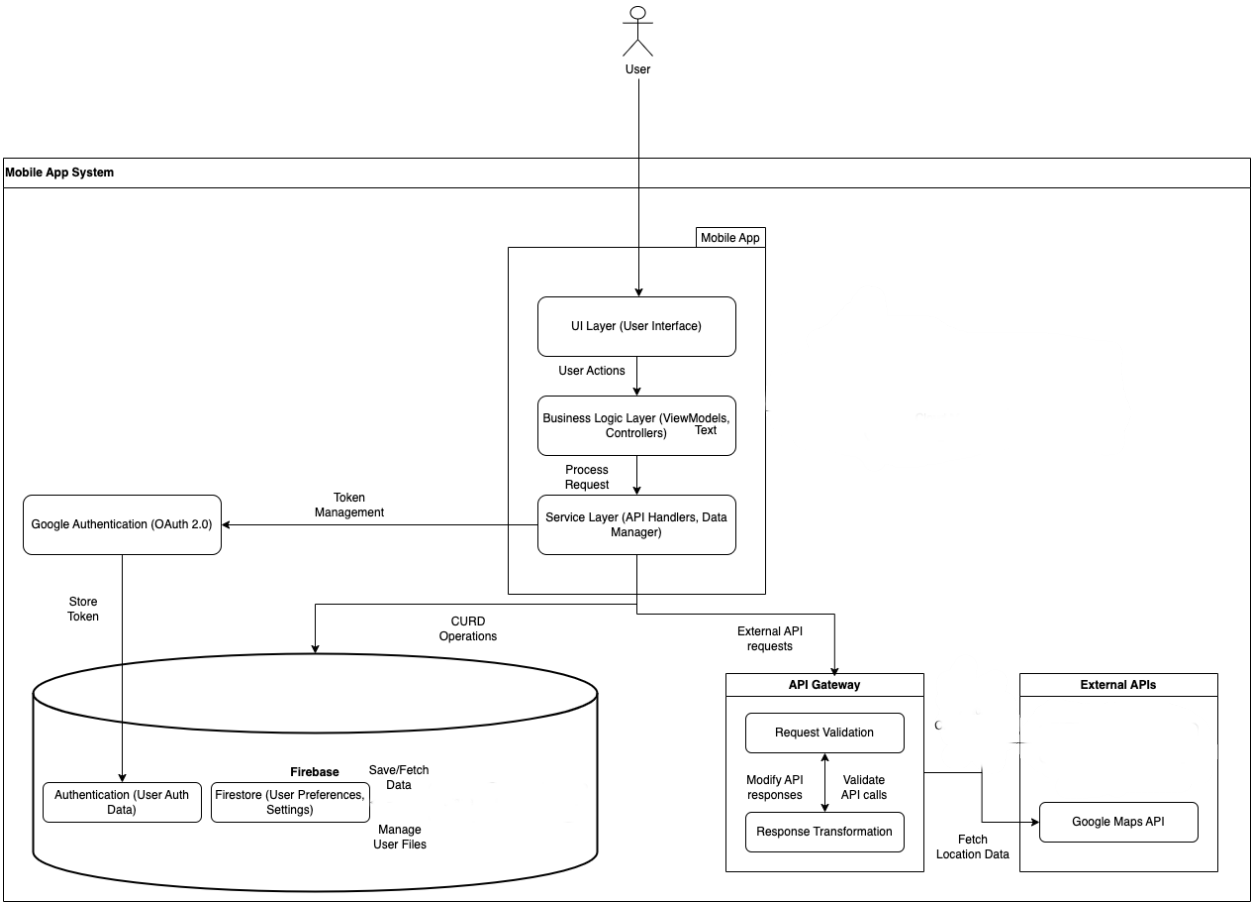


Figure 1: Software Architecture Diagram

3.5 UML Class Diagram

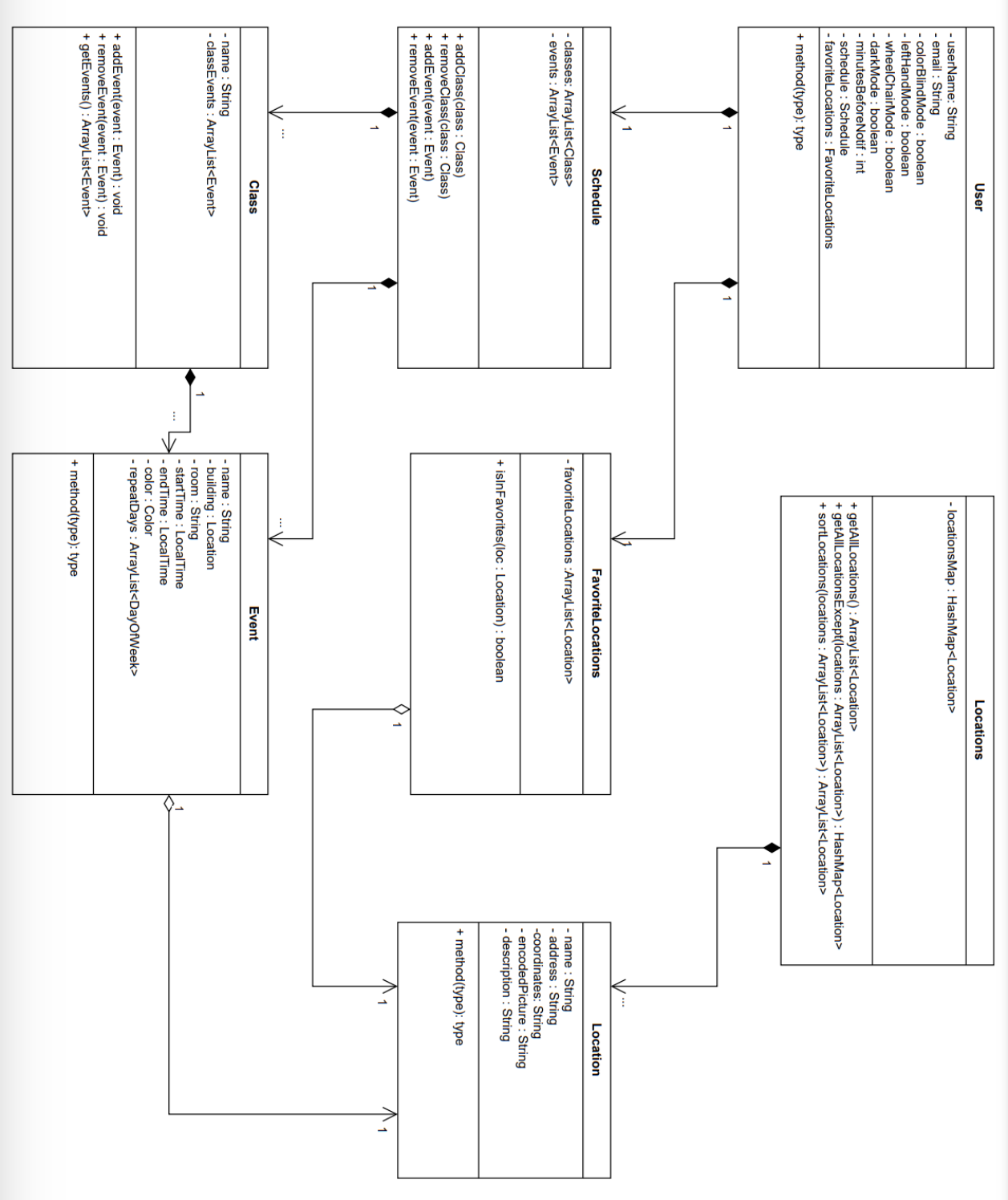


Figure 2: UML Class Diagram

4 Data Design

4.1 Data-flow Diagram

- DATA FLOW DIAGRAM (DFD) for CWU Campus Navigation App

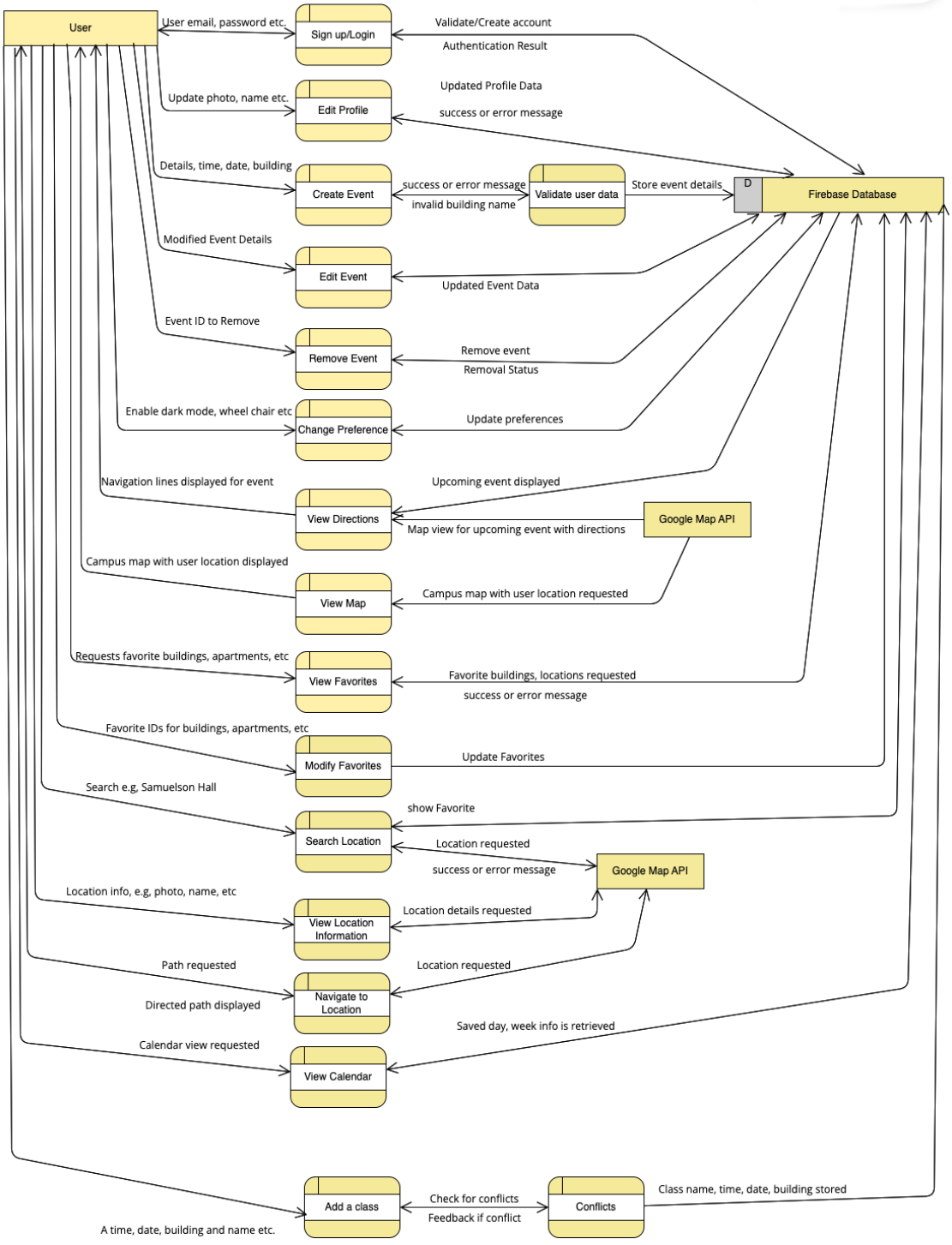


Figure 3: Data Flow Diagram

4.2 Data Dictionary

User		
Field Name	Data Type	Description
userName	String	Username
email	String	E-mail
colorBlindMode	boolean	Toggle for colorblind mode
leftHandMode	boolean	Toggle for lefthand mode
wheelChairMode	boolean	Toggle for wheelchair mode
darkMode	boolean	Toggle for dark mode
minutesBeforeNotif	int	Minutes before event to send notification
schedule	Schedule	The user's Schedule
favoriteLocations	FavoriteLocations	The user's FavoriteLocations

Schedule		
Field Name	Data Type	Description
classes	ArrayListClass	List of classes user is in
eventGroups	ArrayListEventGroup	List of EventGroups
events	ArrayListEvent	List of Event(s)
quarterStart	DateTime	Quarter start time for Schedule
quarterEnd	DateTime	Quarter end time for Schedule

FavoriteLocations		
Field Name	Data Type	Description
favoriteLocations	ArrayListLocation	List of Location(s)

Location		
Field Name	Data Type	Description
name	String	Location name
address	String	Location address
coordinates	String	Location coordinates
encodedPicture	String	Location picture
description	String	Location text description

Locations		
Field Name	Data Type	Description
locationsMap	HashMapLocation	Hashmap of Location(s)

Event		
Field Name	Data Type	Description
name	String	Event name
building	Location	Event building name
room	String	Event room name/ID
color	Color	User picked color for event

OneTimeEvent extends Event		
Field Name	Data Type	Description
name	String	Event name
building	Location	Event building name
room	String	Event room name/ID
color	Color	User picked color for event
startDateTime	DateTime	Event start date
endDateTime	DateTime	Event end date

repeatingEvent extends Event		
Field Name	Data Type	Description
name	String	Event name
building	Location	Event building name
room	String	Event room name/ID
color	Color	User picked color for event
events	ArrayListOneTimeEvent	List of OneTimeEvent(s)
scheduleOnHoliday	boolean	Toggle if events appear on holidays
deleteAtQuarterEnd	boolean	Toggle if events delete after quarter ends
startDate	Date	First date for the events
endDate	Date	Last date for the events
OccuranceDays	ArrayListDayOfWeek	List of OccuranceDays
startLocalTime	LocalTime	Start time in hh:mm:ss
endLocalTime	LocalTime	End time in hh:mm:ss

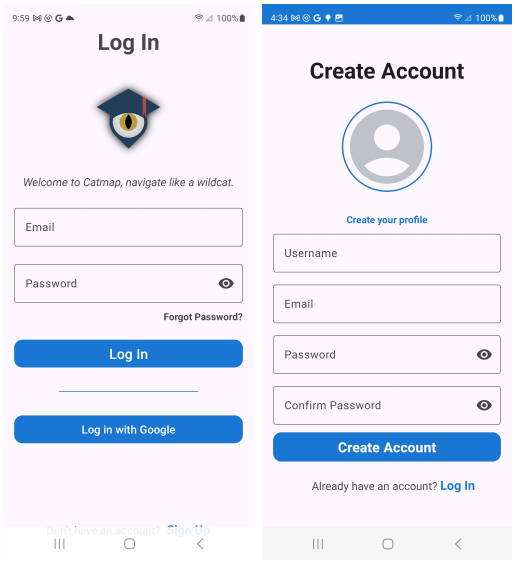
Class/EventGroup		
Field Name	Data Type	Description
events	ArrayListEvent	List of Event(s)
scheduleOnHoliday	boolean	Toggle if EventGroup occurs during holidays
deleteAtQuarterEnd	boolean	Toggle if EventGroup is deleted after quarter ends
groupColor	Color	Color appearance for all Event(s) in this EventGroup

Table 1: Data Dictionary

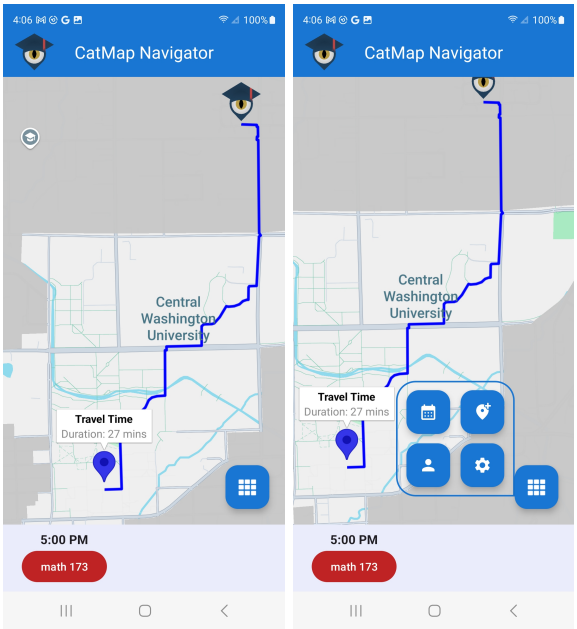
5 User Interface Design

5.1 Overview

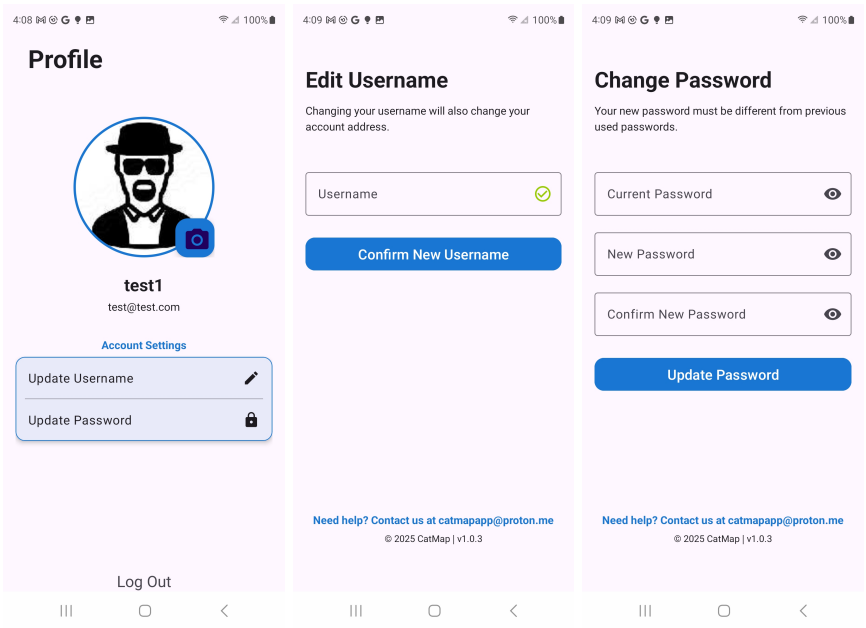
The first thing the user will see is the sign in / sign up the user will be given option to enter their user name and password and the user may have to select a profile if they are signing in. No other information will be necessary other then that from how users will not need to communicate with one another through the app.



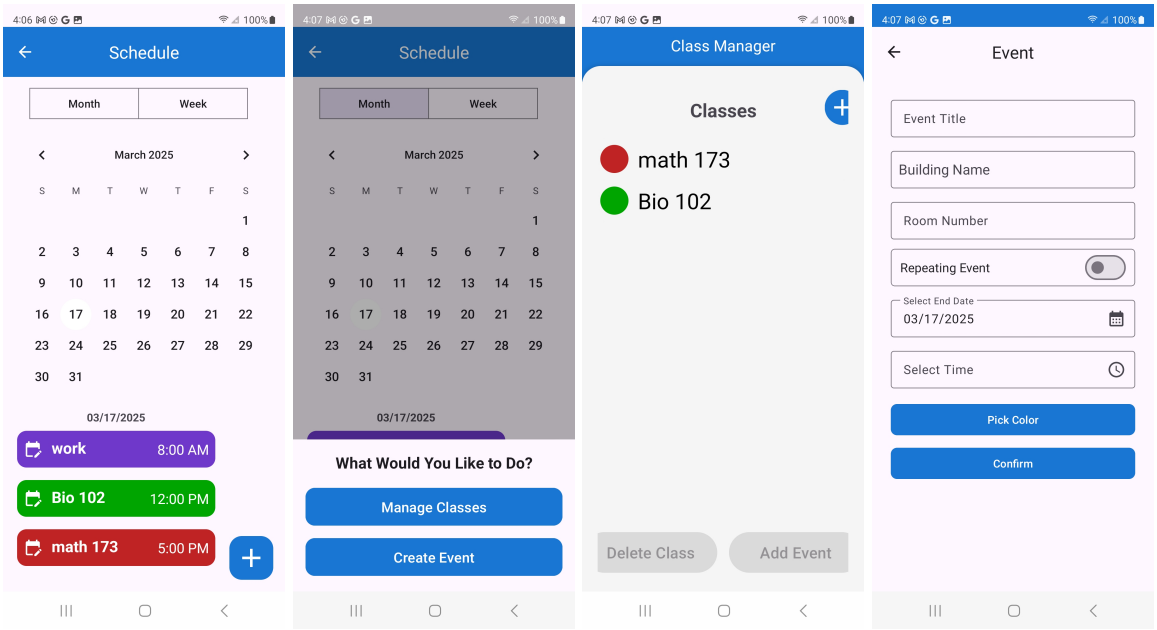
The user will then see the map with the daily schedule place underneath the map. Along with the miscellaneous icon containing the profile, schedule, locations, and the settings of the application.

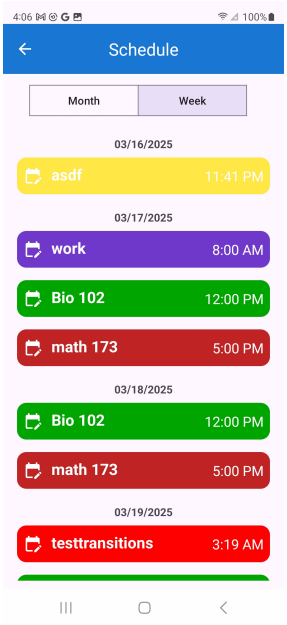


When the profile icon is clicked the user can change their profile information or recover their account.

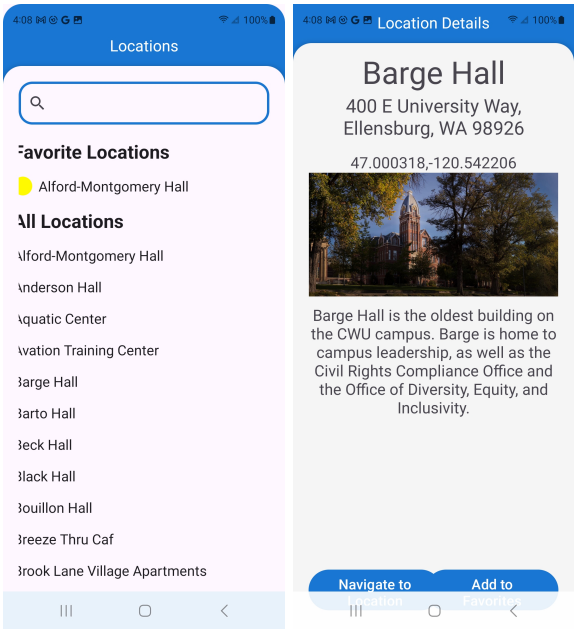


When the miscellaneous icon is clicked they can click on the schedule icon where they can view the calender on top of their schedule for the day, add a new event, add a new class, modify events and view the weekly schedule.

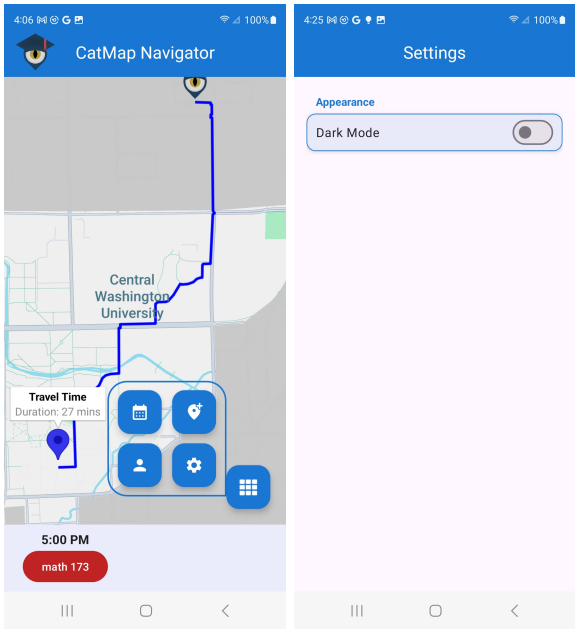




Locations where they an show all their favorites and add/remove from their favorites if necessary; along with searching for specific buildings and seeing what it looks like before they add it along with specific details on said building.



Lastly they can access the settings tab were they can change their preferences such as light and dark mode.



5.2 UI Navigation Flow

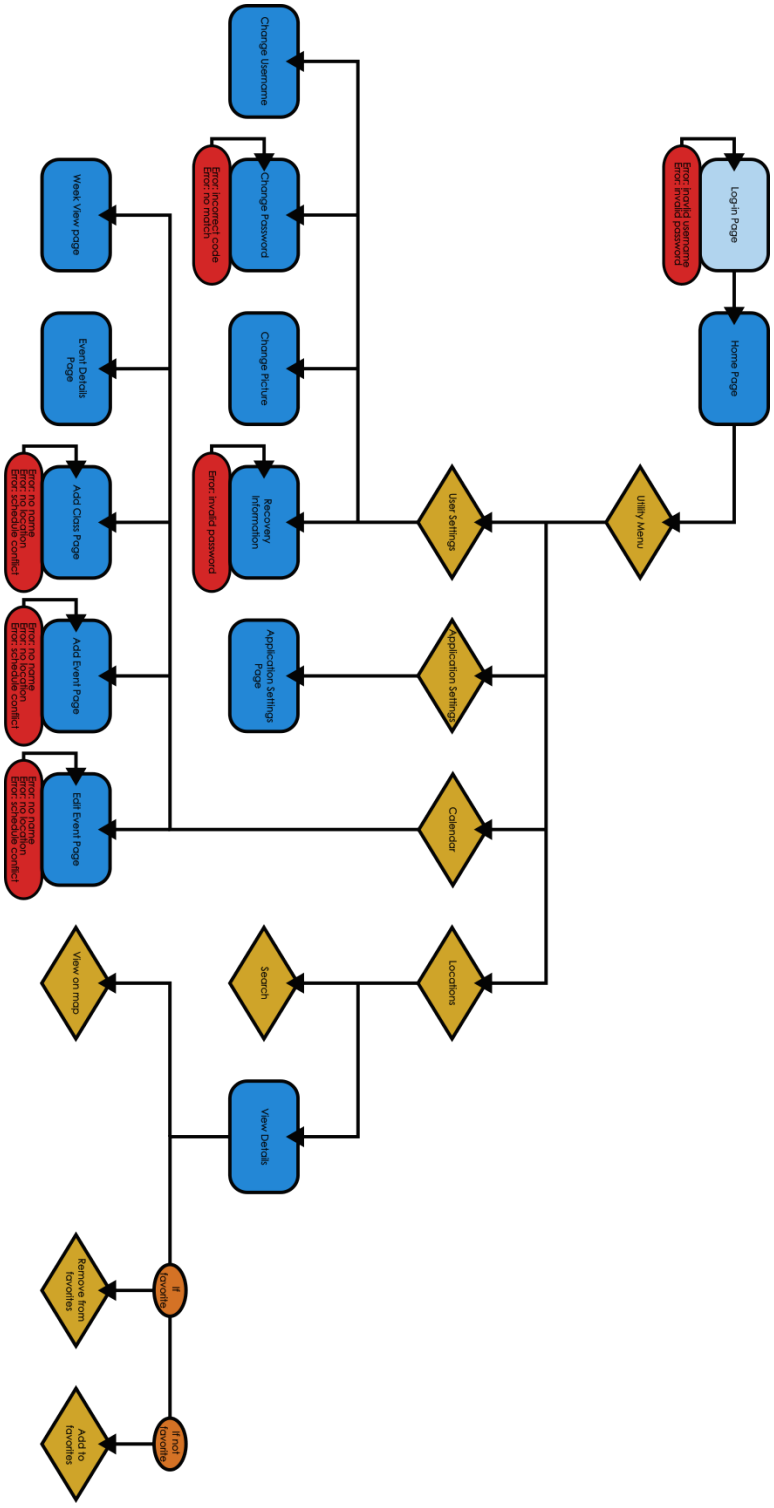


Figure 4: Navigational Flow Diagram

5.3 App Icon

This icon is the logo for the app as well as the marker that shows where the user is on the map:

Figure 5: Cat Map Logo

6 External Interfaces

6.1 Google Maps API

The application will integrate the Google Maps API to display an interactive map of the Central Washington University (CWU) campus on the homepage. This feature will enhance navigation by providing real-time positioning, building locations, and route visualization for users.

6.2 Google Routes API

The Google Routes API will be utilized to fetch accurate paths between locations to guide the user to the desired destination.

6.3 Google Authentication (OAuth 2.0)

The application will implement Google Authentication using OAuth 2.0 to provide a secure and seamless login experience for users.

6.4 Firebase

The application will use Firebase as the storage mechanism to keep user data secure and reliably accessible. This will also allow users to access their data from different devices so long as they are logged in.

7 Traceability Matrix

Use Case	Req. Num	Requirement Description	Design Comp.	Test Case	Status
1. Sign Up / Login	1.1	User can sign up			
	1.2	User can login			
	1.3	User can login with Google account			
	1.4	Saving user account schedule, favorites, etc, with Firestore			
	1.5	User data can be retrieved on any device running the app that user logs into			
	1.6	User created accounts will be stored in and retrieved from the database			
2. Edit Profile	2.1	User can change profile picture			
	2.2	User can change name			
	2.3	User can change e-mail			
	2.4	User can change password			
	2.5	User profile is retrieved from DB			
	2.6	Profile changes will be done on Firestore			
	2.7	Profile changes are done locally, then on the DB			
	2.8	Profile change queries are attempted max 5 times before failure			
	2.9	Profile changes are retrieved from DB			
3. Create Event	3.1	User can only add events if they're not conflicting			
	3.2	Events needs valid dates			
	3.3	Events need a valid building			
	3.4	Events need a title			
	3.5	Event is successfully added			
	3.6	Error message for failing to add events			
	3.7	Adding events interfaces with Google Maps			
4.Edit Event	4.1	Events needs to be on the schedule to edit			
	4.2	Edited event cannot conflict with other events			
	4.3	Event dates can be edited			
	4.4	Event names can be edited			
	4.5	Event buildings can be edited			
	4.6	Edited event details need to be validated			
	4.7	Display error message for invalid event edits			
	4.8	Editing events interfaces with Google Maps			
5. Remove Event	5.1	Events needs to be on the schedule or favorites to be removed			
	5.2	User can remove events by clicking			
	5.3	Events have a removal button			
6. Change Preferences	6.1	Misc. icon is loaded before accessing prefs. settings			
	6.2	Settings icon is loaded before accessing prefs. settings			
	6.3	Clicking misc. icon pulls up schedule, locations, and settings icon			
	6.4	Prefs. setting can be accessed via settings icon			
	6.5	Acessibility options can be turned on via settings			
	6.6	Acessibility options are turned on using a switch or slider			
	6.7	Notification settings are changed through a separate window			
7. View Directions	7.1	Map needs to be loaded before paths			
	7.2	Upcoming events always show a path			
	7.3	Event paths is a dashed line			
	7.4	Users can create custom paths			
	7.5	Building info is retrieved from Google Maps			
	7.6	Building visualization is done with Google Maps			
	7.7	Directions are created from Google Maps			
	7.8	Paths are created from Google Maps			
8. View Map	8.1	Campus Map shows topdown view			
	8.2	Map shows user location			
	8.3	Map restricts to Campus area			
	8.4	Map determines Campus area			
	8.5	Campus buildings are stored from Google Maps			
	8.6	Google Maps helps reconstruct campus visually			
9. View Favorites	9.1	Misc. icon is accessed before favorites window			
	9.2	Clicking Misc. icon shows settings, location, and schedule icon			
	9.3	Clicking Locations icon pops up Location window			
	9.4	The app will depict the users' favorite locations or search for a new one if they desire.			
	9.5	CPU Campus building is stored from Google Map			

10. Modify Favorites	10.1	Misc. icon is loaded before favorites window			
	10.2	Clicking Misc. icon shows settings, location, and schedule icon			
	10.3	Clicking Locations icon pops up Locations window			
	10.4	The app will depict user fav. locations or search for a new one			
	10.5	User can add a location to favorites via button			
	10.6	User can remove a location from their favorites via button			
	10.7	CPU Campus building is stored from Google Map			
11. View Location Information	11.1	User can view detailed location info			
	11.2	User can view location picture			
	11.3	User can view location description			
	11.4	User can view location opening and closing time			
	11.5	User can view path to location			
	11.6	User can add location to favorites via button			
12. Search Locations	12.1	Users can use search bar for specific locations			
	12.2	Search bar redirects to locations page and displays keyboard for search			
	12.3	Search bar lists locations as users type			
13. Navigate to Location	13.1	A path from user location to location is drawn			
	13.2	A pin icon represents the user's location			
	13.3	Locations will have circular pictures			
	13.4	Paths will be segmented			
	13.5	Paths will be to the latest event, or user custom location then latest event			
	13.6	Paths will be drawn using Google Maps			
	13.7	A node-oriented pathfinding algorithm will be used if Google Maps isn't robust			
	13.8	GPS may be used to retrieve user location			
14. View Calender	14.1	Clicking schedule icon shows current month calender			
	14.2	Calender can switch to other months			
	14.3	Clicking on a day will show scheduled events			
	14.4	Clicking a day displays date			
	14.5	Google Account events can be imported			
	14.6	Google Account integration will use Google Calender API			
15. View Weekly Schedule	15.1	Weekly schedule is shown			
	15.2	Weekly schedule lists 7 days vertically			
	15.3	Schedule events will be listed			
	15.4	User can click on an event for info			
16. View Generalized Daily Schedule	16.1	Daily timeline will show upcoming and past events			
	16.2	Events are displayed as rectangles			
	16.3	Daily timeline centers on current time with a red vertical bar			
	16.4	Daily timeline shows a span of 5 hours			
	16.5	Daily timeline is scrollable for all 24 hours			
	16.6	Daily timeline centers back after scrolling in 3s			
	16.7	Daily timeline can be expanded and opens extended daily view			
	16.8	Daily timeline scrolling is smooth			
17. View Extended Daily Schedule	17.1	Daily timeline can be expanded and opens extended daily view			
	17.2	Extended daily view contains all info in daily view, and more specifics			
	17.3	Extended daily view displays events with taller rectangles			
	17.4	Extended daily view displays date and is scrollable			
	17.5	Extended timeline scrolling is smooth			
18. Add Class	18.1	Classes cannot have conflicting times			
	18.2	Classes need a valid date			
	18.3	Classes need a valid building			
	18.4	Class info needs to be validated prior to adding			
	18.5	Event will be added after info validation			
	18.6	Error message for inputting invalid info			
	18.7	Building name is retrieved from Google Maps			

Figure 6: Traceability Matrix