Software Design Document

Guard Dog Mobile Application

Sekou Diaby

Gaymer Barrios

Brock Ryan

Anojan Balendra

Date: 12/09/2021

1.0 Introduction

1.1 Purpose

This software design document describes the architecture and system design of the Guard Dog Mobile application. This application was developed as the final project for Fall CP-317 at Wilfrid Laurier University.

The intended audience of this document is the Professor of CP317 as well as any other individual involved in the marking of the CP317 final project.

2.0 System Overview

The idea of this project was influenced by recent events at Western University campus of the alleged drugging and assault of first-year students on a university residence. Rumours were spread on social media which made it harder for the police to corroborate the accusations and although Western University has a safety app, it was not helpful on this occasion. Thus, we started brainstorming on a mobile application that can prevent or make it easier for students to report these incidents in a safe way that is separated from the university and social media.

The general functionality of the application allows a user to login or register to get access to a map screen showing the locations of reported incidents. Using the navigation bar, the user can then see alerts, access an EMS page, make a report or logout.

The design of the application is divided into 5 sections which are authentication, map view, EMS view, alerts view and incident report submission.

3.0 System Architecture

3.1 Architectural Design

The system is divided into the following subsystems:

1. The Authentication subsystem
2. The Map subsystem
3. The EMS subsystem
4. The Alerts subsystem
5. The Report subsystem

The Authentication subsystem is responsible for allowing the user to register an account, login, and logout out of their account.

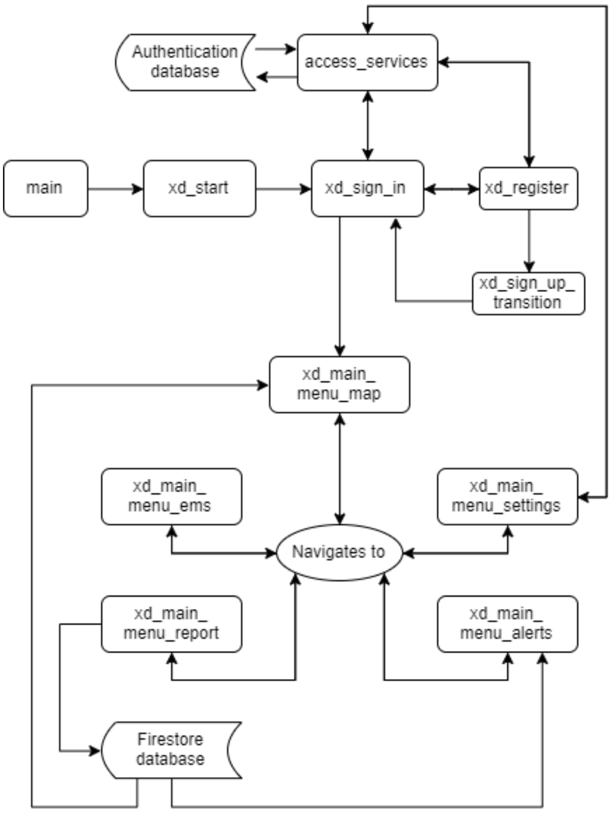
The Map subsystem is responsible for displaying the current location of the user in a map and show any nearby reported incidents with an image pin on the map. It is also responsible for allowing the user to send out an audible distress signal.

The EMS subsystem is responsible for providing a communication channel with emergency personal. This a future feature that can be added to the application.

The Alerts subsystem is responsible for querying a database of user reported incidents and displaying a list ordered by the current distance of the user to the events.

The Report subsystem is responsible for collecting the information necessary to submit an incident report. The report information is saved to a database.

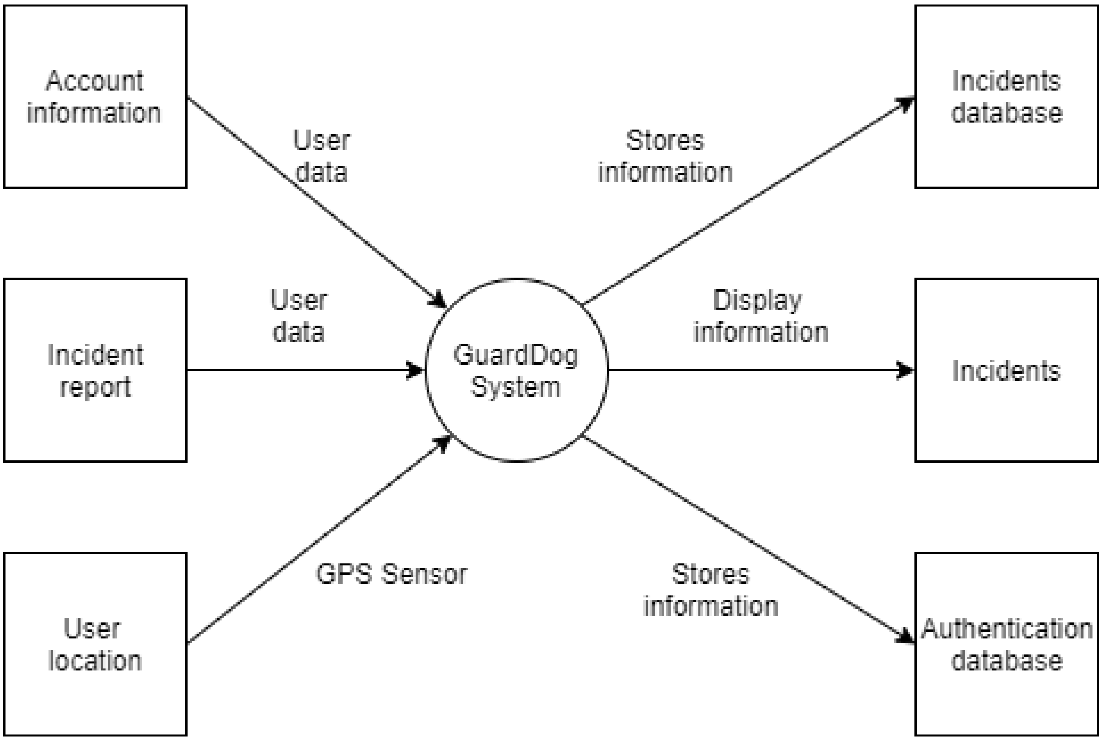
The following diagram shows the major subsystem, data repositories and their interconnections.



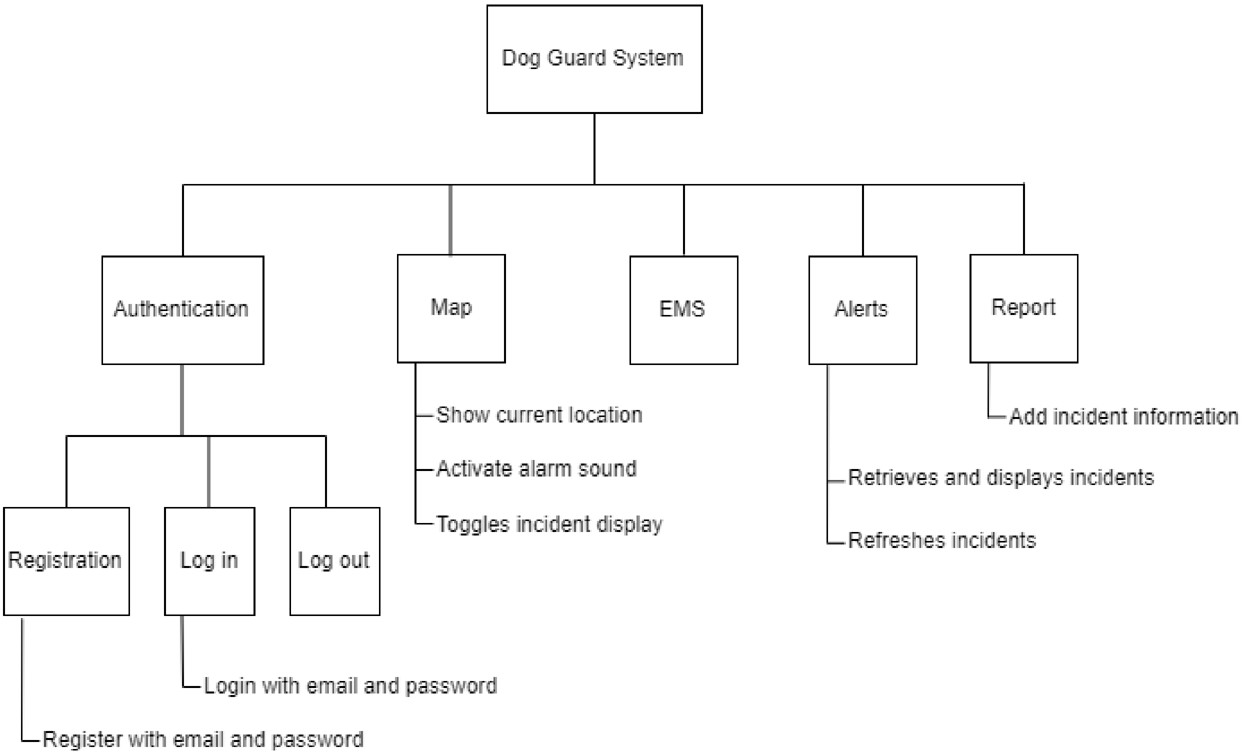
3.2 Decomposition Description

The following is a functional description of the Guard Dog mobile application.

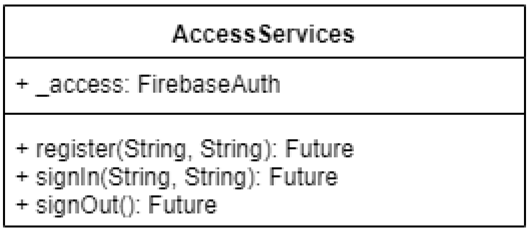
Top-level data flow diagram



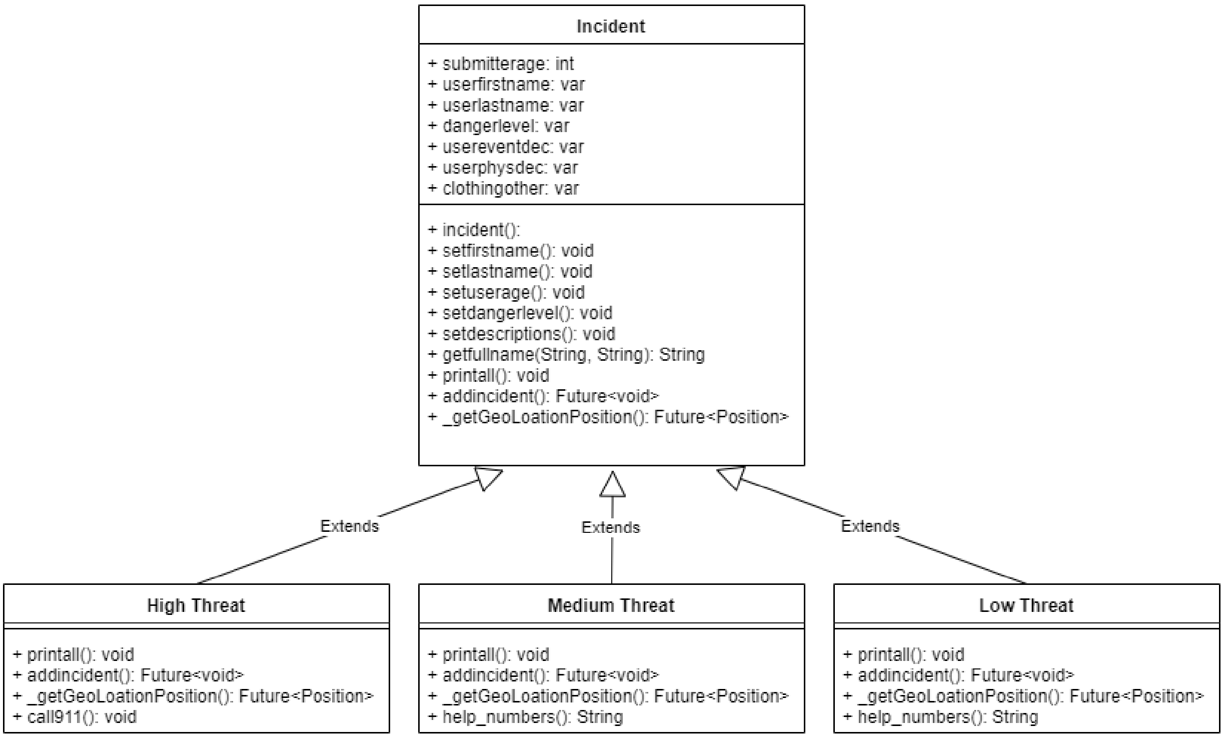
Structural decomposition diagram



AccessServices class diagram


AccessServices
+ _access: FirebaseAuth
+ register(String, String): Future
+ signIn(String, String): Future
+ signOut(): Future

The incident parent class and child classes relationship



4.0 Data Flow

4.1 Data Description

The Guard Dog mobile application uses the Firebase google platform. The Firebase Authentication database is used to manage account registration, account sign in and log out functionality. The Cloud Firestore is used to manage the submitted user incidents. All data that is stored within Firebase is stored in a serialised JSON format. The application shares a common credentials to login into Firebase.

User Data Structure

In the application, a GuardUser is a system entity that uniquely identifies an application user. An email and a password are collected from the user to create a Firebase user. For our application, most of these attributes are not needed so we used the GuardUser class to create an abstraction of the firebase user so that we would only work with the attributes our application required. In this case, we only needed the uid, email and password attribute to uniquely identify users and reported incidents.

Attribute Name	Attribute Type	Attribute Description
email	string	User email address
password	string	User Login password
uid	string	ID assigned to a user that has signed up

Incident Data Structure

An incident object within the application is composed of multiple attributes which are collected and can be used to describe an event of threatening nature. These attributes are as follows:

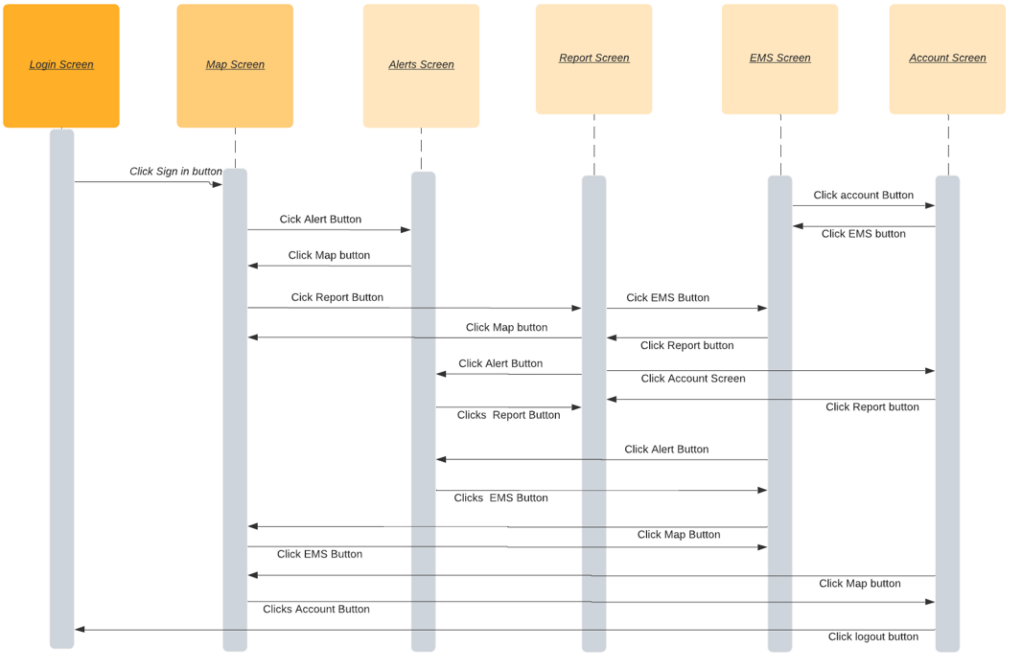
Attribute Name	Attribute Type	Attribute Description
eventid	string	Unique ID of the incident.
userfirstname	string	The first name of the user submitting the incident.
userlastname	string	The last name of the user submitting the incident.
usereventdec	string	The description of the incident.
userphysdec	string	The description of the person(s) of interest.
dangerlevel	string	The level of the threat identified
clothingother	sting	Additional descriptions
locationlat	string	Location latitude
locationlong	string	Location longitude
submitterage	string	The age of the user submitting the incident.
timesubmitted	string	Time and date when incident was submitted

All these fields are assigned a data type mostly integers and string and are collected through text field within the application. A new incident object is created, assigned with these fields, serialized, and sent to the Firebase to be stored into a JSON collection. When storing within Firebase, a new Firebase connection is instantiated with our application's Database login credentials which is shared among all users. The name of the collection, which is synonymous to a table, is used to direct the data to the right collection in Firebase. The events are sent, stored, and sorted by date and time of submission.

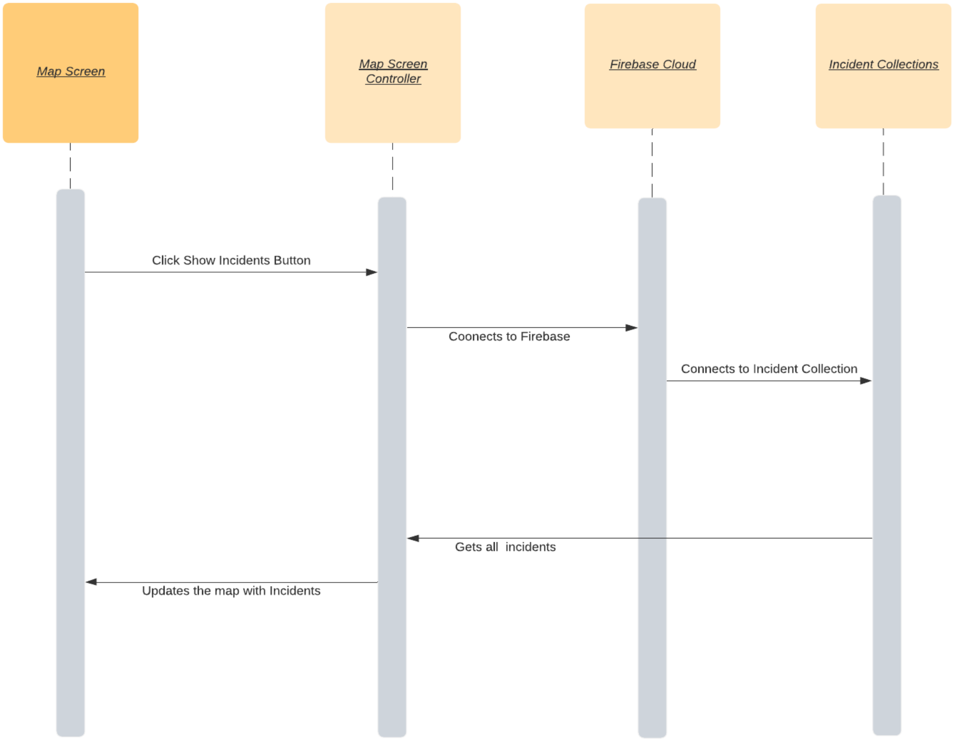
When retrieving an incident from Firebase, the application creates a new Firebase connection, connects to the right collection/table, and retrieves the serialized JSON data into a string. This string is deserialized into an incident object where we loop through the string and assign a value to every attribute of the object. The list of objects that is retrieved from Firebase is also sorted by date and time of submission.

5.0 Use Case Realizations

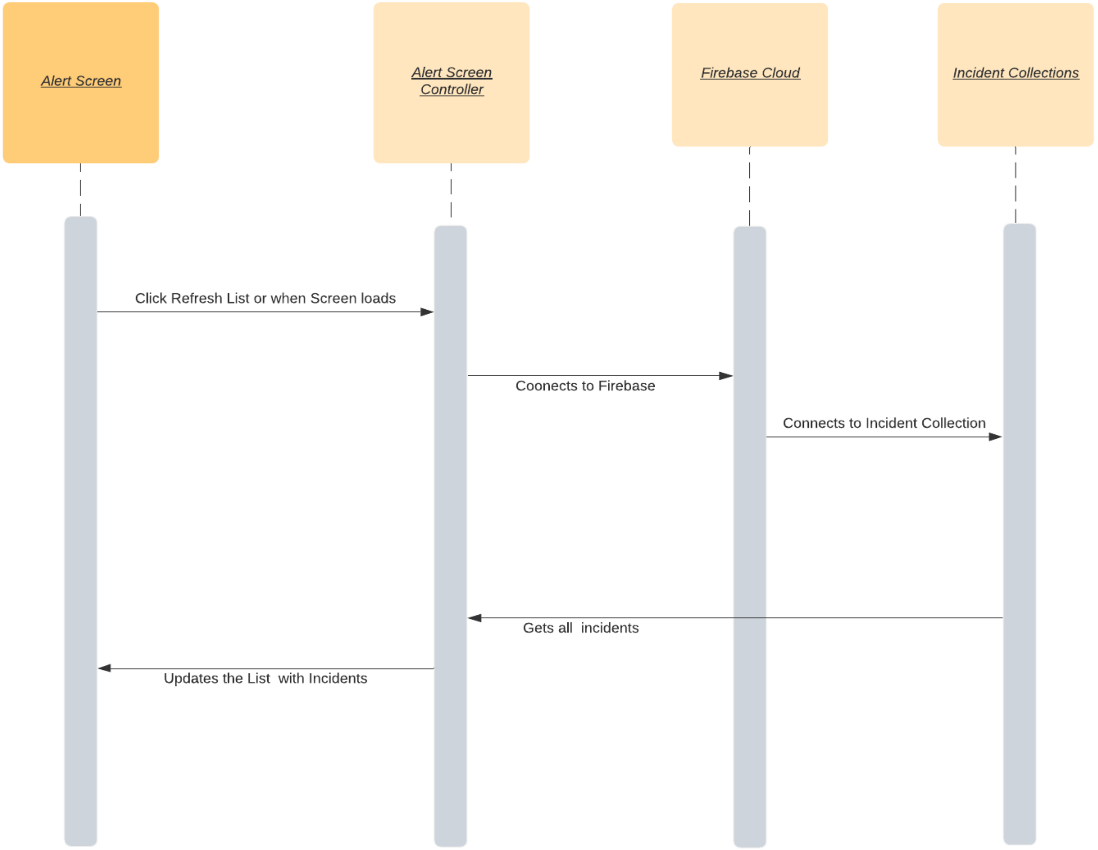
System Sequence Diagram



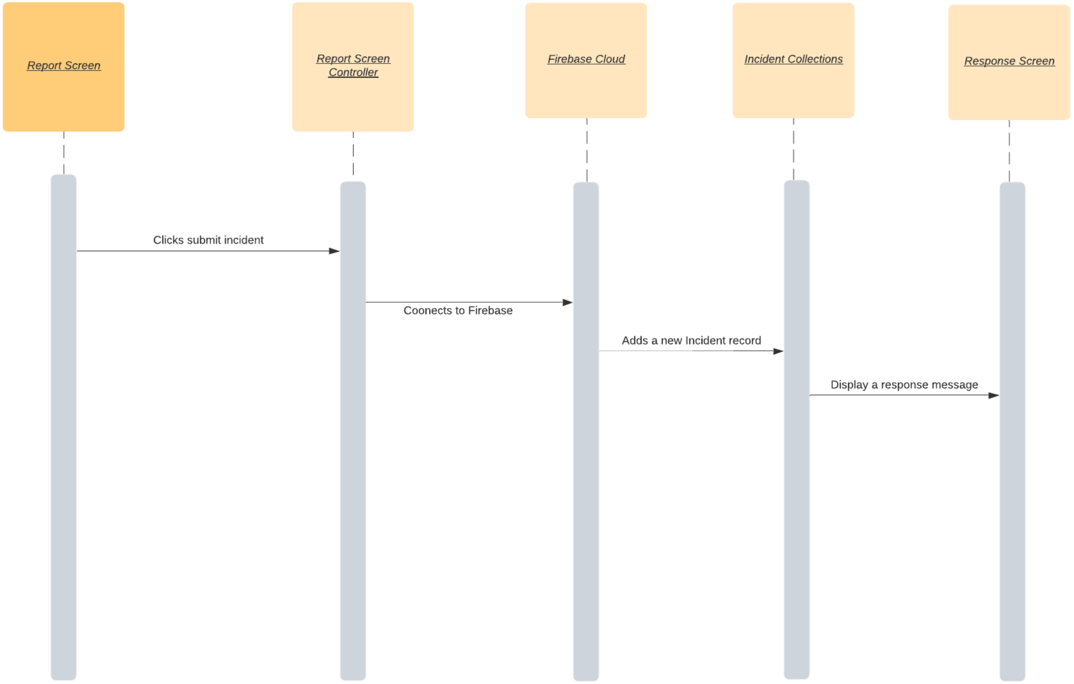
Show Incidents on Map Sequence Diagram



Show Incidents in Alerts Sequence Diagram



Submit an Incident Sequence Diagram



6.0 Human Interface Design

6.1 Overview of User Interface

The first feature the user will use is the Sign-up functionality. The user is presented with a form with two fields, an email field and a password field. If the form is submitted and one of the fields is empty, an error message is displayed underneath the missing field. The only requirement for an email is to have the @ sign, otherwise the user is presented with a “Failed to sign up” message. The password should be at least 6 characters, otherwise an error message is displayed, and the sign-up process fails. If both the email and password are accepted, the registration feedback screen is displayed, and the user can now return to the Sign-in screen.

The second feature the user will use is the Sign-in functionality. The user needs to enter the email and password that they used to Sign-up. If the email is not found in the Firebase Authentication database, a messaged saying “Failed to sign in” is displayed. Likewise, the same message is displayed if the password provided does not match the password in the database.

Once the user successfully signs in, they get access to the Map, EMS, Alerts, Report and Settings functionality. These screens can be accessed through the navigation bar located at the bottom of the screen.

On the Map screen, the user can see their current location displayed by a blue dot. This position is updated as the user’s map position changes. On the bottom of the Map screen is the navigation bar used to move between screens. On top of the navigation bar, there are two icons, an Alarm icon and a Dog icon. These icons provide the following functionality: The Alarm icon will generate a loud sound with the intention of gathering the attention of nearby people. It is meant to be activated by the user when they feel a sudden threat in their surroundings. The Dog icon provides a toggle functionality that displays/hides the current incidents on the map screen. An incident has 3 threat levels: A high threat is showed by a Red Dog pin. A medium threat is a Yellow Dog pin. Finally, a low threat incident is a Gray Dog pin. This visual colour information is meant to provide a quick idea of the severity of nearby threats.

The EMS screen is a placeholder for a future feature where the user can be allowed direct communication with emergency personal.

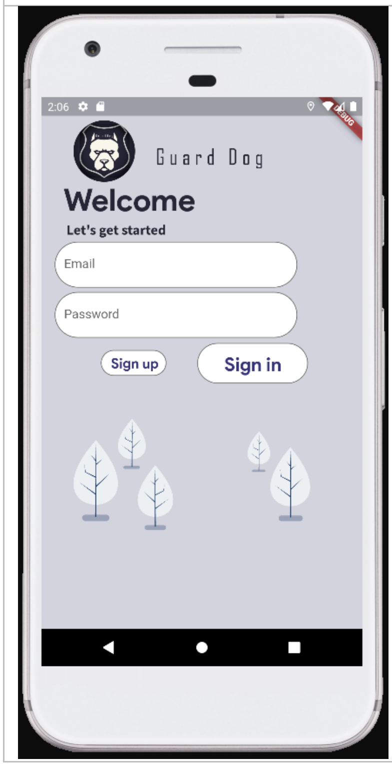
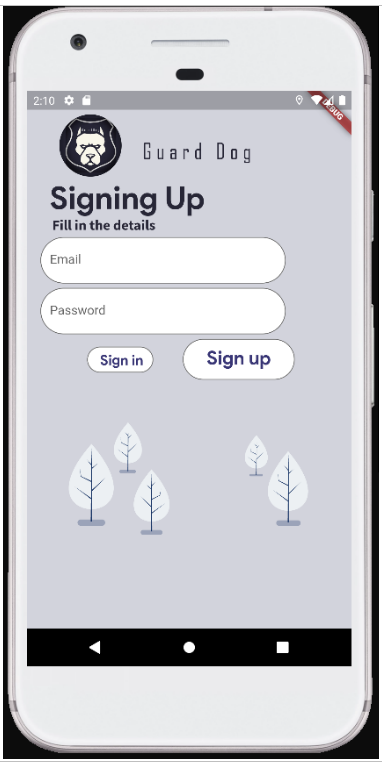
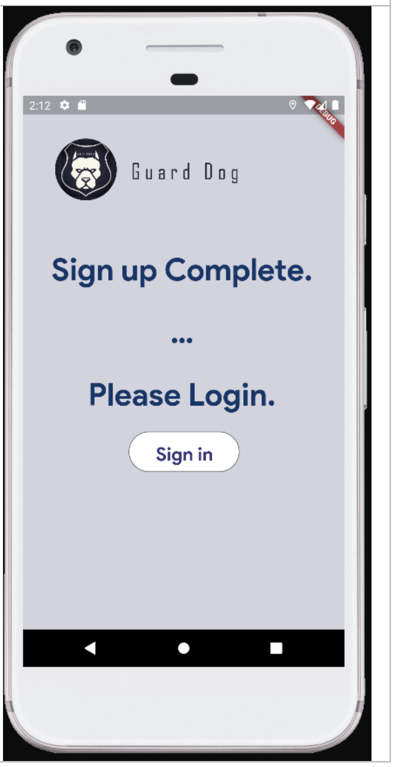
The Alerts screen functionality is to provide the user a list of the current threats that are displayed on the map. Each threat is colour coded with a Dog image and a rectangle of the same colour. The information displayed are the Danger Level, Description and Location. There is also a Refresh button that the user can use to reload the list to check for newly added incidents.

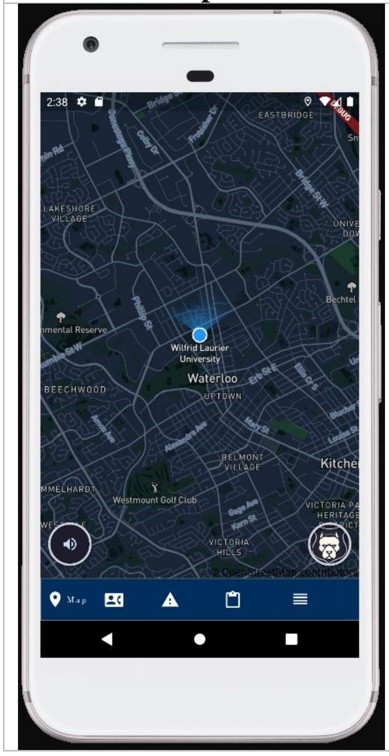
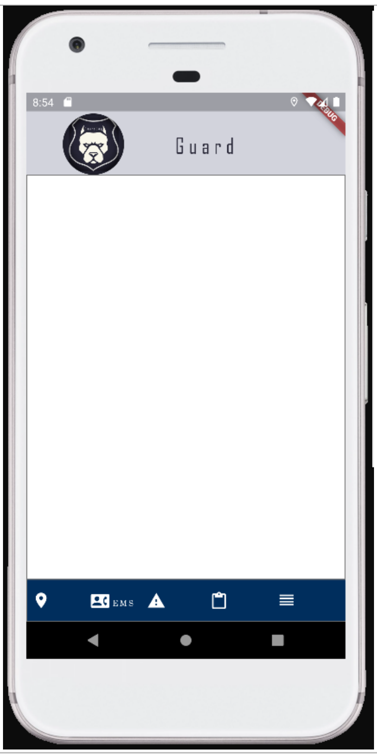
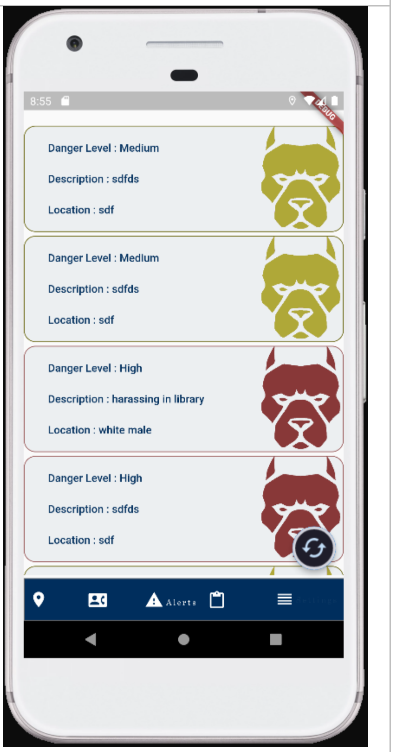
The Incident Report screen functionality is to provide the user with a form to submit an incident report. It is divided into two sections. First, we have the Preliminary Details. It includes fields for entering the first name, last name, age and relationship to victim of the user making the report. It also has a dropdown menu where the user can choose between High, Medium and Low levels of danger. The second section is the for the suspect and event description. The user can provide a description of the event, a description of the suspect’s facial, physical traits and clothing. They can also provide other details that they have deemed as important to be reported. Once all the sections are filled in, the user can use the Submit Report button to complete the form. If the threat level was a Low level, a “Stay Safe” message is displayed. For a Medium level threat, the message is displayed with a non-emergency phone number. All three threats levels give the user an Incident Reported feedback screen where they are given the option of returning to the Map screen. The Incident Report screen also has a cancel button in case the user wants to cancel a report they were making which takes them back to the Map screen.

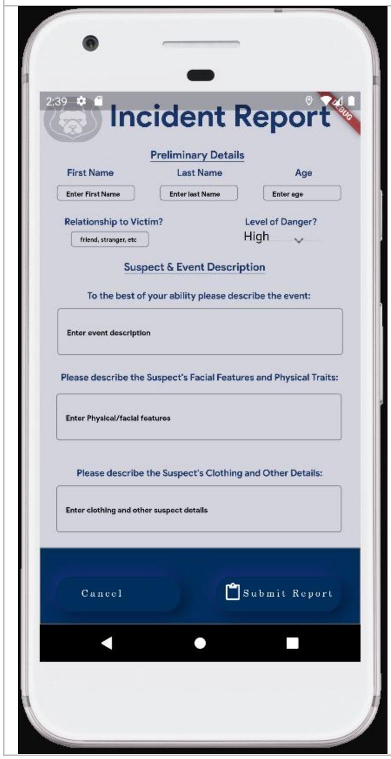
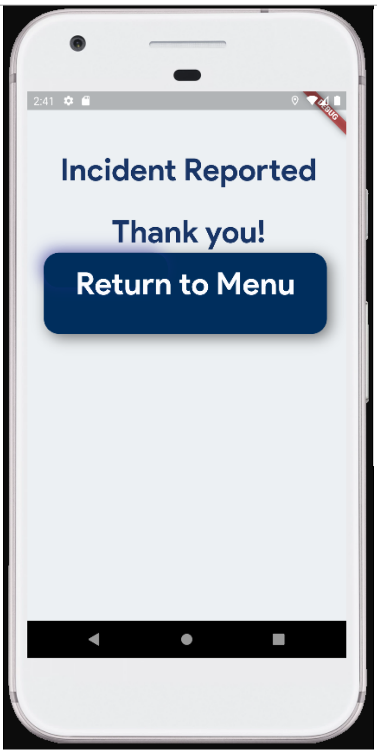
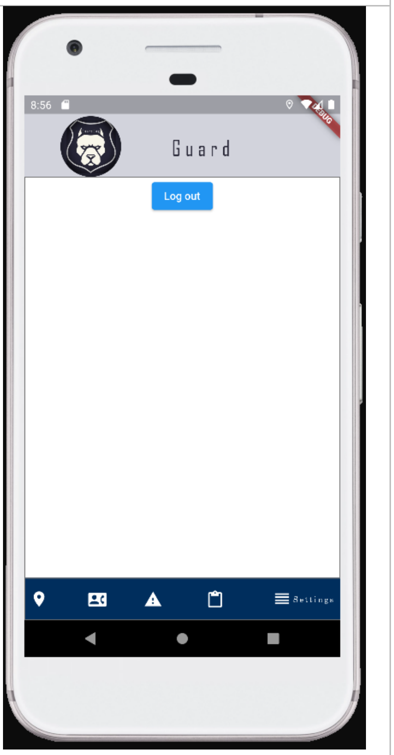
The Settings screen functionality is to provide a Log-out button. After pressing this button, the user will be taken to the Sign-in page concluding the user’s session.

6.2 Screen Images

The following are screen images of the Guard Dog mobile application running on a Pixel android emulator.

The Login screen	**The Registration screen**	**The Registration feedback screen**
		

The Map screen	The EMS screen	The Alerts screen
 2:38 EASTBRIDGE LAKESHORE VILLAGE Wilfrid Laurier University Waterloo UPTOWN BEECHWOOD BELMONT VILLAGE Westmount Golf Club VICTORIA HILLS Map		8:55 Danger Level : Medium Description : sdfds Location : sdf Danger Level : Medium Description : sdfds Location : sdf Danger Level : High Description : harassing in library Location : white male Danger Level : High Description : sdfds Location : sdf Alerts

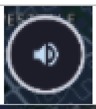
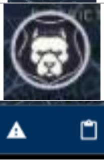
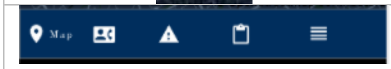
The Report screen	The Incident Reported screen	The Settings screen
2:39 Incident Report Preliminary Details First Name Last Name Age Enter First Name Enter last Name Enter age Relationship to Victim? friend, stranger, etc Level of Danger? High Suspect & Event Description To the best of your ability please describe the event: Enter event description Please describe the Suspect's Facial Features and Physical Traits: Enter Physical/facial features Please describe the Suspect's Clothing and Other Details: Enter clothing and other suspect details Cancel Submit Report	2:41 Incident Reported Thank you! Return to Menu	8:56 Guard Log out Settings

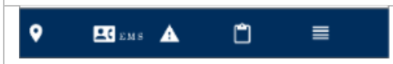
6.3 Screen Objects and Actions

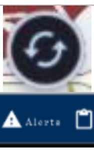
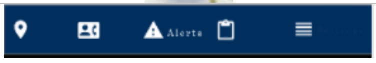
Login Screen	
Sign in	Allows users to sign into the application.
Sign up	Allows users to sign up to gain access to the application.

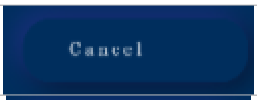
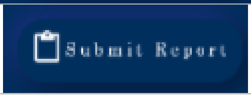
Registration Screen	
Sign up	Allows users to sign up to gain access to the application.
Sign in	Allows users to sign into the application.

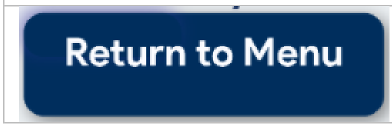
Registration feedback screen	
Sign in	Allows users to sign into the application.

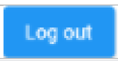
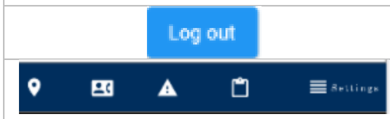
Map screen	
	Generates an audible distress sound.
	Hides or shows the previously recorded incidents on the map.
	Navigation bar that allows the user navigate to the following screens: • Map Screen, • Emergency Call Screen, • Alerts Screen, • Report Incident Screen, • Settings Screen.

EMS screen	
	Navigation bar that allows the user navigate to the following screens: • Map Screen, • Emergency Call Screen, • Alerts Screen, • Report Incident Screen, • Settings Screen.

Alerts screen	
	Check the database for any new incident alert and refreshes the list of incident on the screen.
	Navigation bar that allows the user navigate to the following screens: • Map Screen, • Emergency Call Screen, • Alerts Screen, • Report Incident Screen, • Settings Screen.

Report screen	
	Cancels the submission of an incident and redirects the user the map screen.
	Submits an incident to be stored in the database.

Incident reported screen	
	Redirects the user to the map.

Settings screen	
	Logs the user out of the application
	Navigation bar that allows the user navigate to the following screens: • Map Screen, • Emergency Call Screen, • Alerts Screen, • Report Incident Screen, • Settings Screen.

7.0 Reference

Flutter documentation. Flutter. (n.d.). Retrieved December 9, 2021, from https://docs.flutter.dev/.

Dart documentation. Dart. (n.d.). Retrieved December 9, 2021, from https://dart.dev/guides.

Welcome to the XD User Guide. (n.d.). Retrieved December 9, 2021, from https://helpx.adobe.com/xd/user-guide.html.

Google. (n.d.). *Firebase documentation*. Google. Retrieved December 9, 2021, from https://firebase.google.com/docs.

Commons , C. (2021). *Parse JSON in the background*. Flutter. Retrieved November 1, 2021, from https://docs.flutter.dev/cookbook/networking/background-parsing.

Commons, C. (2019, February 26). *Compute function*. compute function - foundation library - Dart API. Retrieved November 1, 2021, from https://docs-flutter-io.firebaseapp.com/flutter/foundation/compute.html.

Commons, C. (2020). *Floatingactionbutton class Null Safety*. FloatingActionButton class - material library - Dart API. Retrieved September 12, 2021, from https://api.flutter.dev/flutter/material/FloatingActionButton-class.html.

Commons, C. (n.d.). *Isolate class null safety*. dart. Retrieved November 9, 2021, from https://api.dart.dev/stable/2.15.0/dart-isolate/Isolate-class.html.

Dev, F. (2021, July 6). *Flutter_email_sender: Flutter Package*. Dart packages. Retrieved December 3, 2021, from https://pub.dev/packages/flutter_email_sender/install.

Dev, F. (2021, November 30). *Url_launcher: Flutter Package*. Dart packages. Retrieved November 9, 2021, from https://pub.dev/packages/url_launcher.

Rasmussen, M. (2020, July 10). *Creating objects and classes in Dart and flutter*. Dart Academy. Retrieved November 12, 2021, from https://dart.academy/creating-objects-and-classes-in-dart-and-flutter/.

Ryan, J. P. (2021, September 13). *Flutter_map: Flutter Package*. Dart packages. Retrieved December 9, 2021, from https://pub.dev/packages/flutter_map.

server6y, tl. (2020, September 26). *Flutter_map_location_marker: Flutter Package*. Dart packages. Retrieved November 9, 2021, from https://pub.dev/packages/flutter_map_location_marker.

Walrath, K. (2020, January 23). *Dart asynchronous programming: Isolates and event loops*. Medium. Retrieved November 9, 2021, from https://medium.com/dartlang/dart-asynchronous-programming-isolates-and-event-loops-bffc3e296a6a#:~:text=An%20isolate%20is%20what%20all,that%20runs%20an%20event%20loop.&text=Two%20isolates%2C%20each%20with%20its%20own%20memory%20and%20thread%20of%20execution.