Section 1 - Project Description

1.1 Project
Project SafeWalk

1.2 Description
SafeWalk is a mobile application designed to improve user safety during commutes, especially in unfamiliar or unsafe environments. By offering real-time location tracking, emergency notifications, and user-friendly interfaces, the application empowers individuals to feel more secure when walking alone.

1.3 Revision History

Date	Comment	Author
10/15/2023		Paramvir Toor

Team Members

Paramvir Toor

Julian Lamont Prater

Alfredo Gonzalez

Yengtaova Yuatongjerxiong

Mayuka Nozaki

Aaron Anthony Cantu

Contents

Section 2 - Overview

2.1 Purpose

Project SafeWalk is a mobile application designed to enhance user safety during commutes, particularly in unfamiliar or potentially unsafe environments. The application aims to provide real-time location tracking, emergency notifications, and user-friendly interfaces to make users feel secure when walking alone.

2.2 Scope

The project deliverables encompass the following:

- A database infrastructure to store user data and facilitate real-time communication with emergency contacts.
- Comprehensive user interface design focusing on simplicity, accessibility, and intuitive navigation.
- Provide geolocation notifications/updates with trusted contacts based on the user decisions.

2.3 Requirements

The requirements for this project are going to be based on functionality. Below you will find some estimates associated with each of them.

2.3.1 Estimates

#	Description	Hrs. Est.
1	UX.UI Design	4
2	Database	20
3	Front End Implementation	15
4	Login/Signup Feature	10
5	Login/Signup with Facebook	5
6	Location Sharing Feature	15
7	Emergency Contact	8
8	Panic Button	10
9	Phone Contact	10
	TOTAL:	97

2.3.2 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement

Id #	SRS Requirement	SDD Module
1	1.1 Product Scope	2.2 Scope
2	2 Functional Requirements	2.3 Requirements
3	3 External Interface Req	8 Other Interfaces
4	5 Definitions and acronyms	11 Glossary

Section 3 - System Architecture

The architecture of this project makes sure that the flow of data is continuous, performs at a high level, and is reliable while prioritizing user safety. The architecture is a combination of client-side applications, server-side processes, database management systems, and third-party integrations. You can expect to see client-side interfaces trigger backend database operations when applicable in the project.

Section 4 - Data Dictionary

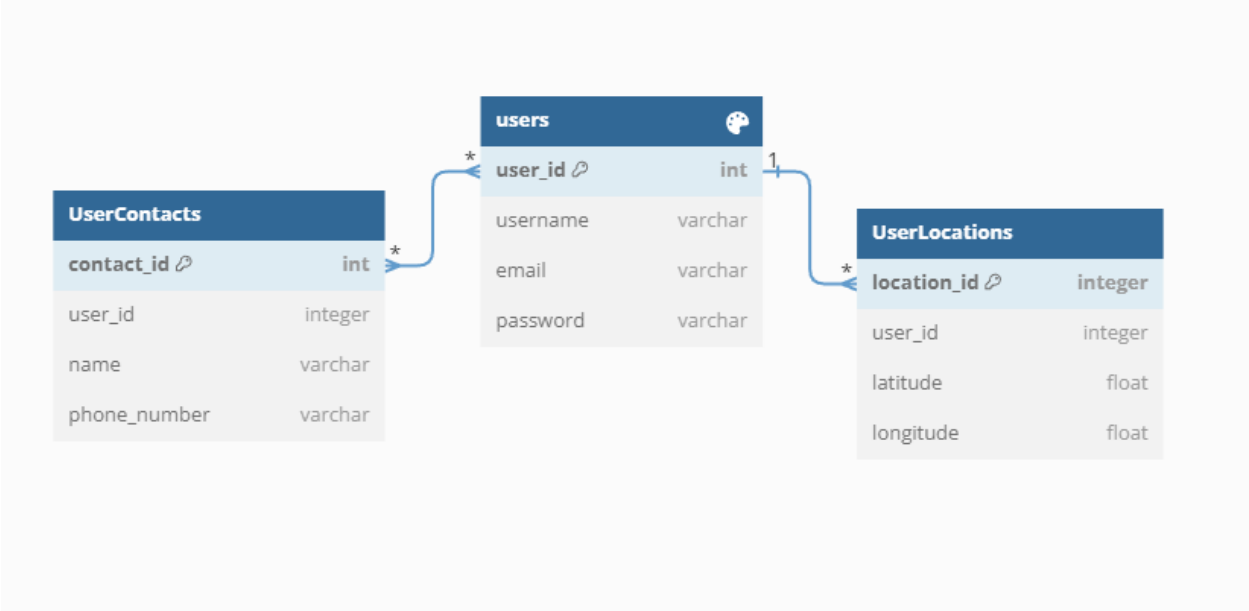
The data dictionary will provide detailed information about the data structures and the elements within the module. Essentially serves as a guide for developers and ensures that they have an understanding of data storage and usage.

Table		
Field	**Notes**	**Type**
user_id	Unique identifier for each user	INT
username	The username chosen by the user	VARCHAR
email	The user's registered email address	VARCHAR
password	The password created by the user	VARCHAR
contact_id	Unique identifier for each contact	INT
name	Contact's name	VARCHAR
phone_number	Contact's phone number	VARCHAR
location_id	Unique identifier for each location entry	INT
latitude	Latitude coordinate of the location	FLOAT
longitude	Longitude coordinate of the location	FLOAT

Section 5 - Software Domain Design

5.1 Software Application Domain Chart

Below is a representation of the software application domains and the relationship between objects. UML diagrams help visualize the architecture and flow within the system.



5.2 Software Application Domain

Within the project SafeWalk, the software application domain is divided into a number of smaller domains that each include a variety of functions. Each domain contains a family of elements that team up to bring functionality.

5.2.1 Domain X

This domain handles everything related to user accounts, including registration, login, profile management, and user settings. It integrates with the database domain to securely store user data.

5.2.1.1 Component Y of Domain X

The component ensures that users are who they claim to be. It also bounds the login and registration procedures, password recovery, and two-factor authentication.

Data Flow

1. User submits a login/registration form.
2. Data is sent to the server.
3. Password encryption before storage or comparison.
4. User authentication status is sent back to the client.

5.2.1.1.1 Task Z of Component Y1 of Domain X

This task Z involves accepting the user's credentials, validating them with the stored data, and return either an authentication message or an error message.

Data Flow

1. Retrieve 'username' and 'password' from client.
2. Use a function that encrypts the password.
3. Validate the credentials against the 'Users' table.
4. If valid, return the authenticated message, else return an error.

Section 6 – Data Design

For Project SafeWalk, here is the design of the data and their relationship with the different functionalities and external dataset that are in place.

6.1 Persistent/Static Data

The following are descriptions of persistent and static datasets with Project SafeWalk:

6.1.1 Dataset

Relationship with trusted/emergency contacts, relationship between security of the database.

6.1.2 Static Data

User preference settings, Geographical locations, Error messages and notifications, Emergency/Trusted contact list.

6.1.3 Persisted data

User account information, Location data, App logs, Activity history.

6.2 Transient/Dynamic Data

Real time status updates, User notifications, Emergency notifications, Session information (meaning when the user initiates a walk within the app).

6.3 External Interface Data

Social media account linking, location Ionic based services.

6.4 Transformation of Data

Geolocation coding, formatting of dynamic data for notifications.

Section 7 - User Interface Design

7.1 User Interface Design Overview

Some of the high level requirements for Project SafeWalk include, but are not limited to:

- User Safety Features
- User registration and Authentication
- Contact Management

- Location Sharing
- User-friendly interface
- Database Management
- Integration with Firebase or other External Services
- Responsive Design
- Compatibility Security

7.2 User Interface Navigation Flow

Diagram the flow from one screen to the next

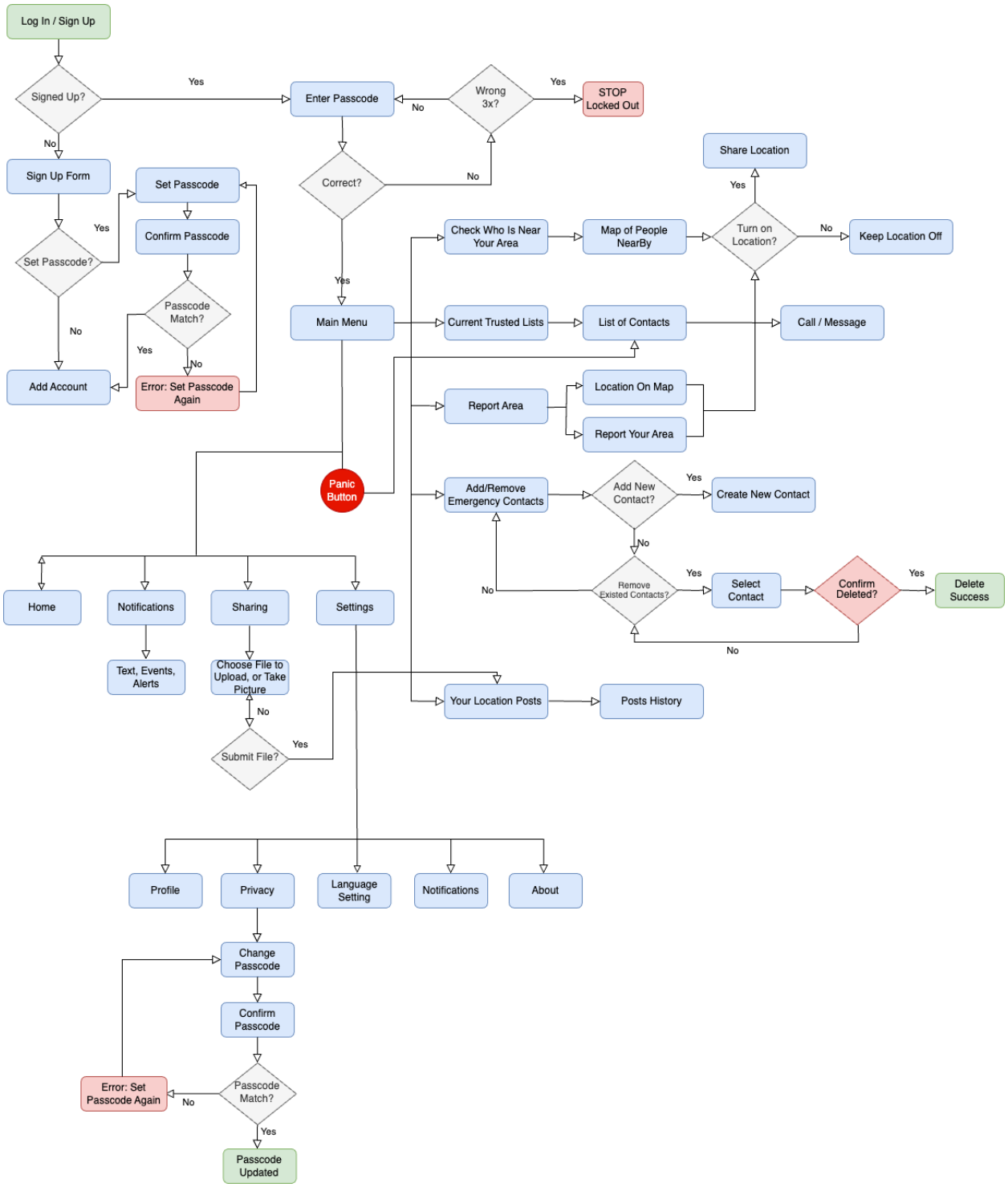


Figure 7.2.1: A ER diagram of how a user interacts and the path that will be followed.

7.3 Use Cases / User Function Description

When using Project SafeWalk, the user can expect to utilize the app's functions in a predictive manner. The following use cases will serve as examples.

1. *User Registration and Login:*
    a. **Description:** A new user wants to utilize the features of SafeWalk. They navigate to the app and register with their email or social media account. They then log in to access the app's features.
    b. **Actors:** User, System.
    c. **Flow:** For login, The user will enter their credentials being an email and password and the system will process their credentials on the back end database and verify if it is an existing user or not. For Sign Up, the user will enter their first and last name, email, username, and password. This will then add them to the database on the backend.
2. *Adding/Removing Emergency Contacts:*
    a. **Description:** A user wants to set up their emergency contacts in the SafeWalk app.
    b. **Actors:** User, System.
    c. **Flow:** User accesses the settings menu, selects the emergency contacts option, and inputs the contact details. The system validates and saves the contact information for future use.
3. *Sharing Real-Time Location:*
    a. **Description:** A user feels unsafe during their commute and wants to share their real-time location with their emergency contacts. They open the app and initiate the location-sharing feature.
    b. **Actors:** Emergency Contacts, User, System.
    c. **Flow:** The user activates the location-sharing feature, the system retrieves the user's location, and the app sends this information to the saved trusted contacts.
4. *Panic Button:*
    a. **Description:** In a potentially dangerous situation, a user presses the panic button to send an immediate alert to their emergency contacts. The app also notifies local authorities with the user's location.
    b. **Actors:** Emergency Services, User, System.
    c. **Flow:** User activates the panic button, the system sends signals to the emergency contacts, and also alerts local authorities with the user's real-time location for immediate assistance.

5. *General User Interface Navigation:*
    a. **Description:** A user wants to explore different features within the SafeWalk app.
    b. **Actors:** User, System.
    c. **Flow:** User interacts with the app interface, navigating through different sections using menu options, buttons, and links to access desired features and information.

Section 8 - Other Interfaces

External interfaces used in the execution of this module include;
Firebase Authentication API (Firebase SDK) that handles user registration and sign-ins.
Firebase Cloud Firestore Database API (Firebase SDK) used to store and retrieve user data, including contacts and locations.
SMS Gateway API (SMS Gateway Provider) used to send SMS alerts to contacts during emergencies.

8.1 Interface X

Using node.js, the library express helps with http requests. Express can handle get and post requests. Middleware libraries like cors, jwt, and body-parser help with getting formats and tokens established. Another library used is firebase admin. With firebase admin we can connect to the firebase api by configuring our secure tokens given by the firebase store database. Firebase handles its database as documents and collections. Documents are json formated. Collections are like folders and can store documents. You can add collections to documents, and access those document variables to get the collections. We have asynchronous functions that are ready to receive get and post requests. When a request is received with a post, the body parser grabs values from the front end. Depending on the post and data received, the firestore api will interact with the data and add or update data to the database. Same with get requests, it will send an http response back to the front end, sent in json format and include cors to successfully respond. Express and middleware will handle json formats back to the client. When working with the firestore database, the api receives data as a promise, we can take the snapshot of that data and find out whether something exists in the database. With promises, we can also check if we have an error, and respond with the error info back to the client as a response. When a login is established, an http response will send a JWT(json web token) as a response so the client can insert it into the header. Any time this user wants to access private information, the backend will check the request and make sure the header includes the right token.

Section 9 - Extra Design Features / Outstanding Issues

Extra features would include an emergency beacon feature that allows users to send out an emergency beacon to all and any nearby SafeWalk users in the vicinity. This feature would be useful for situations where immediate assistance is needed and there is no time for proper authorities to arrive.

Section 10 – References

Firebase Documentation provides details on how to securely and efficiently use Firebase for backend operation.
Ionic Framework Documentation helps understand the framework's capabilities and limitations.

Section 11 – Glossary

API (Application Programming Interface): A set of rules that allows different software entities to communicate with each other.
DBMS (Database Management System): Software system that uses a standard method to store and organize data.
GPS (Global Positioning System): Satellite-based navigation system.
RTLS (Real-time Location System): Systems that automatically identify and track the location of objects or people in real time.
UML (Unified Modeling System) Diagram: Standardized modeling language enabling developers to specify, visualize, construct, and document the artifacts of software systems.
ER (Entity-Relationship) Diagram: A graphical representation that depicts the entities involved, and the relationships between these entities.
JWT (JSON Web Token): A token that defines a compact and self-contained way for securely transmitting information between applications as a JSON object.