Helping Hands

Emergency Help and Safety System

Guided By:

Prof. Abdul-Rahman Mawlood-Yunis

Developed By:

225817200 Ayush Patel
225817210 Rushi Parmar
225816420 Malay Patel
225817220 Smit Panchal
225817180 Kishan Patel

1. Abstract

- Modern societies are confronted with an increasing number of abnormal events, crises, disasters, and accidents. These types of threats can put a person in an emergency.
- To overcome these situations, there arises a need for an Emergency Help and Safety System. Helping Hands is built for the same reason.
- Helping Hands is an android application that increases the chance of help in case of an emergency by notifying nearby volunteers.

2. Introduction

- The increasing crime rate in today's society and unpredictable routine leads to a very insecure daily life, especially for those living alone and away from their family.
- There must be a handy application so that users can help each other in need. Helping hands does the same work through which users can get help from complete strangers.
- Features of Helping Hands:
    - Users can contact government emergency services (i.e. police/medical/fire services)
    - In case of an emergency, the user can trigger an SOS alert which will notify user-selected contacts, and the live location of the user will be sent to them every 50 seconds.
    - Users can see available volunteers within the range of the current location.
    - Users can broadcast help requests which will notify nearby volunteers and volunteers can accept the help request to provide help.
    - The application also supports User Account Management

3. Project Scope

In 2 months of time, 5 graduate students of Wilfrid Laurier University create an Emergency Help and Safety android App that allows users to contact government services, emergency contacts, or nearby volunteers.

4. Tentative Schedule

Key Milestone	Deadline
Form the project team, Finalize the project plan and Documentation	October 2
Basic UI, Login, Signup, Logout, Edit Profile, Call & Text (Iteration One)	October 31
Generate Broadcast, Map segment, Requests segment, Notification modules and Instrumentation Testing (Iteration Two)	December 2
Presentation Slides, Final Project Report	December 7

5. Technology/Tools

- Android Studio
- Cloud Firestore
- Google Maps API Platform
- Java
- XML

6. Software requirement specification:

R.1 Managing User Profile

R.1.1 Create User Profile

Description: Users can register themselves.

Input: Basic Personal Details

Output: Generated user profile

R.1.2 Deactivate User Profile

Description: Users can deactivate their profile.

Input: User selection

Output: Success message

R.1.3 View User Profile

Description: User can view their profile.

Input: User selection

Output: Appropriate User Profile

R.1.4 Update User Profile

Description: Users can update basic profile details.

Input: User Data

Output: Success message and show updated profile

R.2 System forms and Displays GeoLocation Map

R.2.1 System Displays User's Current Location

Description: Show current Location of user in Map

Input: Access Device's Location

Output: Pinned Point Location on Map

R.2.2 System Gets nearby Volunteer's Location

Description: Show Location of Available Volunteer in Map

Input: User Selection

Output: Display Location of Volunteer in Map

R.2.3 System shows nearby Emergency requests of other users

Description: Show Location of nearby incoming request

Input: User Selection

Output: Display Location of requests

R.3 System generates and broadcasts Emergency Help Requests

R.3.1 Generate Help Signals

Description: System Generates Emergency Signals

Input: User selection or Gestures

Output: Emergency Signal data generated on User's Devices

R.3.2 Broadcast Help Signals

R.3.2.1 Broadcast Critical Emergency

Description: System Broadcasts Emergency request to nearby Volunteers and user selected contacts

Input: User Selection

Output: Emergency data sent to nearby volunteers and Emergency Contacts

R.3.2.2 Broadcast Intermediate level Emergency

Description: System Broadcasts Intermediate level Emergency signals and notify user selected contacts

Input: User Selection

Output: Emergency data sent to nearby volunteers and Emergency Contacts

R.3.3 Contact Government Services

Description: System contacts Government Emergency Services

Input: User Selection

Output: Appropriate Government Services are alerted

R.4 System allows Users to respond to Incoming Requests

R.4.1 Receive Emergency Requests

Description: Emergency requests are received and displayed

Input: Incoming emergency request

Output: Display respected request and appropriate message

R.4.2 Accept/Reject incoming Requests

Description: User is given option to accept or reject the request

Input: User selection

Output: Appropriate message is sent

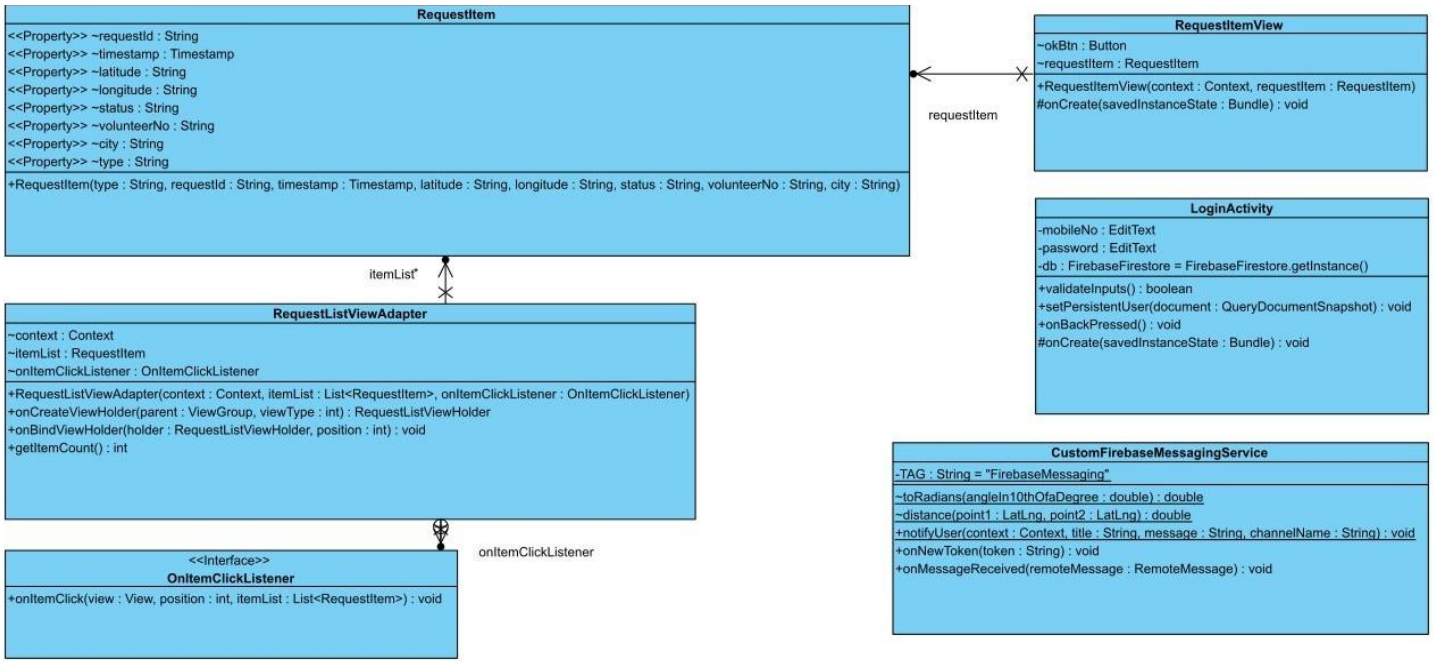
R.4.3 Track Accepted Request

Description: System tracks the requests that are accepted by volunteers

Input: Acceptance of request

Output: Display track of the volunteer

7. Class diagram:



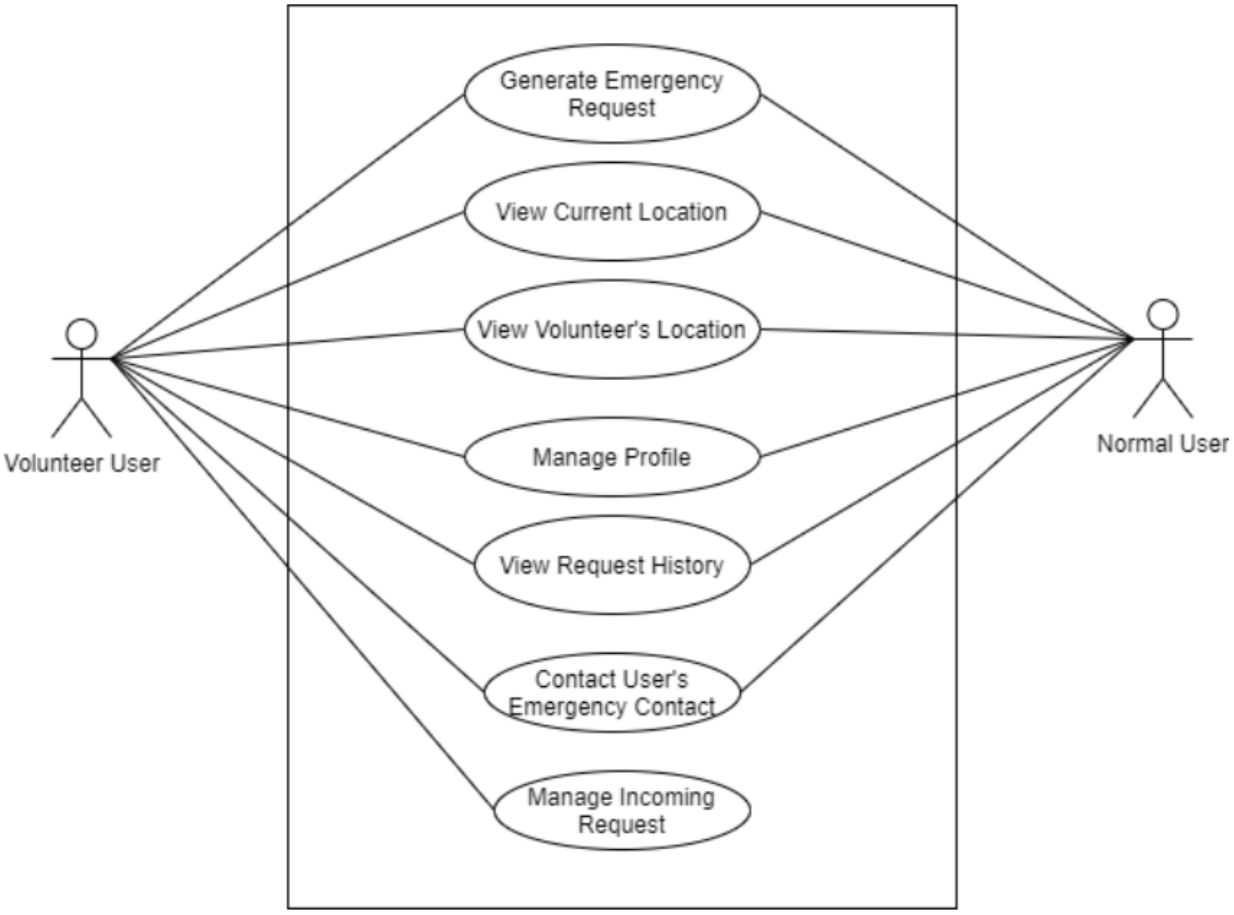
- **Description of important classes:**

1. User
    - Class that represents the current user.
    - Includes attributes and getter-setter methods to store all the necessary details of current users in SharedPreferences.
2. Volunteer
    - Class that represents the volunteer users who are in the nearby area.
    - Includes attributes and getter-setter methods to store all the necessary details of the available volunteers in the area.
3. Requester
    - Class that represents the requester users who have initiated the emergency and who are in the nearby area.
    - Includes attributes and getter-setter methods to store all the necessary details of the possible requester in the area.
4. LoginActivity
    - Used for logging in to the application by validating user-provided account credentials with the database.
    - Creates a session on successful login in SharedPreferences and saves the details of the logged-in user.
5. SignUpActivity
    - Used for creating a new user account.
    - Validates the user-provided necessary profile details and saves all the data in the database. Also logs in the user automatically after a successful sign-up.
6. EditProfileActivity
    - Used for editing user profile details.

- Validates user-provided changes to the profile data and updates the database accordingly.

7. EmergencyContactsActivity
    - Used for validating and saving emergency contacts of the user.
    - User can provide 3 emergency contacts with the relation with them.
8. MainActivity
    - Main root activity once the user is logged in.
    - Holds all of the fragments. Ask for the required app permission.
    - Fetches the live user location and monitors the location updates.
9. HomeFragment
    - Includes implementation of 3 buttons for the following actions:
        a. makeCall: Makes a call to government services
        b. alertContacts: Starts a new thread to call the sendSMS function to text the emergency message with the current location of the user. Keeps sending the live location to the selected emergency contacts until the user stops SOS Alert using the same button.
        c. initiateEmergency: Start an emergency broadcast to notify all the nearby volunteers by a notification about the user emergency. Runs in the background.
10. MapFragment
    - Integrated google map which shows the live current location of the user.
    - Important Function of the MapFragment:
        a. fetchCurrLocation: Fetches the current location of the user using the device's GPS service
        b. discoverVolunteers: Runs on a thread to discover nearby volunteers and updates the volunteer marker on the map.
        c. discoverRequesters: Runs on a thread to discover nearby requesters if the current user is a volunteer and updates the requester marker on the map.
        d. setMarker: Sets marker of the location on the integrated google map of all three: volunteers, requesters and the current user.
11. RequestsFragment
    - Uses recycler view to display the lists of all the emergency requests that the user has generated or responded to based on user type.
    - Provides an option to mark requests as resolved.
12. ProfileFragment
    - Used for the settings tab.
    - Provides actions for logging out, and deactivating the user account.
    - Also redirects to edit profile activity.
13. CustomFirebaseMessagingService
    - Listener for FCM (Firebase Cloud Messaging) which is used for detecting push notifications provided by Firebase.

8. Use-case diagram:



- **Actors / Type of users:**

    1. Normal Users: Normal users are the user who can use the app just to generate the emergency whenever they need help.
    2. Volunteer User: Volunteer users are users who are willing to help others in their emergency situations. Therefore they can generate the emergency as well as accept incoming help requests to provide help to other users.

- **System Overview:**

    1. Contact Government Services:
        - Both types of users (Normal & Volunteer) can contact government services by phone call using a single button in the app when they are in an emergency.

    2. Generate Emergency Request:
        - Both types of users can generate an emergency broadcast request which will notify all the volunteers residing in the area of the user by a notification.

- The live location of the user will be shown to all the nearby volunteers in the integrated google map.
- Volunteers have the option to accept the help request.

3. View Current Location:
    - Both types of users can view their live location in the Google map which is integrated into the application.

4. View Volunteer's Location:
    - Both types of users can view the live location of volunteers which is located near their location in the integrated Google map.

5. Manage Profile:
    - Both types of users can perform actions for following Profile Management Tasks:
        a. Edit Profile Details
        b. Edit Emergency Contacts
        c. Change Password
        d. Logout
        e. Deactivate Account
        f. Access the Help & Feedback section

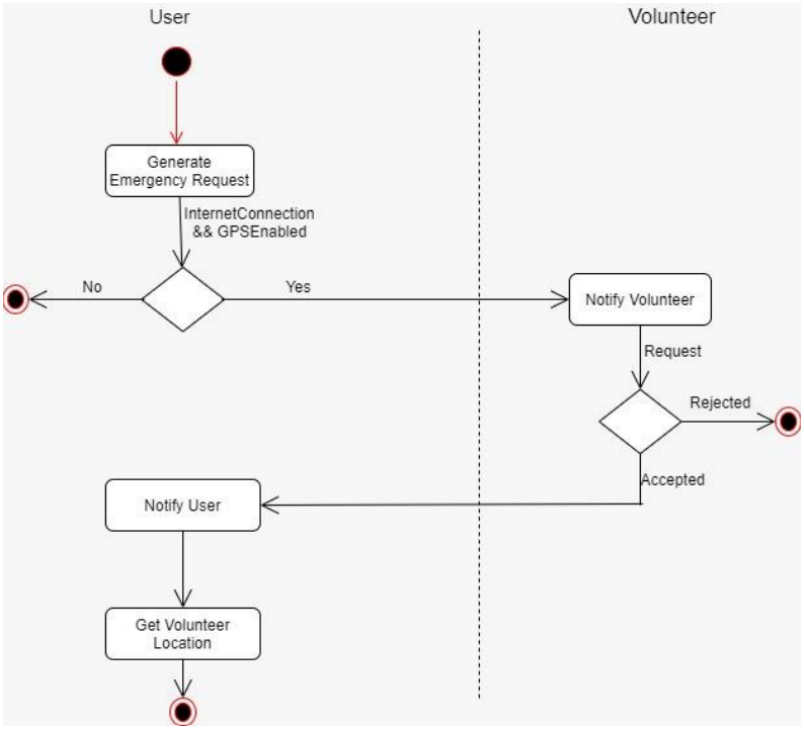
6. View Request History:
    - Both types of users have the option to view the request history using the requests tab.
    - All the details of the requests such as origin location, time and date, the status of the request, an ID of the assigned volunteer, and a unique request ID is shown in the Requests History tab.

7. Contact User's Emergency Contacts:
    - Both types of users can start an SOS alert to contact their emergency contacts via SMS.
    - The SOS alert will keep sending an SMS to the emergency contacts every 5 seconds until the user shuts off the SOS alert from the application.
    - The SMS message contains the live location of the user who is in an emergency.

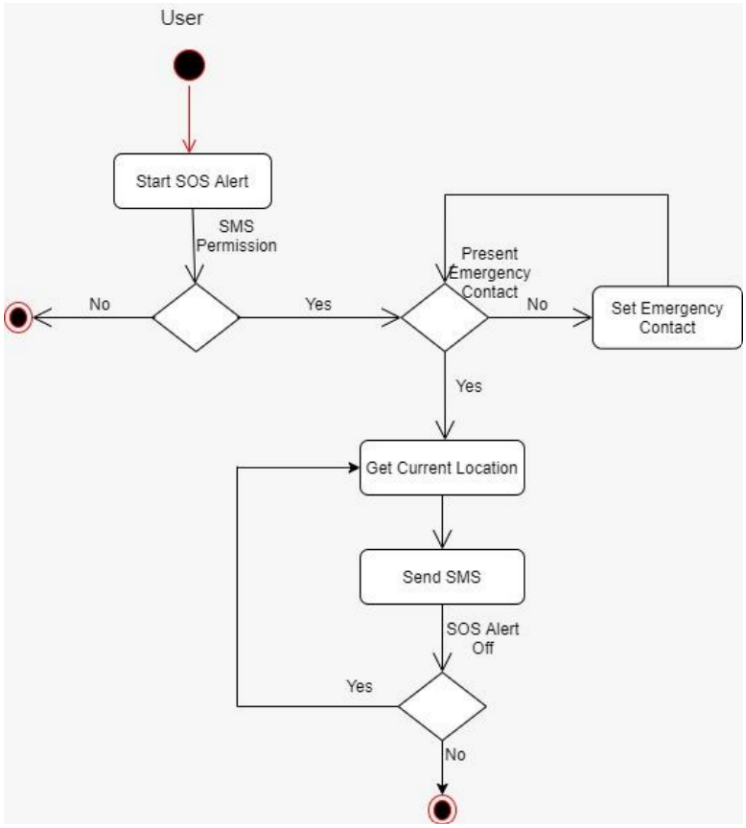
8. Manage Incoming Requests:
    - Only volunteer users can manage incoming requests.
    - Volunteers have the option to accept incoming help requests.
    - If any volunteer accepts the incoming requests, the user in an emergency will get notified and the assigned volunteer will be shown on the user's map in a different icon.

9. Activity Diagram:

a. Start SOS Alert:



b. Broadcast Emergency Request:



10.Testing:

- For manual testing of the System, a mixed approach comprised of Integration Testing and Regression Testing is used.
- Every module is tested using Integration Testing where we test the unit parts and then combine those unit parts to form a module.
- For example, while creating Map Module, first basic functions like getCurrentLocation, getVolunteers and getRequestor are tested, then the Map Module is Created and Tested if it works properly.
- After any module is created, now by the rules of Regression Testing, this module is added to the whole system and then the whole system is tested and it is made sure that the whole system works as desired after adding the module to the system.
- For example, after successfully creating Map Module, it is added to the system and then the whole current system is tested to ensure it is properly working.
- For automation testing, instrumentation testing of Android Studio was implemented using Espresso.

11.Limitation and future extension:

- Limitation: Background processes can be handled more efficiently. Process load due to concurrent thread execution can be reduced.
- Future Extension: The user interface will be improved to provide better interaction with the system. Background processes and Thread execution will be efficient and the processing load will be reduced.

12. Conclusion:

- With the successful deployment of Helping Hands, the team of volunteers will be ready to help those in need whenever required.
- At the time of emergency, users will be able to broadcast emergency signals, the application would send information to the selected contacts of users to get help and the local government services could also be informed.
- Hence saving many lives and helping countless people, helping hands will make sure you get help on time.

13. Bibliography:

- Book:
    - Head First Android Development: by DAVID GRIFFITHS and Dawn Griffiths
- Websites:
    - https://developer.android.com/docs
    - https://www.tutorialspoint.com/android/android_studio.htm