

CHAPTER 1- INTRODUCTION

Here we introduce an app which ensures the safety of women. This helps to identify and call on resources to help the one out of dangerous situations. These reduce risk and bring assistance when we need it and help us to identify the location of the one in danger. While in emergency the user would have to press the signal (SOS) button, it will automatically get generated from the user end and send SMS to those contacts which has been saved at the time of registration.

1.1 AIM:

The primary aim of women safety applications is to enhance the personal security and confidence of women by providing tools and resources to address unsafe situations effectively. These apps often include features like panic buttons, real-time location sharing, and direct alerts to emergency contacts enabling swift responses in critical moments. They also offer safe route suggestions, live tracking, and community support options to help users navigate their surroundings safely. By facilitating instant communication and assistance, these applications act as a safety net, reducing the risk of harm and ensuring that help is readily available when needed.

1.2 ABOUT PROJECT:

The women's safety application project is a technology-driven initiative aimed at enhancing the safety and security of women in various environments. This application is designed to provide a range of features, including emergency assistance through panic buttons, real-time location tracking, and instant alerts to trusted contacts or law enforcement. Users can also access features like safe route navigation, live monitoring, and SOS signals to ensure timely help in distress situations. The app aims to bridge the gap between technology and safety by offering tools that are easy to use, reliable, and effective in both urban and rural settings.

1.3 OBJECTIVE

The primary objective of a woman's safety application is to empower women by providing tools to enhance their personal safety and security in real-time. It aims to create a platform that enables quick communication with trusted contacts, emergency services, or authorities during distress through features like SOS alerts, live location tracking, and instant notification

CHAPTER 2- LITERATURE SURVEY

2.1 EXISTING SYSTEM:

The existing system of women safety applications typically focuses on providing emergency assistance and safety features to address threats effectively. These applications often include a panic button that sends instant distress signals to pre-selected contacts, local authorities, or nearby users. Real-time location tracking is a common feature, allowing users to share their live location with trusted contacts for monitoring their movements. Additionally, some apps provide safe route navigation to help women avoid risky areas, along with geo-fencing alerts to notify when someone enters or exits a designated safe zone. While these systems are helpful, they also have certain limitations. Many existing applications require a stable internet connection or GPS signal, which may not be accessible in all areas.

2.2 PROPOSED SYSTEM:

The proposed system for a women's safety application aims to overcome the limitations of existing systems by integrating advanced technologies and offering a more comprehensive set of features. It will also incorporate real-time location tracking that updates automatically and allows users to share their live location even in low-connectivity areas. Moreover, the app will connect users to local support networks and help lines while ensuring data privacy and encryption to protect user information.

2.3 FEASIBILITY STUDY:

The main objective of the feasibility study is to test the technical, economical and operational feasibility of developing a computer system. This is done by investigating the existing system in the area under investigation and generating idea about a new system.

2.3.1 Technical feasibility:

The technical feasibility of a women safety application lies in leveraging the robust technological infrastructure available today. Modern smart phones are equipped with essential features such as GPS, accelerometers, internet connectivity, and advanced sensors, which can be used for real-time location tracking, safe route navigation, and emergency alert systems. These hardware capabilities, combined with widely accessible mobile networks and internet services, make the development and functionality of such applications achievable.

Additionally, incorporating offline features like SMS-based alerts ensures usability even in areas with limited network coverage, enhancing the app's reliability in diverse scenarios.

2.3.2 Economic feasibility:

The economic feasibility of a women safety application evaluates the costs of development, deployment, and maintenance against the potential benefits and revenue opportunities. Additional costs include marketing, training, and customer support to ensure widespread adoption and user satisfaction. However, the development costs can be managed by leveraging existing technologies, open-source tools, and scalable platforms, making the initial investment more affordable. Furthermore, the growing demand for personal safety applications in urban and rural areas ensures a large potential user base, making it a viable investment. By offering a combination of social impact and financial sustainability, a women safety application can be economically feasible while addressing a critical societal need.

2.3.3 Operational feasibility:

The operational feasibility of a women safety application focuses on its practicality and ability to function effectively in real-world scenarios. It involves evaluating whether the app can be easily integrated into users' daily routines and whether it can be supported and maintained over time. The application should be user-friendly, requiring minimal training or technical knowledge, ensuring it can be easily adopted by women of different age groups and technological proficiency. The app's functionalities, such as panic buttons, real-time location tracking, and emergency alerts, must work seamlessly in both urban and rural environments with varying network coverage.

CHAPTER 3- TOOLS REQUIREMENTS

3.1 HARDWARE REQUIREMENTS:

1. RAM: 8GB or higher
2. Hard Disk: 20GB or higher
3. OS: Windows 10 or higher

3.2 SOFTWARE REQUIREMENTS:

1. Software: Android Studio
2. Programming Language: Java Script
3. Backend: Shared preference

CHAPTER 4 -TECHNOLOGIES USED

4.1 INTRODUCTION TO SOFTWARE:

Android Studio is the official integrated development environment (IDE) for developing Android applications. It is developed by Google and based on JetBrains' IntelliJ IDEA. Launched in May 2013, it offers a range of tools and features tailored specifically for Android development.

Key Features:

1. Code Editor: A powerful editor with smart code completion, refactoring, and linting tools.
2. Layout Editor: A visual design tool for building user interfaces with drag-and-drop functionality.
3. Gradle Build System: A flexible build system for managing dependencies and build configurations.
4. Android Emulator: A built-in emulator for testing apps on virtual devices.
5. Debugging Tools: Advanced debugging features, including logcat, breakpoints, and device profiling.

Supported Platforms:

Android Studio is available for Windows, macOS, and Linux. It requires the Java Development Kit (JDK) and Android SDK to function.

It is widely used by developers worldwide for building Android applications due to its robust features and seamless integration with the Android ecosystem.

Advantages of Android Studio:

1. Official Support: As the official IDE from Google, Android Studio receives regular updates and support, ensuring compatibility with the latest Android features and tools.
2. User-Friendly Interface: Offers a clean and intuitive UI, making it easier for developers to navigate, design, and manage their projects.
3. Rich Code Editor: Provides intelligent code completion, real-time code analysis, and refactoring, which improve coding efficiency and reduce errors.

4.2 INTRODUCTION TO PROGRAMMING LANGUAGE:

JavaScript is a high-level, dynamic, and versatile programming language primarily used for creating interactive and dynamic content on websites. Developed by Brendan Eich in 1995, it has become one of the core technologies of the web, alongside HTML and CSS.

Key Features:

1. **Client-Side Execution:** Runs directly in web browsers, enabling interactive web pages without requiring server-side processing.
2. **Dynamic Typing:** Variables can hold any type of data without strict type declarations.
3. **Event-Driven:** Supports event handling, making it suitable for interactive user interfaces.
4. **Lightweight and Flexible:** Ideal for adding features like animations, form validation, and dynamic content.
5. **Cross-Platform:** Works across all modern browsers and operating systems.

Uses of JavaScript:

1. **Web Development:** Enhances websites with interactivity, animations, and dynamic content updates.
2. **Backend Development:** Through environments like Node.js, JavaScript can also be used for server-side programming.
3. **Mobile App Development:** Frameworks like React Native enable building mobile apps.
4. **Game Development:** Used in 2D and 3D game development with libraries like Three.js.

Advantages:

Easy to Learn: Its syntax is simple and beginner-friendly.

Fast Execution: Runs directly in the browser, reducing server load and latency.

Extensive Ecosystem: Supports numerous frameworks and libraries, such as React, Angular, and Vue.js.

JavaScript has evolved significantly and now powers a wide range of applications, making it one of the most widely used programming languages worldwide

4.3 INTRODUCTION TO BACKEND:

In Android Studio, the Shared Preferences Library is a simple and lightweight mechanism for storing and retrieving small amounts of key-value data. It is primarily used for storing user preferences or app settings, such as user login status, theme selection, or configuration options.

Key Features:

1. Key-Value Pair Storage: Data is stored as a combination of keys and their corresponding values.
2. Persistent Storage: The data persists across app sessions until explicitly removed.
3. Simple API: Easy-to-use methods for storing, retrieving, and deleting data.
4. Private or Shared Access: Data can be restricted to the app or shared across applications.

Use Cases:

Saving user preferences (e.g., dark mode, language settings).

Storing small configurations (e.g., token IDs, last viewed page).

Keeping light weight session data.

Example Usage:

Here's how Shared Preferences works in Android:

1. Saving Data:

```
SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",  
MODE_PRIVATE);
```

```
SharedPreferences.Editor editor = sharedPreferences.edit();
```

```
editor.putString("username", "JohnDoe");
```

```
editor.putInt("userAge", 25);
```

```
editor.apply(); // Save changes asynchronously
```

2. Retrieving Data:

```
SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs",  
MODE_PRIVATE);
```

```
String username = sharedPreferences.getString("username", "DefaultName");
```

```
int userAge = sharedPreferences.getInt("userAge", 0);
```

3. Removing Data:

```
SharedPreferences.Editor editor = sharedPreferences.edit();
```

```
editor.remove("username"); // Remove specific key
```

```
editor.clear(); // Remove all data
```

```
editor.apply();
```

Advantages:

Light weight and easy to use.

No need for a database for small data storage.

Data remains intact even if the app is closed or restarted.

Limitations:

Not suitable for large or complex data structures.

Data is stored in XML format, which may not be as secure for sensitive information (use encryption for sensitive data).

Shared Preferences is ideal for basic configurations and preferences, while more extensive data needs might require a database solution like Room or SQLite.

CHAPTER 5- SOFTWARE REQUIREMENT SPECIFICATION

5.1 INTRODUCTION:

A Software Requirement Specification (SRS) is a comprehensive document that serves as a foundation for software development. It outlines the functional and non-functional requirements of a software system, providing detailed description of what the system should do, how it should behave, and the constraints under which it must operate.

5.2 PURPOSE:

The purpose of this document is to define the software requirements for the Women's Safety Application, aiming to provide safety features like emergency alerts, real-time location tracking, and safe route navigation for women in distress.

5.3 SCOPE:

The app will offer emergency assistance, awareness features, community support, and real-time safety updates to enhance personal security. The application will be available on Android and iOS platforms.

CHAPTER 6- SYSTEM DESIGN

6.1 MODULES:

The system design of a Women's Safety Application is composed of several modules, each focusing on different aspects of user safety, usability, and functionality. Below is a high-level design, outlining the key modules and their interactions:

1. User Management Module

Purpose: Handles user registration, authentication, and profile management.

Key Functions:

User Registration: Users can sign up via email, mobile number, or social media accounts (Facebook, Google, etc.).

Authentication: Users can log in securely using a password or biometric authentication (fingerprint/face recognition).

Profile Management: Allows users to manage their personal details, emergency contacts, and preferences (e.g., notification settings).

2. Emergency Response Module

Purpose: Provides the core safety features, such as panic alerts and live location sharing.

Key Functions:

Panic Button: When pressed, the panic button sends alerts with real-time location data to emergency contacts or local authorities (via SMS or push notifications).

Emergency Contact Management: Users can add, remove, and manage emergency contacts that will receive alerts during a crisis.

Location Tracking: Tracks and updates the user's live location during an emergency situation and shares it with the selected contacts.

3. Safe Route Navigation Module

Purpose: Helps users navigate safely by recommending secure routes and avoiding high-risk areas.

Key Functions:

Route Suggestion: Uses real-time crime data and user feedback to recommend safer routes.

Geofencing: Allows users to set up “safe zones” where they can receive alerts if they enter or exit the area.

Map Integration: Provides real-time navigation using GPS, integrating with mapping services like Google Maps.

4. Notification & Alerts Module

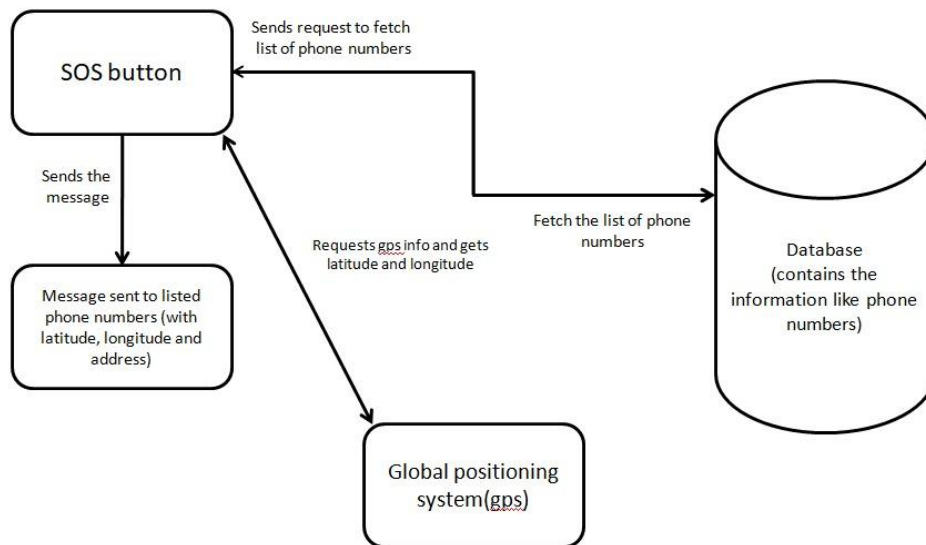
Purpose: Sends notifications and alerts to users for emergencies, safety tips, or updates.

Key Functions:

Push Notifications: Sends real-time alerts when there's an emergency or a safety-related update.

Alert History: Provides users with a history of past alerts, including incidents and updates on reported cases.

Location-based Alerts: Sends alerts based on the user's location or geo fencing settings (e.g., entering a high-risk area).



The diagram illustrates a system flow for an SOS alert mechanism. Here's a quick explanation:

1. **SOS Button:** When pressed, it triggers two actions:

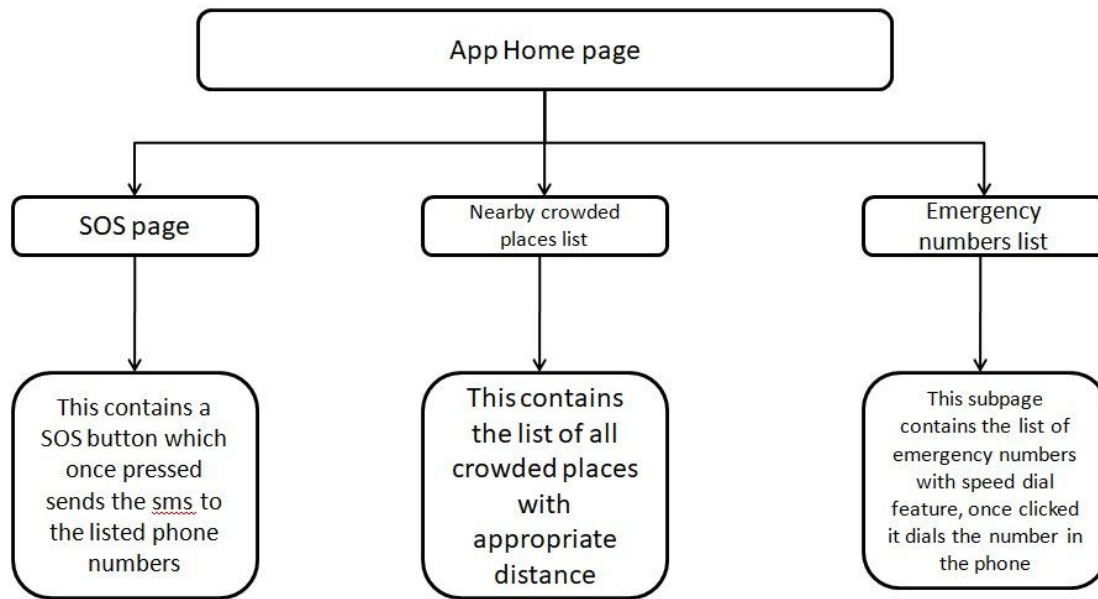
Sends a request to the Database to fetch a list of phone numbers.

Sends a request to the Global Positioning System (GPS) to get the user's location data (latitude and longitude)

2. **Database:** Contains the information such as phone numbers. It responds to the SOS button by sending back the list of contacts.

3. **GPS:** Provides the current location (latitude and longitude) of the user.

4. **Message Sent:** After obtaining the list of contacts and the user's location, a message containing the latitude, longitude, and address is sent to the listed phone numbers.



The figure represents the structure and key features of a women's safety application. Here's the explanation:

App Home Page

The central hub or landing page of the application, from which users can navigate to three main functionalities:

1. SOS Page

This page includes an SOS button.

When the button is pressed, it automatically sends an SMS to pre-listed emergency contacts with critical information like the user's current location.

2. Nearby Crowded Places List

This page provides a list of crowded places near the user, along with the distance to each location.

It helps users identify safer areas nearby in case they feel threatened.

CHAPTER 7- SYSTEM IMPLEMENTATION

7.1 CODING:

7.1.1 Emergency Contact:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

        android:layout_height="match_parent">

<LinearLayout

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:orientation="vertical"

    android:padding="16dp">

<TextView

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:text="Emergency Contacts"

    android:textSize="20sp"

    android:textStyle="bold"

    android:gravity="center"

    android:padding="8dp"
```

```
        tools:ignore="HardcodedText" />

<ListView

    android:id="@+id/listViewContacts"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:dividerHeight="1dp" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.registrationapp;

import android.Manifest;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;


import java.util.ArrayList;

import java.util.List;

public class EmergencyContactActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CALL_PHONE = 1;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_emergency_contact);

        ListView listView = findViewById(R.id.listViewContacts);

        // Fetch registered phone numbers from SharedPreferences

        SharedPreferences sharedPreferences = getSharedPreferences("RegistrationApp",
MODE_PRIVATE);

        String registeredPhones = sharedPreferences.getString("RegisteredPhoneNumbers", "");

        List<String> contacts = new ArrayList<>();

        if (!registeredPhones.isEmpty()) {

            String[] phones = registeredPhones.split(",");

            for (String phone : phones) {

                contacts.add("Registered: " + phone);

            }

        }

    }

}
```



```
// Add default emergency numbers

contacts.add("Police Station: 100");

contacts.add("Ambulance: 108");

contacts.add("Child Helpline: 1098");

contacts.add("Women Helpline: 181");

// Set up the list adapter

ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, contacts);

listView.setAdapter(adapter);

// Add click listener to initiate a phone call

listView.setOnItemClickListener((AdapterView<?> parent, View view, int position,
long id) -> {

    String selectedContact = contacts.get(position);

    String phoneNumber = selectedContact.split(": ")[1]; // Extract the phone number

    makePhoneCall(phoneNumber);

});

}

private void makePhoneCall(String phoneNumber) {

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {

        // Request CALL_PHONE permission

        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CALL_PHONE},
PERMISSION_REQUEST_CALL_PHONE);
```

```
    } else {

        // Make the phone call

        Intent callIntent = new Intent(Intent.ACTION_CALL);

        callIntent.setData(Uri.parse("tel:" + phoneNumber));

        startActivity(callIntent);

    }

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == PERMISSION_REQUEST_CALL_PHONE) {

        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

            Toast.makeText(this, "Permission granted. You can now make calls.",
Toast.LENGTH_SHORT).show();

        } else {

            Toast.makeText(this, "Permission denied. Cannot make calls.",
Toast.LENGTH_SHORT).show();

        }

    }

}

}
```

7.1.2 Home Page:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent">

    <LinearLayout

        android:layout_width="match_parent"

        android:layout_height="match_parent"

        android:orientation="vertical"

        android:gravity="center"

        android:padding="16dp">

        <Button

            android:id="@+id/btnEmergencySOS"

            android:layout_width="match_parent"

            android:layout_height="wrap_content"

            android:text="Emergency SOS"

            android:layout_marginBottom="16dp"

            android:backgroundTint="@color/purple_500"

            android:textColor="@android:color/white"

            tools:ignore="HardcodedText" />

        <Button

            android:id="@+id/btnNearbyCrowdedArea"
```

```
        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:text="Nearby Crowded Area"

        android:layout_marginBottom="16dp"

        android:backgroundTint="@color/purple_500"

        android:textColor="@android:color/white"

        tools:ignore="HardcodedText" />
```

```
<Button
```

```
        android:id="@+id/btnEmergencyContact"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:text="Emergency Contact"

        android:backgroundTint="@color/purple_500"

        android:textColor="@android:color/white"

        tools:ignore="HardcodedText" />
```

```
</LinearLayout>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.registrationapp;

import android.annotation.SuppressLint;

import android.content.Intent;

import android.os.Bundle;

import android.widget.Button;
```

```
import androidx.appcompat.app.AppCompatActivity;

public class HomeActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_home);

        Button btnEmergencySOS = findViewById(R.id.btnEmergencySOS);

        Button btnNearbyCrowdedArea = findViewById(R.id.btnNearbyCrowdedArea);

        Button btnEmergencyContact = findViewById(R.id.btnEmergencyContact);

        btnEmergencySOS.setOnClickListener(v -> {

            Intent intent = new Intent(HomeActivity.this, SOSActivity.class);

            startActivity(intent);

        });

        btnNearbyCrowdedArea.setOnClickListener(v -> {

            Intent intent = new Intent(HomeActivity.this, NearbyActivity.class);

            startActivity(intent);

        });

        // Inside HomeActivity.java

        @SuppressWarnings("CutPasteId") Button emergencyContactButton =
        findViewById(R.id.btnEmergencyContact);

        emergencyContactButton.setOnClickListener(v -> {

            Intent intent = new Intent(HomeActivity.this, EmergencyContactActivity.class);

            startActivity(intent);
```

```
});  
  
}  
  
}
```

COLORS

```
<?xml version="1.0" encoding="utf-8"?>  
  
<resources>  
  
    <color name="black">#FF000000</color>  
  
    <color name="white">#FFFFFFFF</color>  
  
    <color name="purple_500">#6200EE</color>  
  
</resources>
```

STRINGS

```
<resources>  
  
    <string name="app_name">RegistrationApp</string>  
  
    <string name="select_phone_count">Select Number of Phones</string>  
  
    <string name="register">Register</string>  
  
    <string name="emergency_sos">Emergency SOS</string>  
  
    <string name="nearby_crowded_area">Nearby Crowded Area</string>  
  
    <string name="emergency_contact">Emergency Contact</string>  
  
    <string-array name="phone_count_array">  
  
        <item>1</item>  
  
        <item>2</item>  
  
        <item>3</item>
```

```
<item>4</item>

<item>5</item>

</string-array>

</resources>
```

THEMES

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <!-- Base application theme. -->

    <style name="Base.Theme.RegistrationApp"
        parent="Theme.Material3.DayNight.NoActionBar">

        <!-- Customize your light theme here. -->

        <!-- <item name="colorPrimary">@color/my_light_primary</item> -->

    </style>

    <style name="Theme.RegistrationApp" parent="Base.Theme.RegistrationApp" />

</resources>
```

7.1.3 Nearby Activity:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    xmlns:app="http://schemas.android.com/apk/res-auto">
```

<Button

```
    android:id="@+id/btnNearbyCrowdedArea"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Nearby Crowded Area"

    android:layout_marginTop="20dp"

    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    tools:ignore="HardcodedText" />
```

<ListView

```
    android:id="@+id/listView"

    android:layout_width="0dp"

    android:layout_height="0dp"

    app:layout_constraintTop_toBottomOf="@id/btnNearbyCrowdedArea"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintBottom_toBottomOf="parent" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

```
package com.example.registrationapp;
```

```
import android.Manifest;
```



```
import android.annotation.SuppressLint;

import android.content.pm.PackageManager;

import android.location.Location;

import android.os.Bundle;

import android.widget.Button;

import android.widget.ListView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import com.android.volley.Request;

import com.android.volley.RequestQueue;

import com.android.volley.toolbox.JsonObjectRequest;

import com.android.volley.toolbox.Volley;

import com.google.android.gms.location.FusedLocationProviderClient;

import com.google.android.gms.location.LocationServices;

import org.json.JSONArray;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class NearbyActivity extends AppCompatActivity {
```

```
private ListView listView;

private PlacesAdapter adapter;

private final List<Place> placesList = new ArrayList<>();

private Location currentLocation;

private FusedLocationProviderClient fusedLocationClient;

private static final int REQUEST_LOCATION_PERMISSION = 1;

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_nearby);

    listView = findViewById(R.id.listView);

    Button btnNearbyCrowdedArea = findViewById(R.id.btnNearbyCrowdedArea);

    // Initialize FusedLocationProviderClient

    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

    // Set the button click listener to fetch nearby places

    btnNearbyCrowdedArea.setOnClickListener(v -> fetchNearbyPlaces());


    // Fetch current location

    fetchCurrentLocation();

}

// Method to fetch the current location

@SuppressWarnings("MissingPermission")

private void fetchCurrentLocation() {
```

```
// Check for location permissions

if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&

    ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

    // Request location permissions if not already granted

    ActivityCompat.requestPermissions(this, new String[]{

        Manifest.permission.ACCESS_FINE_LOCATION,

        Manifest.permission.ACCESS_COARSE_LOCATION

    }, REQUEST_LOCATION_PERMISSION);

    return;

}

// Fetch the last known location

fusedLocationClient.getLastLocation()

    .addOnSuccessListener(location -> {

        if (location != null) {

            currentLocation = location;

            Toast.makeText(this, "Location fetched: " + location.getLatitude() + ", " +
location.getLongitude(), Toast.LENGTH_SHORT).show();

        } else {

            Toast.makeText(this, "Unable to fetch location",
Toast.LENGTH_SHORT).show();

        }

    })
```

```

    })

    .addOnFailureListener(e -> Toast.makeText(this, "Failed to fetch location: " +
e.getMessage(), Toast.LENGTH_SHORT).show());

}

// Method to fetch nearby places from the Overpass API

private void fetchNearbyPlaces() {

    if (currentLocation == null) {

        Toast.makeText(NearbyActivity.this, "Unable to get location",
Toast.LENGTH_SHORT).show();

        return;

    }

    String latitude = String.valueOf(currentLocation.getLatitude());

    String longitude = String.valueOf(currentLocation.getLongitude());

    String radius = "5000"; // 5km radius

    // Corrected Overpass API query URL

    String url = "https://overpass-api.de/api/interpreter?data=[out:json];(" +

        "node[amenity=hospital](around:" + radius + "," + latitude + "," + longitude + ");"

+

        "node[amenity=police](around:" + radius + "," + latitude + "," + longitude + ");" +

        "node[shop=mall](around:" + radius + "," + latitude + "," + longitude + ");" +

        ");out;";

    // Create a RequestQueue to send the request

    RequestQueue queue = Volley.newRequestQueue(this);

    // Create the request to fetch the data

```

```
JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null,

    response -> {

        try {

            JSONArray elements = response.getJSONArray("elements");

            placesList.clear();

            // Parse the response and add places to the list

            for (int i = 0; i < elements.length(); i++) {

                JSONObject placeObject = elements.getJSONObject(i);

                String name = placeObject.optJSONObject("tags") != null &&
Objects.requireNonNull(placeObject.optJSONObject("tags")).has("name")

                    ? placeObject.getJSONObject("tags").optString("name")

                    : "Unnamed Place";

                double lat = placeObject.getDouble("lat");

                double lon = placeObject.getDouble("lon");

                // Calculate distance from current location

                float[] results = new float[1];

                android.location.Location.distanceBetween(currentLocation.getLatitude(),
currentLocation.getLongitude(), lat, lon, results);

                float distance = results[0];

                placesList.add(new Place(name, lat, lon, distance));

            }

            // Sort places by distance (ascending)

            placesList.sort((place1, place2) -> Float.compare(place1.getDistance(),
place2.getDistance()));
```

```
// Update the ListView with the fetched places

adapter = new PlacesAdapter(NearbyActivity.this, placesList);

listView.setAdapter(adapter);

} catch (Exception e) {

    Toast.makeText(NearbyActivity.this, "Error parsing data",
Toast.LENGTH_SHORT).show();

}

},

error -> Toast.makeText(NearbyActivity.this, "Failed to fetch data",
Toast.LENGTH_SHORT).show());

// Add the request to the RequestQueue

queue.add(request);

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == REQUEST_LOCATION_PERMISSION) {

        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

            fetchCurrentLocation();

        } else {

            Toast.makeText(this, "Location permission denied",
Toast.LENGTH_SHORT).show();


```

```
    }  
    }  
    }  
}
```

7.1.4 SOS Activity:

```
<?xml version="1.0" encoding="utf-8"?>  
  
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
  
    xmlns:tools="http://schemas.android.com/tools"  
  
    android:layout_width="match_parent"  
  
    android:layout_height="match_parent">  
  
    <LinearLayout  
  
        android:layout_width="match_parent"  
  
        android:layout_height="match_parent"  
  
        android:orientation="vertical"  
  
        android:padding="16dp"  
  
        android:gravity="center"  
  
        android:background="#EFEFEF">  
  
        <!-- SOS Button -->  
  
        <Button  
  
            android:id="@+id/btnSendSOS"  
  
            android:layout_width="match_parent"  
  
            android:layout_height="wrap_content"
```

```
        android:text="Send SOS"

        android:backgroundTint="@color/purple_500"

        android:textColor="@android:color/white"

        android:padding="12dp"

        android:textSize="18sp"

        tools:ignore="HardcodedText" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.registrationapp;

import android.Manifest;

import android.content.SharedPreferences;

import android.content.pm.PackageManager;

import android.location.Location;

import android.location.LocationManager;

import android.os.Bundle;

import android.telephony.SmsManager;

import android.widget.Button;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;
```



```
public class SOSActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CODE = 1;

    private String[] phoneNumbers;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_sos);

        // Retrieve registered phone numbers from SharedPreferences

        SharedPreferences sharedPreferences = getSharedPreferences("RegistrationApp",
MODE_PRIVATE);

        String phoneNumbersString = sharedPreferences.getString("RegisteredPhoneNumbers",
        "");

        phoneNumbers = phoneNumbersString.isEmpty() ? new String[0] :
phoneNumbersString.split(",");

        Button btnSendSOS = findViewById(R.id.btnSendSOS);

        btnSendSOS.setOnClickListener(v -> {

            // Request permissions if not granted

            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED ||

                ActivityCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS) !=
PackageManager.PERMISSION_GRANTED) {

                requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.SEND_SMS}, PERMISSION_REQUEST_CODE);
```

```
        } else {

            sendSOS();

        }

    });

}

private void sendSOS() {

    if (phoneNumbers.length == 0) {

        Toast.makeText(this, "No registered phone numbers found!",
Toast.LENGTH_SHORT).show();

        return;

    }

    LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

        Toast.makeText(this, "Location permission not granted!",
Toast.LENGTH_SHORT).show();

        return;

    }

    Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

    if (location != null) {

        String message = "SOS! My location: https://maps.google.com/?q=" +
location.getLatitude() + "," + location.getLongitude();
```

```
try {

    SmsManager smsManager = SmsManager.getDefault();

    for (String phoneNumber : phoneNumbers) {

        smsManager.sendTextMessage(phoneNumber, null, message, null, null);

    }

    Toast.makeText(this, "SOS message sent successfully!",
Toast.LENGTH_SHORT).show();

    } catch (Exception e) {

        Toast.makeText(this, "Failed to send SMS: " + e.getMessage(),
Toast.LENGTH_SHORT).show();

    }

    } else {

        Toast.makeText(this, "Unable to fetch location. Make sure GPS is enabled.",
Toast.LENGTH_SHORT).show();

    }

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == PERMISSION_REQUEST_CODE) {

        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED && grantResults[1] ==
PackageManager.PERMISSION_GRANTED) {

            sendSOS();

        }

    }

}
```

```
        } else {  
  
            Toast.makeText(this, "Permissions denied. Cannot send SOS.",  
Toast.LENGTH_SHORT).show();  
  
        }  
  
    }  
  
}
```

7.1.5 Place item:

```
<?xml version="1.0" encoding="utf-8"?>  
  
<androidx.constraintlayout.widget.ConstraintLayout  
xmlns:android="http://schemas.android.com/apk/res/android"  
  
    xmlns:tools="http://schemas.android.com/tools"  
  
    android:layout_width="match_parent"  
  
    android:layout_height="match_parent">  
  
    <LinearLayout  
  
        android:layout_width="match_parent"  
  
        android:layout_height="wrap_content"  
  
        android:orientation="vertical"  
  
        android:padding="10dp"  
  
        tools:ignore="MissingConstraints">  
  
        <TextView  
  
            android:id="@+id/place_name"  
  
            android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"

        android:text="Place Name"

        android:textSize="16sp"

        android:textStyle="bold"

        android:paddingBottom="4dp"

        tools:ignore="HardcodedText" />

<TextView

        android:id="@+id/place_distance"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:text="Distance"

        android:textSize="14sp"

        android:textColor="#555555"

        tools:ignore="HardcodedText" />

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.registrationapp;

public class Place {

    private final String name;

    private final double latitude;

    private final double longitude;

    private final float distance;
```

```
public Place(String name, double latitude, double longitude, float distance) {  
  
    this.name = name;  
  
    this.latitude = latitude;  
  
    this.longitude = longitude;  
  
    this.distance = distance;  
  
}  
  
public String getName() {  
  
    return name;  
  
}  
  
public double getLatitude() {  
  
    return latitude;  
  
}  
  
public double getLongitude() {  
  
    return longitude;  
  
}  
  
public float getDistance() {  
  
    return distance;  
  
}  
  
}
```

```
package com.example.registrationapp;  
  
import android.annotation.SuppressLint;  
  
import android.content.Context;
```

```
import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.TextView;

import java.util.List;

public class PlacesAdapter extends BaseAdapter {

    private final Context context;

    private final List<Place> places;

    public PlacesAdapter(Context context, List<Place> places) {

        this.context = context;

        this.places = places;

    }

    @Override

    public int getCount() {

        return places.size();

    }

    @Override

    public Object getItem(int position) {

        return places.get(position);

    }

    @Override

    public long getItemId(int position) {
```

```
        return position;
    }

    @SuppressWarnings("DefaultLocale")
    @Override

    public View getView(int position, View convertView, ViewGroup parent) {

        if (convertView == null) {

            convertView = LayoutInflater.from(context).inflate(R.layout.place_item, parent,
false);

        }

        Place place = places.get(position);

        TextView nameTextView = convertView.findViewById(R.id.place_name);

        TextView distanceTextView = convertView.findViewById(R.id.place_distance);

        nameTextView.setText(place.getName());

        distanceTextView.setText(String.format("%.2f meters away", place.getDistance()));

        return convertView;

    }

}
```

7.1.6 Register:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"
```



```
    android:layout_width="match_parent"

    android:layout_height="match_parent"

    xmlns:app="http://schemas.android.com/apk/res-auto">

<ScrollView

    android:layout_width="0dp"

    android:layout_height="0dp"

    android:padding="16dp"

    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintEnd_toEndOf="parent">

<LinearLayout

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:orientation="vertical"

    android:layout_gravity="center">

    <!-- Name Input -->

    <EditText

        android:id="@+id/etName"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_marginBottom="8dp"

        android:hint="Enter your name"
```

```
        android:importantForAutofill="no"

        android:inputType="textPersonName"

        android:padding="12dp"

        tools:ignore="HardcodedText" />

<!-- Age Input -->

<EditText

        android:id="@+id/etAge"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Enter your age"

        android:importantForAutofill="no"

        android:inputType="number"

        android:padding="12dp"

        android:layout_marginBottom="8dp"

        tools:ignore="HardcodedText" />

<!-- Address Input -->

<EditText

        android:id="@+id/etAddress"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Enter your address"

        android:importantForAutofill="no"

        android:inputType="textPostalAddress"
```

```
        android:padding="12dp"

        android:layout_marginBottom="8dp"

        tools:ignore="HardcodedText" />

<!-- Spinner to select phone count -->

<Spinner

    android:id="@+id/spinnerPhoneCount"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_marginBottom="8dp"

    android:spinnerMode="dropdown"

    tools:ignore="HardcodedText" />

<!-- Dynamic container for phone fields -->

<LinearLayout

    android:id="@+id/phoneFieldsContainer"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:orientation="vertical"

    android:layout_marginBottom="16dp" />

<!-- Register Button -->

<Button

    android:id="@+id/btnRegister"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"
```

```
        android:text="Register"

        android:backgroundTint="@color/purple_500"

        android:textColor="@android:color/white"

        android:padding="12dp"

        tools:ignore="HardcodedText" />

    </LinearLayout>

</ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
package com.example.registrationapp;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.text.InputType;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.Spinner;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
```

```
import java.util.List;

public class RegisterActivity extends AppCompatActivity {

    private EditText etName, etAge, etAddress;

    private LinearLayout phoneFieldsLayout;

    private List<EditText> phoneFields;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        // Check if the user has already registered

        SharedPreferences sharedPreferences = getSharedPreferences("RegistrationApp",
MODE_PRIVATE);

        boolean isRegistered = sharedPreferences.getBoolean("isRegistered", false);

        if (isRegistered) {

            // If already registered, navigate to HomeActivity

            Intent intent = new Intent(RegisterActivity.this, HomeActivity.class);

            startActivity(intent);

            finish();

            return;

        }

        setContentView(R.layout.register_main);

        // Initialize views

        etName = findViewById(R.id.etName);
```

```
etAge = findViewById(R.id.etAge);

etAddress = findViewById(R.id.etAddress);

Spinner spinnerPhoneCount = findViewById(R.id.spinnerPhoneCount);

phoneFieldsLayout = findViewById(R.id.phoneFieldsContainer);

Button btnRegister = findViewById(R.id.btnRegister);

phoneFields = new ArrayList<>();

// Set up Spinner with phone count options (1-5)

ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(

    this,

    R.array.phone_count_array,

    android.R.layout.simple_spinner_item

);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

spinnerPhoneCount.setAdapter(adapter);

// Listener to dynamically add phone fields based on spinner selection

spinnerPhoneCount.setOnItemClickListener(new

AdapterView.OnItemClickListener() {

    @Override

    public void onItemClick(AdapterView<?> parent, View view, int position, long id)

    {

        int count = Integer.parseInt(parent.getItemAtPosition(position).toString());

        addPhoneFields(count);

    }

}
```

```
@Override

public void onNothingSelected(AdapterView<?> parent) {

    // Do nothing

}

});

// Register button logic

btnRegister.setOnClickListener(v -> {

    String name = etName.getText().toString().trim();

    String age = etAge.getText().toString().trim();

    String address = etAddress.getText().toString().trim();

    List<String> phoneNumbers = new ArrayList<>();

    for (EditText phoneField : phoneFields) {

        String phone = phoneField.getText().toString().trim();

        if (!phone.isEmpty()) {

            phoneNumbers.add(phone);

        }

    }

    if (name.isEmpty() || age.isEmpty() || address.isEmpty() || phoneNumbers.isEmpty())

{

    // Ensure all fields are filled

    etName.setError("Required");

    etAge.setError("Required");
```

```
        etAddress.setError("Required");

        return;
    }

    // Save user data to SharedPreferences

    SharedPreferences.Editor editor = sharedPreferences.edit();

    StringBuilder phoneNumbersString = new StringBuilder();

    for (String phone : phoneNumbers) {

        phoneNumbersString.append(phone).append(",");

    }

    if (phoneNumbersString.length() > 0) {

        phoneNumbersString.deleteCharAt(phoneNumbersString.length() - 1);

    }

    editor.putString("RegisteredName", name);

    editor.putString("RegisteredAge", age);

    editor.putString("RegisteredAddress", address);

    editor.putString("RegisteredPhoneNumbers", phoneNumbersString.toString());

    editor.putBoolean("isRegistered", true);

    editor.apply();

    // Navigate to HomeActivity

    Intent intent = new Intent(RegisterActivity.this, HomeActivity.class);

    startActivity(intent);

    finish();

});
```



```
}

/**
 * Dynamically adds phone input fields based on the count.
 */

private void addPhoneFields(int count) {

    phoneFieldsLayout.removeAllViews();

    phoneFields.clear();

    for (int i = 0; i < count; i++) {

        EditText phoneField = new EditText(this);

        phoneField.setHint("Enter phone number " + (i + 1));

        phoneField.setInputType(InputType.TYPE_CLASS_PHONE);

        phoneFields.add(phoneField);

        phoneFieldsLayout.addView(phoneField);

    }

}

}
```

7.1.8 Main Activity:

```
package com.example.registrationapp;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        // Check if the user is already registered

        SharedPreferences sharedPreferences = getSharedPreferences("RegistrationApp",
            MODE_PRIVATE);

        boolean isRegistered = sharedPreferences.getBoolean("isRegistered", false);

        Intent intent;

        if (isRegistered) {

            // If already registered, navigate to HomeActivity

            intent = new Intent(MainActivity.this, HomeActivity.class);

        } else {

            // If not registered, navigate to RegisterActivity

            intent = new Intent(MainActivity.this, RegisterActivity.class);

        }

        startActivity(intent);

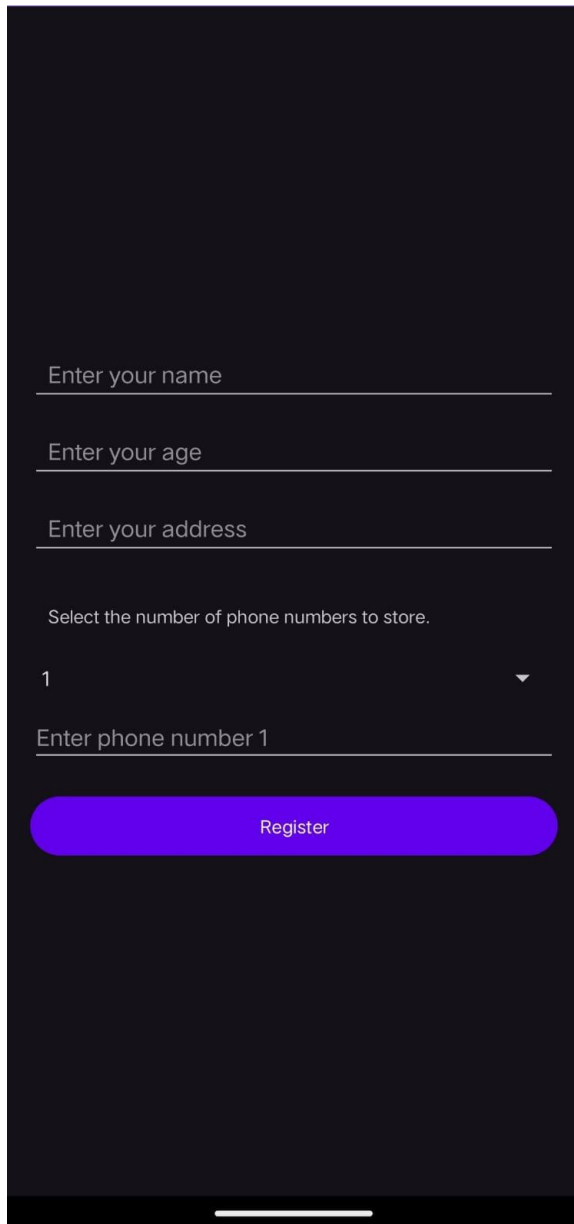
        finish();

    }

}
```

CHAPTER 8- SNAPSHOTS

Registration Page:

A screenshot of the registration page of a mobile application. The page has a dark background. It features four text input fields with light gray placeholder text: "Enter your name", "Enter your age", "Enter your address", and "Enter phone number 1". Below the address field is a dropdown menu with the text "Select the number of phone numbers to store." and the value "1". At the bottom of the form is a large, rounded, orange button with the text "Register".

Enter your name

Enter your age

Enter your address

Select the number of phone numbers to store.

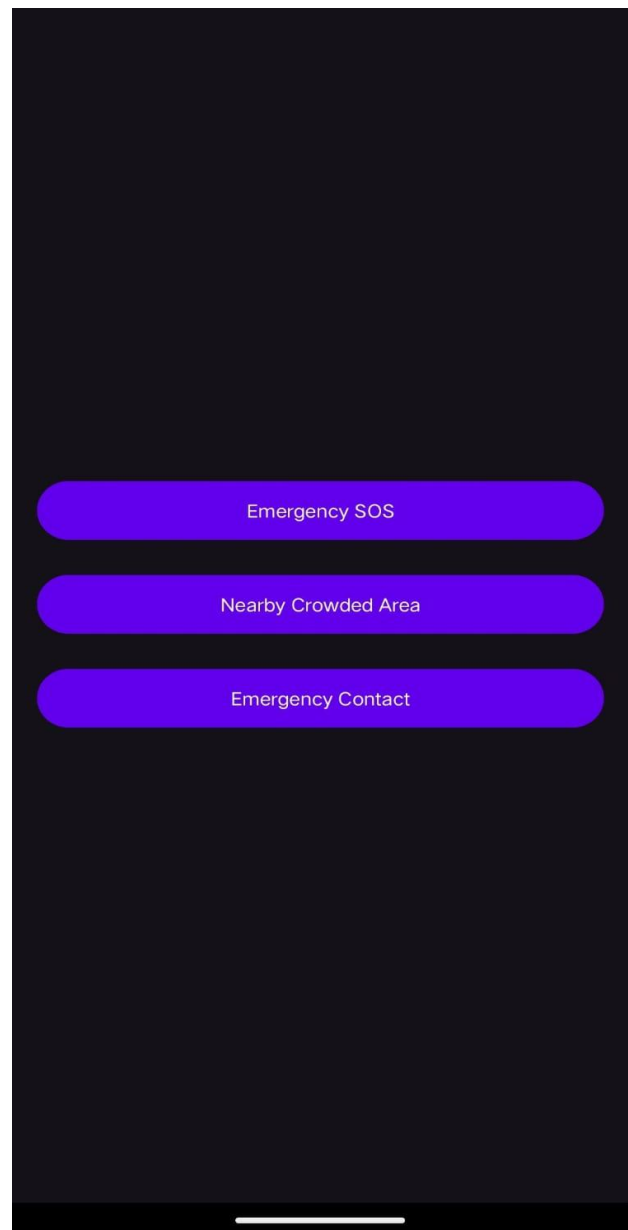
1

Enter phone number 1

Register

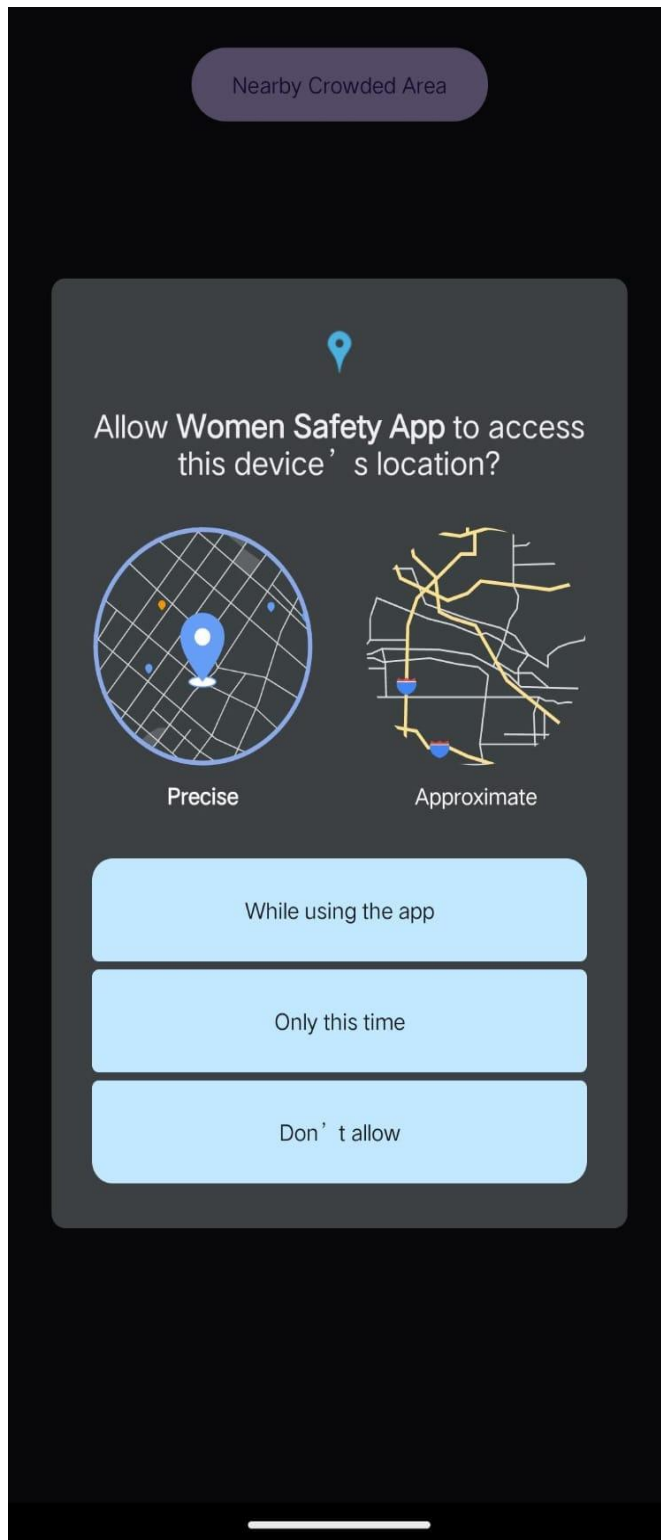
This page enables the users to register into application

Emergency Assistant Page:



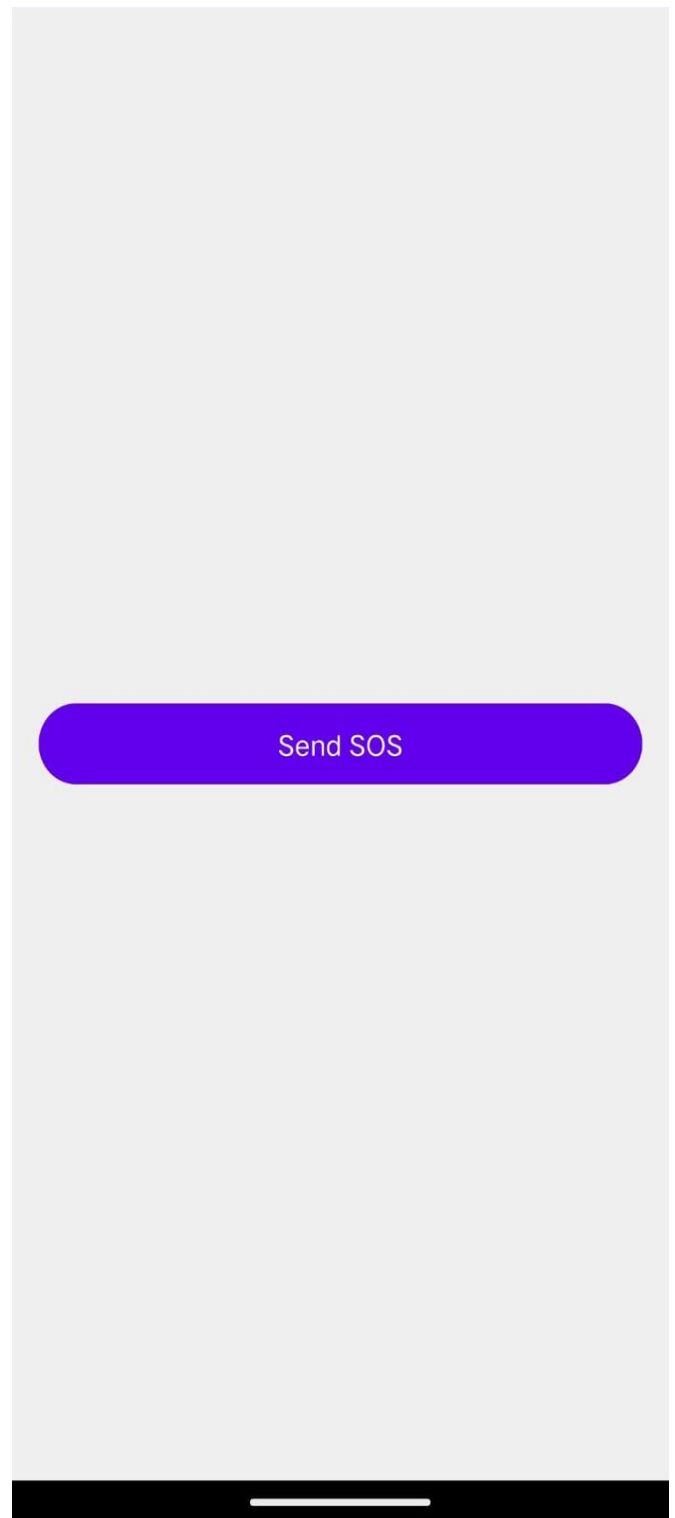
This page provides emergency assistant application

Location Page:



Users are asked for accessing the location

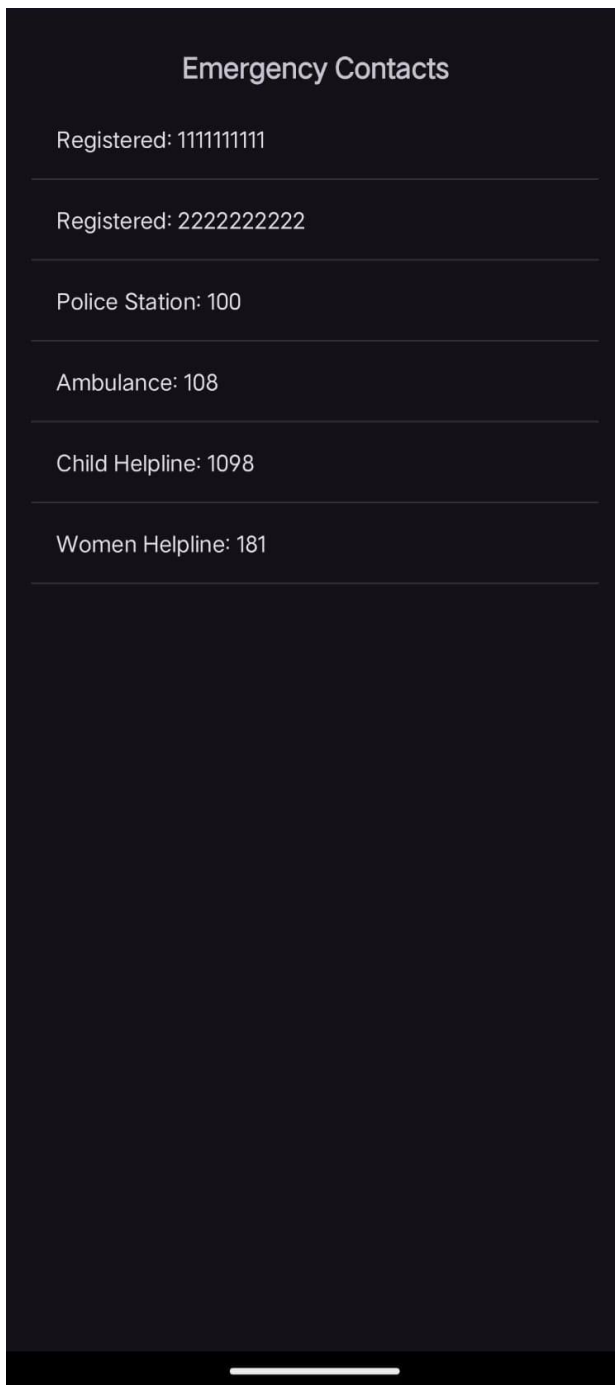
SOS Page:



This page provides the SOS button

which Sends your location to contact

Emergency Contacts Page:



Users are provided with emergency helplines

CHAPTER 9- FUTURE ENHANCEMENT

Here are some future enhancements for a Women's Safety Application to make it more advanced, user-friendly, and effective:

1. AI and Machine Learning Integration

Predictive Analysis: Analyze user movement patterns and identify unsafe areas based on crime data.

Behavioral Alerts: Detect unusual user activity or sudden stops in movement and automatically send alerts.

Personalized Safety Suggestions: Provide personalized routes and suggestions based on user preferences and real-time data.

2. Advanced Location Tracking

Real-Time GPS Tracking: Enhance location accuracy to allow family or friends to track the user.

Geo fencing: Notify trusted contacts if the user enters high-risk areas.

Indoor Location Tracking: Implement technologies like Bluetooth or Wi-Fi for precise location tracking in buildings.

3. Voice and Gesture Commands

Integrate voice or gesture-based SOS activation for hands-free access, such as saying a code word or shaking the phone.

4. Wearable Device Integration

Connect with smart watches, fitness bands, or other wearable devices to activate safety features without needing a phone.

Enable panic button features on wearables.

CHAPTER 10- CONCLUSION

The Women's Safety Application is a crucial tool designed to enhance the security and empowerment of women by leveraging technology. It provides essential features such as SOS alerts, real-time location tracking, emergency contact integration, and quick access to safety resources. By combining user-friendly design with advanced functionalities like GPS tracking, emergency numbers, and nearby safety information, the app ensures timely assistance during emergencies.

With future enhancements like AI-based predictive analysis, wearable device integration, voice commands, and community-driven safety networks, the application has the potential to become even more effective in preventing and responding to unsafe situations.

In conclusion, the Women's Safety Application not only addresses immediate safety concerns but also fosters a sense of security, confidence, and independence among women. By continuously evolving with technological advancements, it stands as a proactive solution toward building a safer environment for women globally.

CHAPTER 11- BIBLIOGRAPHY

Books:

1. Android App Development in Android Studio - J. Paul Cardle
2. A Smarter Way to Learn JavaScript - Mark Myers

Site:

https://www.researchgate.net/publication/352386903_Android_App_for_Women_Safety

https://youtube.com/playlist?list=PLS1QulWo1RIbb1cYyzZpLFCKvdYV_yJ-E&si=yKr9VgvRo7oc8zHt

https://youtube.com/playlist?list=PLGjplNEQ1it_oTvuLRNqXfz_v_0pq6unW&si=Ah4fb4W1uvvo9W1J