ISSN 2277-2685
IJESR/April-June. 2025/ Vol-15/Issue-2s/1564-1570
Nune Sindhu *et. al.*, / International Journal of Engineering & Science Research

# SilentMate: Automatic Phone Silencer

**Amtul Shanaz, Nune Sindhu, Badavath Rajeswari**

[1]Associate Professor, Department Of Cse, Bhoj Reddy Engineering College For Women, India.

[2,3,4]B. Tech Students, Department Of Cse, Bhoj Reddy Engineering College For Women, India.

***ABSTRACT***

*SilentMate is an innovative mobile application designed to enhance user convenience and maintain social etiquette by automatically silencing a device based on specific locations and user-defined schedules. The app aims to minimize disturbances in locations such as libraries, temples, offices, schools, colleges, and hospitals by using geofencing technology. Additionally, it ensures safety on highways by monitoring vehicle speed and sending alerts when speed limits are exceeded.*

## 1-INTRODUCTION

SilentMate is an intelligent Android-based mobile application designed to automatically switch a user's phone to silent mode based on real-world contextual triggers such as current speed, geographical location, and scheduled tasks. The application aims to reduce disturbances in sensitive environments. By automating this process, SilentMate enhances user convenience, encourages responsible phone usage, and supports social etiquette—without requiring constant manual input.

**Proposed System**

SilentMate leverages location-based services, time-based scheduling, and speed monitoring to create a user-friendly experience:

1.Location-Based Automation: Using GPS, the app identifies user locations like libraries or hospitals and silences the device.

2.Time-Based Scheduling: Users can pre-set silent periods for meetings or important events.

3.Speed Monitoring: The app tracks the device's movement speed. If the user exceeds a predefined speed limit, it sends an alert message and emits a beeping sound to warn them.

## 2-REQUIREMENT ANALYSIS

The requirement analysis identifies both functional and non-functional specifications critical to the development and successful operation of the **SilentMate** application. This ensures that the system performs reliably, securely, and efficiently while meeting user expectations.

**Functional Requirements**

These are the essential features and behaviors that **SilentMate** must support:

1. **User Registration and Authentication**
   - Users can **register** and **log in** securely using **Firebase Authentication** (email/password or other providers).
   - Ensures personalized task management and data synchronization.
2. **Task Scheduling**
   - Users can **schedule silence tasks** with:
     - Specific **start and end times**
     - Associated **location coordinates**
   - Supports **one-time** and **recurring** task events.
3. **Location Tracking**
   - Utilizes the device's **GPS** to monitor the user's real-time location.
   - Triggers silent mode when entering or exiting predefined zones.

**Non-Functional Requirements**

These requirements ensure the system is **usable, performant, and secure**:

1. **User-Friendly Interface**
   - Simple, intuitive UI/UX suitable for all user demographics.
   - Clean layout for task creation and management.
2. **Quick Response Time**
   - Fast execution of task triggers and mode changes.
   - Near-instant app load and smooth navigation.
3. **Reliable Background Services**
   - Application services must run **persistently** in the background to monitor conditions.
   - Should continue functioning even if the app is not in the foreground.

## 3. DESIGN

**Architectures**

Design architecture refers to the overall structure and framework of a system, outlining how its components interact, communicate, and work together to meet functional and non-functional requirements. It serves as a blueprint for the system's development, ensuring it is scalable, maintainable, and efficient.

Software Architecture

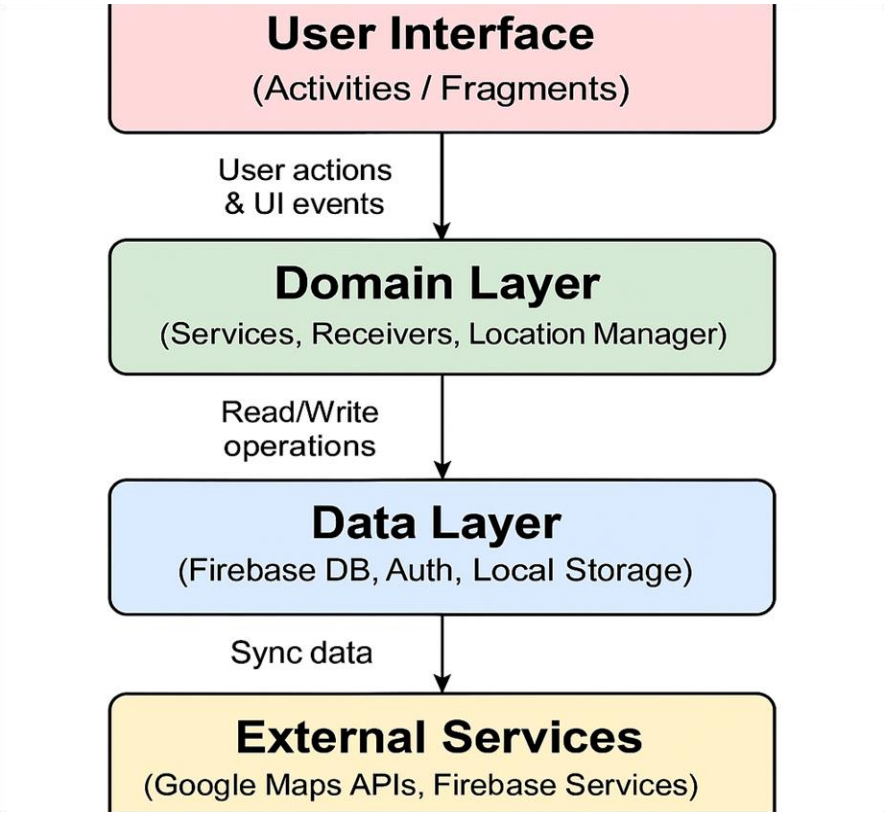



**Fig.no: 3.1**

**Technical Architecture**

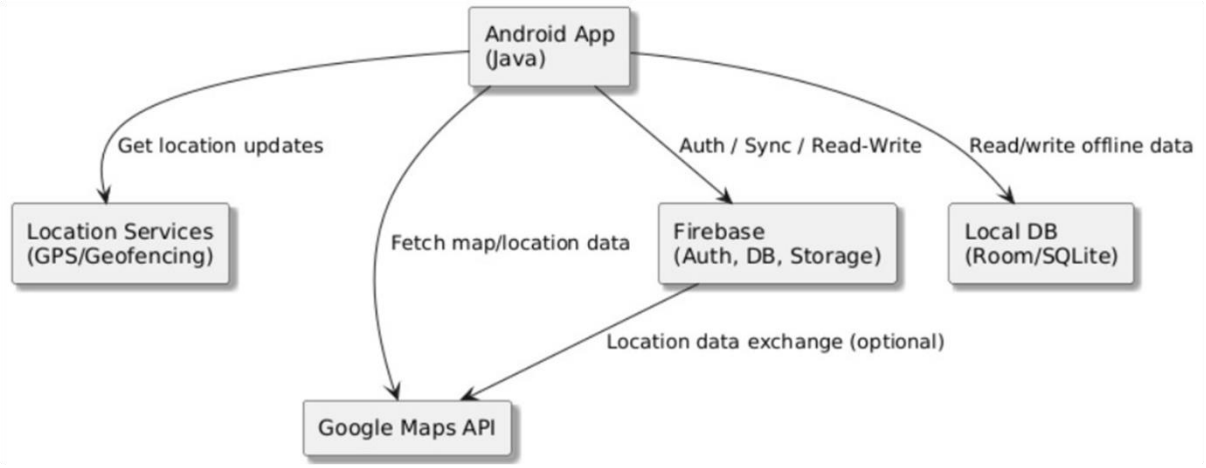



**Fig.no: 3.2**

## 4-IMPLEMENTATION (PSEUDO CODE)

Implementation in software development refers to the actual building and integration of all the components defined during design into a working application. It involves writing code, setting up databases, connecting APIs, and ensuring all modules interact seamlessly. In the context of this application, *implementation* includes realizing all functional parts such as user authentication, task management, location tracking, and silent mode activation

**User Authentication (Login & Registration)**

**Platform:** Firebase Authentication

**Steps**

- Setup Firebase Project.
- Enable Email/Password sign-in method.
- Design XML layouts for LoginActivity and RegisterActivity.
- On form submission:
- Redirect authenticated user to the main screen.

1. **Task Scheduling Module**

**Platform:** Firebase Realtime Database + SharedPreferences

**Steps:**

- Create a data model: MyDoes.java.
- Create task form screen (NewTaskAct) to input title, time, date, and location.
- Save tasks:
  - Temporarily with TempTaskStorage using Gson + SharedPreferences.
  - Permanently to Firebase:
- Load and display tasks in TaskSchedule.java using DoesAdapter with RecyclerView.

2. **Location-Based Triggering**

**Platform:** FusedLocationProviderClient (Google Play Services)

**Steps:**

- Request permissions (ACCESS_FINE_LOCATION and ACCESS_BACKGROUND_LOCATION).
- Fetch current location using:
- Compare fetched location to task coordinates.
- If within defined radius, trigger silent mode.

3. **Speed-Based Decision Making**

**Module:** SpeedChecker.java

**Steps:**

- Get speed from location object: location.getSpeed().
- Use logic:
- This mock logic can be replaced with Google Roads API in future.

4. **Silent Mode Trigger**

- **API:** AudioManager
- **Steps:**
- Access AudioManager and set ringer mode:

5. **Persistent Storage of Tasks and Locations**

- **Platform:** SharedPreferences with Gson
- **Steps:**
- Save all tasks locally:

6. **Background Task and Notification**

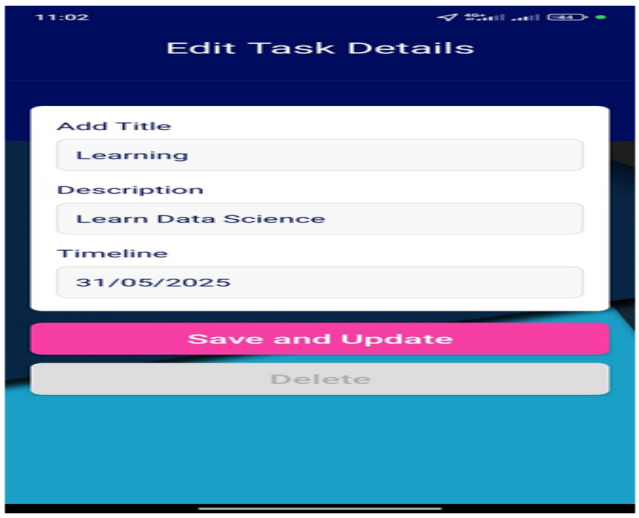
- **Platform:** Background Services + NotificationManager
- **Steps:**
- When a location or time matches a task:
- Use NotificationManager to display it.

7. **UI & Navigation Flow**

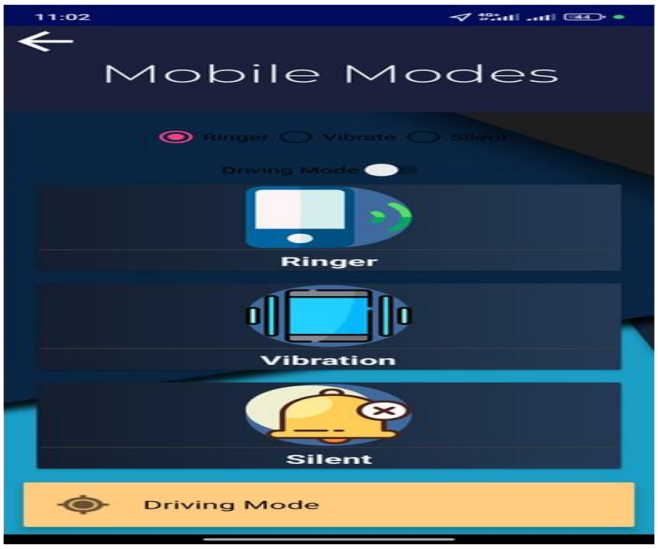
- **Steps:**
- Use Intent to navigate between screens:
- Pass required data using intent.putExtra(...).

## 5-SCREENSHOTS

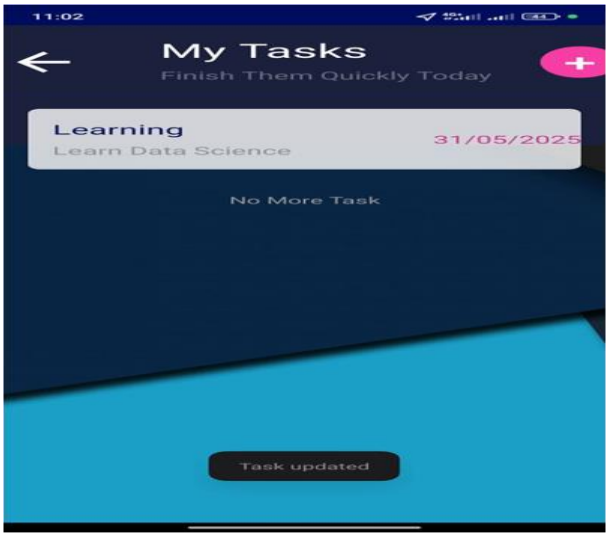

**ScreenShot No:1**




**ScreenShot No:6.2**

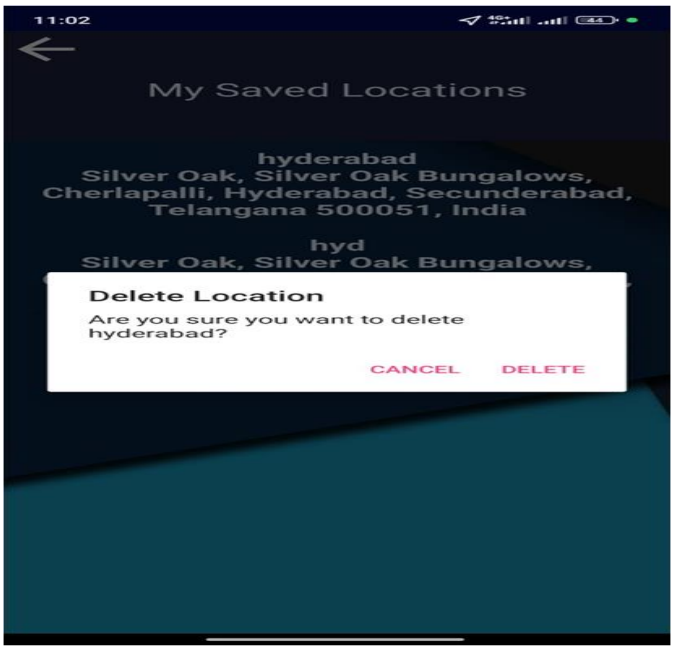



**SreenShot No:3**




**ScreenShot No:4**


**ScreenShot No:5**




**ScreenShot No:6**

## 6-CONCLUSION

The project SilentMate: Automatic Phone Silencer aims to solve the problem of manual phone silencing in sensitive or planned environments like meetings, classrooms, or religious places. It combines location-based services, scheduling, and real-time task handling using Firebase and local storage. The system is stable, user-friendly, and adaptable.

## REFERENCES

· **Android Developers Documentation**

This documentation provides details on Android development, including activity lifecycle methods, UI elements like RecyclerView, and best practices for permissions and location services.

URL: https://developer.android.com/docs

· **Firebase Documentation**

This resource is crucial for setting up Firebase Realtime Database to store and retrieve tasks remotely, implementing ValueEventListener, and handling data binding with POJO models.

URL: https://firebase.google.com/docs

- **Google Play Services - Location API**

  This API is used for accessing a user's location coordinates and implementing location-based triggers for silent mode, as well as for the mock logic in SpeedChecker.java for demonstration purposes.

  URL: https://developers.google.com/android/reference/com/google/android/gms/location

- **Geofencing API - Android Developers**

  This API is referenced for designing the logic to detect entry into predefined locations to trigger silent mode and for planning future scope features for location-based automation.

  URL: https://developer.android.com/training/location/geofencing

- **Gson Library - Google GitHub**

  This library is used for serializing and deserializing task lists in TempTaskStorage.java, converting List<MyDoes> to JSON and vice versa for local storage.

  URL: https://github.com/google/gson

- **Google Material Design Guidelines**

  These guidelines are utilized for designing consistent and responsive UI/UX elements, including choosing typefaces, button styles, and visual hierarchy for various app screens.

  URL: https://m3.material.io/

- **Android Geo Services and Maps SDK (Planned Scope)**

  This resource is referenced for the future implementation of accurate geofencing and route-based silent mode, as outlined in the design documentation and future scope section.

  URL: https://developers.google.com/maps/documentation