EVENT MANAGEMENT SYSTEM

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

GULLIPALLI DEVI VARA PRASAD NAIDU

12205408

CSE 226

ANDROID APP DEPLOYMENT

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

November 2024

TABLE OF CONTENTS

Introduction

The Event Management System Android application is designed to streamline the organization of events, creating a seamless and intuitive experience for users from the moment they launch the app. It begins with a captivating splash screen that introduces the user to the interface, followed by a secure login and signup system that provides tailored access, including a special admin role with additional privileges. Once authenticated, users are guided to a comprehensive dashboard that divides events into three distinct sections: Today's Events, All Events, and Completed Events, making it easy to track and manage both upcoming and past events at a glance. To enhance user interaction, the dashboard includes an Info button for contact details, a Rating button that encourages feedback to improve the app, and a floating action button (FAB) that expands to reveal quick links to key features, including a Maps activity and the Create Event activity. The Create Event activity itself offers a user-friendly form with fields to input critical event information such as the event name, location, date, time, type, and a brief description, allowing users to fully customize each entry. Once created, events are displayed in a ListView that makes it easy to review and access saved details. The app architecture is supported by two dedicated databases: one for managing administrator credentials and another for storing events, ensuring that data remains organized and secure. A set of custom adapters seamlessly fetches and displays event details from the database, contributing to the app's responsive and efficient performance. Altogether, this Event Management System app combines ease of use, thoughtful organization, and robust functionality, making it an ideal solution for users who need to manage events effectively and stay organized with minimal effort on their Android devices.

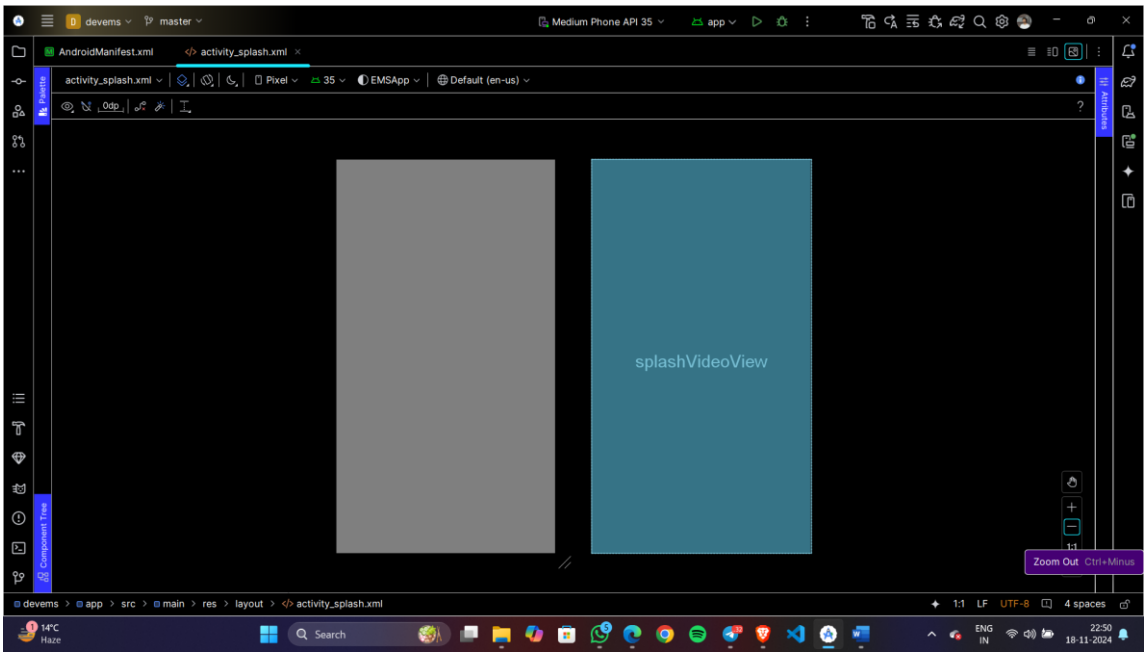
Key Features and Interfaces of the Event Management System App

1. **User-Friendly Event Creation and Management Interface**
    - Allows users to create events with essential details like name, location, date, time, type, and description.
    - Easy-to-use form layout with input fields enhanced by DatePicker and TimePicker.
2. **SQLite Database for Persistent Event Storage**
    - Uses an SQLite database to store event information, supporting CRUD (Create, Read, Update, Delete) operations.
    - Persistent storage ensures events are saved across app sessions.
3. **Event Listing and Adapter Integration**
    - A custom EventsAdapter fetches and displays events in a ListView.
    - Provides a clean, structured list of all events with quick access to event details.
4. **Location Navigation with Google Maps Integration**
    - Utilizes UserNavigation activity to facilitate location-based navigation for events.
    - Supports location entry through a search box and displays selected locations on an embedded MapView.
    - Allows users to open Google Maps for detailed navigation to specific event locations.
5. **Real-Time Search Functionality in Fragments**
    - Filters displayed events based on user-entered queries, enhancing usability and event accessibility.

6. **Customizable Event Type Selection**
    - Provides a dropdown for event types, such as Birthday, Conference, Wedding, etc., making the app versatile for different event categories.
7. **Dynamic Long-Press Options for Event Modification**
    - Long-press functionality within the ListView allows users to edit or delete events directly, adding flexibility to event management.
8. **Login and Signup with Admin Role-Based Access**
    - Features a login and signup system with SQLite, allowing only admin role access to the event management functionalities.
    - Secure storage of user credentials and contact details for authenticated access.
9. **Notification Reminders for Upcoming Events**
    - Notifies users based on event date and time, keeping them informed of upcoming events.
10. **UI Enhancements for Better User Experience**
    - Includes vector icons for input fields, unique backgrounds, and card layouts to provide a visually appealing and user-friendly interface.

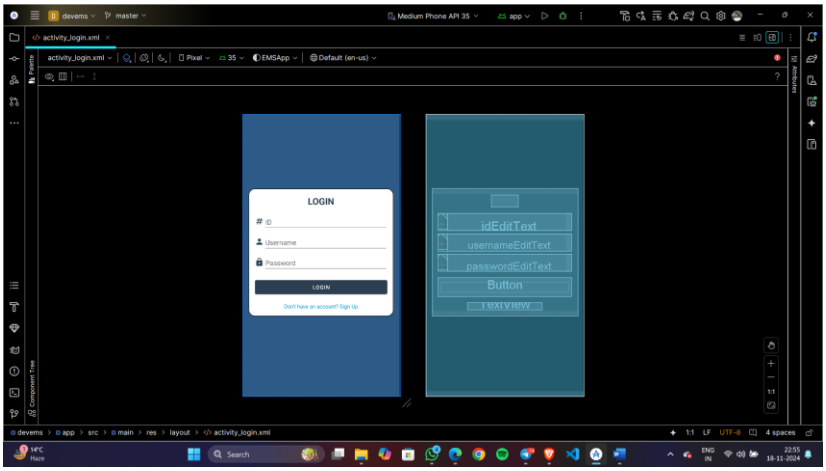
Splash Activity

The SplashActivity module in the Event Management System app delivers an engaging introductory experience with a video-based splash screen. Implemented in `SplashActivity.kt`, this activity features a full-screen `VideoView` that plays a short clip from the app's raw folder (`event_management_system.mp4`) upon launch, creating a professional and visually appealing entry point. The `onCreate` method sets the content view to `activity_splash.xml`, where the `VideoView` is defined within a `ConstraintLayout` for seamless alignment with screen dimensions. A URI retrieves the video resource, and the video playback is accompanied by a handler that triggers a 1.5-second delay (`SPLASH_DURATION`). Upon completion, the app transitions smoothly to `LoginActivity` using an `Intent` while calling `finish()` to remove the splash screen from the back stack, offering a polished and user-friendly experience.



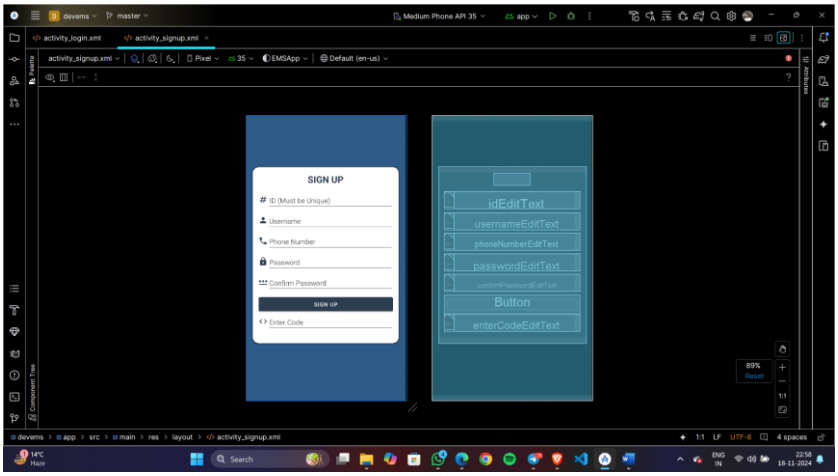
Login Activity

The LoginActivity module of the Event Management System app offers a user-friendly interface for secure account authentication, enabling access to the app's core features. The layout, defined in `activity_login.xml`, features a visually appealing `CardView` containing a login form with fields for ID, username, and password, each enhanced with vector icons for better usability. Alongside the `EditText` fields are a `Login` button and a `TextView` linking to the signup page. In `LoginActivity.kt`, the login process validates user inputs against stored credentials using the `UserDatabaseHelper`. During `onCreate`, UI elements are initialized, and inputs are checked for completeness and correctness. If the credentials match, the app transitions to `DashboardActivity`; otherwise, a toast message prompts users to retry. New users can navigate to `SignupActivity` via the signup link. This streamlined login functionality ensures secure access and directs users to the appropriate section based on their status.



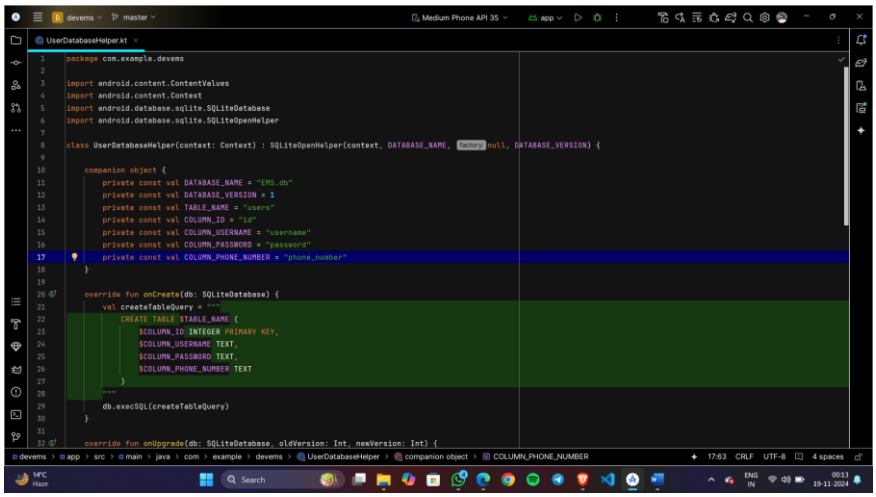
SignUpActivity

The sign-up screen for the Android application provides a clean, user-friendly interface and robust functionality for secure user registration. The layout, designed in XML, features a `LinearLayout` that arranges input fields vertically within a `CardView` for a polished appearance, complemented by a consistent color scheme using `colorPrimaryDark`. The form includes fields for ID, username, phone number, password, confirm password, and a verification code, where "4444" is required for admin users. Vector images accompany each field for visual clarity, enhancing the user experience. The Kotlin code validates inputs by checking that all fields are filled, passwords match, the ID is a valid and unique number, and the admin code is correct. Upon successful validation, the user is added to the database, a success message is displayed, and the app redirects to the login screen. If an error occurs, an appropriate message notifies the user. This implementation ensures a secure, intuitive, and visually appealing registration process with seamless navigation to the login screen.



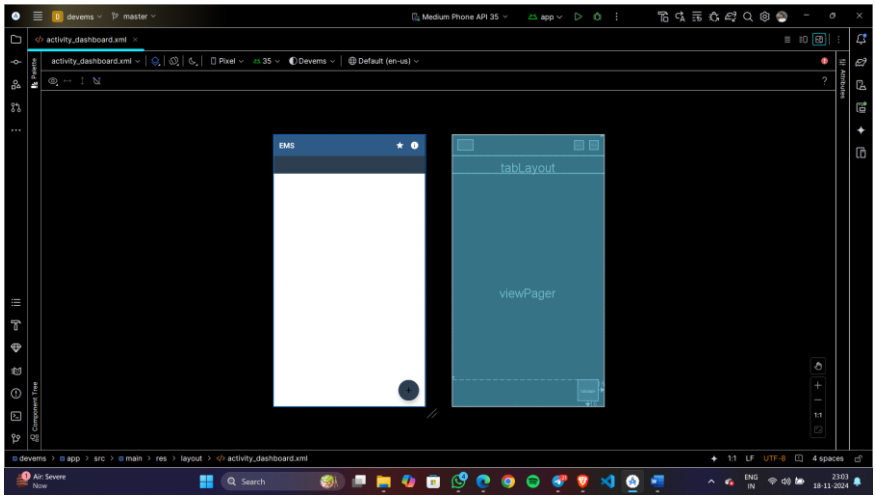
UserDatabaseHelper

The `UserDatabaseHelper` class manages a simple SQLite database for user data in an event management system Android app. Extending `SQLiteOpenHelper`, it facilitates database creation, version management, and user authentication processes. The database, named `EMS.db`, contains a single table, `users`, storing details such as ID, username, password, and phone number. The `onCreate` method defines the table schema with an SQL query, while `onUpgrade` handles database updates by dropping and recreating the table as needed. Key methods include `isIdExists`, which checks for duplicate user IDs to prevent redundancy, `addUser` for inserting new user records, and `validateUser` to verify user credentials during login. This class ensures secure storage and efficient handling of user data for registration and authentication, forming a backbone for the app's user management functionality.



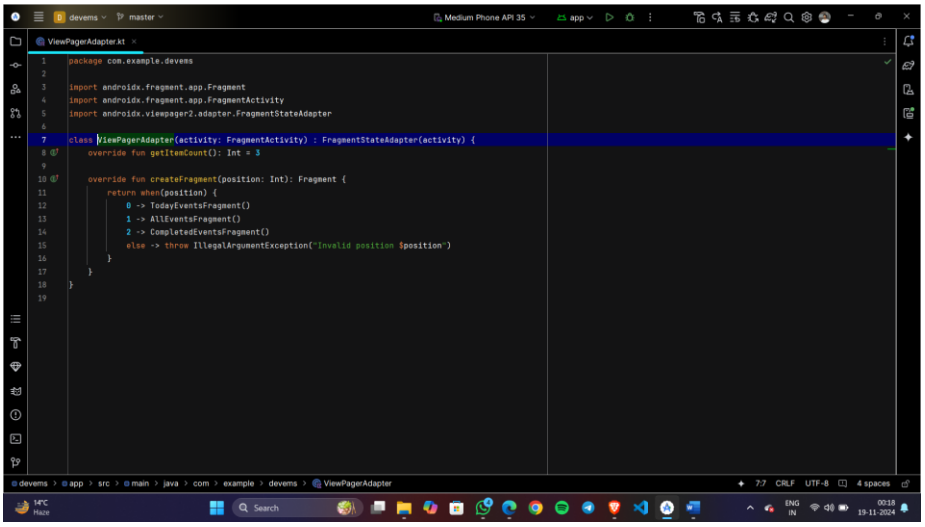
Dashboard Activity

The `DashboardActivity` class is a pivotal part of the Event Management System (EMS) Android app, offering a centralized, user-friendly interface with diverse interactive features. It utilizes `ViewPager2` and `TabLayout` for a tabbed layout, enabling smooth navigation between "Today," "All Events," and "Completed" event categories, managed via a `TabLayoutMediator`. A dynamic Floating Action Button (FAB) menu enhances functionality, with the primary FAB toggling visibility of secondary FABs for creating new events (`fabCreateEvent`) and navigating to user locations via maps (`fabMaps`). The toolbar includes an `infoButton` for displaying informational dialogs and a `ratingButton` for app rating, implemented with a `RatingBar` that provides personalized feedback messages based on user ratings. The activity's XML layout integrates the toolbar, FABs, and `ViewPager2` within a `RelativeLayout`, ensuring a responsive and modern design. By combining intuitive navigation, event management, and user interaction features, `DashboardActivity` serves as the central hub of the EMS app, offering quick access to key functionalities and enhancing overall usability.



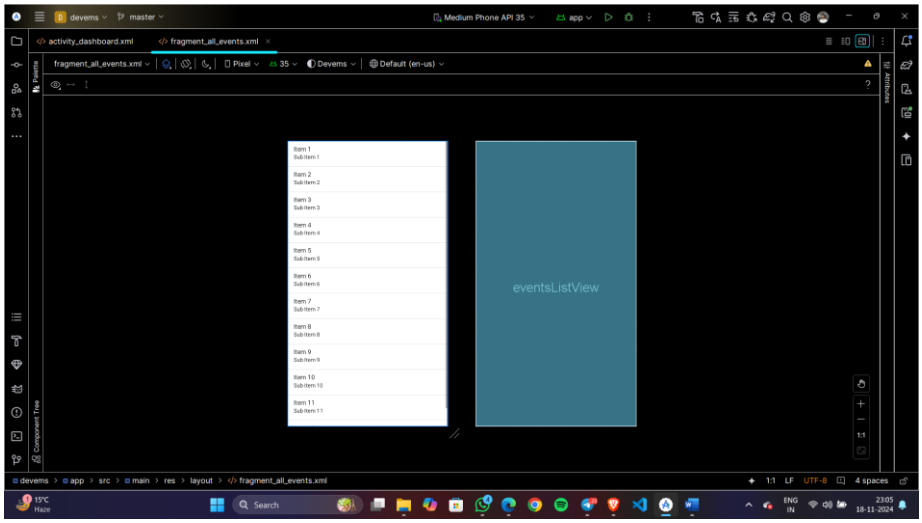
ViewPageAdapter

The `ViewPagerAdapter` class is a custom adapter for managing fragment navigation in a `ViewPager2` within an event management app. Extending `FragmentStateAdapter`, it works with a `FragmentActivity` to seamlessly switch between three fragments: `TodayEventsFragment`, `AllEventsFragment`, and `CompletedEventsFragment`, each representing different views of the app's events. The `getItemCount` method specifies that there are three fragments, while the `createFragment` method returns the appropriate fragment based on the selected position—position 0 shows `TodayEventsFragment`, position 1 displays `AllEventsFragment`, and position 2 returns `CompletedEventsFragment`. An `IllegalArgumentException` is thrown for invalid positions to maintain stability. This adapter facilitates smooth swiping between event views in the `ViewPager2`, providing an intuitive and efficient user experience for navigating event data.



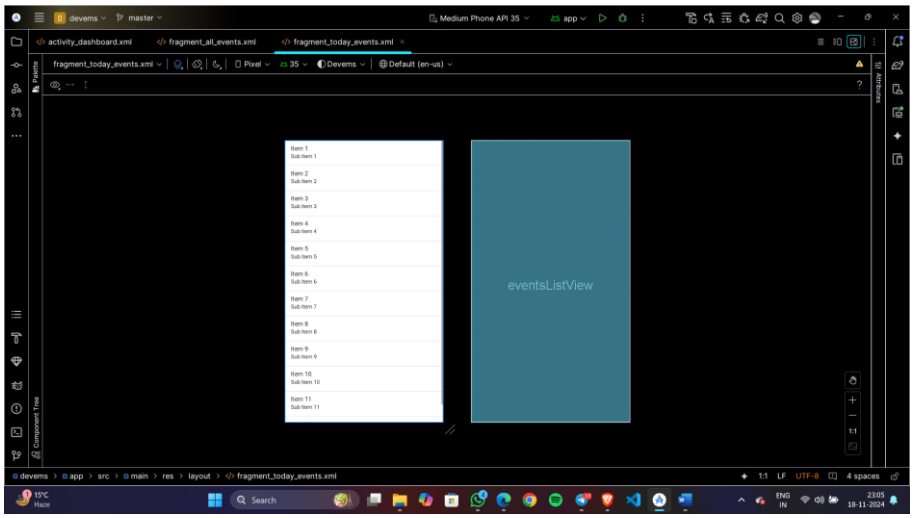
AllEventsFragment

The `AllEventsFragment` class, extending `BaseEventsFragment`, is designed to dynamically display all events in the Event Management System (EMS) application. The `loadEvents()` method is overridden to retrieve all events from the database (`dbHelper.getAllEvents()`), clear the existing event list, and populate it with the fetched results. Once updated, the associated `eventsAdapter` is notified to refresh the UI. The fragment's XML layout includes a `FrameLayout` containing a `ListView` (ID: `eventsListView`) for displaying events and a `TextView` (ID: `emptyView`) as a placeholder message when no events are available. The placeholder text, "No events found," is centrally aligned and dynamically displayed if the event list is empty, ensuring clear feedback for users. By seamlessly handling event data retrieval and display, this fragment provides an intuitive and responsive user experience, maintaining functionality and usability even in the absence of events.



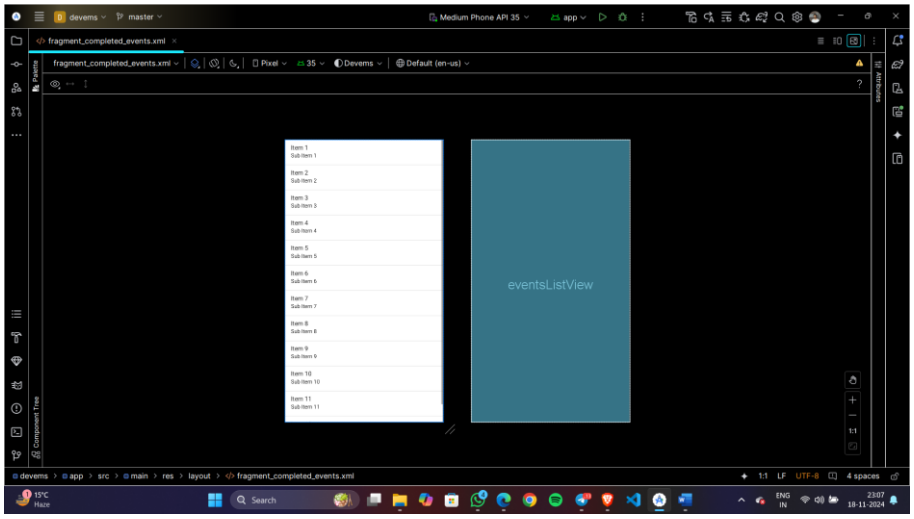
Today’sEventsFragment

The `TodayEventsFragment` class, extending `BaseEventsFragment`, is tailored to display events scheduled for the current day in the Event Management System (EMS). Its `loadEvents()` method clears the existing event list, retrieves all events from the database (`dbHelper.getAllEvents()`), and filters them to match the current date, formatted consistently using the `SimpleDateFormat` pattern "dd/MM/yyyy." Only events occurring today are added to the list, after which the `eventsAdapter` is notified to update the UI. The fragment's XML layout includes a `FrameLayout` with a `ListView` (ID: `eventsListView`) for displaying today's events and a `TextView` (ID: `emptyView`) that shows a "No events found" message if no events are scheduled. This dynamic display and empty state handling ensure a clear and user-friendly interface, providing seamless feedback on whether any events are planned for the current day.



CompletedEventsFragment

The `CompletedEventsFragment` class, part of the Event Management System (EMS), displays events that have already occurred—those with dates earlier than the current date. Extending `BaseEventsFragment`, it overrides the `loadEvents()` method to filter and show completed events. The method begins by clearing the current events list and fetching all events from the database. Using the `SimpleDateFormat` class, the current date is formatted and stored in a `today` variable. Each event’s date is parsed and compared with the current date, adding only those events to the list that occurred before today. Once the list is updated, the `eventsAdapter` is notified to refresh the UI. The fragment's XML layout includes a `FrameLayout` containing a `ListView` (ID: `eventsListView`) for displaying completed events and a `TextView` (ID: `emptyView`) for showing a "No events found" message when there are no completed events. This design ensures a user-friendly interface by clearly indicating when no completed events are available while seamlessly presenting relevant event data.



BaseEventsFragment

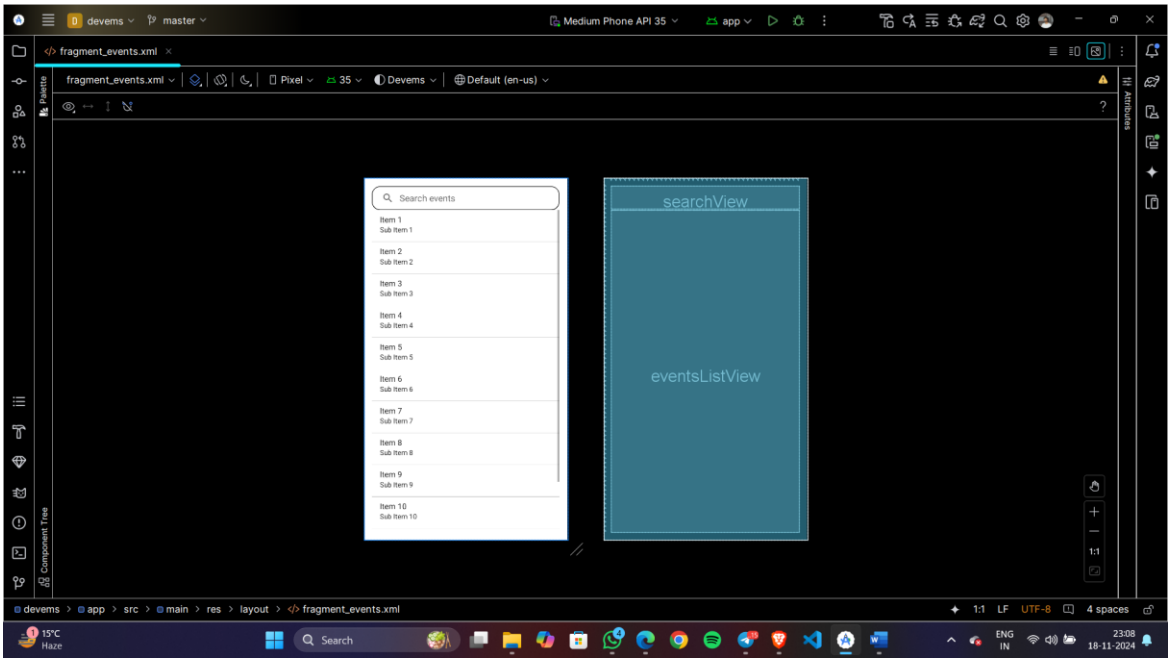
The given code implements the foundational architecture of a fragment-based event management system in an Android application. Specifically, it defines the `BaseEventsFragment` class, an abstract fragment that acts as a base for managing and displaying a list of events using a `ListView`. The class integrates a search functionality, event filtering, and an edit dialog for event management, making it a comprehensive and modular component for handling event-related operations.

The `BaseEventsFragment` is designed to handle event data dynamically by interfacing with an SQLite database through the `EventDatabaseHelper`. The fragment initializes critical components like a `ListView` for displaying events, a `SearchView` for filtering events, and an `EventsAdapter` to manage the binding of event data to the list view. The `onCreateView` method inflates the fragment layout, and `onViewCreated` sets up the user interface components. It incorporates an empty view (`TextView`) to provide user feedback when no events are available, ensuring a seamless user experience.

One of the key features is the search functionality, implemented using the `SearchView`'s `OnQueryTextListener`. It enables real-time filtering of events based on user input by comparing the search query with event names in a case-insensitive manner. This makes the system highly responsive and user-friendly.

The fragment also includes an `AlertDialog` for editing or deleting events, providing an intuitive interface for event management. Users can edit various attributes of an event, such as its name, location, date, time, type, and description, using pre-filled `EditText` fields. The dialog incorporates date and time pickers, allowing users to conveniently update event schedules. The `AlertDialog` also provides options to save changes, cancel edits, or delete the event entirely. Upon performing these operations, the database is updated, and the event list is refreshed to reflect the changes dynamically.

The fragment ensures data consistency by overriding lifecycle methods like `onResume` to reload events whenever the fragment becomes active and `onDestroy` to close the database connection when the fragment is destroyed. The use of abstract methods, like `loadEvents`, allows child classes to define specific behaviors for loading data, promoting reusability and flexibility in the application design.In summary, the `BaseEventsFragment` exemplifies a robust and extensible approach to managing event-related operations in an Android app. It encapsulates functionalities for displaying, searching, editing, and deleting events, all while maintaining a clean and modular code structure. This makes it a cornerstone for developing an efficient and user-centric event management system.



CreateEvent

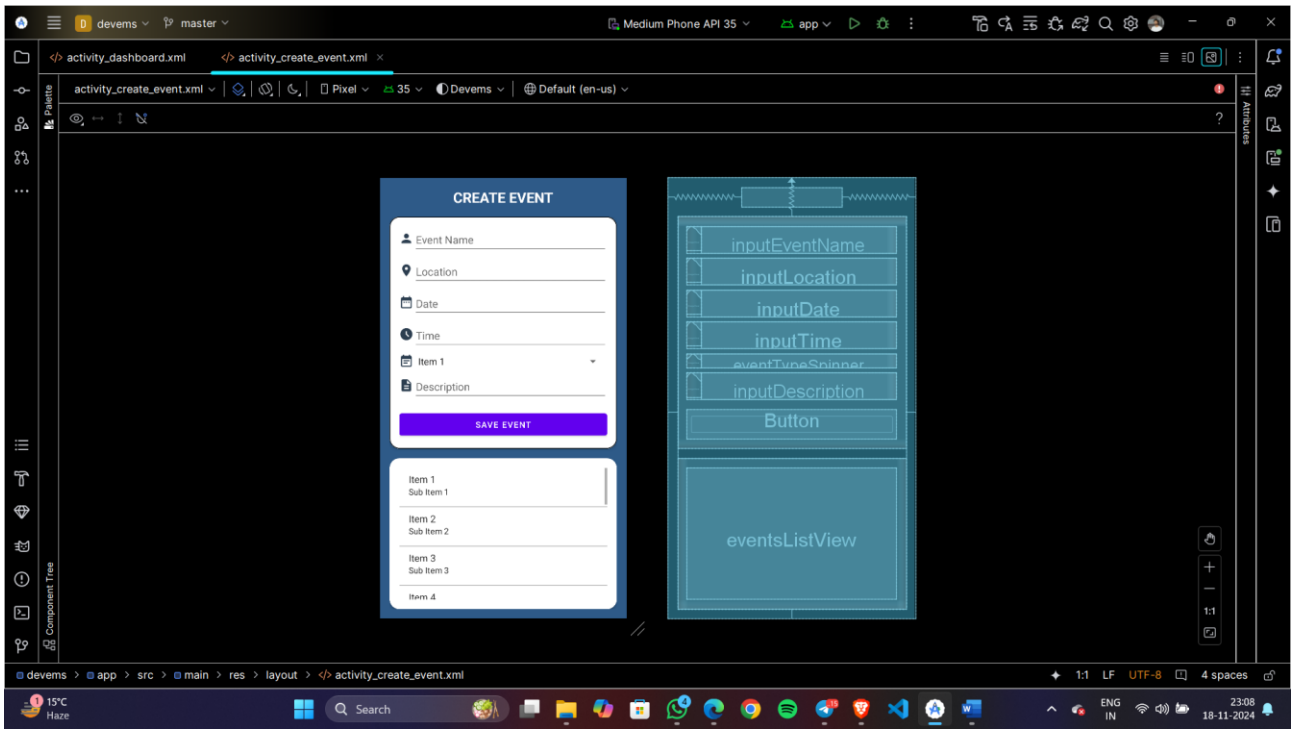
This code represents an Android activity that enables users to create and manage events. The CreateEvent activity allows users to input event details, including the name, location, date, time, event type, and description, and save them into a local database. It includes functionality for selecting the event's date and time using DatePickerDialog and TimePickerDialog respectively. Additionally, users can select the event type from a dropdown spinner, which contains options like Birthday, Marriage, Festivals, and others.

Upon saving an event, the application uses the EventDatabaseHelper class to insert the event data into an SQLite database. A confirmation toast is shown to notify the user that the event has been saved successfully. Furthermore, a notification is scheduled for the event's specified date and time using Android's AlarmManager, ensuring that the user receives a reminder for the event. The event data is displayed in a list view, and users can interact with events by editing or deleting them.

The activity includes a notification permission request for devices running Android versions higher than Android 13 (API level 33), ensuring the app has the necessary permissions to send notifications. The UI of the CreateEvent activity is built using a combination of LinearLayout, CardView, and EditText components, with vector images used as icons before each EditText field. The background of the layout is styled with a custom drawable, providing a visually appealing design.

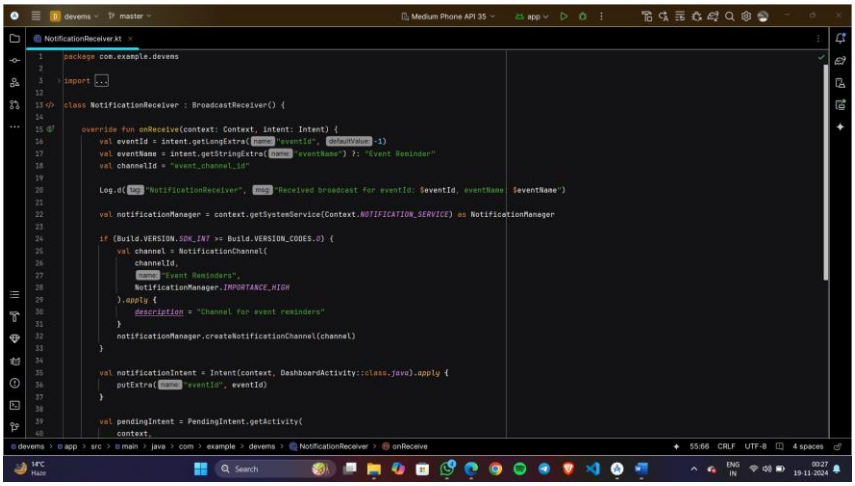
In addition, the code provides a method for displaying an editable dialog where users can modify or delete events. This dialog is customized with pre-filled data for the selected event, and any changes made are updated in the database. If an event is deleted, the corresponding notification is also canceled. The app also ensures that notifications are properly updated whenever an event is modified.

Overall, this implementation demonstrates a well-organized approach to creating, editing, and managing events, providing users with a smooth and interactive experience for managing their schedules.



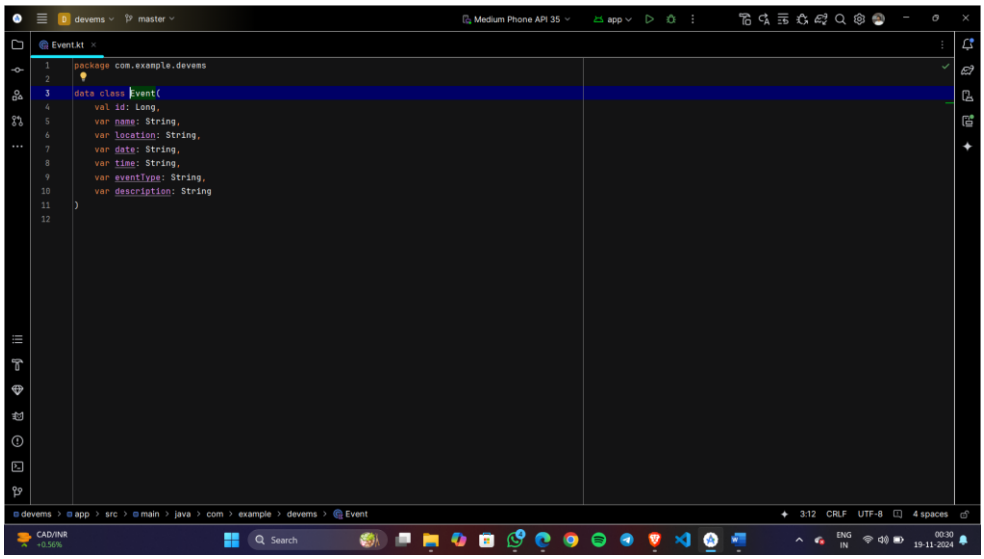
NotificationReceiver

The NotificationReceiver class is an integral part of the event management system, responsible for handling event reminder notifications. It extends BroadcastReceiver to listen for broadcast messages containing event-specific information such as the event ID and name. Upon receiving a broadcast, the class extracts this data and logs it for debugging purposes. It then uses a NotificationManager to create and display a notification. If the device is running Android Oreo or later, a notification channel is created to ensure compatibility with modern Android features. The notification itself is customized with a title, message, and an icon, while a PendingIntent is used to allow the user to interact with the notification and open the DashboardActivity when tapped. The notification is designed to have high priority and auto-cancel once interacted with, ensuring it catches the user's attention. The event ID is used as a unique identifier for each notification, allowing individual management of reminders. This class ensures that users receive timely notifications about their events, keeping them engaged and informed in a user-friendly manner.



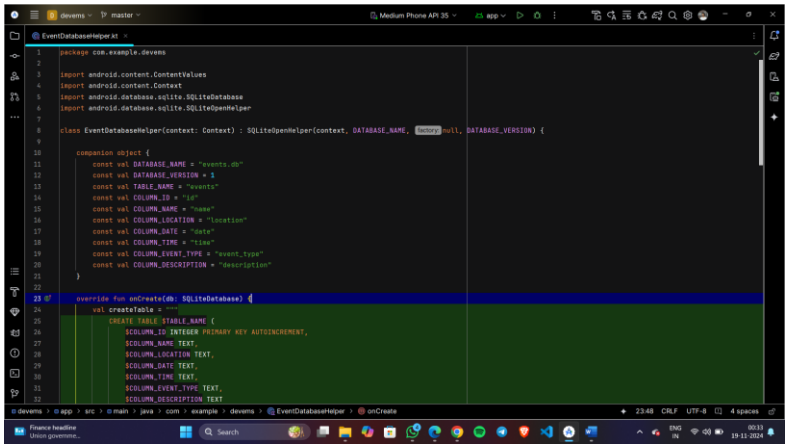
Event

The `Event` data class is a crucial component in the event management system of the app, encapsulating essential attributes like `id`, `name`, `location`, `date`, `time`, `eventType`, and `description` to define and manage events. The `id` serves as a unique identifier, while `name`, `location`, and `description` provide key details about the event. The `date` and `time` fields store the event's schedule, and `eventType` categorizes the event, enabling easy filtering. By structuring all event-related information into a single class, it simplifies database interactions, improves data organization, and ensures scalability and maintainability. This class facilitates smooth event handling by enabling efficient storage, retrieval, and manipulation of event data, contributing to a seamless user experience and robust system architecture.



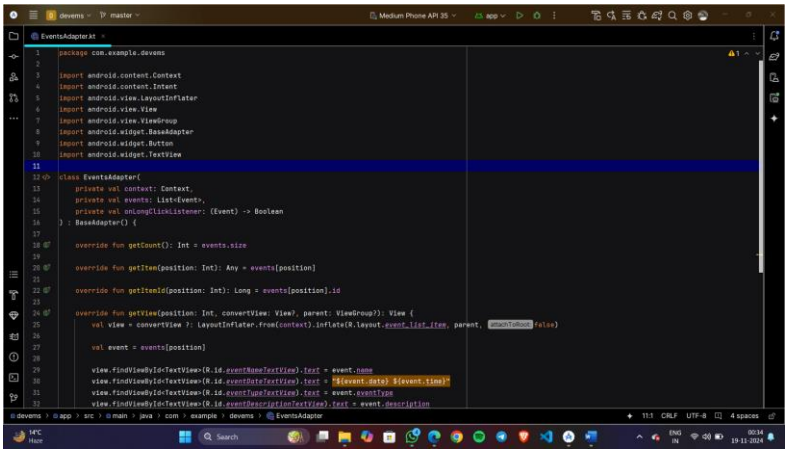
EventDatabaseHelper

The `EventDatabaseHelper` class manages the SQLite database for storing and handling events in the event management app, extending `SQLiteOpenHelper` to create and upgrade the database schema efficiently. It defines constants for the database name, version, table name, and column names for easy reference. In the `onCreate` method, it specifies an SQL query to create the events table with columns such as `id`, `name`, `location`, `date`, `time`, `event_type`, and `description`, while the `onUpgrade` method ensures smooth schema updates by dropping and recreating the table when the database version changes. The class includes CRUD operations: `insertEvent` adds events using `ContentValues` and `db.insert`; `getAllEvents` retrieves all events via a raw SQL query, returning them as a list of `Event` objects; `updateEvent` modifies event details using `db.update`; and `deleteEvent` removes events by `id` using `db.delete`, with a success flag. This class ensures robust local data storage, supports key app features like adding, displaying, updating, and deleting events, and leverages SQLite for persistent event storage, even across app closures or device restarts.



EventAdapter

The `EventsAdapter` class is a custom adapter for displaying event data in a `ListView` within the event management app, extending `BaseAdapter` to bind event data to views efficiently. It takes three parameters: the context, a list of events, and a lambda function `onLongClickListener` for handling long-press actions on event items. The `getCount` method specifies the number of events, `getItem` retrieves an event at a given position, and `getItemId` returns the unique ID of an event for operations like updating or deleting. The `getView` method inflates a custom layout (`event_list_item.xml`) for each list item, populating `TextView` elements with event details such as name, date, time, type, description, and location. It also configures a button (`btn_navigate_event`) to navigate users to the `UserNavigation` activity, passing the event's location via an `Intent` for interaction with maps. The class implements a long-press listener, triggering the provided `onLongClickListener` lambda for actions like editing or deleting events. By binding data to the UI and enabling interactivity such as location navigation and long-press actions, the `EventsAdapter` is essential for managing and displaying events within the app.

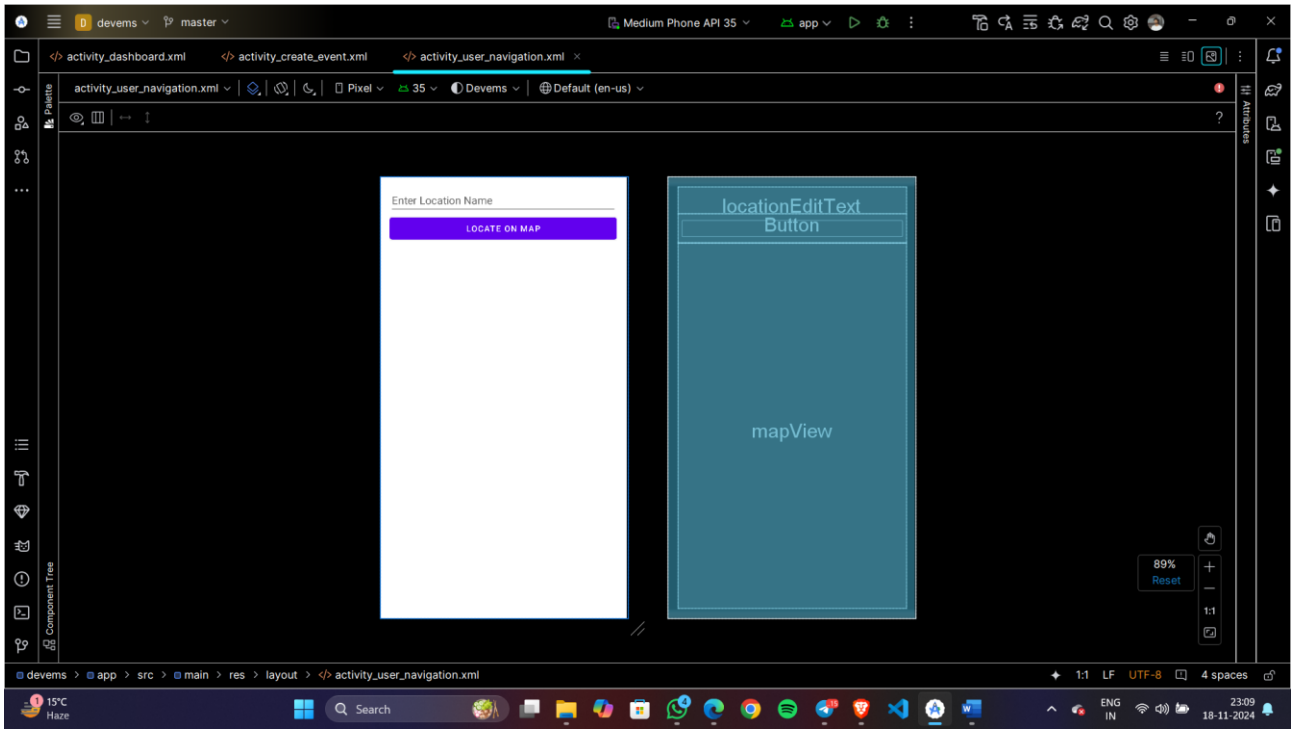


UserNavigation

The `UserNavigation` activity in the Android application enables users to search, view, and navigate to locations using Google Maps, providing an interactive navigation experience. It integrates Google Maps API to display a `MapView` alongside an `EditText` for entering location names and a `Button` for initiating searches. In the `onCreate` method, it initializes components like the input field, search button, and `MapView`, and uses `FusedLocationProviderClient` to manage location-based operations, including retrieving the user's current location. If a location is provided via an `Intent`, it pre-fills the input field and attempts navigation. The activity uses `Geocoder` to convert the entered location name into geographical coordinates (latitude and longitude), placing a marker on the map and adjusting the camera to focus on the location. Clicking the "Locate on Map" button constructs a `Uri` for Google Maps and opens the app using an `Intent`, displaying an error if the input is invalid or Google Maps is unavailable.

The `getCurrentLocation` method fetches the user’s current position, displays it with a marker, and zooms the map accordingly. The `onMapReady` callback ensures map readiness and displays the current location if permission is granted. Lifecycle methods (`onResume`, `onPause`, `onDestroy`, and `onLowMemory`) are overridden for proper `MapView` management. Permissions for `ACCESS_FINE_LOCATION` are checked and requested dynamically. The XML layout features a `LinearLayout` with an `EditText`, a `Button`, and a `MapView`, with layout configurations ensuring efficient space usage. Overall, the `UserNavigation` activity combines geocoding, real-time location updates, and seamless Google Maps integration, offering a robust and user-friendly navigation solution.

This activity not only enhances usability by enabling precise location searches but also provides flexibility with real-time updates and map interactivity. By integrating Google Maps' `Intent`-based navigation, users can seamlessly transition to the native Maps app for advanced navigation features. The use of `Geocoder` for address-to-coordinate conversion, along with permission handling for location services, ensures reliability and security. Its clean layout design and lifecycle management ensure smooth operation across different device states, making `UserNavigation` a core feature for any location-based functionality in the app.



AndroidManifest

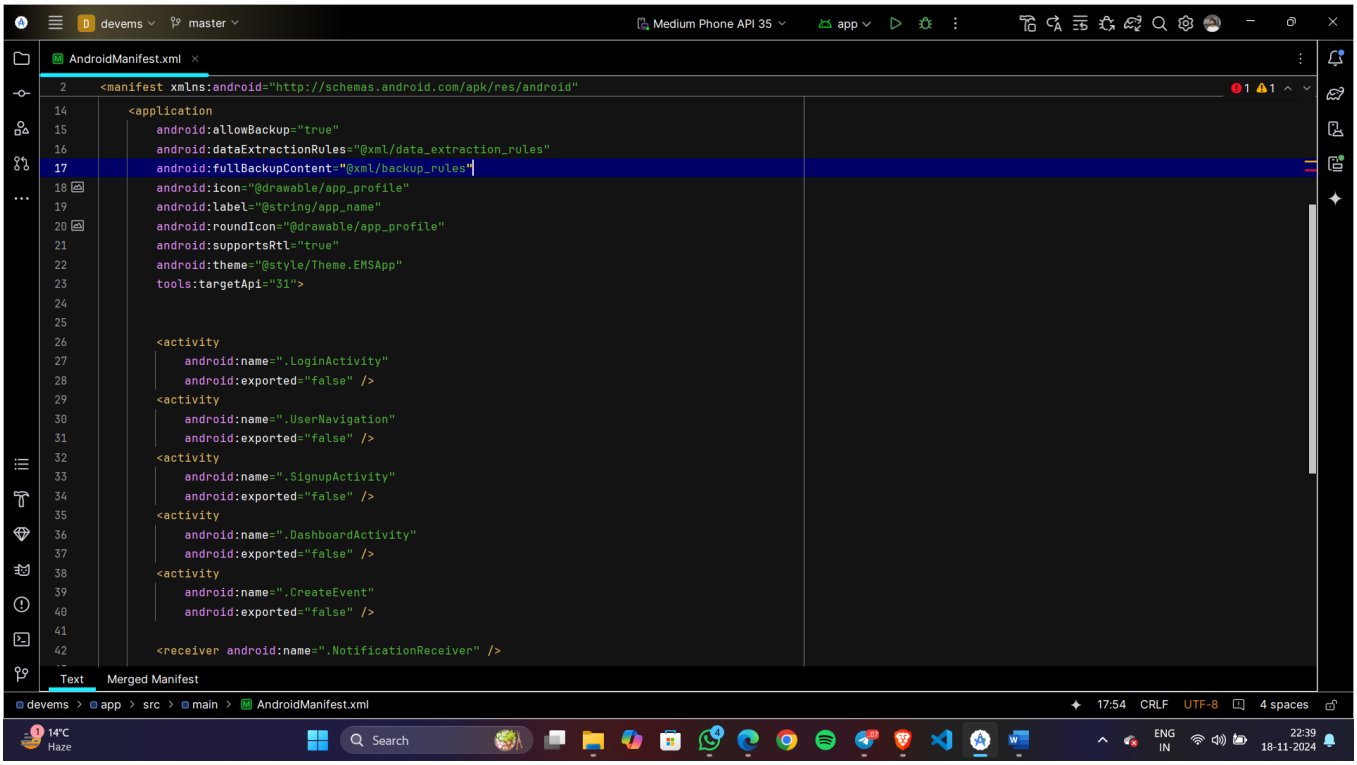
The provided Android manifest file serves as the configuration blueprint for an event management application. It defines the application's essential permissions, components, and metadata, ensuring the proper functioning of its features. This manifest begins by declaring several permissions to enable core functionalities. Permissions such as `INTERNET` and location-based permissions (`ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION`) are necessary for integrating Google Maps and location-based event management. Permissions for alarms (`SET_ALARM` and `SCHEDULE_EXACT_ALARM`) and notifications (`POST_NOTIFICATIONS` and `VIBRATE`) ensure that the app can schedule reminders and alert users effectively. The `RECEIVE_BOOT_COMPLETED` permission allows the app to restore scheduled alarms and notifications even after a device reboot, ensuring consistent functionality.

Within the `<application>` tag, the app's appearance and theme are specified, including its label, icons, and backup settings. The app is configured to support right-to-left (RTL) layouts for better global compatibility. The target API level is set using the `tools:targetApi` attribute, reflecting adherence to modern Android development standards.

The application declares several activities, each corresponding to a specific feature of the app. For example, `LoginActivity`, `SignupActivity`, and `DashboardActivity` represent user authentication and dashboard functionalities. `CreateEvent` allows users to create and manage events, while `SplashActivity` is configured as the launcher activity, displaying an introductory screen when the app starts. The `exported` attribute, set to `false` for most activities, restricts external apps from accessing these components, enhancing security. Only `SplashActivity` is marked `exported="true"` with an intent filter defining it as the app's entry point.

The manifest also includes a `NotificationReceiver`, which is a broadcast receiver responsible for handling notifications or alarms. Additionally, metadata for the Google Maps API key is declared using the `com.google.android.geo.API_KEY` tag, enabling the app to use map functionalities like location search and navigation.

In summary, this manifest file defines a secure, feature-rich structure for the event management app, ensuring the seamless integration of its functionalities while adhering to Android's best practices and security guidelines. It highlights a well-organized architecture for permissions, activities, and key components, forming the backbone of the app's operational framework.



Code – Module wise or Activity Wise

SplashActivity.kt

```
package com.example.devems
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.os.Handler
import android.widget.VideoView
import androidx.appcompat.app.AppCompatActivity


class SplashActivity : AppCompatActivity() {
    private val SPLASH_DURATION: Long = 1500
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
        val videoView: VideoView = findViewById(R.id.splashVideoView)
        val videoUri =
Uri.parse("android.resource://${packageName}/raw/event_management_system")
        videoView.setVideoURI(videoUri)
        videoView.start()
        Handler().postDelayed({
            startActivity(Intent(this, LoginActivity::class.java))
            finish()
        }, SPLASH_DURATION)
    }
}
```

activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<   ndroid.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashActivity">

    <VideoView
        android:id="@+id/splashVideoView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_centerInParent="true" />

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

LoginActivity.kt

```
package com.example.devems
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class LoginActivity : AppCompatActivity() {
    private lateinit var idEditText: EditText
    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var loginButton: Button
    private lateinit var signupTextView: TextView
    private lateinit var dbHelper: UserDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        dbHelper = UserDatabaseHelper(this)

        idEditText = findViewById(R.id.idEditText)
        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        loginButton = findViewById(R.id.loginButton)
        signupTextView = findViewById(R.id.signupTextView)

        loginButton.setOnClickListener { handleLogin() }

        signupTextView.setOnClickListener {
            startActivity(Intent(this, SignupActivity::class.java))
        }
    }

    private fun handleLogin() {
        val idText = idEditText.text.toString().trim()
        val username = usernameEditText.text.toString().trim()
        val password = passwordEditText.text.toString().trim()

        if (idText.isEmpty() || username.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please enter all details", Toast.LENGTH_SHORT).show()
            return
        }
```

```
        val id = idText.toIntOrNull()
        if (id == null) {
            Toast.makeText(this, "ID must be a valid number", Toast.LENGTH_SHORT).show()
            return
        }

        val isValidUser = dbHelper.validateUser(id, username, password)
        if (isValidUser) {
            val intent = Intent(this, DashboardActivity::class.java)
            startActivity(intent)
            finish()
        } else {
            Toast.makeText(this, "Invalid credentials", Toast.LENGTH_SHORT).show()
        }
    }
}
```

activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center"
    android:background="@color/colorPrimaryDark">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        app:cardBackgroundColor="@android:color/white"
        app:cardCornerRadius="16dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="16dp"
            android:background="@drawable/curved_background">

            <TextView
                android:id="@+id/titleLOGIN"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="LOGIN"
                android:textSize="24sp"
```

```
    android:textColor="@color/colorPrimary"
    android:textStyle="bold"
    android:layout_marginBottom="16dp"
    android:layout_gravity="center" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_numbers_24"
        app:tint="@color/colorPrimary" />

    <EditText
        android:id="@+id/idEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="ID"
        android:inputType="number" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_person_24"
        app:tint="@color/colorPrimary" />

    <EditText
        android:id="@+id/usernameEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Username" />
</LinearLayout>
```

```
<LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginBottom="16dp">

            <ImageView
               android:layout_width="24dp"
               android:layout_height="40dp"
               android:src="@drawable/baseline_lock_24"
               app:tint="@color/colorPrimary" />

            <EditText
               android:id="@+id/passwordEditText"
               android:layout_width="0dp"
               android:layout_height="wrap_content"
               android:layout_weight="1"
               android:hint="Password"
               android:inputType="textPassword" />
         </LinearLayout>

         <Button
            android:id="@+id/loginButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Login" />

         <TextView
            android:id="@+id/signupTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Don't have an account? Sign Up"
            android:textColor="@android:color/holo_blue_dark"
            android:layout_gravity="center"
            android:layout_marginTop="16dp" />

      </LinearLayout>
   </androidx.cardview.widget.CardView>

</LinearLayout>
```

SignupActivity.kt

```
package com.example.devems

import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class SignupActivity : AppCompatActivity() {
    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var confirmPasswordEditText: EditText
    private lateinit var enterCodeEditText: EditText
    private lateinit var idEditText: EditText
    private lateinit var phoneNumberEditText: EditText
    private lateinit var signupButton: Button

    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_signup)

        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        confirmPasswordEditText = findViewById(R.id.confirmPasswordEditText)
        enterCodeEditText = findViewById(R.id.enterCodeEditText)
        idEditText = findViewById(R.id.idEditText)
        phoneNumberEditText = findViewById(R.id.phoneNumberEditText)
        signupButton = findViewById(R.id.signupButton)

        signupButton.setOnClickListener { handleSignup() }
    }

    private fun handleSignup() {
        val username = usernameEditText.text.toString().trim()
        val password = passwordEditText.text.toString().trim()
        val confirmPassword = confirmPasswordEditText.text.toString().trim()
        val enterCode = enterCodeEditText.text.toString().trim()
        val idText = idEditText.text.toString().trim()
        val phoneNumber = phoneNumberEditText.text.toString().trim()

        if (username.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() || idText.isEmpty()
|| phoneNumber.isEmpty()) {
            Toast.makeText(this, "Please enter all details", Toast.LENGTH_SHORT).show()
```

```
            return
        }

        if (password != confirmPassword) {
            Toast.makeText(this, "Passwords do not match", Toast.LENGTH_SHORT).show()
            return
        }

        val id = idText.toIntOrNull()
        if (id == null) {
            Toast.makeText(this, "ID must be a valid number", Toast.LENGTH_SHORT).show()
            return
        }

        if (enterCode.isEmpty() || enterCode != "4444") {
            Toast.makeText(this, "Invalid admin code", Toast.LENGTH_SHORT).show()
            return
        }

        val dbHelper = UserDatabaseHelper(this)
        if (dbHelper.isIdExists(id)) {
            Toast.makeText(this, "ID already exists", Toast.LENGTH_SHORT).show()
            return
        }

        val isAdded = dbHelper.addUser(id, username, password, phoneNumber)
        if (isAdded) {
            Toast.makeText(this, "Signup successful! Please login.", Toast.LENGTH_SHORT).show()
            startActivity(Intent(this@SignupActivity, LoginActivity::class.java))
            finish()
        } else {
            Toast.makeText(this, "Signup failed, please try again.", Toast.LENGTH_SHORT).show()
        }
    }
}
```

activity_signup.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center"
    android:background="@color/colorPrimaryDark">

    <androidx.cardview.widget.CardView
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginBottom="16dp"
app:cardBackgroundColor="@android:color/white"
app:cardCornerRadius="16dp">

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:padding="16dp"
   android:background="@drawable/curved_background">

   <TextView
      android:id="@+id/titleLOGIN"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="SIGN UP"
      android:textSize="24sp"
      android:textColor="@color/colorPrimary"
      android:textStyle="bold"
      android:layout_marginBottom="16dp"
      android:layout_gravity="center"/>

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:layout_marginBottom="8dp">

      <ImageView
         android:layout_width="24dp"
         android:layout_height="40dp"
         android:src="@drawable/baseline_numbers_24"
         app:tint="@color/colorPrimary" />

      <EditText
         android:id="@+id/idEditText"
         android:layout_width="0dp"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:hint="ID (Must be Unique)" />
   </LinearLayout>

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
```

```
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_person_24"
        app:tint="@color/colorPrimary" />

    <EditText
        android:id="@+id/usernameEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Username" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_phone_24"
        app:tint="@color/colorPrimary" />

    <EditText
        android:id="@+id/phoneNumberEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Phone Number"
        android:inputType="phone" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_lock_24"
        app:tint="@color/colorPrimary" />
```

```
    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Password"
        android:inputType="textPassword" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="8dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_password_24"
        app:tint="@color/colorPrimary" />

    <EditText
        android:id="@+id/confirmPasswordEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Confirm Password"
        android:inputType="textPassword" />
</LinearLayout>

<Button
    android:id="@+id/signupButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Sign Up" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginBottom="16dp">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="40dp"
        android:src="@drawable/baseline_code_24"
        app:tint="@color/colorPrimary" />
```

```
            <EditText
                android:id="@+id/enterCodeEditText"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:hint="Enter Code"
                android:inputType="number" />
        </LinearLayout>

    </LinearLayout>
</androidx.cardview.widget.CardView>

</LinearLayout>
```

UserDatabaseHelper.kt

```
package com.example.devems

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME,
null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "EMS.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "users"
        private const val COLUMN_ID = "id"
        private const val COLUMN_USERNAME = "username"
        private const val COLUMN_PASSWORD = "password"
        private const val COLUMN_PHONE_NUMBER = "phone_number"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTableQuery = """
            CREATE TABLE $TABLE_NAME (
                $COLUMN_ID INTEGER PRIMARY KEY,
                $COLUMN_USERNAME TEXT,
                $COLUMN_PASSWORD TEXT,
                $COLUMN_PHONE_NUMBER TEXT
            )
        """
        db.execSQL(createTableQuery)
    }
```

```
    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun isIdExists(id: Int): Boolean {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID = ?"
        val cursor = db.rawQuery(query, arrayOf(id.toString()))
        val exists = cursor.count > 0
        cursor.close()
        db.close()
        return exists
    }

    fun addUser(id: Int, username: String, password: String, phoneNumber: String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_ID, id)
            put(COLUMN_USERNAME, username)
            put(COLUMN_PASSWORD, password)
            put(COLUMN_PHONE_NUMBER, phoneNumber)
        }

        val result = db.insert(TABLE_NAME, null, values)
        db.close()
        return result != -1L
    }

    fun validateUser(id: Int, username: String, password: String): Boolean {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID = ? AND
$COLUMN_USERNAME = ? AND $COLUMN_PASSWORD = ?"
        val cursor = db.rawQuery(query, arrayOf(id.toString(), username, password))
        val isValid = cursor.count > 0
        cursor.close()
        db.close()
        return isValid
    }
}
```

DashboardActivity.kt

```
package com.example.devems

import android.content.Intent
import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import androidx.viewpager2.widget.ViewPager2
import com.google.android.material.floatingactionbutton.FloatingActionButton
import com.google.android.material.tabs.TabLayout
import com.google.android.material.tabs.TabLayoutMediator
import android.app.AlertDialog
import android.view.LayoutInflater
import android.widget.Button
import android.widget.ImageButton
import android.widget.RatingBar
import android.widget.Toast

class DashboardActivity : AppCompatActivity() {
    private lateinit var viewPager: ViewPager2
    private lateinit var tabLayout: TabLayout
    private lateinit var fabMain: FloatingActionButton
    private lateinit var fabCreateEvent: FloatingActionButton
    private lateinit var fabMaps: FloatingActionButton
    private lateinit var fabMenu: View
    private var isFabMenuOpen = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dashboard)

        viewPager = findViewById(R.id.viewPager)
        tabLayout = findViewById(R.id.tabLayout)
        fabMain = findViewById(R.id.fabMain)
        fabCreateEvent = findViewById(R.id.fabCreateEvent)
        fabMaps = findViewById(R.id.fabMaps)
        fabMenu = findViewById(R.id.fabMenu)



        viewPager.adapter = ViewPagerAdapter(this)

        TabLayoutMediator(tabLayout, viewPager) { tab, position ->
            tab.text = when(position) {
                0 -> "Today"
                1 -> "All Events"
                2 -> "Completed"
                else -> null
```

```
            }
        }.attach()

        fabMain.setOnClickListener {
            toggleFabMenu()
        }

        fabCreateEvent.setOnClickListener {
            val intent = Intent(this, CreateEvent::class.java)
            startActivity(intent)
            toggleFabMenu()
        }

        fabMaps.setOnClickListener {
            val intent = Intent(this, UserNavigation::class.java)
            startActivity(intent)
            toggleFabMenu()
        }

        findViewById<ImageButton>(R.id.infoButton).setOnClickListener {
            showInfoDialog()
        }

        findViewById<ImageButton>(R.id.ratingButton).setOnClickListener {
            showRatingDialog()
        }

    }

    private fun toggleFabMenu() {
        if (isFabMenuOpen) {
            fabMenu.visibility = View.GONE
            isFabMenuOpen = false
        } else {
            fabMenu.visibility = View.VISIBLE
            isFabMenuOpen = true
        }
    }

    private fun showInfoDialog() {
        val dialogView = LayoutInflater.from(this).inflate(R.layout.dialog_info, null)
        val builder = AlertDialog.Builder(this)
            .setView(dialogView)
            .setPositiveButton("OK", null)
        builder.create().show()
    }

    private fun showRatingDialog() {
```

```
        val dialogView = LayoutInflater.from(this).inflate(R.layout.dialog_rate, null)
        val ratingBar = dialogView.findViewById<RatingBar>(R.id.ratingBar)

        val builder = AlertDialog.Builder(this)
            .setView(dialogView)
            .setCancelable(true)

        val dialog = builder.create()
        dialog.show()

        dialogView.findViewById<Button>(R.id.rateButton).setOnClickListener {
            val rating = ratingBar.rating
            val message = when (rating) {
                1f -> "We will try our best to impress you next time!"
                2f -> "Thank you! We appreciate your feedback."
                3f -> "Thanks for your feedback!"
                4f -> "Thank you! We are glad you liked it."
                5f -> "Thank you so much! We're thrilled to impress you!"
                else -> "Thank you for rating!"
            }
            Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
            dialog.dismiss()
        }
    }

}
```

activity_dashboard.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DashboardActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimaryDark"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <TextView
            android:id="@+id/emsLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_marginStart="16dp"
        android:text="EMS"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center_vertical" />

    <ImageButton
        android:id="@+id/infoButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/baseline_info_24"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:layout_gravity="right|center_vertical"
        android:contentDescription="Toolbar button"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="16dp" />

    <ImageButton
        android:id="@+id/ratingButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/baseline_star_24"
        android:background="?attr/selectableItemBackgroundBorderless"
        android:layout_gravity="right|center_vertical"
        android:contentDescription="Toolbar Rating Button"
        android:layout_toStartOf="@id/infoButton"
        android:layout_marginEnd="16dp" />

</androidx.appcompat.widget.Toolbar>

<com.google.android.material.tabs.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/toolbar"
    app:tabMode="fixed"
    app:tabGravity="fill"
    app:tabTextColor="@android:color/white"
    app:tabSelectedTextColor="@android:color/white"
    app:tabIndicatorColor="@android:color/white"
    android:background="@color/colorPrimary" />

<androidx.viewpager2.widget.ViewPager2
    android:id="@+id/viewPager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/tabLayout" />
```

```
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabMain"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_margin="16dp"
        android:src="@drawable/baseline_add_24"
        android:tint="@android:color/white"
        app:backgroundTint="@color/colorPrimary" />

    <LinearLayout
        android:id="@+id/fabMenu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/fabMain"
        android:layout_alignParentEnd="true"
        android:visibility="gone"
        android:orientation="vertical"
        android:padding="8dp">

        <com.google.android.material.floatingactionbutton.FloatingActionButton
            android:id="@+id/fabCreateEvent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:src="@drawable/baseline_event_available_24"
            android:tint="@android:color/white"
            app:backgroundTint="@color/colorPrimary"
            android:contentDescription="Create Event" />

        <com.google.android.material.floatingactionbutton.FloatingActionButton
            android:id="@+id/fabMaps"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:src="@drawable/baseline_map_24"
            android:tint="@android:color/white"
            app:backgroundTint="@color/colorPrimary"
            android:contentDescription="Search User" />
    </LinearLayout>

</RelativeLayout>
```

ViewPageAdapter.kt

```
package com.example.devems

import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentActivity
import androidx.viewpager2.adapter.FragmentStateAdapter

class ViewPagerAdapter(activity: FragmentActivity) : FragmentStateAdapter(activity) {
    override fun getItemCount(): Int = 3

    override fun createFragment(position: Int): Fragment {
        return when(position) {
            0 -> TodayEventsFragment()
            1 -> AllEventsFragment()
            2 -> CompletedEventsFragment()
            else -> throw IllegalArgumentException("Invalid position $position")
        }
    }
}
```

AllEventsFragment.kt

```
package com.example.devems

class AllEventsFragment : BaseEventsFragment() {
    override fun loadEvents() {
        allEvents.clear()
        allEvents.addAll(dbHelper.getAllEvents())
        filteredEvents.clear()
        filteredEvents.addAll(allEvents)
        eventsAdapter.notifyDataSetChanged()
    }
}
```

fragment_all_events.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/eventsListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/emptyView"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="No events found"
        android:visibility="gone" />

</FrameLayout>
```

TodayEventsFragment.kt

```
package com.example.devems

import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

class TodayEventsFragment : BaseEventsFragment() {
    override fun loadEvents() {
        allEvents.clear()
        val today = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(Date())
        allEvents.addAll(dbHelper.getAllEvents().filter { it.date == today })
        filteredEvents.clear()
        filteredEvents.addAll(allEvents)
        eventsAdapter.notifyDataSetChanged()
    }
}
```

fragment_today_events.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/eventsListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/emptyView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="No events found"
        android:visibility="gone" />

</FrameLayout>
```

CompletedEventsFragment.kt

```
package com.example.devems

import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

class CompletedEventsFragment : BaseEventsFragment() {
    override fun loadEvents() {
        allEvents.clear()
        val today = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).parse(
            SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(Date())
        )

        allEvents.addAll(dbHelper.getAllEvents().filter { event ->
            val eventDate = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).parse(event.date)
            eventDate?.before(today) ?: false
        })
        filteredEvents.clear()
        filteredEvents.addAll(allEvents)
        eventsAdapter.notifyDataSetChanged()
    }
}
```

fragment_completed_events.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/eventsListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/emptyView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="No events found"
        android:visibility="gone" />

</FrameLayout>
```

BaseEventsFragment.kt

```
package com.example.devems

import android.app.DatePickerDialog
import android.app.TimePickerDialog
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.ListView
import android.widget.SearchView
import android.widget.TextView
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import java.util.Calendar
import java.util.Locale

abstract class BaseEventsFragment : Fragment() {
    private lateinit var eventsListView: ListView
    private lateinit var searchView: SearchView
    protected lateinit var dbHelper: EventDatabaseHelper
    protected val allEvents = mutableListOf<Event>()
    protected val filteredEvents = mutableListOf<Event>()
    protected lateinit var eventsAdapter: EventsAdapter

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_events, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        eventsListView = view.findViewById(R.id.eventsListView)
        searchView = view.findViewById(R.id.searchView)
        dbHelper = EventDatabaseHelper(requireContext())
        val emptyView: TextView = view.findViewById(R.id.emptyView)
        eventsListView.emptyView = emptyView

        eventsAdapter = EventsAdapter(requireContext(), filteredEvents) { event ->
            showEditDialog(event)
            true
        }
```

```
    eventsListView.adapter = eventsAdapter

    loadEvents()


    searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
        override fun onQueryTextSubmit(query: String?): Boolean {
            filterEvents(query)
            return true
        }

        override fun onQueryTextChange(newText: String?): Boolean {
            if (newText.isNullOrEmpty()) {
                filterEvents("")
            } else {
                filterEvents(newText)
            }
            return true
        }
    })
}

abstract fun loadEvents()

private fun filterEvents(query: String?) {
    filteredEvents.clear()
    if (query.isNullOrEmpty()) {
        filteredEvents.addAll(allEvents)
    } else {
        val lowerCaseQuery = query.lowercase(Locale.getDefault())
        filteredEvents.addAll(allEvents.filter { event ->
            event.name.lowercase(Locale.getDefault()).contains(lowerCaseQuery)
        })
    }
    eventsAdapter.notifyDataSetChanged()
}

private fun showEditDialog(event: Event) {
    val dialogView = LayoutInflater.from(requireContext())
        .inflate(R.layout.dialog_edit_event, null)

    val editTextName = dialogView.findViewById<EditText>(R.id.editTextEventName)
    val editTextLocation = dialogView.findViewById<EditText>(R.id.editTextEventLocation)
    val editTextDate = dialogView.findViewById<EditText>(R.id.editTextEventDate)
    val editTextTime = dialogView.findViewById<EditText>(R.id.editTextEventTime)
    val editTextType = dialogView.findViewById<EditText>(R.id.editTextEventType)
    val editTextDescription =
```

```
dialogView.findViewById<EditText>(R.id.editTextEventDescription)

        editTextName.setText(event.name)
        editTextLocation.setText(event.location)
        editTextDate.setText(event.date)
        editTextTime.setText(event.time)
        editTextType.setText(event.eventType)
        editTextDescription.setText(event.description)

        editTextDate.setOnClickListener {
            val calendar = Calendar.getInstance()
            val datePickerDialog = DatePickerDialog(
                requireContext(),
                { _, year, month, day ->
                    val selectedDate = String.format(
                        Locale.getDefault(),
                        "%02d/%02d/%04d",
                        day, month + 1, year
                    )
                    editTextDate.setText(selectedDate)
                },
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH)
            )
            datePickerDialog.show()
        }

        editTextTime.setOnClickListener {
            val calendar = Calendar.getInstance()
            val timePickerDialog = TimePickerDialog(
                requireContext(),
                { _, hour, minute ->
                    val selectedTime = String.format(
                        Locale.getDefault(),
                        "%02d:%02d",
                        hour, minute
                    )
                    editTextTime.setText(selectedTime)
                },
                calendar.get(Calendar.HOUR_OF_DAY),
                calendar.get(Calendar.MINUTE),
                true
            )
            timePickerDialog.show()
        }

        val dialog = AlertDialog.Builder(requireContext())
```

```
            .setView(dialogView)
            .setTitle("Edit Event")
            .setPositiveButton("Save") { _, _ ->
                val updatedEvent = Event(
                    id = event.id,
                    name = editTextName.text.toString(),
                    location = editTextLocation.text.toString(),
                    date = editTextDate.text.toString(),
                    time = editTextTime.text.toString(),
                    eventType = editTextType.text.toString(),
                    description = editTextDescription.text.toString()
                )

                dbHelper.updateEvent(updatedEvent)
                loadEvents()
            }
            .setNegativeButton("Cancel", null)
            .setNeutralButton("Delete") { _, _ ->
                dbHelper.deleteEvent(event.id)
                loadEvents()
            }
            .create()

        dialog.show()
    }

    override fun onResume() {
        super.onResume()
        loadEvents()
    }

    override fun onDestroy() {
        super.onDestroy()
        dbHelper.close()
    }
}
```

fragment_events.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <SearchView
        android:id="@+id/searchView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false"
```

```
        android:queryHint="Search events"
        android:background="@drawable/curved_background"/>

    <ListView
        android:id="@+id/eventsListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="0.5dp"
        android:layout_below="@id/searchView" />

    <TextView
        android:id="@+id/emptyView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:text="No events found"
        android:textSize="18sp"
        android:visibility="gone" />
</RelativeLayout>
```

CreateEvents.kt

```
package com.example.devems

import android.app.DatePickerDialog
import android.app.TimePickerDialog
import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import java.util.*
import android.app.AlarmManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.os.Build
import android.widget.Toast
import java.text.SimpleDateFormat
import android.util.Log
import android.provider.Settings
import android.app.AlertDialog
import android.view.LayoutInflater

class CreateEvent : AppCompatActivity() {

    private lateinit var dbHelper: EventDatabaseHelper
    private lateinit var inputName: EditText
```

```
private lateinit var inputLocation: EditText
private lateinit var inputDate: EditText
private lateinit var inputTime: EditText
private lateinit var eventTypeSpinner: Spinner
private lateinit var inputDescription: EditText
private lateinit var btnSaveEvent: Button
private lateinit var eventsListView: ListView
private lateinit var eventsAdapter: EventsAdapter
private val events = mutableListOf<Event>()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_create_event)

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        val alarmManager = getSystemService(AlarmManager::class.java)
        if (!alarmManager.canScheduleExactAlarms()) {
            val intent = Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM)
            startActivity(intent)
        }
    }

    dbHelper = EventDatabaseHelper(this)

    inputName = findViewById(R.id.inputEventName)
    inputLocation = findViewById(R.id.inputLocation)
    inputDate = findViewById(R.id.inputDate)
    inputTime = findViewById(R.id.inputTime)
    eventTypeSpinner = findViewById(R.id.eventTypeSpinner)
    inputDescription = findViewById(R.id.inputDescription)
    btnSaveEvent = findViewById(R.id.btnSaveEvent)
    eventsListView = findViewById(R.id.eventsListView)

    eventsAdapter = EventsAdapter(this, events) { event ->
        showEditDialog(event)
        true
    }
    eventsListView.adapter = eventsAdapter

    loadEvents()

    setupDatePicker()
    setupTimePicker()
    setupSpinner()
    requestNotificationPermission()

    btnSaveEvent.setOnClickListener {
        saveEvent()
```

```
        }
    }


    private fun loadEvents() {
        Log.d("CreateEvent", "Loading events from database...")

        events.clear()
        val loadedEvents = dbHelper.getAllEvents()

        if (loadedEvents.isNotEmpty()) {
            events.addAll(loadedEvents)
            eventsAdapter.notifyDataSetChanged()
            Log.d("CreateEvent", "Events loaded successfully.")
        } else {
            Log.d("CreateEvent", "No events found or error retrieving events.")
        }
    }


    private fun setupDatePicker() {
        inputDate.setOnClickListener {
            val calendar = Calendar.getInstance()
            val year = calendar.get(Calendar.YEAR)
            val month = calendar.get(Calendar.MONTH)
            val day = calendar.get(Calendar.DAY_OF_MONTH)

            val datePickerDialog = DatePickerDialog(this, { _, selectedYear, selectedMonth,
selectedDay ->
                inputDate.setText("$selectedDay/${selectedMonth + 1}/$selectedYear")
            }, year, month, day)

            datePickerDialog.show()
        }
    }

    private fun setupTimePicker() {
        inputTime.setOnClickListener {
            val calendar = Calendar.getInstance()
            val hour = calendar.get(Calendar.HOUR_OF_DAY)
            val minute = calendar.get(Calendar.MINUTE)

            val timePickerDialog = TimePickerDialog(this, { _, selectedHour, selectedMinute ->
                inputTime.setText("$selectedHour:$selectedMinute")
            }, hour, minute, true)

            timePickerDialog.show()
        }
```

```
    }

    private fun setupSpinner() {
        val eventTypes = arrayOf("Birthday", "Marriage", "Festivals", "Weddings", "Party",
"Conferences")
        val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_dropdown_item,
eventTypes)
        eventTypeSpinner.adapter = adapter
    }

    private fun saveEvent() {
        val name = inputName.text.toString()
        val location = inputLocation.text.toString()
        val date = inputDate.text.toString()
        val time = inputTime.text.toString()
        val eventType = eventTypeSpinner.selectedItem.toString()
        val description = inputDescription.text.toString()

        val eventId = dbHelper.insertEvent(name, location, date, time, eventType, description)

        if (eventId != -1L) {
            Toast.makeText(this, "Event saved successfully!", Toast.LENGTH_SHORT).show()
            loadEvents()
            scheduleNotification(eventId, name, date, time)
        } else {
            Toast.makeText(this, "Error saving event", Toast.LENGTH_SHORT).show()
        }
    }


    private fun scheduleNotification(eventId: Long, eventName: String, date: String, time: String) {
        try {
            val alarmManager = getSystemService(Context.ALARM_SERVICE) as? AlarmManager
            val intent = Intent(this, NotificationReceiver::class.java).apply {
                putExtra("eventId", eventId)
                putExtra("eventName", eventName)
            }

            val pendingIntent = PendingIntent.getBroadcast(
                this, eventId.toInt(), intent,
                PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE
            )

            val dateTimeString = "$date $time"
            val dateTimeFormat = SimpleDateFormat("dd/MM/yyyy HH:mm", Locale.getDefault())
            val dateTime = dateTimeFormat.parse(dateTimeString)

            if (dateTime != null && alarmManager != null) {
```

```
            alarmManager.setExact(AlarmManager.RTC_WAKEUP, dateTime.time, pendingIntent)
            Toast.makeText(this, "Notification scheduled for $dateTimeString",
Toast.LENGTH_SHORT).show()
            Log.d("scheduleNotification", "Notification scheduled for $dateTimeString")
        } else {
            Log.e("scheduleNotification", "Failed to parse date or time")
        }
    } catch (e: Exception) {
        Log.e("scheduleNotification", "Error scheduling notification", e)
    }
}


private fun requestNotificationPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        requestPermissions(arrayOf(android.Manifest.permission.POST_NOTIFICATIONS), 101)
    }
}

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>,
grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == 101) {
        if ((grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
            Toast.makeText(this, "Notification permission granted", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(this, "Notification permission denied", Toast.LENGTH_SHORT).show()
        }
    }
}

private fun showEditDialog(event: Event) {
    val dialogView = LayoutInflater.from(this).inflate(R.layout.dialog_edit_event, null)

    val editTextName = dialogView.findViewById<EditText>(R.id.editTextEventName)
    val editTextLocation = dialogView.findViewById<EditText>(R.id.editTextEventLocation)
    val editTextDate = dialogView.findViewById<EditText>(R.id.editTextEventDate)
    val editTextTime = dialogView.findViewById<EditText>(R.id.editTextEventTime)
    val editTextType = dialogView.findViewById<EditText>(R.id.editTextEventType)
    val editTextDescription =
dialogView.findViewById<EditText>(R.id.editTextEventDescription)

    editTextName.setText(event.name)
    editTextLocation.setText(event.location)
    editTextDate.setText(event.date)
    editTextTime.setText(event.time)
    editTextType.setText(event.eventType)
```

```
        editTextDescription.setText(event.description)

        editTextDate.setOnClickListener {
            val calendar = Calendar.getInstance()
            val dateStr = event.date.split("/")
            if (dateStr.size == 3) {
                calendar.set(dateStr[2].toInt(), dateStr[1].toInt() - 1, dateStr[0].toInt())
            }

            DatePickerDialog(this,
                { _, year, month, day ->
                    editTextDate.setText("$day/${month + 1}/$year")
                },
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH)
            ).show()
        }

        editTextTime.setOnClickListener {
            val calendar = Calendar.getInstance()
            val timeStr = event.time.split(":")
            if (timeStr.size == 2) {
                calendar.set(Calendar.HOUR_OF_DAY, timeStr[0].toInt())
                calendar.set(Calendar.MINUTE, timeStr[1].toInt())
            }

            TimePickerDialog(this,
                { _, hour, minute ->
                    editTextTime.setText("$hour:$minute")
                },
                calendar.get(Calendar.HOUR_OF_DAY),
                calendar.get(Calendar.MINUTE),
                true
            ).show()
        }

        val dialog = AlertDialog.Builder(this)
            .setView(dialogView)
            .setTitle("Edit Event")
            .setPositiveButton("Save", null)
            .setNegativeButton("Cancel", null)
            .setNeutralButton("Delete", null)
            .create()

        dialog.setOnShowListener { dialogInterface ->
            val positiveButton = (dialogInterface as
AlertDialog).getButton(AlertDialog.BUTTON_POSITIVE)
```

```
        val neutralButton = dialogInterface.getButton(AlertDialog.BUTTON_NEUTRAL)

        positiveButton.setOnClickListener {
            val updatedEvent = Event(
                id = event.id,
                name = editTextName.text.toString(),
                location = editTextLocation.text.toString(),
                date = editTextDate.text.toString(),
                time = editTextTime.text.toString(),
                eventType = editTextType.text.toString(),
                description = editTextDescription.text.toString()
            )

            if (dbHelper.updateEvent(updatedEvent)) {
                Toast.makeText(this, "Event updated successfully!", Toast.LENGTH_SHORT).show()
                cancelExistingNotification(event.id)
                scheduleNotification(event.id, updatedEvent.name, updatedEvent.date,
updatedEvent.time)
                loadEvents()
                dialog.dismiss()
            } else {
                Toast.makeText(this, "Failed to update event", Toast.LENGTH_SHORT).show()
            }
        }

        neutralButton.setOnClickListener {
            AlertDialog.Builder(this)
                .setTitle("Delete Event")
                .setMessage("Are you sure you want to delete this event?")
                .setPositiveButton("Yes") { _, _ ->
                    if (dbHelper.deleteEvent(event.id)) {
                        Toast.makeText(this, "Event deleted successfully!",
Toast.LENGTH_SHORT).show()
                        cancelExistingNotification(event.id)
                        loadEvents()
                        dialog.dismiss()
                    } else {
                        Toast.makeText(this, "Failed to delete event", Toast.LENGTH_SHORT).show()
                    }
                }
                .setNegativeButton("No", null)
                .show()
        }
    }

    dialog.show()
  }
```

```
    private fun cancelExistingNotification(eventId: Long) {
        val alarmManager = getSystemService(Context.ALARM_SERVICE) as AlarmManager
        val intent = Intent(this, NotificationReceiver::class.java)
        val pendingIntent = PendingIntent.getBroadcast(
            this,
            eventId.toInt(),
            intent,
            PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE
        )
        alarmManager.cancel(pendingIntent)
    }

}
```

activity_create_events.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:padding="16dp"
    android:background="@color/colorPrimaryDark"
    tools:context=".CreateEvent">

    <TextView
        android:id="@+id/titleEMS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="CREATE EVENT"
        android:textSize="24sp"
        android:textColor="@android:color/white"
        android:textStyle="bold"
        android:layout_marginBottom="16dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_below="@id/titleEMS"
        android:layout_centerInParent="true">

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            app:cardBackgroundColor="@android:color/white"
```

```
app:cardCornerRadius="16dp">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="@drawable/curved_background">

    <!-- Event Name with Vector -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageView
            android:layout_width="24dp"
            android:layout_height="40dp"
            android:src="@drawable/baseline_person_24"
            app:tint="@color/colorPrimary" />

        <EditText
            android:id="@+id/inputEventName"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Event Name" />
    </LinearLayout>

    <!-- Location with Vector -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">

        <ImageView
            android:layout_width="24dp"
            android:layout_height="40dp"
            android:src="@drawable/baseline_location_on_24"
            app:tint="@color/colorPrimary"/>

        <EditText
            android:id="@+id/inputLocation"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Location" />
```

```
</LinearLayout>

<!-- Date with Vector -->
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:layout_marginTop="8dp">

   <ImageView
      android:layout_width="24dp"
      android:layout_height="40dp"
      android:src="@drawable/baseline_date_range_24"
      app:tint="@color/colorPrimary"/>

   <EditText
      android:id="@+id/inputDate"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:hint="Date"
      android:focusable="false" />
</LinearLayout>

<!-- Time with Vector -->
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:layout_marginTop="8dp">

   <ImageView
      android:layout_width="24dp"
      android:layout_height="40dp"
      android:src="@drawable/baseline_access_time_filled_24"
      app:tint="@color/colorPrimary"/>

   <EditText
      android:id="@+id/inputTime"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:hint="Time"
      android:focusable="false" />
</LinearLayout>

<!-- Event Type Spinner with Vector -->
<LinearLayout
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="8dp">

            <ImageView
                android:layout_width="24dp"
                android:layout_height="24dp"
                android:src="@drawable/baseline_event_note_24"
                app:tint="@color/colorPrimary"/>

            <Spinner
                android:id="@+id/eventTypeSpinner"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>

        <!-- Description with Vector -->
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="8dp">

            <ImageView
                android:layout_width="24dp"
                android:layout_height="40dp"
                android:src="@drawable/baseline_description_24"
                app:tint="@color/colorPrimary"/>

            <EditText
                android:id="@+id/inputDescription"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:hint="Description" />
        </LinearLayout>

        <Button
            android:id="@+id/btnSaveEvent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Save Event"
            android:layout_marginTop="16dp" />

    </LinearLayout>
</androidx.cardview.widget.CardView>
```

```
        <!-- ListView for Events -->
        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:cardBackgroundColor="@android:color/white"
            app:cardCornerRadius="16dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="16dp">

                <ListView
                    android:id="@+id/eventsListView"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:divider="@android:color/darker_gray"
                    android:dividerHeight="1dp" />
            </LinearLayout>
        </androidx.cardview.widget.CardView>
    </LinearLayout>
</RelativeLayout>
```

NotificationReceiver.kt

```
package com.example.devems

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.os.Build
import android.util.Log
import androidx.core.app.NotificationCompat

class NotificationReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        val eventId = intent.getLongExtra("eventId", -1)
        val eventName = intent.getStringExtra("eventName") ?: "Event Reminder"
        val channelId = "event_channel_id"

        Log.d("NotificationReceiver", "Received broadcast for eventId: $eventId, eventName:
$eventName")

        val notificationManager = context.getSystemService(Context.NOTIFICATION_SERVICE) as
```

NotificationManager

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channel = NotificationChannel(
                channelId,
                "Event Reminders",
                NotificationManager.IMPORTANCE_HIGH
            ).apply {
                description = "Channel for event reminders"
            }
            notificationManager.createNotificationChannel(channel)
        }

        val notificationIntent = Intent(context, DashboardActivity::class.java).apply {
            putExtra("eventId", eventId)
        }

        val pendingIntent = PendingIntent.getActivity(
            context,
            eventId.toInt(),
            notificationIntent,
            PendingIntent.FLAG_IMMUTABLE
        )

        val notification = NotificationCompat.Builder(context, channelId)
            .setSmallIcon(R.drawable.app_profile)
            .setContentTitle("Event Reminder")
            .setContentText("Event Started: $eventName")
            .setContentIntent(pendingIntent)
            .setAutoCancel(true)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .build()

        notificationManager.notify(eventId.toInt(), notification)
    }
}
```

Event.kt

```
package com.example.devems
data class Event(
    val id: Long,
    var name: String,
    var location: String,
    var date: String,
    var time: String,
    var eventType: String,
    var description: String
)
```

EventDatabaseHelper.kt

```
package com.example.devems

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class EventDatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME,
null, DATABASE_VERSION) {

    companion object {
        const val DATABASE_NAME = "events.db"
        const val DATABASE_VERSION = 1
        const val TABLE_NAME = "events"
        const val COLUMN_ID = "id"
        const val COLUMN_NAME = "name"
        const val COLUMN_LOCATION = "location"
        const val COLUMN_DATE = "date"
        const val COLUMN_TIME = "time"
        const val COLUMN_EVENT_TYPE = "event_type"
        const val COLUMN_DESCRIPTION = "description"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE $TABLE_NAME (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_NAME TEXT,
                $COLUMN_LOCATION TEXT,
                $COLUMN_DATE TEXT,
                $COLUMN_TIME TEXT,
                $COLUMN_EVENT_TYPE TEXT,
                $COLUMN_DESCRIPTION TEXT
            )
        """
        db.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun deleteEvent(eventId: Long): Boolean {
        val db = writableDatabase
        val rowsDeleted = db.delete(TABLE_NAME, "$COLUMN_ID = ?",
```

```
        arrayOf(eventId.toString()))
        return rowsDeleted > 0
    }

    fun insertEvent(name: String, location: String, date: String, time: String, eventType: String,
description: String): Long {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_NAME, name)
        values.put(COLUMN_LOCATION, location)
        values.put(COLUMN_DATE, date)
        values.put(COLUMN_TIME, time)
        values.put(COLUMN_EVENT_TYPE, eventType)
        values.put(COLUMN_DESCRIPTION, description)
        return db.insert(TABLE_NAME, null, values)
    }

    fun getAllEvents(): List<Event> {
        val eventsList = mutableListOf<Event>()
        val db = this.readableDatabase
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)

        if (cursor.moveToFirst()) {
            do {
                val event = Event(
                    id = cursor.getLong(cursor.getColumnIndexOrThrow(COLUMN_ID)),
                    name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME)),
                    location =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_LOCATION)),
                    date = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_DATE)),
                    time = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_TIME)),
                    eventType =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_EVENT_TYPE)),
                    description =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_DESCRIPTION))
                )
                eventsList.add(event)
            } while (cursor.moveToNext())
        }
        cursor.close()
        return eventsList
    }

    fun updateEvent(event: Event): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, event.name)
            put(COLUMN_LOCATION, event.location)
```

```
            put(COLUMN_DATE, event.date)
            put(COLUMN_TIME, event.time)
            put(COLUMN_EVENT_TYPE, event.eventType)
            put(COLUMN_DESCRIPTION, event.description)
        }
        val rowsUpdated = db.update(TABLE_NAME, values, "$COLUMN_ID = ?",
arrayOf(event.id.toString()))
        return rowsUpdated > 0
    }
}
```

EventsAdapter.kt

```
package com.example.devems

import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.BaseAdapter
import android.widget.Button
import android.widget.TextView

class EventsAdapter(
    private val context: Context,
    private val events: List<Event>,
    private val onLongClickListener: (Event) -> Boolean
) : BaseAdapter() {

    override fun getCount(): Int = events.size

    override fun getItem(position: Int): Any = events[position]

    override fun getItemId(position: Int): Long = events[position].id

    override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View {
        val view = convertView ?: LayoutInflater.from(context).inflate(R.layout.event_list_item,
parent, false)

        val event = events[position]

        view.findViewById<TextView>(R.id.eventNameTextView).text = event.name
        view.findViewById<TextView>(R.id.eventDateTextView).text = "${event.date}
${event.time}"
        view.findViewById<TextView>(R.id.eventTypeTextView).text = event.eventType
        view.findViewById<TextView>(R.id.eventDescriptionTextView).text = event.description
        view.findViewById<TextView>(R.id.eventLocationTextView).text = event.location
```

```
        view.findViewById<Button>(R.id.btn_navigate_event).setOnClickListener {
            val intent = Intent(context, UserNavigation::class.java)
            intent.putExtra("LOCATION_NAME", event.location)
            context.startActivity(intent)
        }

        view.setOnLongClickListener {
            onLongClickListener(event)
        }

        return view
    }
}
```

UserNavigation.kt

```
package com.example.devems

import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.location.Geocoder
import android.location.Location
import android.net.Uri
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.MapView
import com.google.android.gms.maps.MapsInitializer
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions
import java.io.IOException

class UserNavigation : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var locationEditText: EditText
    private lateinit var searchButton: Button
    private lateinit var mapView: MapView
```

```
    private lateinit var fusedLocationClient: FusedLocationProviderClient
    private var googleMap: GoogleMap? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_user_navigation)

        locationEditText = findViewById(R.id.locationEditText)
        searchButton = findViewById(R.id.searchButton)
        mapView = findViewById(R.id.mapView)

        val locationName = intent.getStringExtra("LOCATION_NAME")
        if (!locationName.isNullOrEmpty()) {
            locationEditText.setText(locationName)
            navigateToLocation(locationName)
        }

        mapView.onCreate(savedInstanceState)
        mapView.getMapAsync(this)

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
        requestLocationPermission()

        searchButton.setOnClickListener {
            val location = locationEditText.text.toString().trim()
            if (location.isNotEmpty()) {
                openGoogleMaps(location)
            } else {
                Toast.makeText(this, "Please enter a location", Toast.LENGTH_SHORT).show()
            }
        }

        locationEditText.addTextChangedListener(object : TextWatcher {
            override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}
            override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}
            override fun afterTextChanged(s: Editable?) {
                val locationName = s.toString().trim()
                if (locationName.isNotEmpty()) {
                    navigateToLocation(locationName)
                }
            }
        })
    }

    private fun requestLocationPermission() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED
```

```
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                LOCATION_PERMISSION_REQUEST_CODE
            )
        } else {
            getCurrentLocation()
        }
    }

    private fun getCurrentLocation() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED
        ) {
            fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
                location?.let {
                    val currentLatLng = LatLng(it.latitude, it.longitude)
                    googleMap?.apply {
                        addMarker(MarkerOptions().position(currentLatLng).title("You are here"))
                        moveCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng, 15f))
                    }
                }
            }
        }
    }

    private fun navigateToLocation(locationName: String) {
        val geocoder = Geocoder(this)
        try {
            val addresses = geocoder.getFromLocationName(locationName, 1)
            if (addresses != null && addresses.isNotEmpty()) {
                val address = addresses[0]
                val latLng = LatLng(address.latitude, address.longitude)
                googleMap?.apply {
                    clear()
                    addMarker(MarkerOptions().position(latLng).title(locationName))
                    animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15f))
                }
            }
        } catch (e: IOException) {
            e.printStackTrace()
            Toast.makeText(this, "Location not found", Toast.LENGTH_SHORT).show()
        }
    }

    private fun openGoogleMaps(location: String) {
```

```
        val gmmIntentUri = Uri.parse("geo:0,0?q=" + Uri.encode(location))
        val mapIntent = Intent(Intent.ACTION_VIEW, gmmIntentUri)
        mapIntent.setPackage("com.google.android.apps.maps")
        if (mapIntent.resolveActivity(packageManager) != null) {
            startActivity(mapIntent)
        } else {
            Toast.makeText(this, "Google Maps app is not installed.", Toast.LENGTH_LONG).show()
        }
    }

    override fun onMapReady(map: GoogleMap) {
        MapsInitializer.initialize(this)
        googleMap = map
        getCurrentLocation()
    }

    override fun onResume() {
        super.onResume()
        mapView.onResume()
    }

    override fun onPause() {
        super.onPause()
        mapView.onPause()
    }

    override fun onDestroy() {
        super.onDestroy()
        mapView.onDestroy()
    }

    override fun onLowMemory() {
        super.onLowMemory()
        mapView.onLowMemory()
    }

    companion object {
        private const val LOCATION_PERMISSION_REQUEST_CODE = 1
    }
}
```

activity_user_navigation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/locationEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Location Name"
        android:inputType="text" />

    <Button
        android:id="@+id/searchButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Locate on Map" />

    <com.google.android.gms.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>
</LinearLayout>
```

dialog_edit_event.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Edit Event Details"
        android:textSize="18sp"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal"
        android:paddingBottom="10dp" />

    <EditText
```

```
        android:id="@+id/editTextEventName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Name"
        android:inputType="text" />

    <EditText
        android:id="@+id/editTextEventLocation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Location"
        android:inputType="text" />

    <EditText
        android:id="@+id/editTextEventDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Date"
        android:inputType="date" />

    <EditText
        android:id="@+id/editTextEventTime"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Time"
        android:inputType="time" />

    <EditText
        android:id="@+id/editTextEventType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Type"
        android:inputType="text" />

    <EditText
        android:id="@+id/editTextEventDescription"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Event Description"
        android:inputType="textMultiLine"
        android:minLines="3" />

</LinearLayout>
```

dialog_info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Contact:"
        android:textStyle="bold"
        android:textSize="18sp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="contactme@gmail.com"
        android:textSize="16sp" />
</LinearLayout>
```

dialog_rate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <RatingBar
        android:id="@+id/ratingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:numStars="5"
        android:stepSize="1.0"
        android:layout_gravity="center"/>

    <Button
        android:id="@+id/rateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Rate"
        android:layout_gravity="center"/>
</LinearLayout>
```

event_list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">

    <TextView
        android:id="@+id/eventNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Name"
        android:textStyle="bold"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/eventDateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Date"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/eventTypeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Type"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/eventDescriptionTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Description"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/eventLocationTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Event Location"
        android:textSize="14sp" />

    <Button
        android:id="@+id/btn_navigate_event"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Navigate to Event"
        android:textStyle="bold"
        android:layout_marginStart="16dp"
        android:layout_gravity="center"/>

</LinearLayout>
```

curved_background.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@android:color/white"/>
    <stroke android:width="1dp" android:color="@android:color/black"/>
    <corners android:radius="16dp"/> <!-- Increased radius for smoother curve -->
</shape>
```

colors.xml

```
<color name="blue">#00008B</color>
<color name="light_blue">#2E5A88</color>
<color name="colorPrimary">#2c3e50</color>
<color name="colorPrimaryDark">#2E5A88</color>
<color name="colorAccent">#2234AB</color>
```

strings.xml

```
<string name="api_key">AIzaSyDeNXKJCFQ9jTkEExHYLNEvBJ73xzAghBY</string>
```

themes.xml

```
<resources >

<style name="Theme.EMSApp" parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:windowNoTitle">true</item>
        <item name="android:windowActionBar">false</item>

        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/white</item>

    </style>

    <style name="ToolbarTitleStyle" parent="TextAppearance.MaterialComponents.Headline6">
        <item name="android:textStyle">bold</item>
    </style>
</resources>
```

build.gradle.kts

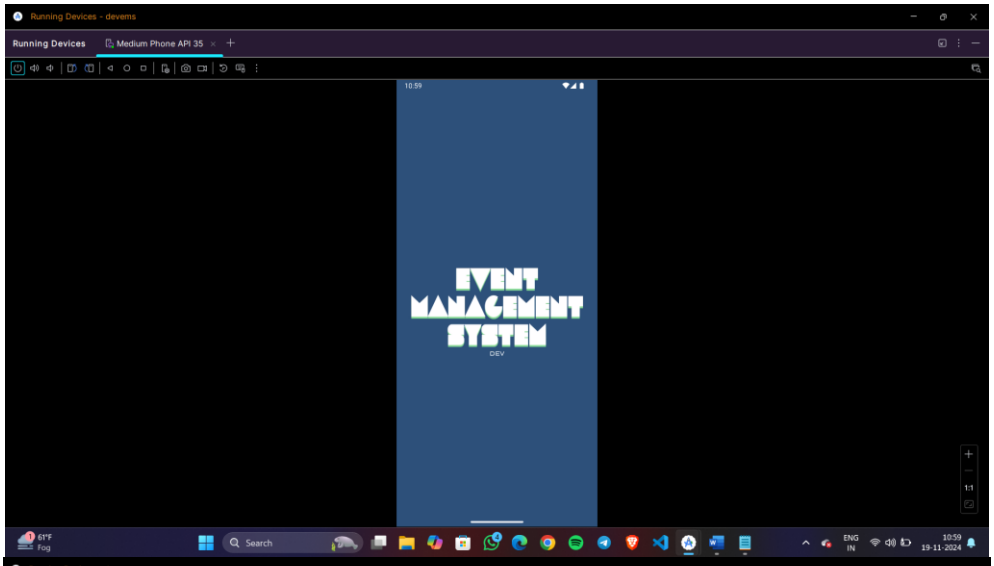
implementation ("com.google.android.gms:play-services-location:21.0.1")
implementation ("com.google.android.gms:play-services-maps:18.1.0")
implementation ("com.google.android.material:material:1.9.0")
implementation ("androidx.viewpager2:viewpager2:1.0.0")

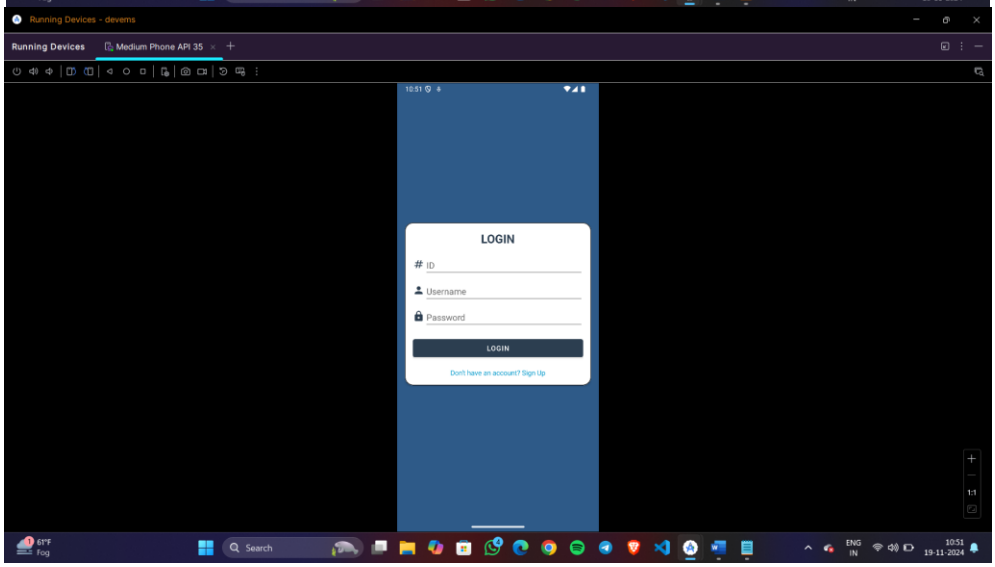
AndroidManifest.xml

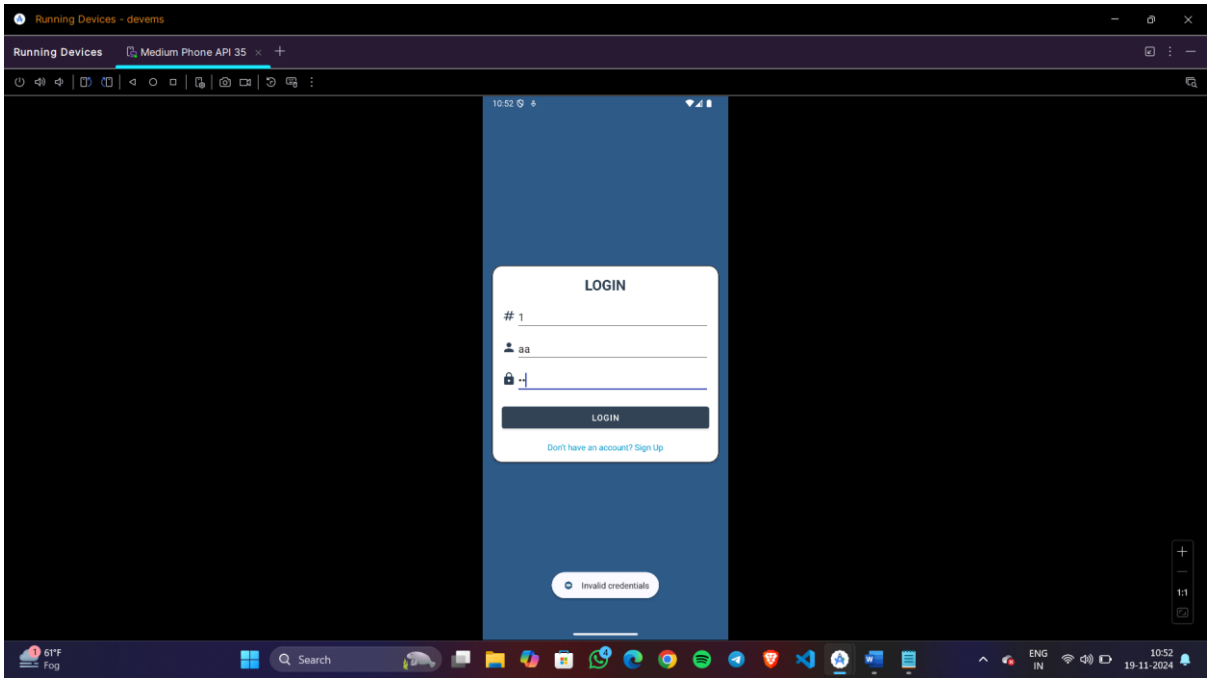
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.SET_ALARM" />
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/app_profile"
        android:label="@string/app_name"
        android:roundIcon="@drawable/app_profile"
        android:supportsRtl="true"
        android:theme="@style/Theme.EMSApp"
        tools:targetApi="31">


        <activity
            android:name=".LoginActivity"
            android:exported="false" />
        <activity
            android:name=".UserNavigation"
            android:exported="false" />
        <activity
            android:name=".SignupActivity"
            android:exported="false" />
        <activity
            android:name=".DashboardActivity"
            android:exported="false" />
        <activity
            android:name=".CreateEvent"
            android:exported="false" />
```

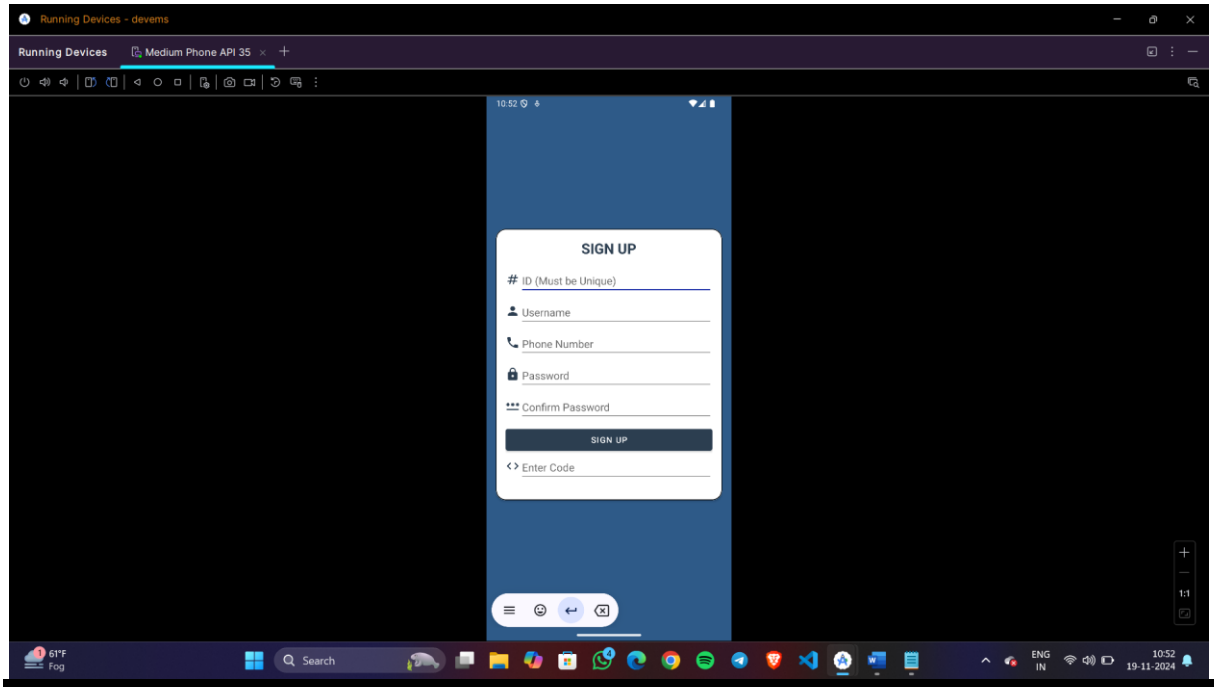
```
        <receiver android:name=".NotificationReceiver" />

        <activity
            android:name=".SplashActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/api_key" />
    </application>

</manifest>
```

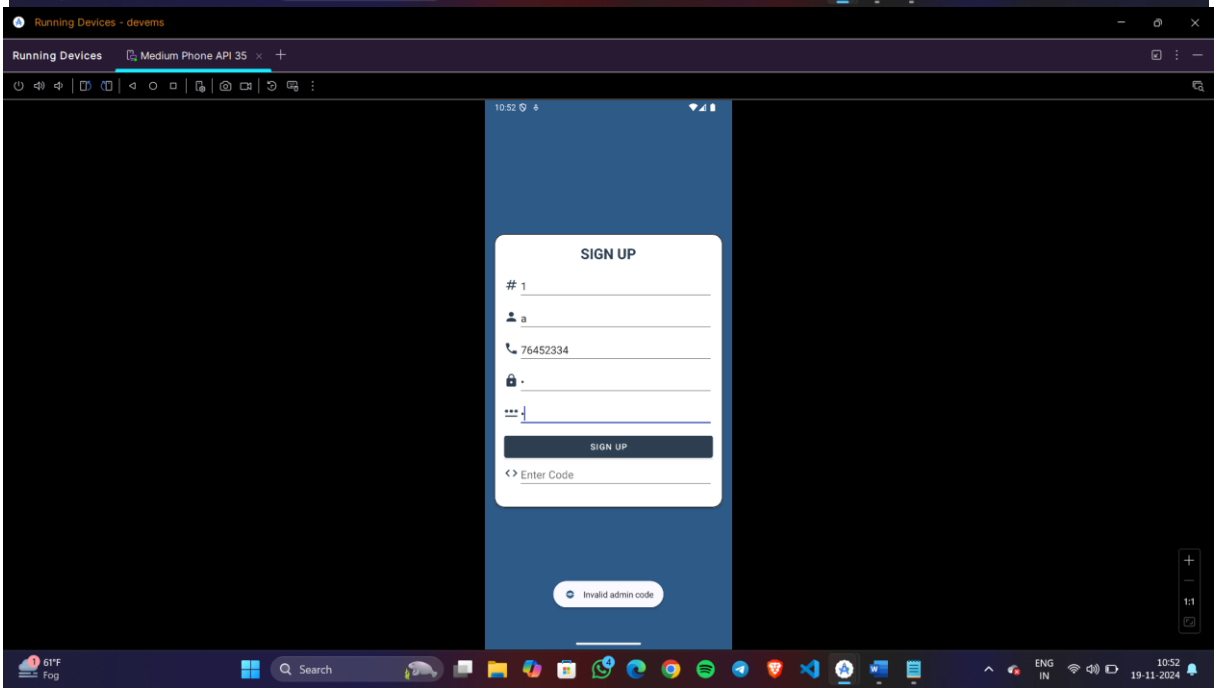
Emulator Screenshots

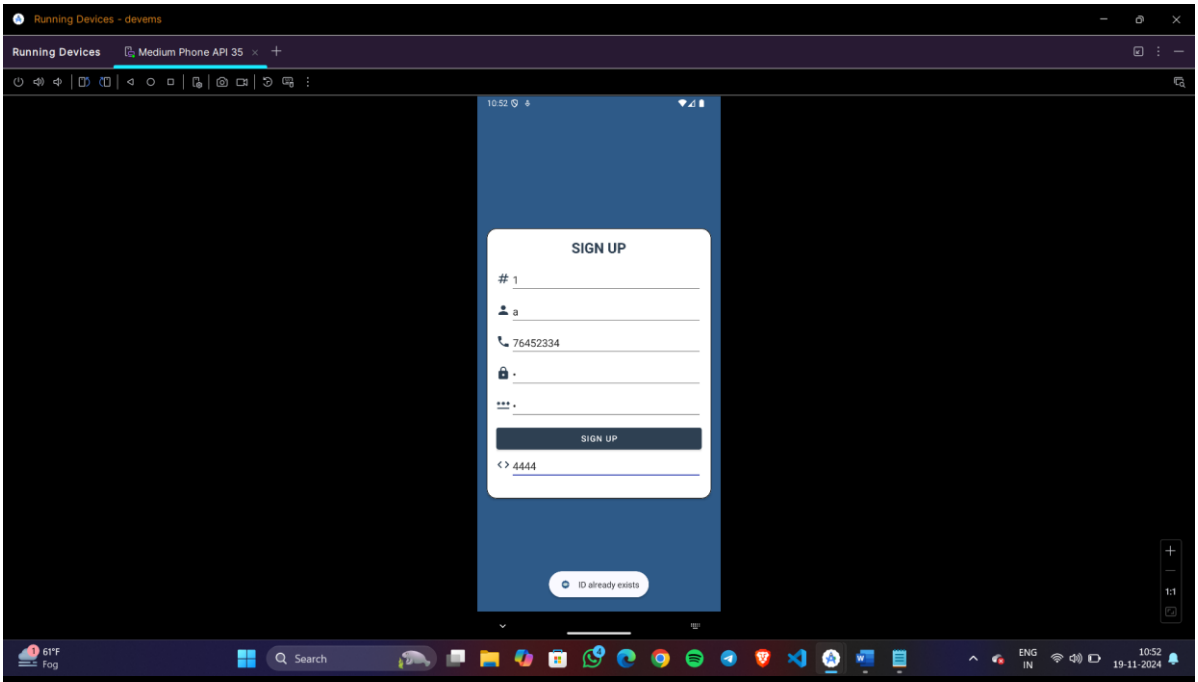


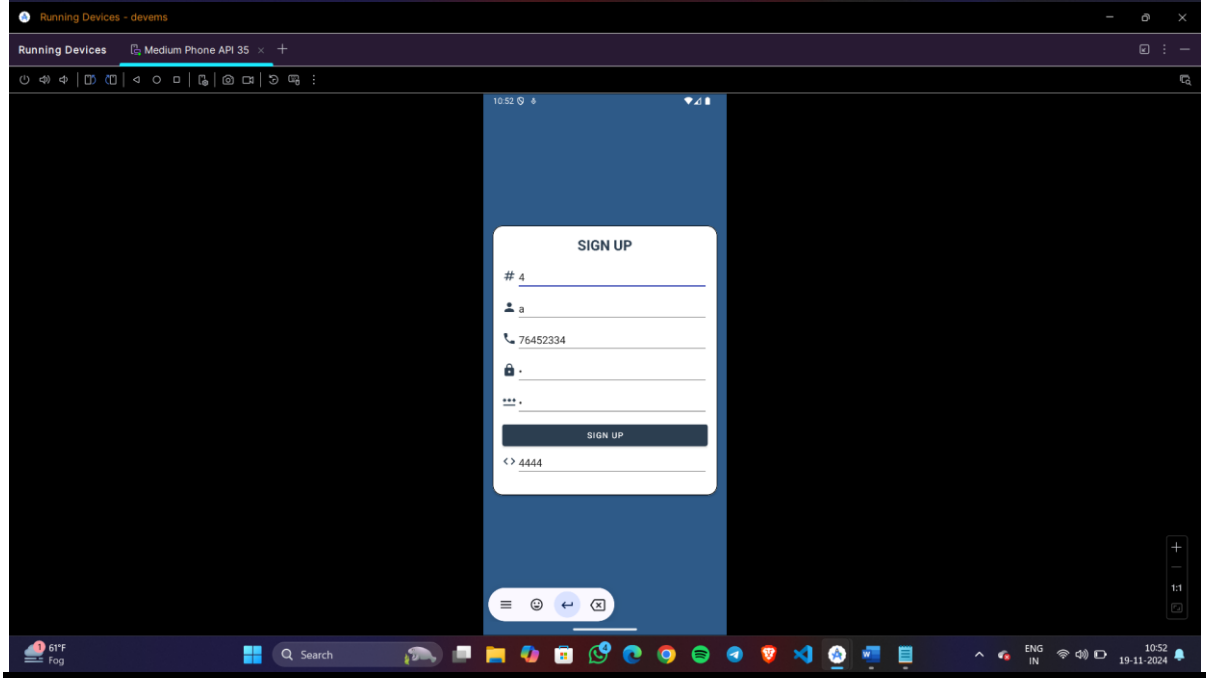


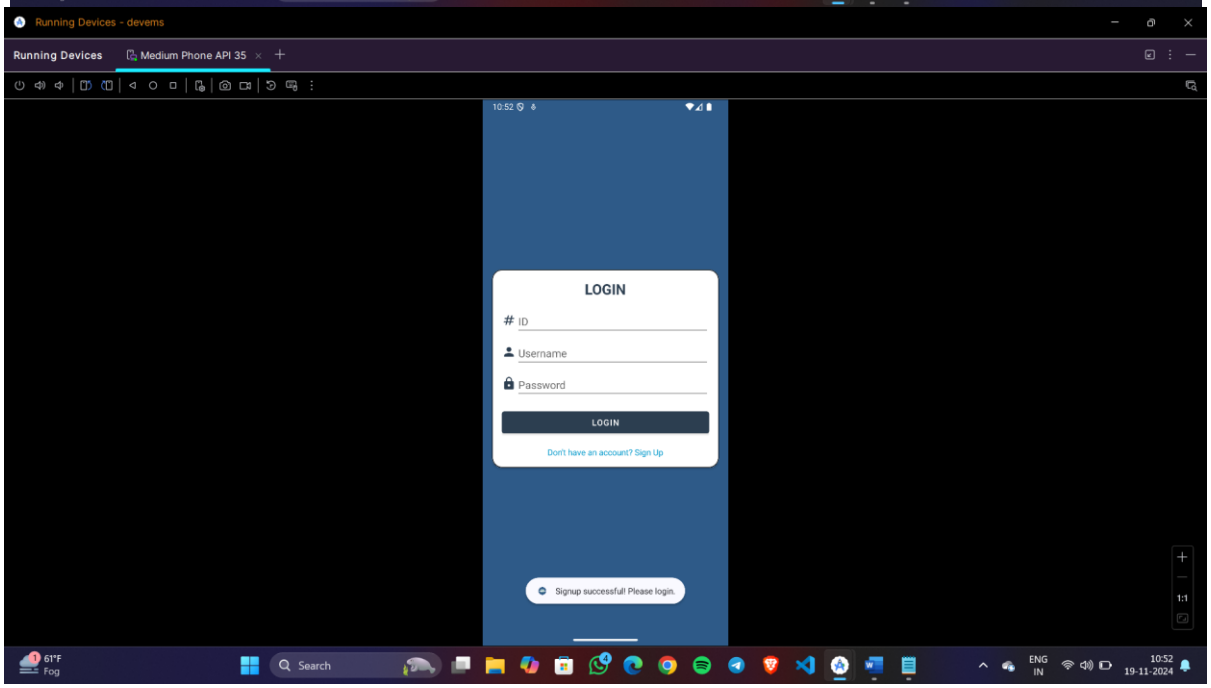

Running Devices - devems
Running Devices
Medium Phone API 35
LOGIN
1
aa
LOGIN
Don't have an account? Sign Up
Invalid credentials

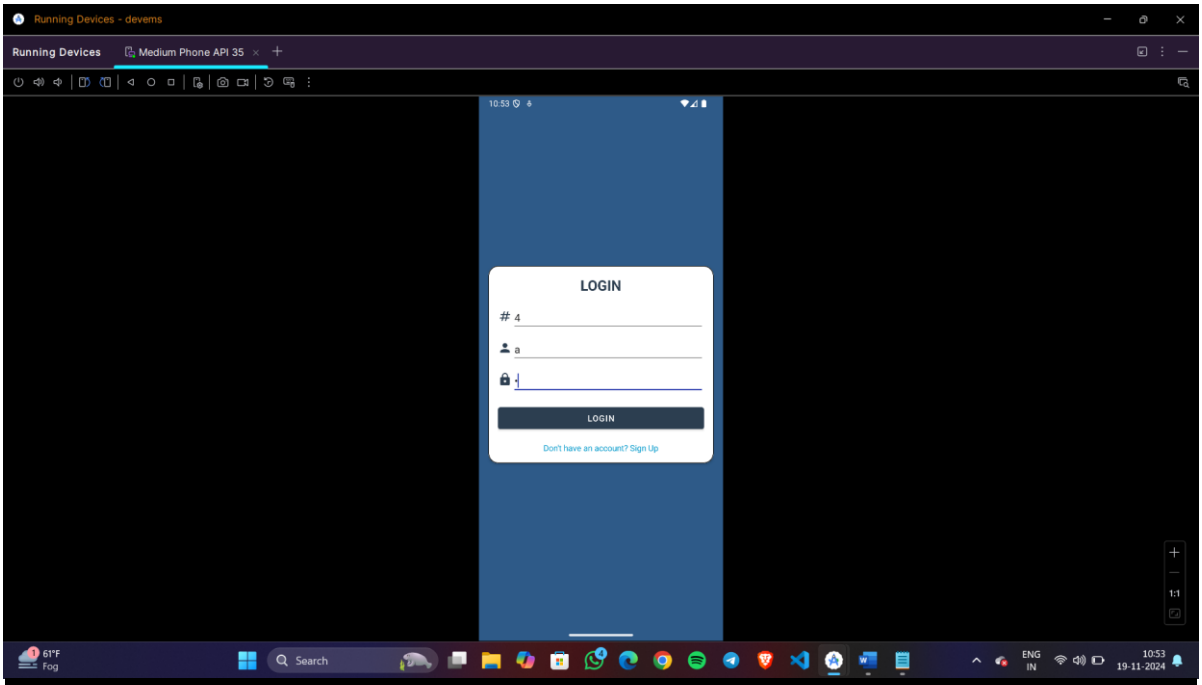

Running Devices - devems
Running Devices
Medium Phone API 35
SIGN UP
ID (Must be Unique)
Username
Phone Number
Password
Confirm Password
SIGN UP
Enter Code

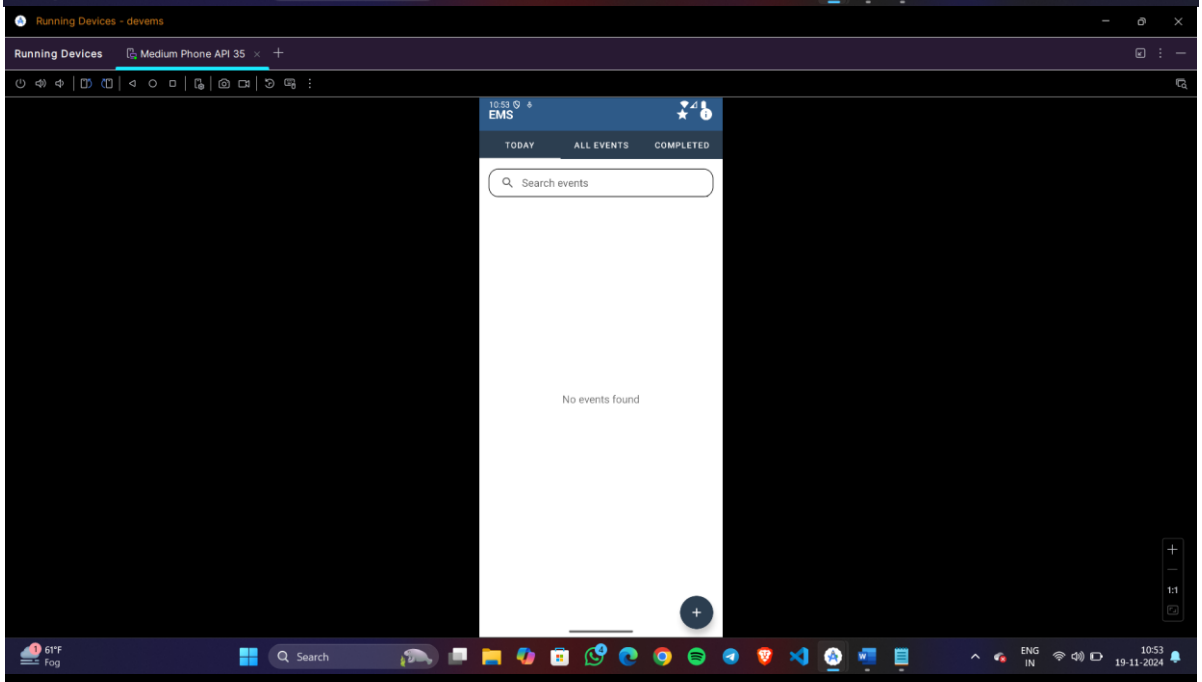

Running Devices - devems
Running Devices
Medium Phone API 35
SIGN UP
1
a
76452334
SIGN UP
Enter Code
Invalid admin code

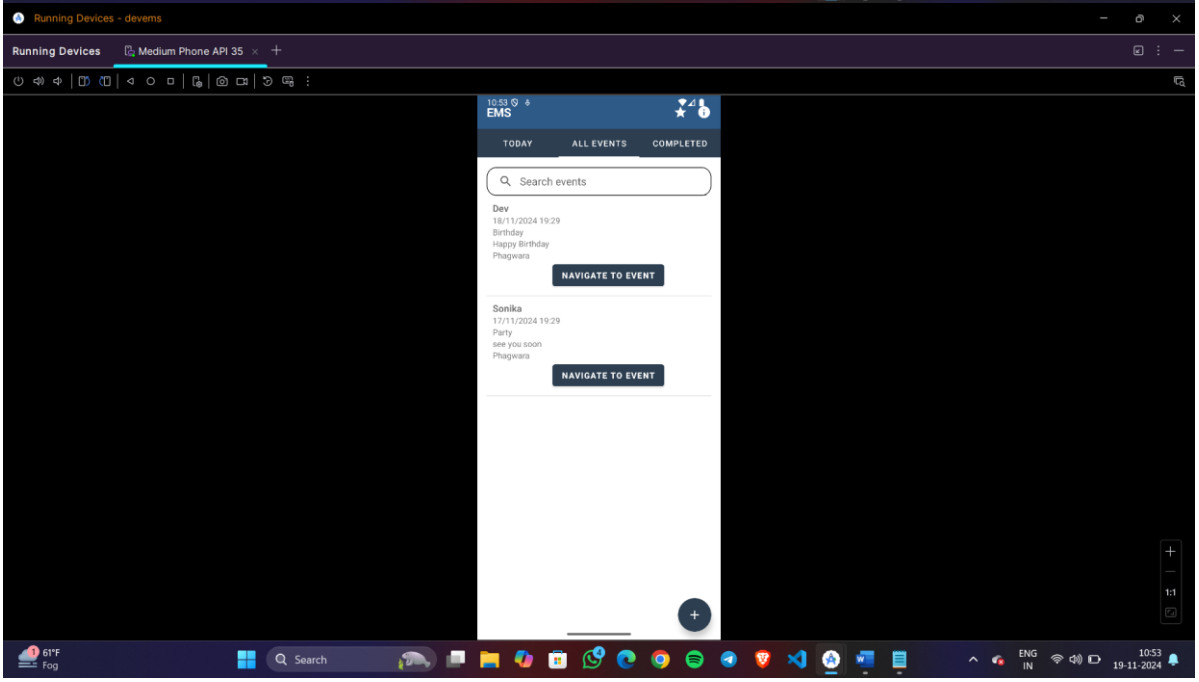

Running Devices - devems
Running Devices
Medium Phone API 35
10:52
SIGN UP
1
a
76452334
SIGN UP
4444
ID already exists
1:1
61°F
Fog
Search
ENG
IN
10:52
19-11-2024

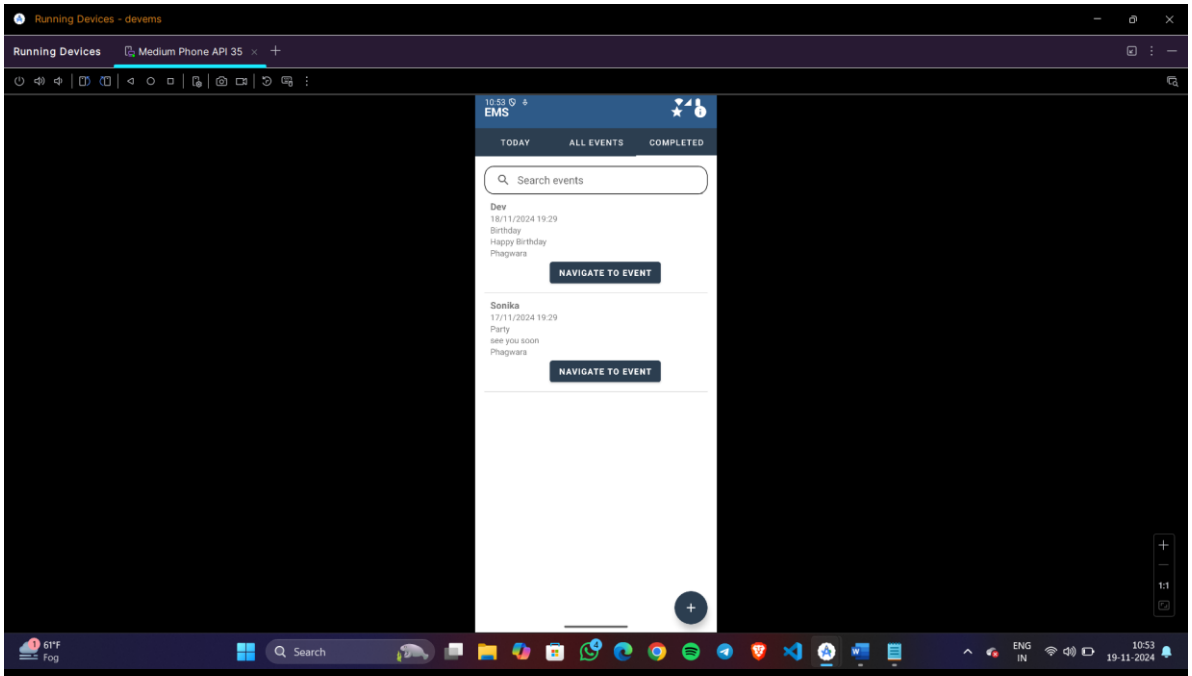

Running Devices - devems
Running Devices
Medium Phone API 35
10:52
SIGN UP
4
a
76452334
SIGN UP
4444
1:1
61°F
Fog
Search
ENG
IN
10:52
19-11-2024

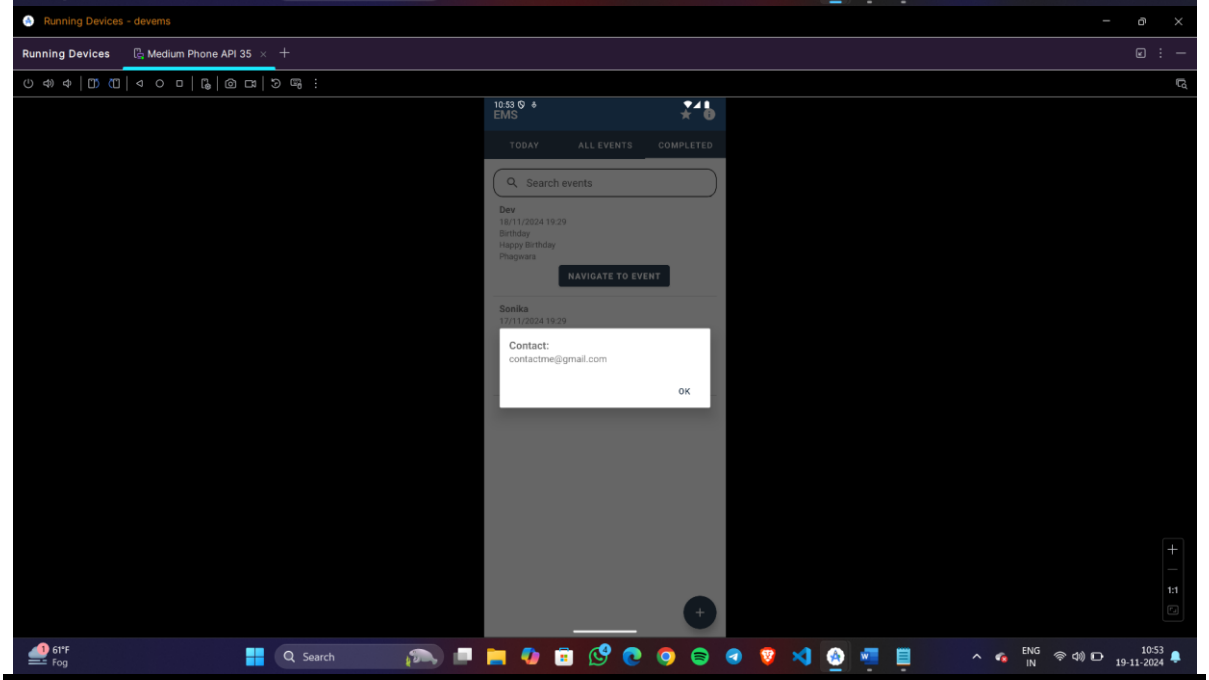

Running Devices - devems
Running Devices
Medium Phone API 35
10:52
LOGIN
ID
Username
Password
LOGIN
Don't have an account? Sign Up
Signup successful! Please login.
1:1
61°F
Fog
Search
ENG
IN
10:52
19-11-2024

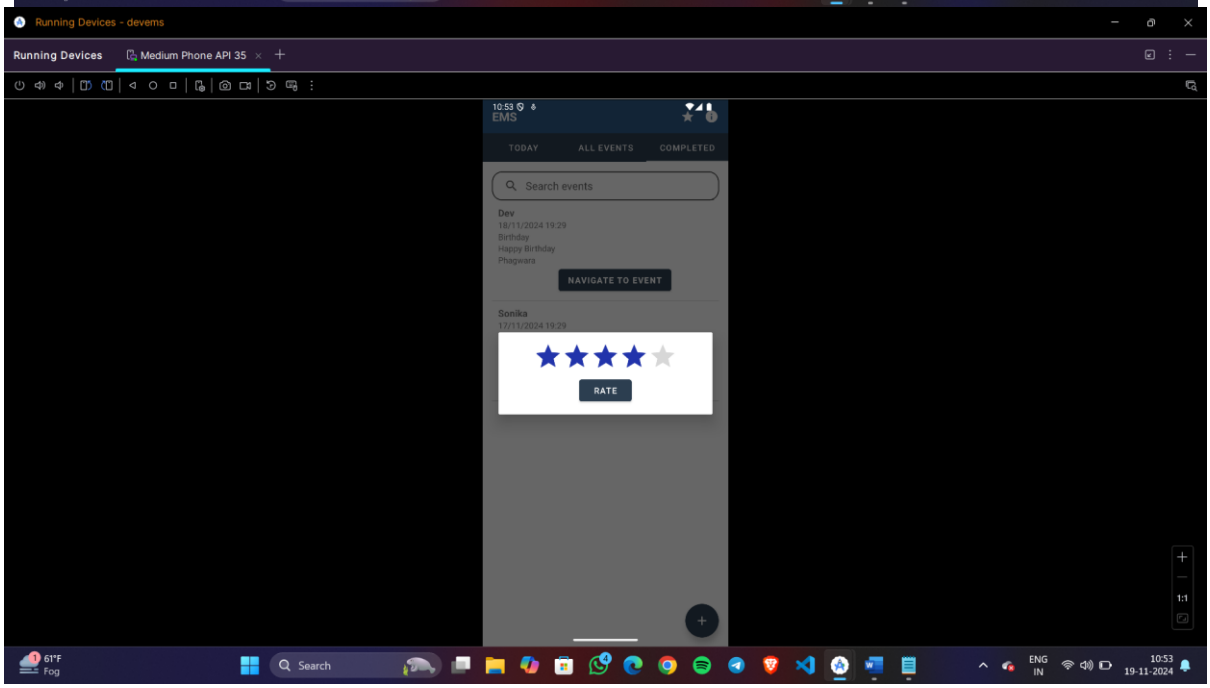

Running Devices - devems
Running Devices
Medium Phone API 35
LOGIN
4
a
LOGIN
Don't have an account? Sign Up

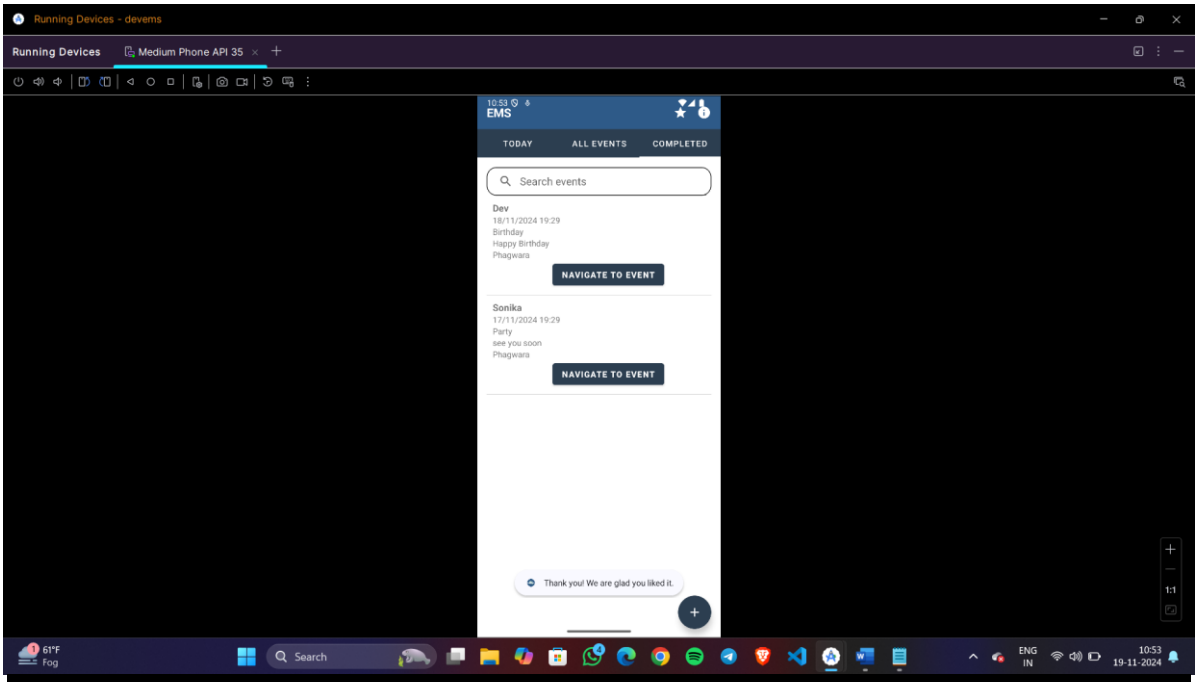

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
No events found

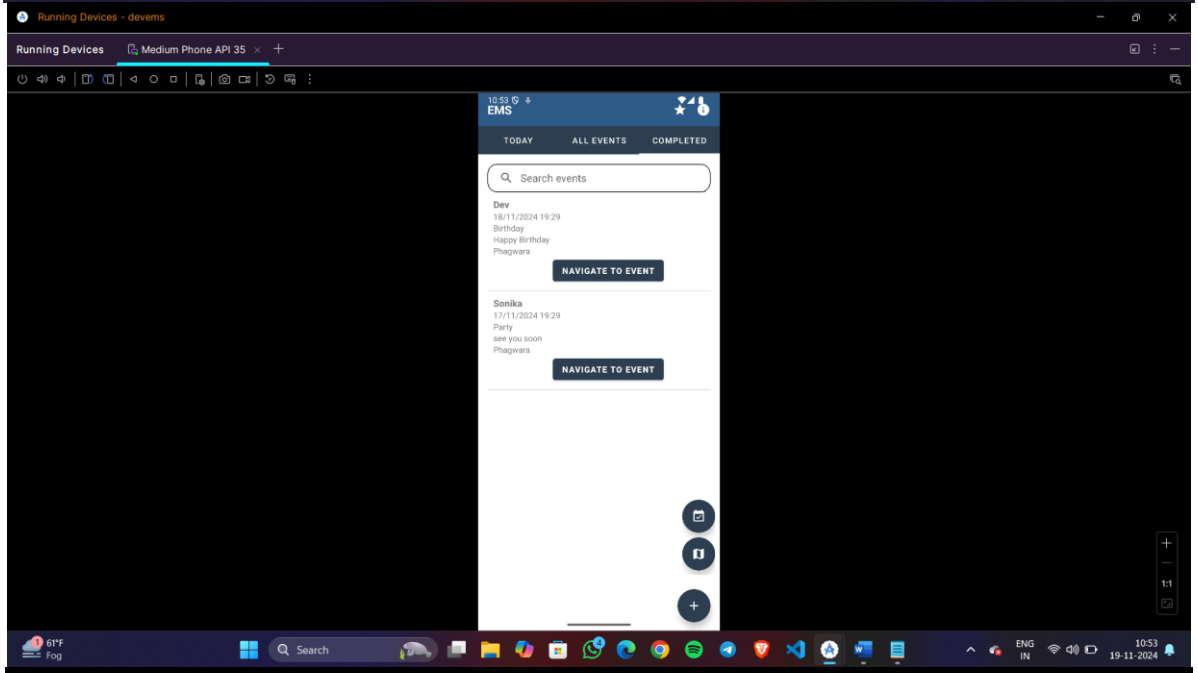

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
NAVIGATE TO EVENT

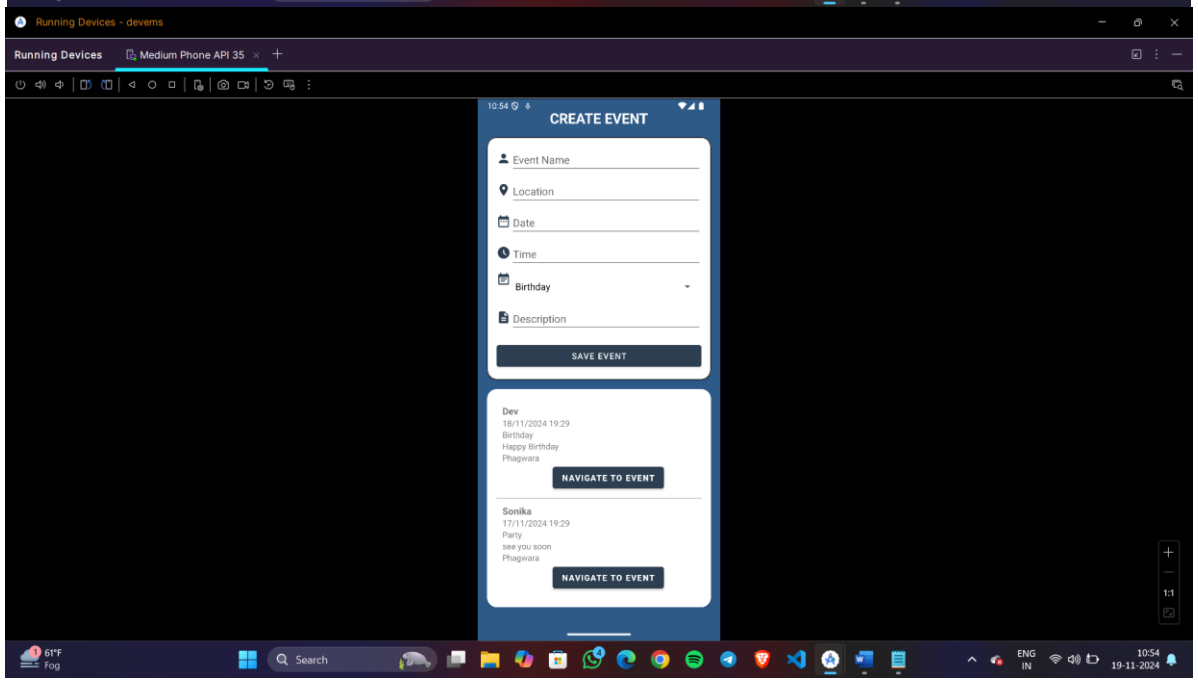

Running Devices - devems
Running Devices
Medium Phone API 35
10:53
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
NAVIGATE TO EVENT
61°F
Fog
Search
ENG
IN
10:53
19-11-2024

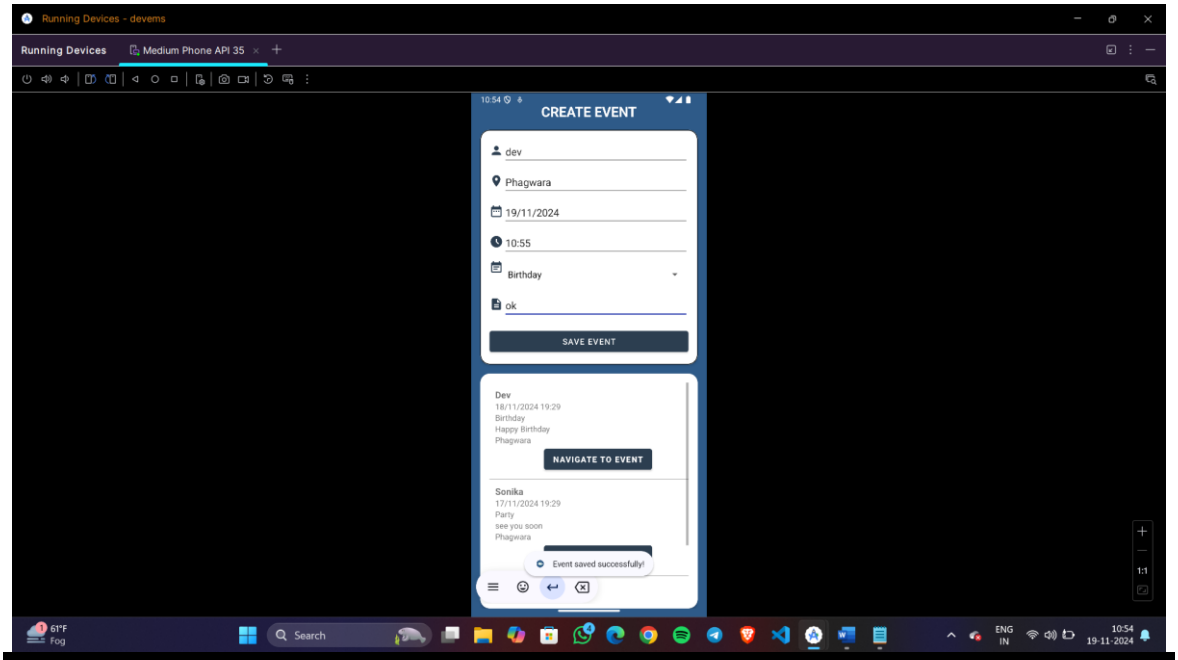

Running Devices - devems
Running Devices
Medium Phone API 35
10:53
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Contact:
contactme@gmail.com
OK
61°F
Fog
Search
ENG
IN
10:53
19-11-2024

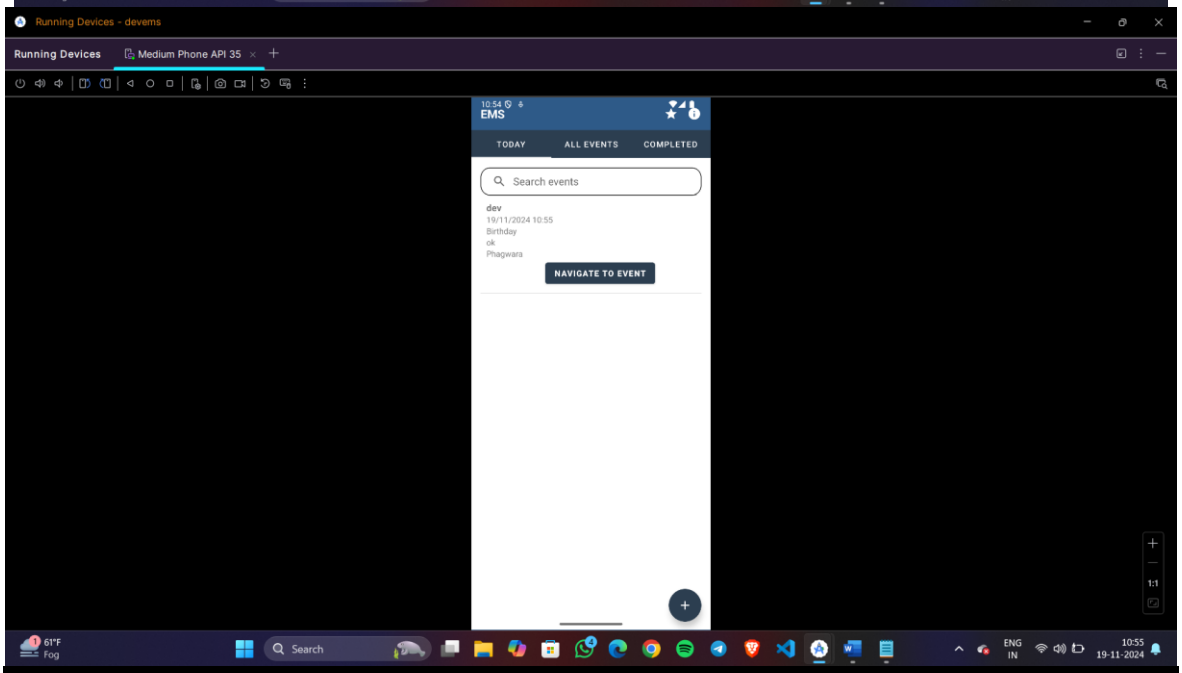

Running Devices - devems
Running Devices
Medium Phone API 35
10:53
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
RATE
61°F
Fog
Search
ENG
IN
10:53
19-11-2024

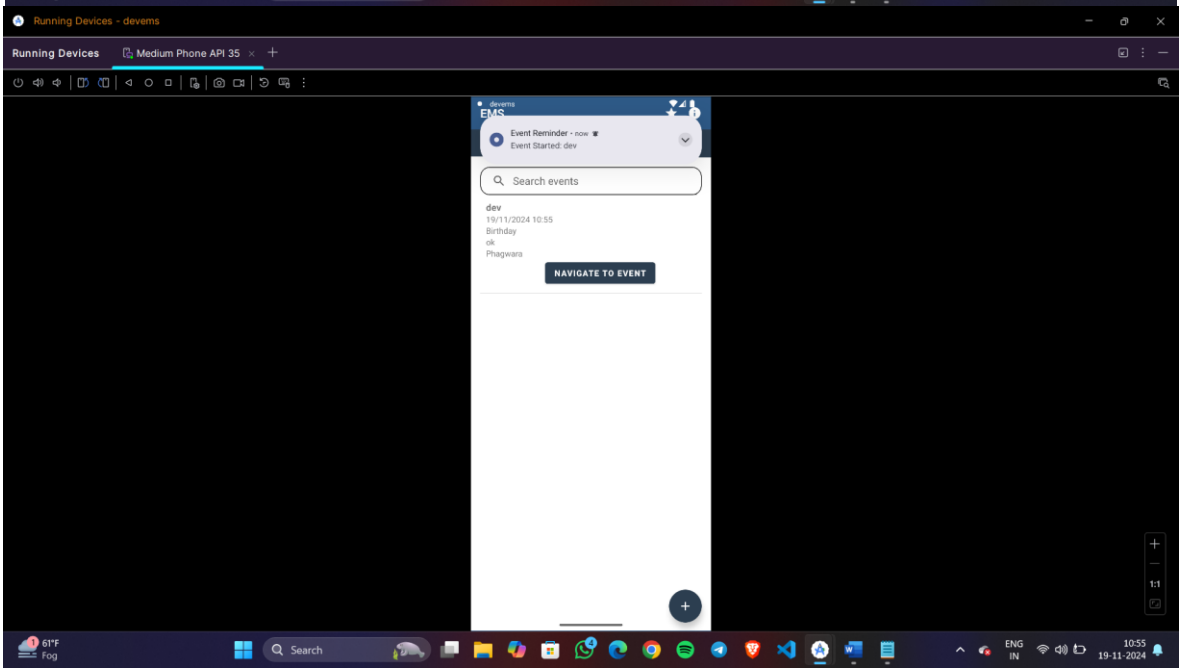

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
NAVIGATE TO EVENT
Thank you! We are glad you liked it.

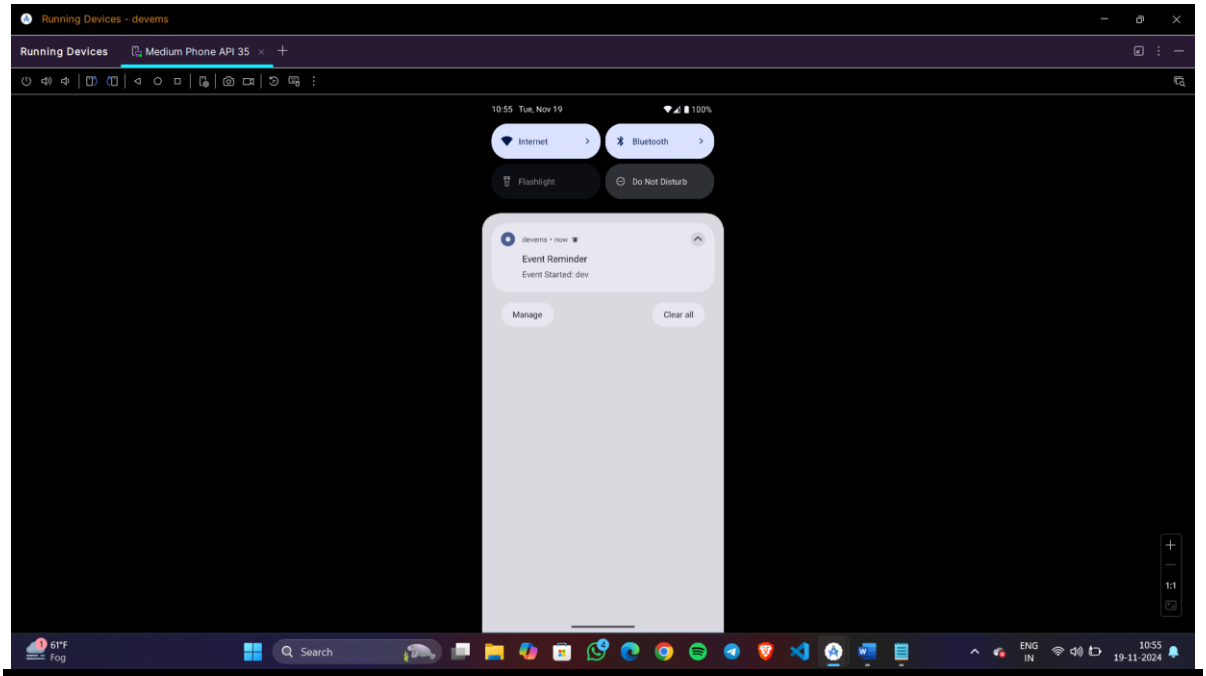

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
NAVIGATE TO EVENT

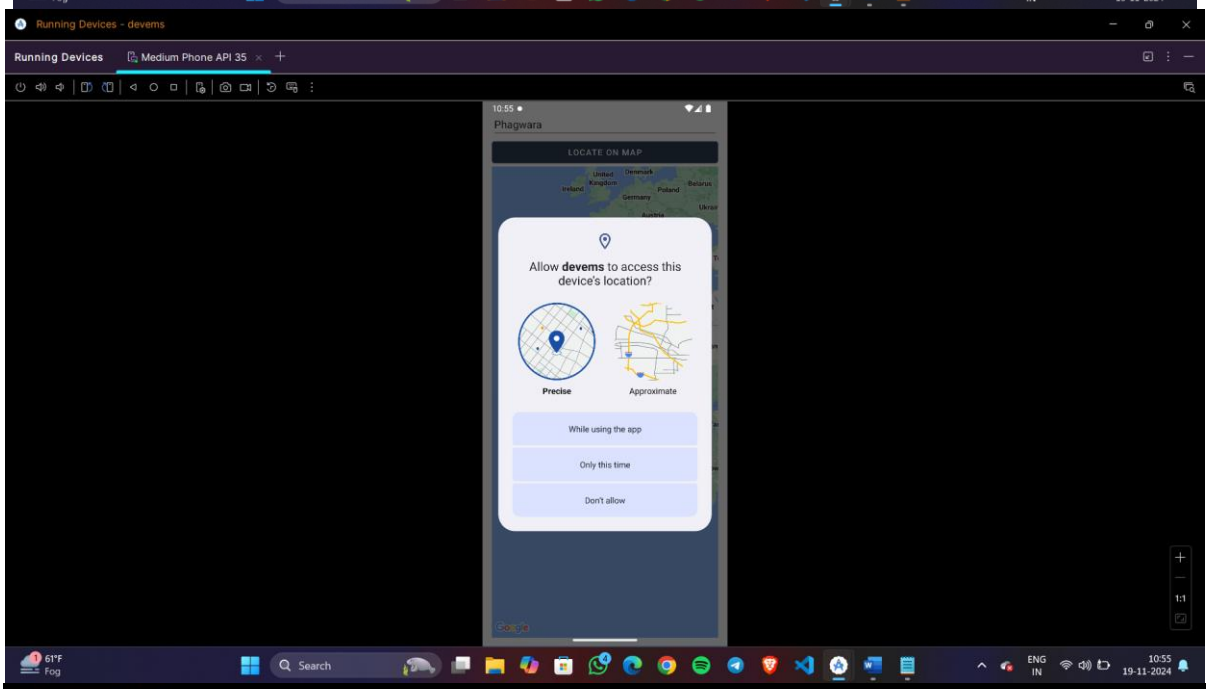

Running Devices - devems
Running Devices
Medium Phone API 35
CREATE EVENT
Event Name
Location
Date
Time
Birthday
Description
SAVE EVENT
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
NAVIGATE TO EVENT

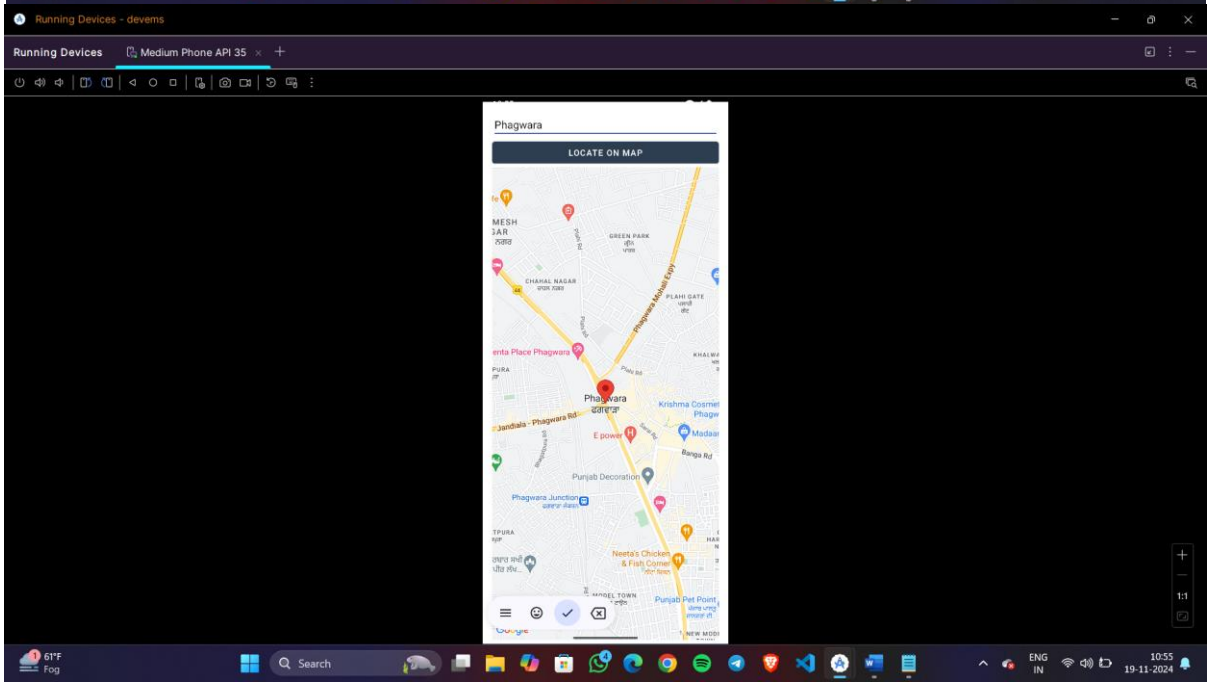

Running Devices - devems
Running Devices
Medium Phone API 35
CREATE EVENT
dev
Phagwara
19/11/2024
10:55
Birthday
ok
SAVE EVENT
Dev
18/11/2024 19:29
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
Sonika
17/11/2024 19:29
Party
see you soon
Phagwara
Event saved successfully!
61°F
Fog
Search
ENG
IN
10:54
19-11-2024

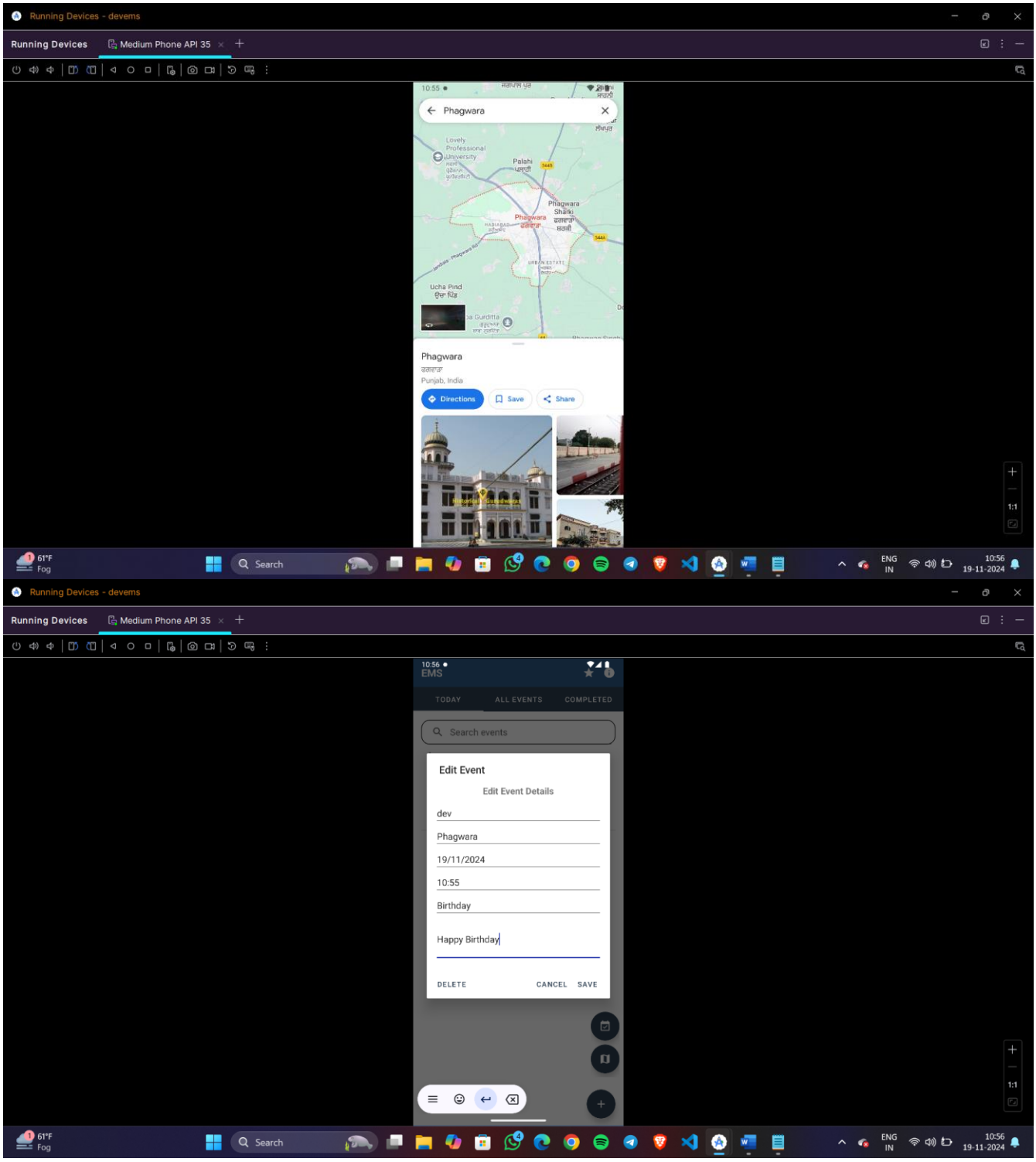

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
dev
19/11/2024 10:55
Birthday
ok
Phagwara
NAVIGATE TO EVENT
61°F
Fog
Search
ENG
IN
10:55
19-11-2024

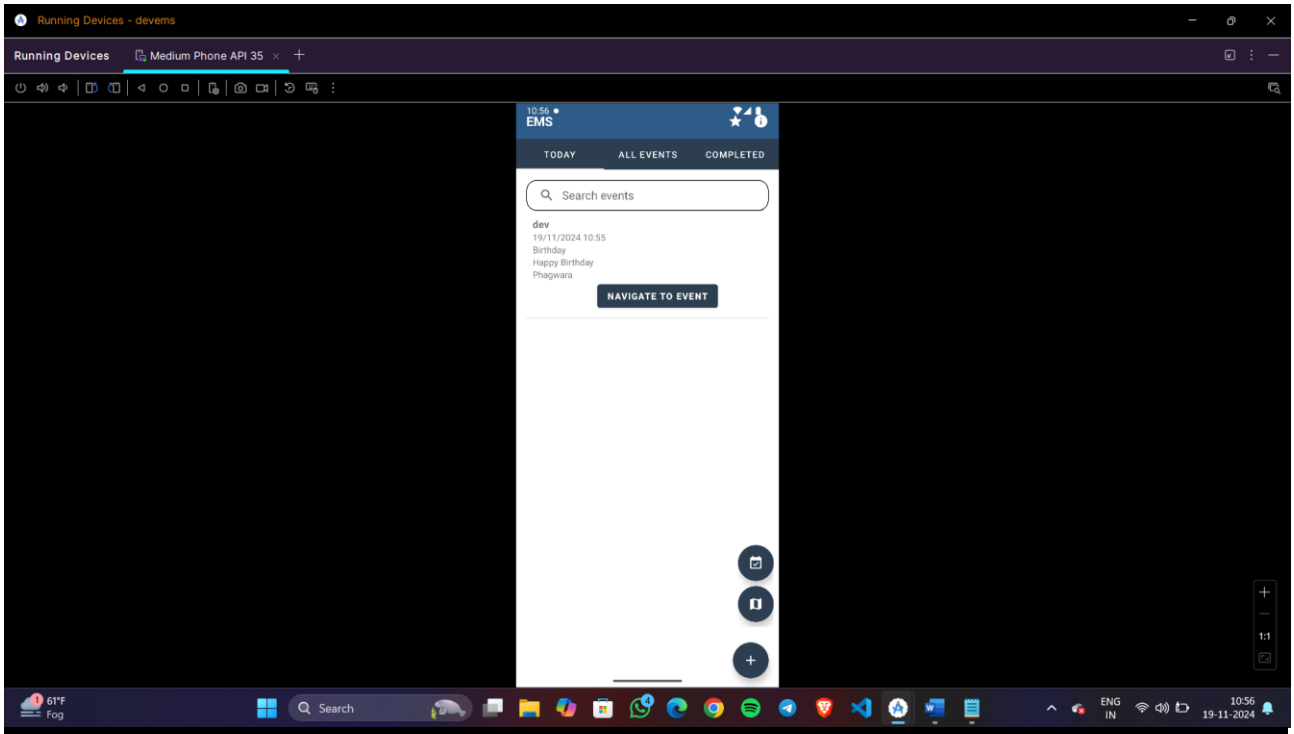

Running Devices - devems
Running Devices
Medium Phone API 35
EMS
Event Reminder · now
Event Started: dev
Search events
dev
19/11/2024 10:55
Birthday
ok
Phagwara
NAVIGATE TO EVENT
61°F
Fog
Search
ENG
IN
10:55
19-11-2024

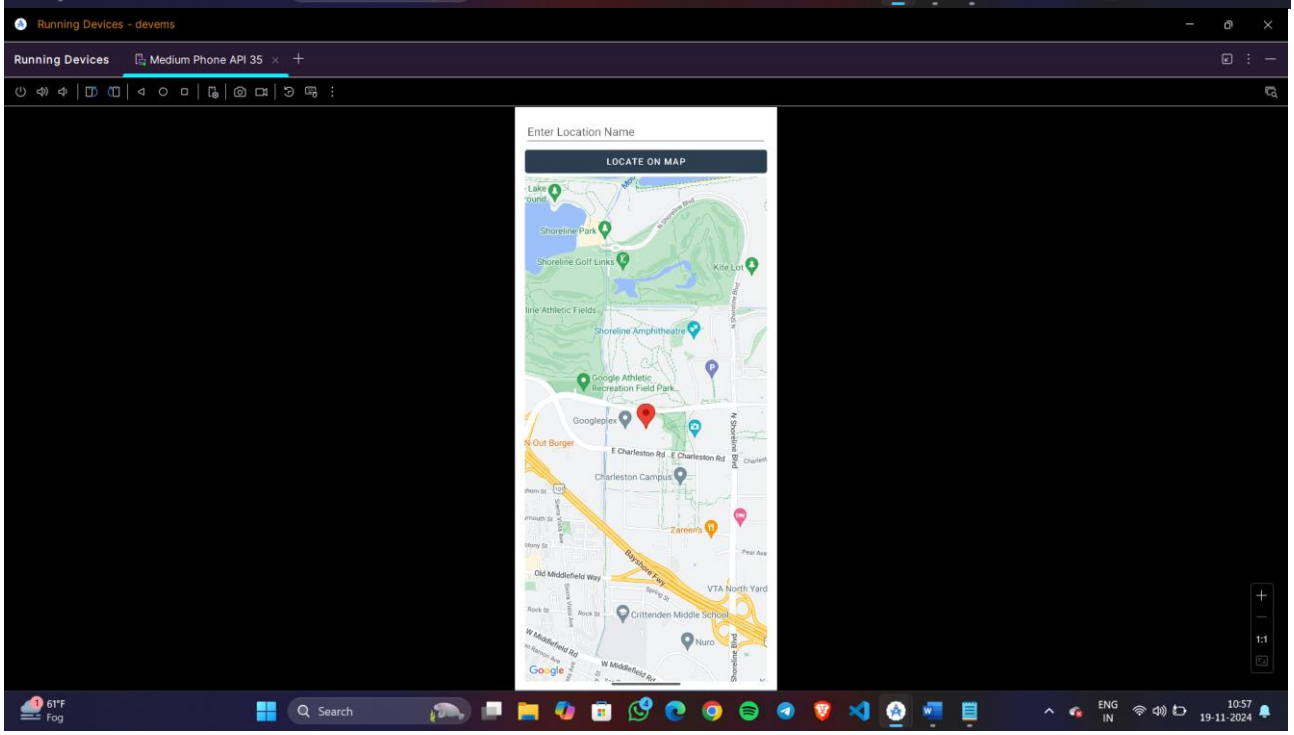

Running Devices - devems
Running Devices
Medium Phone API 35
10:55 Tue, Nov 19
100%
Internet
Bluetooth
Flashlight
Do Not Disturb
devems • now
Event Reminder
Event Started: dev
Manage
Clear all
61°F
Fog
Search
ENG
IN
10:55
19-11-2024


Running Devices - devems
Running Devices
Medium Phone API 35
Phagwara
LOCATE ON MAP
Allow devems to access this device's location?
Precise
Approximate
While using the app
Only this time
Don't allow
61°F
Fog
Search
ENG
IN
10:55
19-11-2024


Running Devices - devems
Running Devices
Medium Phone API 35
Phagwara
LOCATE ON MAP
Phagwara
Punjab Decoration
Phagwara Junction
61°F
Fog
Search
ENG
IN
10:55
19-11-2024


Running Devices - devems
Running Devices
Medium Phone API 35
10:55
Phagwara
Lovely Professional University
Palahi
Phagwara Shanki
Phagwara
Ucha Pind
Phagwara
Punjab, India
Directions
Save
Share
Running Devices - devems
Running Devices
Medium Phone API 35
10:56
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
Edit Event
Edit Event Details
dev
Phagwara
19/11/2024
10:55
Birthday
Happy Birthday
DELETE
CANCEL
SAVE
61°F
Fog
Search
ENG
IN
10:56
19-11-2024


Running Devices - devems
Running Devices
Medium Phone API 35
10:56
EMS
TODAY
ALL EVENTS
COMPLETED
Search events
dev
19/11/2024 10:55
Birthday
Happy Birthday
Phagwara
NAVIGATE TO EVENT
61°F
Fog
Search
ENG
IN
10:56
19-11-2024


Running Devices - devems
Running Devices
Medium Phone API 35
Enter Location Name
LOCATE ON MAP
Shoreline Park
Shoreline Golf Links
Kite Lot
Shoreline Amphitheatre
Google Athletic Recreation Field Park
Googleplex
In-N-Out Burger
E Charleston Rd
Charleston Campus
Zareen's
Old Middlefield Way
VTA North Yard
Crittenden Middle School
Nuro
W Middlefield Rd
Google
61°F
Fog
Search
ENG
IN
10:57
19-11-2024

Conclusion & Future Scope

The Event Management System (EMS) app has successfully demonstrated its ability to streamline the organization, management, and navigation of event-related data. With features such as categorized views for today's events, all events, and completed events, the app provides an efficient mechanism for users to keep track of their schedules. The seamless integration of SQLite for persistent storage, coupled with a modular architecture using fragments and adapters, ensures that the app is both scalable and maintainable. By offering functionalities like search filters, edit/delete operations, date and time pickers, and Google Maps navigation, the EMS app caters to a wide range of user needs, delivering an intuitive and user-friendly experience. The use of design elements such as `TabLayout`, `ViewPager2`, and visually appealing layouts enhances the overall usability and visual appeal of the app.

Looking ahead, the EMS app holds significant potential for further development. Cloud-based synchronization can be introduced to allow users to access their events across multiple devices, making the system more versatile. Integration with push notification services can enable real-time alerts for upcoming events, ensuring that users stay informed and never miss an important schedule. The app could also benefit from adding collaborative features, such as shared event management and participant communication, allowing multiple users to contribute to and manage the same events. To enhance security, advanced authentication mechanisms like biometric login and role-based access control can be implemented. Moreover, incorporating advanced technologies like machine learning could enable predictive event suggestions based on user habits and preferences, further improving usability. With multilingual support and customization options, the app can reach a wider audience and adapt to diverse user requirements. By embracing these future advancements, the EMS app has the potential to transform into a comprehensive, smart event management platform suitable for both personal and professional applications.

Github Link

https://github.com/Gdvprasadnaidu/Event-Management-System

Screen Recording Drive Link

https://drive.google.com/file/d/1rG1u5R_ip2iRP3JFKnv2rmKZX6p03aGQ/view?usp=sharing