MEDICINE MANAGER

CS19611 - MOBILE APPLICATION DEVELOPMENT LABORATORY

PROJECT REPORT

Submitted by

SRIRAMAN K (2116220701288)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Project titled “**MEDICINE MANAGER”** is the bonafide work of **“SRIRAMAN K (2116220701288)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. P. Kumar., M.E., Ph.D.,	Dr. N. Duraimurugan., M.E., Ph.D.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Associate Professor
Department of Computer Science	Department of Computer Science
and Engineering,	and Engineering,
Rajalakshmi Engineering College,	Rajalakshmi Engineering
Chennai - 602 105.	College, Chennai-602 105.

Submitted to Mini Project Viva-Voce Examination held on _______________

Internal Examiner **External Examiner**

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide **Dr. N. DURAIMURUGAN**, We are very glad to thank our Project Coordinator, **Dr. N. DURAIMURUGAN** Associate Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

SRIRAMAN K 2116220701288

ABSTRACT

The **Medicine Manager** Android application is a user-friendly tool designed to help individuals manage their daily medications efficiently. The application allows users to add medicines by providing essential details such as the medicine name, dosage information, total quantity, expiry date, and the number of dosages required per day. Based on the number of dosages entered, the app prompts the user to input the specific times at which the medicine should be taken, ensuring personalized scheduling for each user.

All the medicine-related data is stored locally using a **SQLite database**, allowing for quick and offline access to information. Users can view a list of all the added medicines and perform actions such as editing or deleting existing entries. This provides flexibility in managing changing prescriptions or adjusting to updated routines.

A key feature of the application is its **alert system**. It notifies users through in-app alerts when a medicine is nearing its expiry date, helping prevent accidental consumption of expired medications. Additionally, the app sends **SMS reminders** to the user's registered mobile number at the scheduled dosage times. This ensures that users receive timely notifications to take their medicines, even if they are not actively using the app at that moment.
The project leverages core Android components such as SQLiteOpenHelper for database operations, AlarmManager for setting timed reminders, and BroadcastReceiver for handling background notifications and SMS alerts. Overall, the Medicine Manager application offers a practical and reliable solution for personal medicine tracking, enhancing adherence to prescribed schedules and promoting better health management.

TABLE OF CONTENTS

1. INTRODUCTION

The **Medicine Manager** app is an Android-based application designed to simplify and automate the process of managing personal medication schedules. In daily life, people often struggle with remembering to take their medicines on time, tracking dosages, or monitoring expiry dates — especially when managing multiple prescriptions. Manual tracking methods can lead to missed doses or consumption of expired medicines, negatively affecting health outcomes.

The Medicine Manager addresses these challenges by offering a centralized solution where users can add medicines with details like name, dosage, quantity, expiry date, and the number of dosages per day. Based on this information, the app intelligently schedules medicine intake reminders and expiry alerts. Users receive in-app notifications and SMS alerts at designated times, ensuring they are reminded to take their medicine even when the app is not actively in use. The application stores all data locally using SQLite, ensuring fast performance and full offline functionality.

By combining core Android components such as SQLiteOpenHelper, AlarmManager, and BroadcastReceiver, the app offers a powerful yet simple interface for effective medication management. It is aimed at users of all ages, from elderly patients who need daily reminders to caretakers and individuals managing complex prescriptions.

.

2. Objectives

- The key objectives of the Medicine Manager project are as follows:
- **Simplify Medicine Tracking:** Allow users to add, view, edit, and delete medicines effortlessly through a user-friendly interface.
- **Timely Reminders:** Ensure users take their medication on time via SMS and in-app alerts
- **Expiry Management:** Notify users when medicines are about to expire, reducing health risks.
- **Offline Functionality:** Use a local SQLite database for full app functionality without internet connectivity.
- **Efficient Scheduling:** Allow custom dosage timing based on the number of dosages per day.
- **Clean UI:** Deliver a responsive and visually simple interface suitable for users of all ages.

- **Low Resource Usage:** Ensure smooth performance even on devices with limited hardware.

3. Modules

- The app is organized into the following modules:
  Splash Module: Shows a branded splash screen during app launch, creating a professional impression and initializing necessary components before navigating to the main interface.
  Add Medicine Module: Enables users to input medicine details including name, dosage, quantity, expiry date, number of dosages per day, and dosage times. Dynamically adjusts the UI based on dosage count (e.g., one or two time pickers).
  Medicine List Module: Displays a list of all added medicines using a RecyclerView. Allows users to edit or delete medicines with intuitive buttons and confirmation dialogs.
- **Database Module:** Handles all operations related to the SQLite database such as insert, update, delete, and fetch. Ensures that medicine data persists locally on the device.
- **Alert & Notification Module:** Uses AlarmManager to schedule dosage reminders and expiry alerts. BroadcastReceiver listens for alarms and triggers notifications and SMS messages using SmsManager.
- **UI Module:** Comprises all XML layout files that define the visual appearance of screens including splash, add/edit forms, and medicine list. Ensures clean and adaptive design across devices.

CHAPTER 2

1. Software Description

The Medicine Manager app is developed for the Android platform using Java as the primary programming language and SQLite for local database storage. The development is carried out in Android Studio, which provides a comprehensive environment for designing user interfaces, managing resources, and compiling APKs. The app is optimized for both performance and simplicity, ensuring smooth usage on a wide range of Android devices. Its core functionality revolves around storing medicine data, scheduling alerts, and sending reminders via in-app notifications and SMS. The user interface is built using XML for responsive design, while Java handles logic such as alarm scheduling, database operations, and lifecycle management. By operating entirely offline and storing data locally, the app guarantees reliability in environments with limited or no internet access

2. Software Description

2.1 Kotlin

Kotlin is a modern, statically typed programming language developed by JetBrains and officially supported by Google for Android development. It offers features like null safety, coroutines for asynchronous tasks, and concise syntax. In the Medicine Manager app, Kotlin is used to implement all business logic including database interactions, alarm scheduling, SMS sending, and UI control.

2.2 XML

XML (Extensible Markup Language) is used in Android to define user interface layouts. All UI screens in the Medicine Manager app—such as the splash screen, add medicine screen, and medicine list—are designed using XML, ensuring responsive and structured layouts. This separation of design and logic allows for easier maintenance and scalability.

2.3 SQLite

SQLite is a lightweight embedded relational database system included with Android. It stores user-entered data such as medicine name, dosage, quantity, expiry date, number of dosages per day, and specific times for each dose. The local database ensures fast and offline access to all records, and supports CRUD operations needed for managing the medicine list.

2.4 AlarmManager & BroadcastReceiver

The app uses Android's AlarmManager to schedule medicine reminders based on the user's input. BroadcastReceiver components are used to receive these alarms and trigger in-app notifications and SMS alerts. This setup ensures that reminders work reliably even when the app is not running in the foreground.

3. Requirements And Analysis

3.1 Requirement Specification

Functional Requirements:

- Users can add medicine details including name, dosage, quantity, expiry date, and number of dosages per day.
- Depending on the number of dosages per day, the app prompts users to enter specific times.
- Medicines can be edited or deleted from the list.
- In-app alerts are triggered for each scheduled dose.
- SMS alerts are sent to the user's mobile number at dose times.
- Expiry reminders are generated before the medicine expires.

Non-functional Requirements:

- Responsive, user-friendly interface with support for a wide range of Android devices.
- Accurate and timely notifications and SMS reminders.
- Offline functionality with persistent local data.
- Optimized performance for smooth UI interaction and low battery usage.
- Compatibility with Android API level 21 and above.

Analysis:

- User Workflow: Designed for simplicity so users of all ages can input medicines and dosage schedules without confusion.
- Data Model: Optimized SQLite schema to support multiple dosage times, SMS log, and expiry alerts for each medicine.

3.2 Hardware and Software Requirements

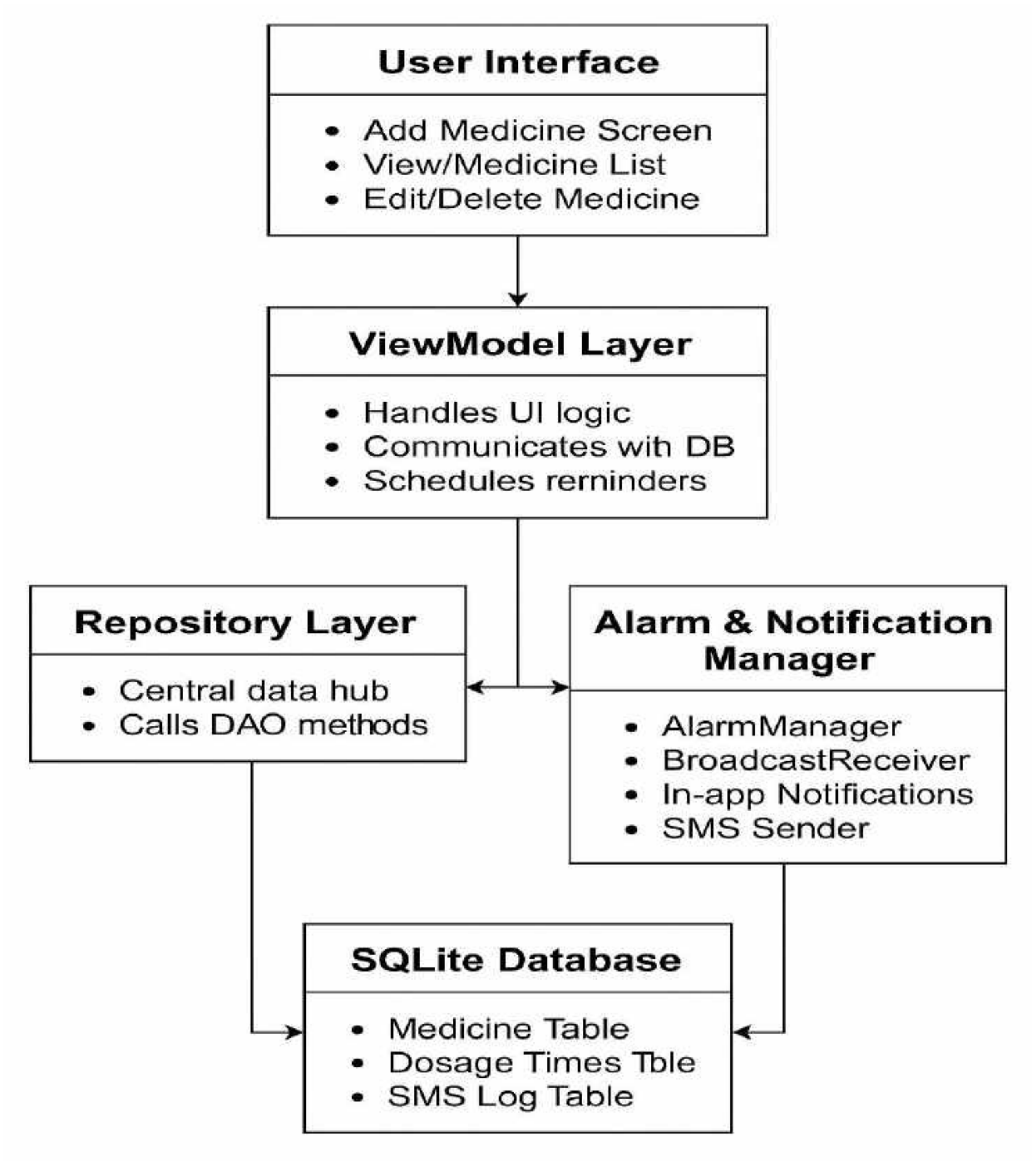
Hardware Requirements:

- Android device (phone or emulator) running API level 21 or higher.
- At least 1 GB RAM and 100 MB free internal storage.
- Development PC with minimum 4 GB RAM for development and testing.

Software Requirements:

- Android device (phone or emulator) running API level 21 or higher.
- At least 1 GB RAM and 100 MB free internal storage.
- Development PC with minimum 4 GB RAM for development and testing.

3.3 Architecture Diagram



Presentation Layer (UI Layer)

- Displays UI for add/view/edit/delete medicine.
- Handles user input for dosage times.

ViewModel Layer

- Manages UI-related data.
- Handles logic between UI and data.

Repository Layer

- Abstracts data access from multiple sources.
- Coordinates between ViewModel and Database.

Database Layer (SQLite)

- Stores medicine details locally.
- Supports CRUD operations.

Notification Layer

- Sends in-app alerts for expiry.
- Triggers SMS reminders for dosage.

Utility Layer

- Handles time selection, validations, and SMS logic.

3.4. ER Diagram

Entities:

.

1.Medicine

Represents each medicine added by the user.

Attributes:

- medicine_id (Primary Key)
- name
- dosage (e.g., 500mg)
- quantity (e.g., number of tablets)
- expiry_date
- dosages_per_day

2. DosageSchedule

Stores the dosage times for each medicine per day.

Attributes:

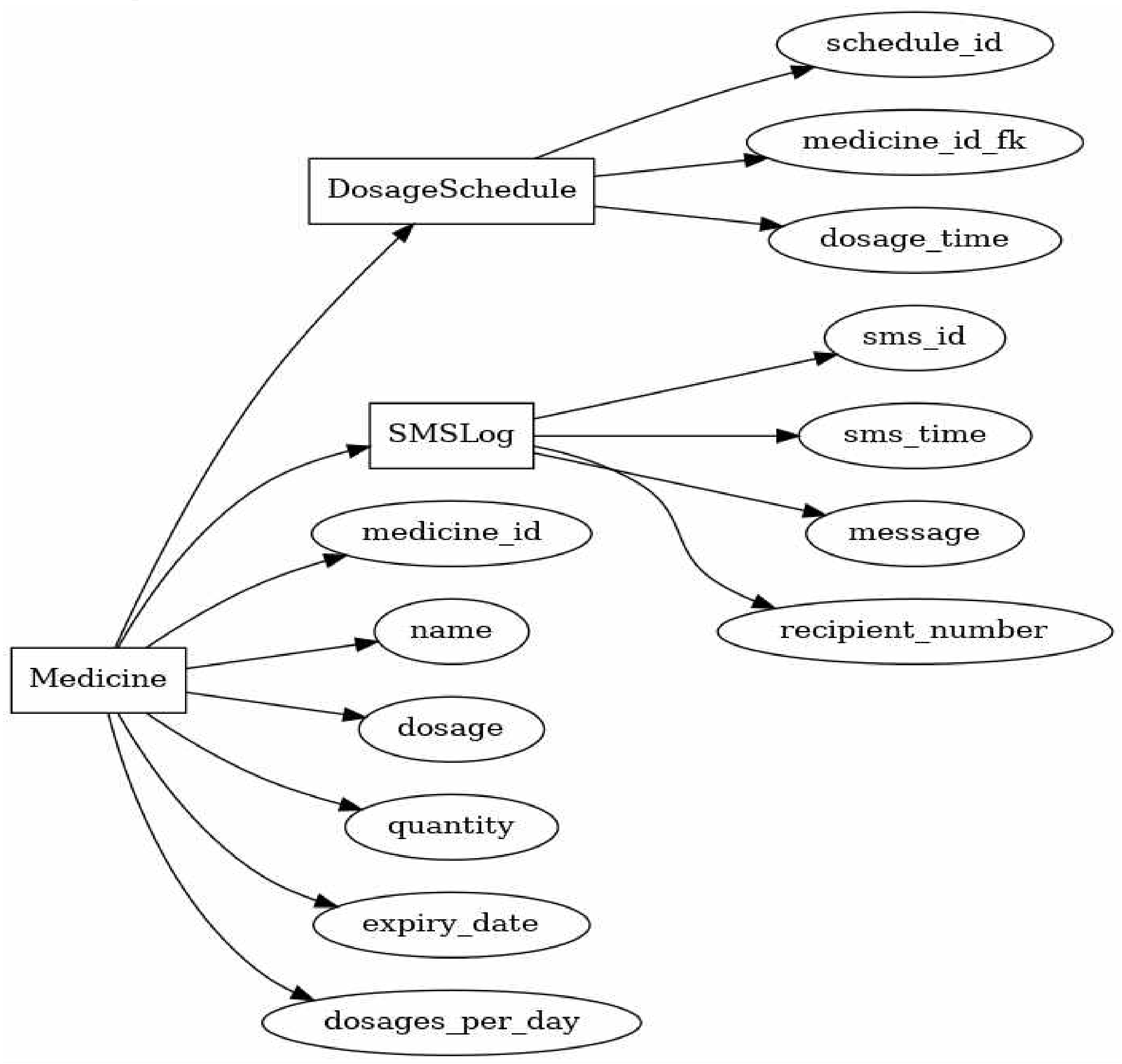
- schedule_id (Primary Key)
- medicine_id_fk (Foreign Key to Medicine)
- dosage_time (e.g., "08:00 AM", "06:00 PM")

3. SMSLog

Tracks the SMS reminders sent to the user.

Attributes:

- sms_id (Primary Key)
- sms_time
- message
- recipient_number



3.5. Normalization

1NF – First Normal Form

- Rule: No repeating groups; each field contains only atomic (indivisible) values.
    - Having one medicine per row in the Medicine table.
    - Storing each dosage time as a separate row in DosageSchedule.

2NF – Second Normal Form

- Rule: Must be in 1NF and all non-key attributes must be fully functionally dependent on the primary key.
    - In DosageSchedule, dosage_time depends on schedule_id, not partially on medicine_id_fk.
    - No partial dependencies exist in composite keys (or you avoided composite keys by assigning schedule_id as PK).

3NF – Third Normal Form

- Rule: Must be in 2NF and no transitive dependencies (non-key fields depending on other non-key fields).
    - In Medicine, attributes like name, dosage, quantity, etc., depend only on medicine_id.
    - In SMSLog, fields like message and recipient_number depend solely on sms_id

This ensures minimal redundancy and efficient query performance.

4.PROGRAM CODE

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.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Medicinemanager"
        tools:targetApi="31">
        <activity android:name=".HomeActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.Medicinemanager">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</application>

</manifest>
```

HomeActivity.Kt

```
package com.example.medicinemanager


import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.widget.Button

class HomeActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_home)

        val addMedicineButton = findViewById<Button>(R.id.btnAddMedicine)
        val showBtn = findViewById<Button>(R.id.btnShowItems)
        addMedicineButton.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }
        showBtn.setOnClickListener {
            val intent = Intent(this, MedicineListActivity::class.java)
            startActivity(intent)
        }
    }
}
```

MainActivity.Kt

```

package com.example.medicinemanager



import android.app.DatePickerDialog
import android.app.TimePickerDialog
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
import java.util.Calendar
import android.Manifest
import android.content.pm.PackageManager
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import android.telephony.SmsManager
import androidx.appcompat.app.AlertDialog
import java.text.SimpleDateFormat
import java.util.Locale

class MainActivity : AppCompatActivity() {

    private var medicineNameToEdit: String? = null
    private var selectedDoseTimes: String = "" // Storing dose times as comma-
separated string

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        supportActionBar?.hide()
        setContentView(R.layout.activity_main)

        val dbHelper = MedicineDatabaseHelper(this)

        val nameEditText = findViewById<EditText>(R.id.editTextMedicineName)
        val dosageEditText = findViewById<EditText>(R.id.editTextDosage)
        val quantityEditText = findViewById<EditText>(R.id.editTextQuantity)
        val expiryDateEditText = findViewById<EditText>(R.id.editTextExpiryDate)
        val doseCountEditText = findViewById<EditText>(R.id.editTextDoseCount)
        val doseTimesButton = findViewById<Button>(R.id.buttonSelectDoseTimes)
        val addButton = findViewById<Button>(R.id.buttonAddMedicine)
        val showItemsButton = findViewById<Button>(R.id.buttonShowItems)

        showExpiryAlertIfAny()

        // Request SMS permission if not granted
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS)
            != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.SEND_SMS), 1)
        }

        sendReminderSMS()

        medicineNameToEdit = intent.getStringExtra("medicine_name_to_edit")
        if (medicineNameToEdit != null) {
            val existingMedicine = dbHelper.getMedicineDetails(medicineNameToEdit!!)
            nameEditText.setText(existingMedicine.name)
            dosageEditText.setText(existingMedicine.dosage)
            quantityEditText.setText(existingMedicine.quantity)
            expiryDateEditText.setText(existingMedicine.expiry)
            doseCountEditText.setText(existingMedicine.doseCount.toString())
            selectedDoseTimes = existingMedicine.doseTimes.joinToString(", ")
            doseTimesButton.text = if (selectedDoseTimes.isNotEmpty())
selectedDoseTimes else "Set Dose Times"
            addButton.text = "Update Medicine"
        }

        expiryDateEditText.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 datePicker = DatePickerDialog(this, { _, y, m, d ->
                val formattedDate = "${"%02d".format(d)}/${"%02d".format(m + 1)}/$y"
                expiryDateEditText.setText(formattedDate)
            }, year, month, day)

            datePicker.show()
        }

        doseTimesButton.setOnClickListener {
            val doseCountStr = doseCountEditText.text.toString().trim()

            if (doseCountStr.isEmpty()) {
                Toast.makeText(this, "Please enter Dose Count first",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            val doseCount = doseCountStr.toIntOrNull()
            if (doseCount == null || doseCount <= 0) {
                Toast.makeText(this, "Invalid Dose Count",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            val doseTimesList = mutableListOf<String>()
            fun showTimePickerDialog(currentCount: Int) {
                val calendar = Calendar.getInstance()
                val hour = calendar.get(Calendar.HOUR_OF_DAY)
                val minute = calendar.get(Calendar.MINUTE)

                val timePickerDialog = TimePickerDialog(
                    this,
                    { _, selectedHour, selectedMinute ->
                        val formattedTime = Calendar.getInstance().apply {
                            set(Calendar.HOUR_OF_DAY, selectedHour)
                            set(Calendar.MINUTE, selectedMinute)
                        }.let {
                            val sdf = SimpleDateFormat("h:mm a", Locale.getDefault())
                            sdf.format(it.time)
                        }

                        doseTimesList.add(formattedTime)

                        if (currentCount < doseCount) {
                            showTimePickerDialog(currentCount + 1)
                        } else {
                            selectedDoseTimes = doseTimesList.joinToString(", ")
                            doseTimesButton.text = selectedDoseTimes
                        }
                    },
                    hour,
                    minute,
                    false
                )
                timePickerDialog.show()
            }

            showTimePickerDialog(1)
        }

        showItemsButton.setOnClickListener {
            startActivity(Intent(this, MedicineListActivity::class.java))
        }

        addButton.setOnClickListener {
            val name = nameEditText.text.toString().trim()
            val dosage = dosageEditText.text.toString().trim()
            val quantity = quantityEditText.text.toString().trim()
            val expiry = expiryDateEditText.text.toString().trim()
            val doseCount = doseCountEditText.text.toString().trim()

            if (name.isEmpty() || dosage.isEmpty() || quantity.isEmpty() || expiry.isEmpty()
    || doseCount.isEmpty() || selectedDoseTimes.isEmpty()) {
                Toast.makeText(this, "Please fill all fields",
    Toast.LENGTH_SHORT).show()
            } else {
                val doseTimesList = selectedDoseTimes.split(",").map { it.trim() }

                val success = if (medicineNameToEdit != null) {
                    dbHelper.updateMedicine(
                        medicineNameToEdit!!,
                        dosage,
                        quantity,
                        expiry,
                        doseCount.toInt(),
                        doseTimesList
                    )
                } else {
                    dbHelper.insertMedicine(
                        name,
                        dosage,
                        quantity,
                        expiry,
                        doseCount.toInt(),
                        doseTimesList
                    )
                }

                if (success) {
                    Toast.makeText(this, if (medicineNameToEdit != null) "Item Updated"
    else "Item Added", Toast.LENGTH_SHORT).show()

                    if (medicineNameToEdit == null) {
                        nameEditText.text.clear()
                        dosageEditText.text.clear()
                        quantityEditText.text.clear()
                        expiryDateEditText.text.clear()
                        doseCountEditText.text.clear()
                        doseTimesButton.text = "Set Dose Times"
                        selectedDoseTimes = ""
                        nameEditText.requestFocus()
                    } else {
                        startActivity(Intent(this, MedicineListActivity::class.java))
                        finish()
                    }
                } else {
                    Toast.makeText(this, "Failed to add/update item",
    Toast.LENGTH_SHORT).show()
                }
            }
        }
    }

    private fun showExpiryAlertIfAny() {
        val dbHelper = MedicineDatabaseHelper(this)
        val cursor = dbHelper.getAllMedicines()

        val today = Calendar.getInstance()
        val tomorrow = Calendar.getInstance()
        tomorrow.add(Calendar.DAY_OF_YEAR, 1)

        val expiringMeds = mutableListOf<String>()

        while (cursor.moveToNext()) {
            val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))
            val expiryStr = cursor.getString(cursor.getColumnIndexOrThrow("expiry"))

            try {
                val sdf = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())
                val expiryDate = sdf.parse(expiryStr)

                expiryDate?.let {
                    val cal = Calendar.getInstance()
                    cal.time = it

                    if (isSameDay(cal, today) || isSameDay(cal, tomorrow)) {
                        expiringMeds.add("$name - Exp: $expiryStr")
                    }
                }
            } catch (e: Exception) {
                e.printStackTrace()
            }
        }

        cursor.close()

        if (expiringMeds.isNotEmpty()) {
            val message = expiringMeds.joinToString("\n")
```

```
          AlertDialog.Builder(this)
              .setTitle("Medicine Expiry Alert")
              .setMessage("The following medicines are expiring today or
tomorrow:\n\n$message")
              .setPositiveButton("OK", null)
              .show()
      }
   }

   private fun isSameDay(cal1: Calendar, cal2: Calendar): Boolean {
      return cal1.get(Calendar.YEAR) == cal2.get(Calendar.YEAR) &&
              cal1.get(Calendar.DAY_OF_YEAR) ==
cal2.get(Calendar.DAY_OF_YEAR)
   }

   private fun sendReminderSMS() {
      val dbHelper = MedicineDatabaseHelper(this)
      val cursor = dbHelper.getAllMedicines()

      if (cursor.moveToFirst()) {
          do {
              val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))
              val dosage = cursor.getString(cursor.getColumnIndexOrThrow("dosage"))

              val message = "Reminder: Take your medicine '$name' - Dosage: $dosage"
              val smsManager = SmsManager.getDefault()
              smsManager.sendTextMessage("9342568667", null, message, null, null)

          } while (cursor.moveToNext())

          Toast.makeText(this, "Reminder SMS sent to 9342568667 ",
Toast.LENGTH_SHORT).show()
      }

      cursor.close()
   }
}
```

Medicine.kt

```
package com.example.medicinemanager

data class Medicine(
    val name: String,
    val dosage: String,
    val quantity: String,
    val expiry: String,
    val doseCount: Int,
    val doseTimes: List<String>
)
```

MedicineDatabaseHelper.Kt

```
package com.example.medicinemanager

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

class MedicineDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, "MedicineDB", null, 2) { // 🚨 VERSION UPDATED
to 2 (was 1)

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE Medicine (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT,
                dosage TEXT,
                quantity TEXT,
                expiry TEXT,
                doseCount INTEGER,            -- NEW
                doseTimes TEXT                -- NEW
            );
        """
        db.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        // 🚨 Upgrade: Add columns if version updates
        if (oldVersion < 2) {
            db.execSQL("ALTER TABLE Medicine ADD COLUMN doseCount
INTEGER DEFAULT 0")
            db.execSQL("ALTER TABLE Medicine ADD COLUMN doseTimes TEXT
DEFAULT ''")
        }
    }

    fun insertMedicine(name: String, dosage: String, quantity: String, expiry: String,
doseCount: Int, doseTimes: List<String>): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put("name", name)
            put("dosage", dosage)
            put("quantity", quantity)
            put("expiry", expiry)
            put("doseCount", doseCount)                         // NEW
            put("doseTimes", doseTimes.joinToString(","))        // NEW - save list as
comma-separated
        }
        val result = db.insert("Medicine", null, values)
        return result != -1L
    }

    fun deleteMedicine(name: String): Int {
        val db = this.writableDatabase
        return db.delete("Medicine", "name=?", arrayOf(name))
    }

    fun getMedicineDetails(name: String): Medicine {
        val db = readableDatabase
        val cursor = db.query(
            "Medicine",
            null,
            "name = ?",
            arrayOf(name),
            null,
            null,
            null
        )

        return if (cursor.moveToFirst()) {
            val dosage = cursor.getString(cursor.getColumnIndexOrThrow("dosage"))
            val quantity =
cursor.getString(cursor.getColumnIndexOrThrow("quantity"))
```

```
            val expiry = cursor.getString(cursor.getColumnIndexOrThrow("expiry"))
            val doseCount =
cursor.getInt(cursor.getColumnIndexOrThrow("doseCount"))        // NEW
            val doseTimesString =
cursor.getString(cursor.getColumnIndexOrThrow("doseTimes")) // NEW
            val doseTimes = if (doseTimesString.isNullOrEmpty()) emptyList() else
doseTimesString.split(",")

            cursor.close()
            Medicine(name, dosage, quantity, expiry, doseCount, doseTimes)
        } else {
            cursor.close()
            Medicine("", "", "", "", 0, emptyList())
        }
    }

    fun updateMedicine(name: String, newDosage: String, newQuantity: String,
newExpiry: String, newDoseCount: Int, newDoseTimes: List<String>): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put("dosage", newDosage)
            put("quantity", newQuantity)
            put("expiry", newExpiry)
            put("doseCount", newDoseCount)                          // NEW
            put("doseTimes", newDoseTimes.joinToString(","))        // NEW
        }
        val result = db.update(
            "Medicine",
            values,
            "name = ?",
            arrayOf(name)
        )
        return result > 0
    }

    fun getAllMedicines(): Cursor {
        return readableDatabase.rawQuery("SELECT * FROM Medicine", null)
    }
}
```

MedicineListActivity.kt

```
package com.example.medicinemanager

import android.content.Intent
import android.database.Cursor
import android.os.Bundle
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.ListView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MedicineListActivity : AppCompatActivity() {

    private lateinit var dbHelper: MedicineDatabaseHelper
    private lateinit var listView: ListView
    private lateinit var list: MutableList<String>

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

        dbHelper = MedicineDatabaseHelper(this)
        listView = findViewById(R.id.medicineListView)

        loadMedicineList()

        listView.onItemClickListener = AdapterView.OnItemClickListener { _, _,
position, _ ->
            val selectedText = list[position] // "Paracetamol - 500mg - 10 - Exp:
25/04/2025"
            val parts = selectedText.split(" - ")

            if (parts.size >= 6) { // Now expecting 6 parts
                val name = parts[0].trim()
                val dosage = parts[1].trim()
                val quantity = parts[2].trim()
                val expiry = parts[3].replace("Exp: ", "").trim()
                val doseCount = parts[4].trim()  // New field
                val doseTimes = parts[5].trim()  // New field

                val builder = AlertDialog.Builder(this)
                builder.setTitle("Choose Action")
            builder.setItems(arrayOf("Edit", "Delete")) { _, which ->
                when (which) {
                    0 -> {
                        // Edit
                        val intent = Intent(this, MainActivity::class.java).apply {
                            putExtra("medicine_name_to_edit", name)
                            putExtra("dosage_to_edit", dosage)
                            putExtra("quantity_to_edit", quantity)
                            putExtra("expiry_to_edit", expiry)
                            putExtra("dose_count_to_edit", doseCount)  // Pass new fields
                            putExtra("dose_times_to_edit", doseTimes)  // Pass new fields
                        }
                        startActivity(intent)
                    }
                    1 -> {
                        // Delete
                        val rowsDeleted = dbHelper.deleteMedicine(name)
                        if (rowsDeleted > 0) {
                            Toast.makeText(this, "Medicine deleted",
Toast.LENGTH_SHORT).show()
                            list.removeAt(position)
                            (listView.adapter as ArrayAdapter<*>).notifyDataSetChanged()
                        } else {
                            Toast.makeText(this, "Failed to delete medicine",
Toast.LENGTH_SHORT).show()
                        }
                    }
                }
            }
            builder.show()
        } else {
            Toast.makeText(this, "Unable to parse item",
Toast.LENGTH_SHORT).show()
        }
    }
}

private fun loadMedicineList() {
    val cursor: Cursor = dbHelper.getAllMedicines()
    list = mutableListOf()

    while (cursor.moveToNext()) {
        val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))
        val dosage = cursor.getString(cursor.getColumnIndexOrThrow("dosage"))
        val quantity =
cursor.getString(cursor.getColumnIndexOrThrow("quantity"))
        val expiry = cursor.getString(cursor.getColumnIndexOrThrow("expiry"))
        val doseCount =
cursor.getString(cursor.getColumnIndexOrThrow("doseCount"))  // New field
        val doseTimes =
cursor.getString(cursor.getColumnIndexOrThrow("doseTimes"))  // New field
        list.add("$name - $dosage - $quantity - Exp: $expiry - Doses: $doseCount -
Times: $doseTimes")
    }
    cursor.close()

    val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, list)
    listView.adapter = adapter
}
}
```

Activity_home.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:gravity="center"
    android:padding="24dp"
    android:background="#F9F9F9">

    <TextView
        android:id="@+id/appTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Medicine Manager"
        android:textSize="26sp"
        android:textStyle="bold"
        android:textColor="#2E7D32"
        android:layout_marginBottom="48dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        android:background="#FFFFFF"
        android:padding="20dp"
        android:elevation="4dp"
        android:layout_margin="16dp"
        android:layout_marginTop="0dp"
        android:layout_marginBottom="0dp"
        android:backgroundTint="#FFFFFF"
        android:clipToPadding="false"
        android:layout_gravity="center_horizontal">

        <Button
            android:id="@+id/btnAddMedicine"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Add Medicine"
            android:layout_marginBottom="20dp" />

        <Button
            android:id="@+id/btnShowItems"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Show Items" />
    </LinearLayout>

</LinearLayout>
```

Activity_main.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:paddingTop="60dp">
    <EditText
        android:id="@+id/editTextMedicineName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Medicine Name" />

    <EditText
        android:id="@+id/editTextDosage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Dosage ( e.g.How much mg the tablet is )"
        android:inputType="number"
        android:layout_marginTop="8dp" />

    <EditText
        android:id="@+id/editTextQuantity"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Quantity"
        android:inputType="number"
        android:layout_marginTop="8dp" />

    <EditText
        android:id="@+id/editTextExpiryDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Expiry Date (dd/MM/yyyy)"
        android:focusable="false"
        android:layout_marginTop="8dp" />

    <!-- New: Number of doses per day -->
    <EditText
        android:id="@+id/editTextDoseCount"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="How many times to take daily?"
        android:inputType="number"
        android:layout_marginTop="8dp" />

    <!-- Button to select times -->
    <Button
        android:id="@+id/buttonSelectDoseTimes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
          android:text="Set Dose Times"
          android:layout_marginTop="8dp"
          android:layout_gravity="center_horizontal" />

      <!-- TextView to display selected times -->
      <TextView
          android:id="@+id/textViewSelectedTimes"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="Selected times will appear here"
          android:layout_marginTop="8dp"
          android:textAlignment="center" />

      <Button
          android:id="@+id/buttonAddMedicine"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="Add Medicine"
          android:layout_marginTop="16dp" />

      <Button
          android:id="@+id/buttonShowItems"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Show Items"
          android:layout_marginTop="16dp"
          android:layout_gravity="center_horizontal" />

    </LinearLayout>
</ScrollView>
```

Activity_main_list.xml

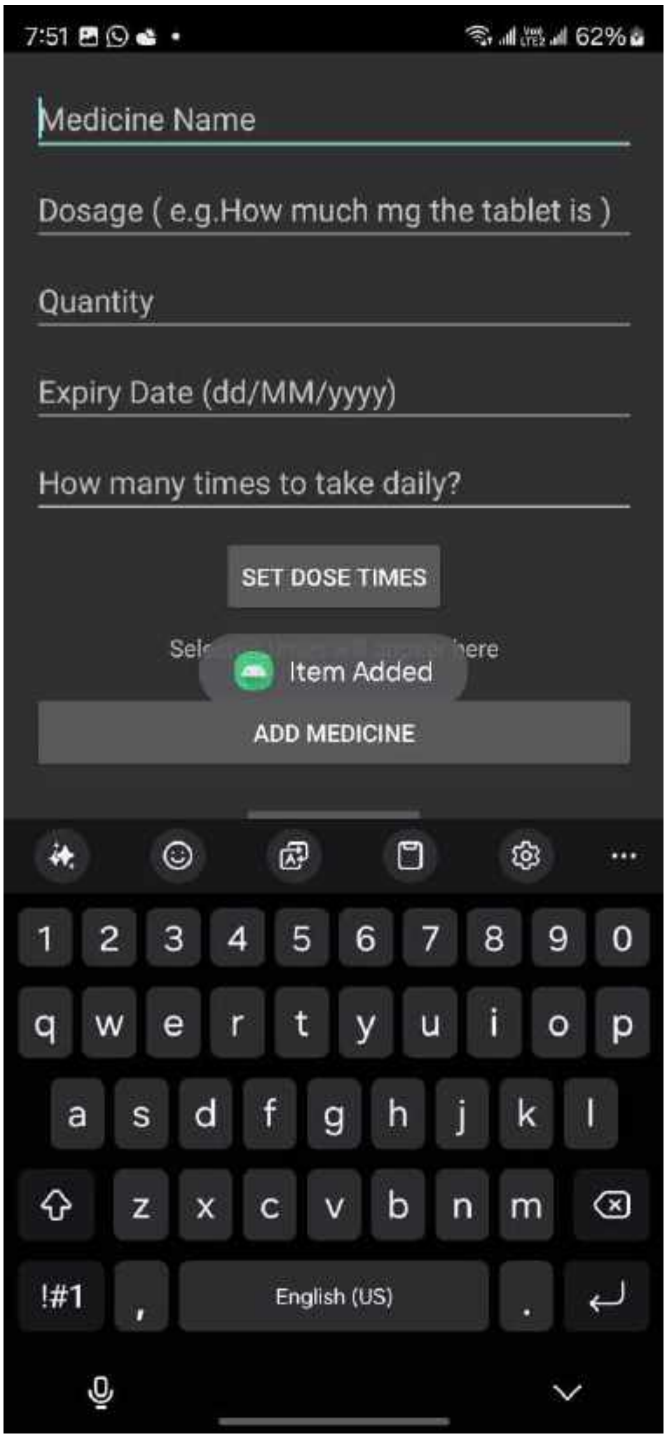
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

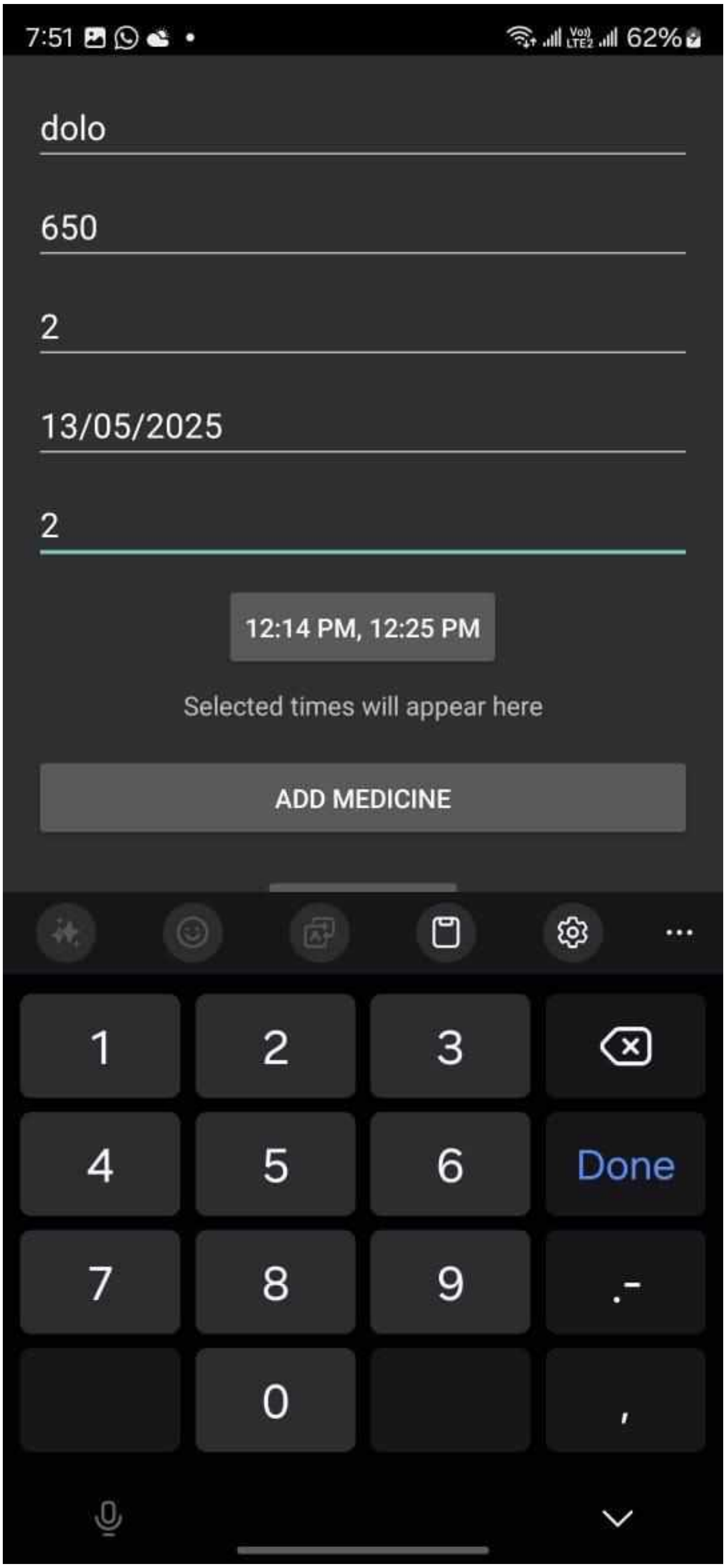
```xml
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="All Medicines"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_marginBottom="12dp" />

    <ListView
        android:id="@+id/medicineListView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="1dp" />
</LinearLayout>
```

4 OUTPUT


7:51
62%
Medicine Name
Dosage (e.g.How much mg the tablet is)
Quantity
Expiry Date (dd/MM/yyyy)
How many times to take daily?
SET DOSE TIMES
Item Added
ADD MEDICINE
English (US)



7:51 62%

dolo

650

2

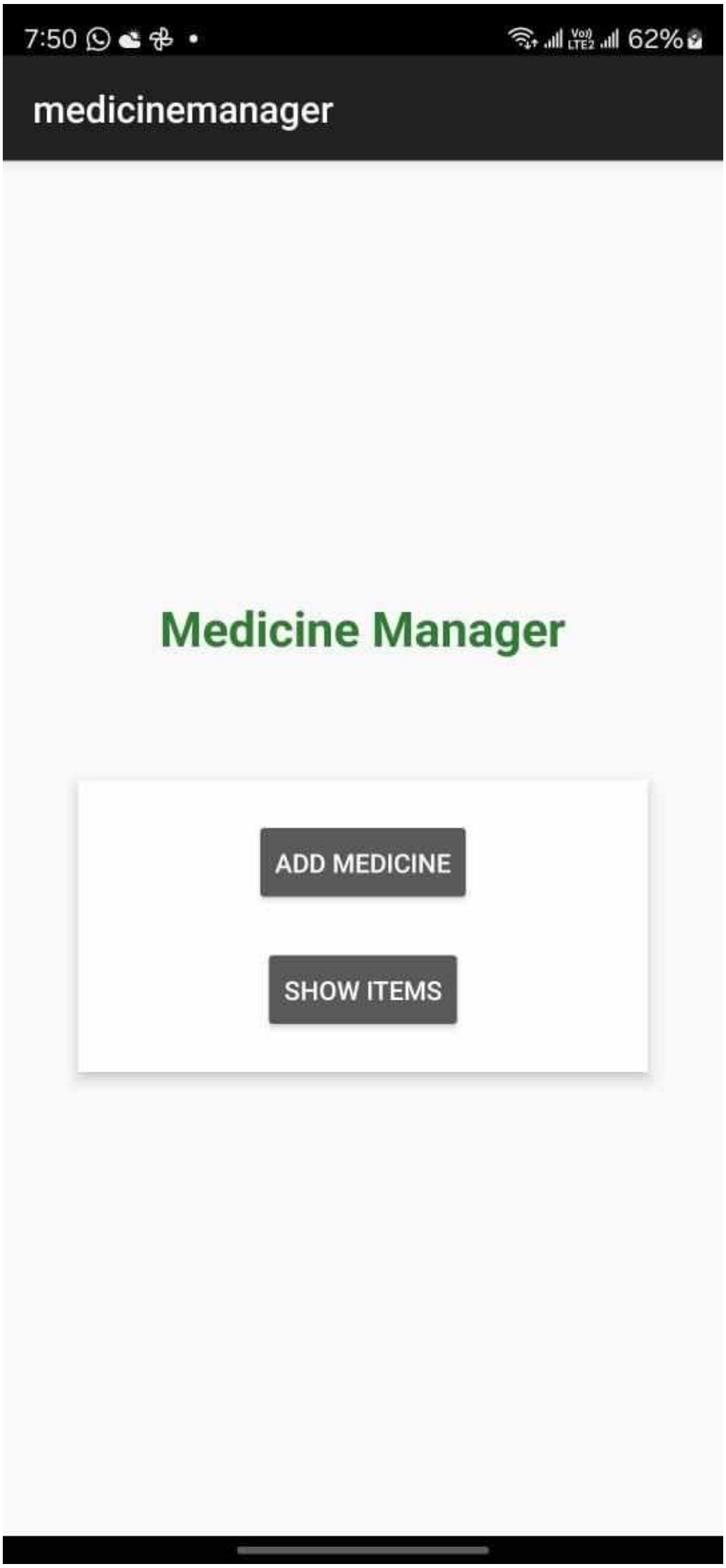
13/05/2025

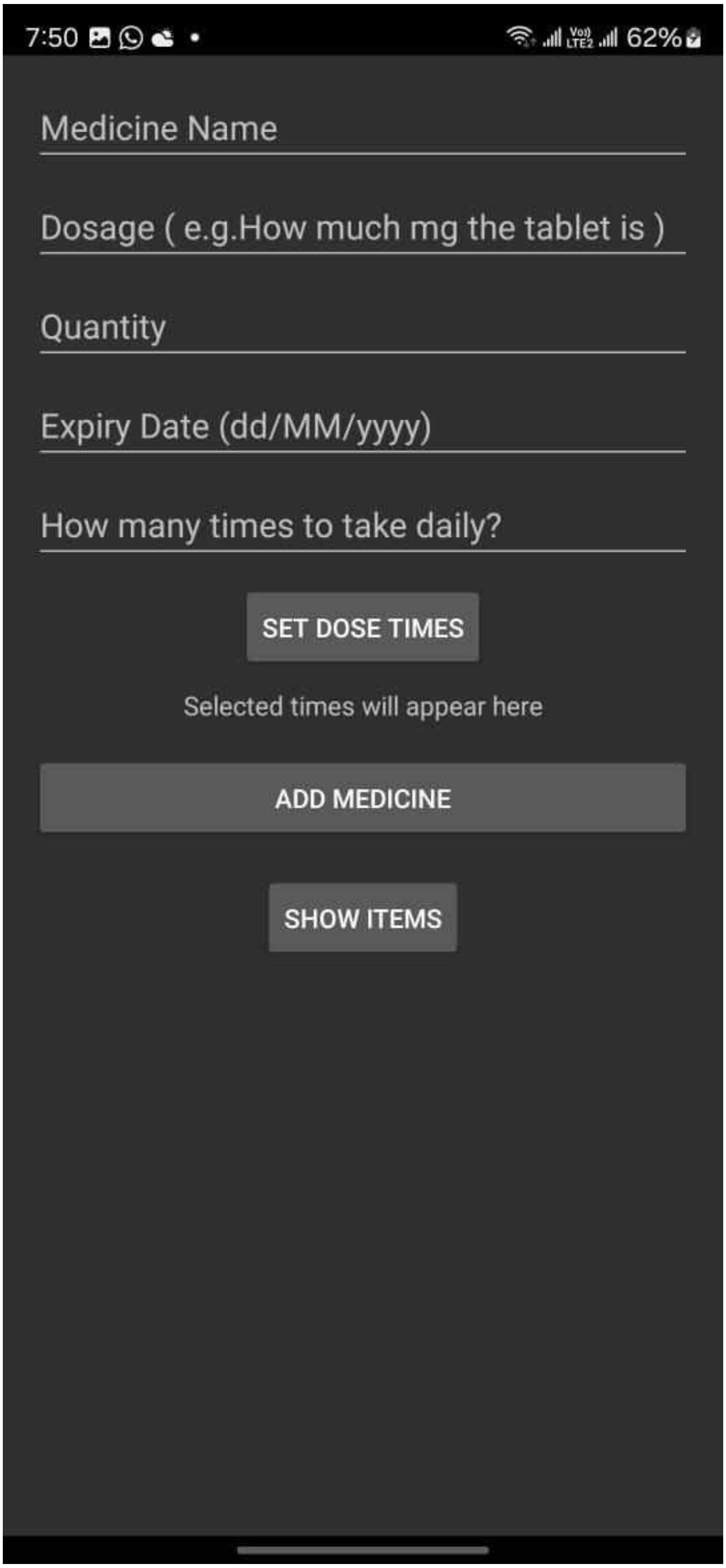
2

12:14 PM, 12:25 PM

Selected times will appear here

ADD MEDICINE


7:50
62%
medicinemanager
Medicine Manager
ADD MEDICINE
SHOW ITEMS


7:50 • 62%

Medicine Name

Dosage (e.g.How much mg the tablet is)

Quantity

Expiry Date (dd/MM/yyyy)

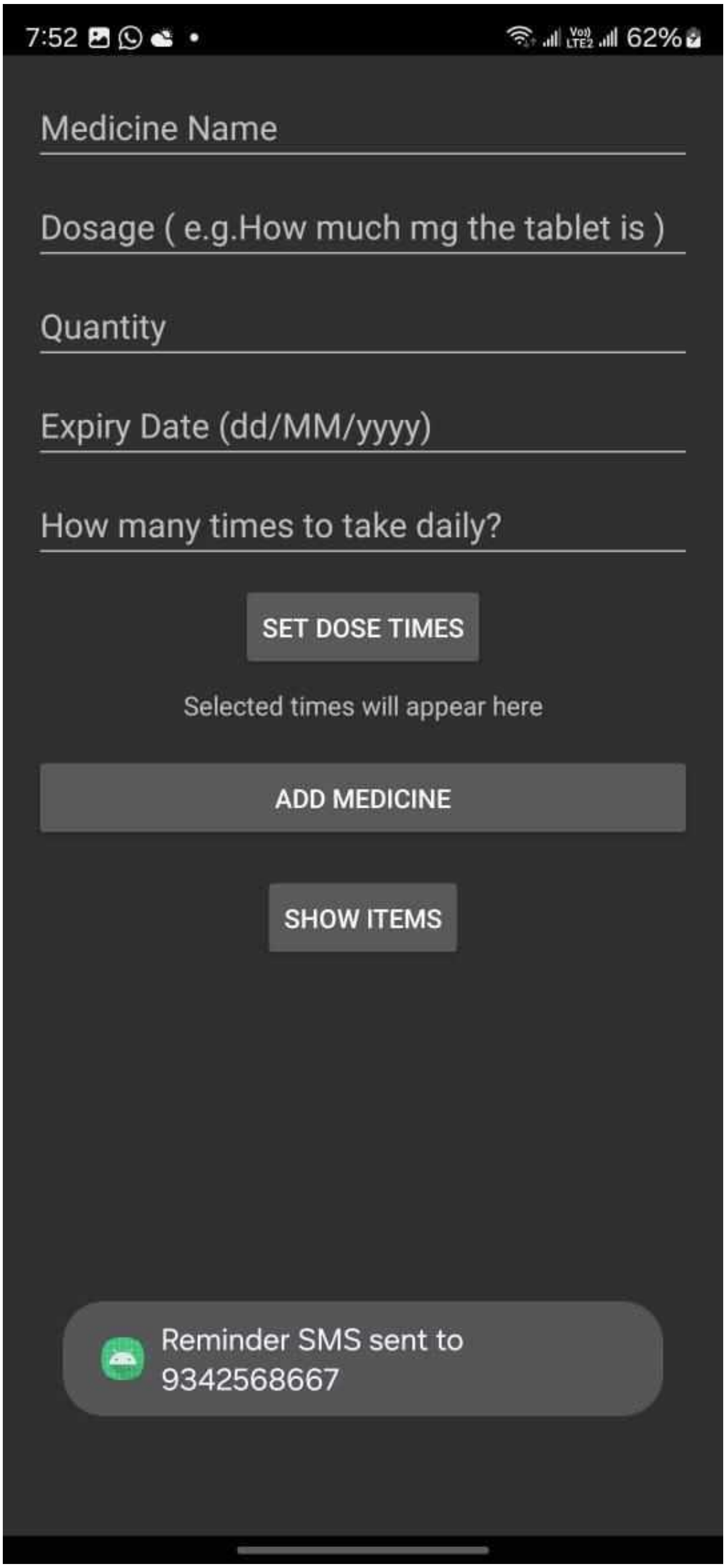
How many times to take daily?

SET DOSE TIMES

Selected times will appear here

ADD MEDICINE

SHOW ITEMS


7:52
62%
Medicine Name
Dosage (e.g.How much mg the tablet is)
Quantity
Expiry Date (dd/MM/yyyy)
How many times to take daily?
SET DOSE TIMES
Selected times will appear here
ADD MEDICINE
SHOW ITEMS

Reminder SMS sent to 9342568667

5 RESULTS AND DISCUSSION

Upon deployment and testing, the Medicine Manager app fulfilled its core objectives effectively. The interface was responsive and user-friendly, allowing smooth navigation between adding, viewing, editing, and deleting medicines. The dosage time input logic performed accurately, adjusting dynamically based on the number of doses per day.

Key highlights:

- Expiry Alerts: In-app alerts for nearing expiry dates worked reliably, helping users avoid consuming expired medicines.
- SMS Reminders: SMS alerts were sent successfully to the user’s registered number, reminding them of their scheduled doses.
- Local Storage: SQLite ensured fast, consistent access to stored medicine data, even on low-end devices.
- Dynamic UI: Kotlin’s data binding allowed seamless UI updates, reducing errors and improving maintainability.

Potential improvements:

1. Integrate cloud sync to back up and restore medicine data.
2. Enable full search and filter features for large medicine lists.
3. Add notification history and user confirmation for taken medicines.

6 CONCLUSION

The **Medicine Manager** app provides a complete and user-centric solution for tracking medicine intake and expiry. By combining Kotlin, XML, and SQLite, the app offers offline functionality, dosage reminders, and expiry alerts—all within a simple, intuitive interface. The modular design supports future scalability and enhancement. Future versions can expand its capabilities with advanced search, medicine interaction warnings, and cloud-based backup features to further assist users in managing their health more effectively.

7 REFERENCES

1. Android Developer Documentation: https://developer.android.com/

2. Kotlin Official Documentation: https://kotlinlang.org/docs/

3. SQLite Documentation: https://www.sqlite.org/

4. Git SCM Documentation: https://git-scm.com/doc

5. JSON.org: https://www.json.org/json-en.html
6. https://www.mysqltutorial.org/mysql-triggers.aspx