FRESH TRACK -TRACKING FRESHNESS AND EXPIRY OF FOOD ITEMS

CS19611 - MOBILE APPLICATION DEVELOPMENT LABORATORY

PROJECT REPORT

Submitted by

SHREYA MRIDULA G **(2116220701271)**

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 **"FRESH TRACK-TRACKING FRESHNESS AND EXPIRY OF FOOD ITEMS"** is the bonafide work of **"SHREYA MRIDULA G (2116220701271)"** 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
Dr. P. Kumar., M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College,
Chennai - 602 105.

SIGNATURE
Dr. N. Duraimurugan., M.E., Ph.D.,
SUPERVISOR
Associate Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering 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.

SHREYA MRIDULA 2116220701271

ABSTRACT

The Fridge Manager app is an Android-based solution aimed at helping users efficiently manage the contents of their refrigerator. Built using Kotlin and XML in Android Studio, the app allows users to add, update, and delete food items with associated details such as quantity and expiry date. A key feature of the app is its ability to send alerts via Toast, SMS, or dialog messages to notify users of upcoming expiries. Additionally, the app includes a smart recipe suggestion feature that recommends recipes based on expiring items, ensuring minimal food waste. The user-friendly interface, powered by RecyclerView and SQLite database integration, provides a smooth and intuitive experience for tracking and managing food inventory. The system also stores user contact info for sending expiry alerts, enhancing its utility and personalization. The application has been tested on Android emulators and is designed to work offline, making it practical and accessible. This project demonstrates the integration of various Android components such as database handling, UI design, telephony services, and activity lifecycle management

TABLE OF CONTENTS

CHAPTER 1

1. INTRODUCTION

In today's fast-paced lifestyle, people often forget the items stored in their fridge, leading to unnecessary food spoilage and wastage. The Fridge Manager app is developed as a solution to this problem, providing a user-friendly interface for tracking all stored food products along with their expiry dates. It sends timely alerts to remind users before items expire and even suggests recipes that can be made using these ingredients. The app also collects basic contact information to send SMS/email alerts, ensuring users are always informed. This utility-oriented application serves as a personal assistant for kitchen management and promotes responsible food usage.

2. Objectives

- To build an intuitive app that helps users track and manage fridge items.
- To notify users about expiring items via SMS and dialogs.
- To suggest useful recipes based on ingredients nearing expiry.
- To maintain a persistent local database of food items and contact info.

3. Modules

The app is organized into the following modules:

1. **Add Item Module**
   Enables users to input food items with name, quantity, and expiry date.

2. **View Items Module**
   Displays a list of all saved food items using ListView.

3. **Update/Delete Module**
   Allows editing or removing existing items from the database.

4. **Expiry Alert Module**
   Triggers daily checks and shows an alert dialog for items expiring today or tomorrow.

5. SMS/Email Module
   Sends alerts about expiring items to the user via SMS or email.

6. **Recipe Suggestion Module**
   Recommends recipes using ingredients that are about to expire.

7. **Database Module**
   Uses SQLite for storing and retrieving all food item data persistently.

CHAPTER 2 SURVEY OF TECHNOLOGY

1. Study of technology

To build the Fridge Manager Android application, a combination of reliable technologies and tools was used. The selection was driven by the need for simplicity, offline functionality, and efficient performance. The primary technologies used include the Android SDK, Kotlin programming language, SQLite for local database management, and XML for UI design. Android Studio was selected as the development environment due to its rich feature set, excellent support for Kotlin, and integrated tools for emulators and debugging. The app also leverages system services like AlarmManager for periodic checks and SmsManager for sending SMS reminders. The use of native Android components allows the app to work efficiently without external dependencies, ensuring speed and low resource consumption.

2. Software Description

- Android Studio: Android Studio is the official IDE for Android development and was used for building, testing, and debugging the app. It provides features like visual layout editors, Gradle build system, emulator integration, and version control.
- **SQLite**: A lightweight embedded database used for local storage. It helps in storing food items along with their expiry dates, quantities, and related recipe data.
- **AlarmManager**: Used to set daily expiry checks and trigger background alerts.
- **Android SDK**: Provides APIs for building apps that can run on all Android devices.
- **Gradle**: The app uses Gradle for project building and dependency management.

3. Requirements And Analysis

The Fridge Manager app is designed to provide a practical solution to users who struggle with managing perishable food items efficiently. The system performs automatic tracking of expiry dates, suggests recipes using expiring ingredients, and notifies users via alerts or SMS to reduce waste and improve planning. This section details the functional and non-functional requirements, system needs, and visual models to understand how the system operates internally.

The functional requirements of the Fridge Manager app ensure that users can effectively manage their perishable food items. The app allows users to add food items along with their name, quantity, and expiry date, storing all this data locally using SQLite for offline access. It automatically checks the expiry dates daily and alerts the user through Toast messages or SMS. Users can edit or delete existing items and input their contact details such as phone number and email to receive notifications. Furthermore, the app suggests recipes using items that are nearing expiry and displays all recipe suggestions upon clicking the “Suggestions” button. In terms of non-functional requirements, the app is designed to function entirely offline, without any dependency on internet connectivity. It offers a clean, user-friendly interface and relies on Android’s AlarmManager for performing efficient daily expiry checks. Additionally, all data operations including insertion, deletion, and updates are executed quickly and reliably to ensure smooth user experience.

3.1 Hardware and Software Requirements

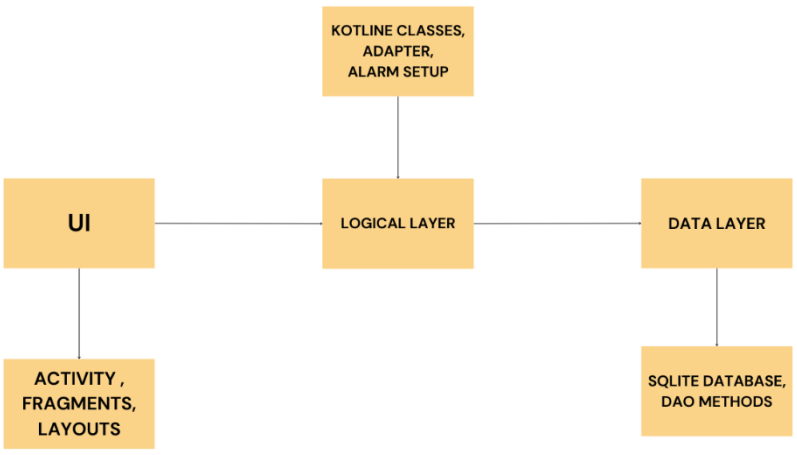
Hardware Requirements:

- Android smartphone or emulator.
- Minimum 1 GB RAM and 100 MB of available storage.
- Optional: Development machine with at least 4 GB RAM for building the APK.

Software Requirements:

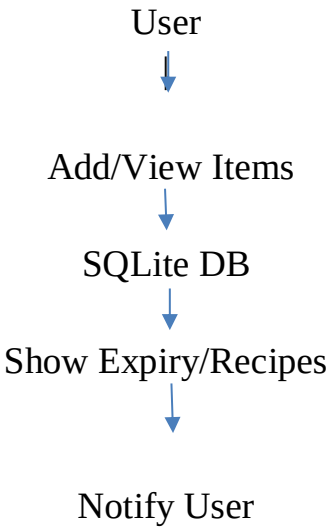
- Android Studio Arctic Fox or later
- JDK 11 or later
- Android SDK
- Kotlin Plugin
- SQLite Database
- Gradle 7.0 or above
- Emulator or physical Android device

3.2Architecture Diagram



3.3 DATA FLOW DIAGRAM:

LEVEL 1:



Level 1 DFD:

- Add Item Activity → DB Insert
- View/Update/Delete Activity → DB Query/Delete
- Alert Module → Reads DB → Triggers SMS/Email
- Suggestion Module → Reads DB → Filters & Displays Recipes

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-feature
        android:name="android.hardware.telephony"
        android:required="false" />

    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <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.FridgeManager1"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

    </application>

</manifest>
```

2.activity_main.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"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/home_linear_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <!-- App title -->
    <TextView
        android:id="@+id/appTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textAlignment="center"
        android:textColor="@color/purple_700"
        android:layout_marginBottom="16dp" />

    <!-- Empty view -->
    <TextView
        android:id="@+id/emptyView"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/empty_message"
    android:textSize="16sp"
    android:textAlignment="center"
    android:textColor="#888888"
    android:layout_marginBottom="16dp"
    android:visibility="gone" />

<!-- RecyclerView -->
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/fridgeItemsRecyclerView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:contentDescription="Fridge item list"
    tools:listitem="@layout/item_fridge" />

<Button
    android:id="@+id/btnSuggestions"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Suggestions"
    android:layout_margin="16dp"
    android:backgroundTint="@color/purple_200"
    android:textColor="@android:color/white"
    android:layout_gravity="end" />

<ListView
    android:id="@+id/listViewRecipes"
    android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp" />

    <!-- FAB -->
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/addItemFab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="8dp"
        android:contentDescription="@string/add_item"
        android:layout_gravity="center_horizontal"
        app:srcCompat="@android:drawable/ic_input_add"
        app:tint="@android:color/white"
        app:backgroundTint="@color/purple_500" />

</LinearLayout>
```

3.activity_suggestions.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="16dp">

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

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Your Suggestions Will Appear Here"
        android:textSize="18sp"
        android:padding="16dp"/>
</LinearLayout>
```

4.item_receipe.xml

```
?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="12dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/recipeTitle"
        android:textStyle="bold"
        android:textSize="18sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/recipeIngredients"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

5.item_fridge.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="horizontal"
    android:padding="12dp"
    android:background="@android:color/white"
    android:gravity="center_vertical">

    <!-- Icon or image for the item -->
    <ImageView
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:src="@drawable/ic_food"

        android:background="#CCCCCC" /> <!-- Light gray background -->



    <!-- Vertical layout for name and date -->
    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">

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

```
            android:text="Milk"
            android:textSize="18sp"
            android:textStyle="bold"
            android:textColor="#000000" />
        <TextView
            android:id="@+id/itemQuantityText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="1L"
            android:textSize="18sp"
            android:textStyle="bold"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/itemDateText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Expires: 25 Apr"
            android:textSize="14sp"
            android:textColor="#666666" />
    </LinearLayout>

</LinearLayout>
```

6.mainactivity.kt

```
package com.example.fridgemanager1

import android.Manifest
import android.app.AlarmManager
import android.app.PendingIntent
```

```
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.os.Bundle
import android.telephony.SmsManager
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.widget.*
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton
import java.text.SimpleDateFormat
import java.util.*


class MainActivity : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var fridgeAdapter: FridgeAdapter
    private val itemList = mutableListOf<FridgeItem>()
    private lateinit var dbHelper: FridgeDatabaseHelper
    private lateinit var btnSuggestions: Button
    private lateinit var listViewRecipes: ListView

    private var userPhone: String = ""
    private var userEmail: String = ""
```

```
    private val allRecipes = listOf(
        "Grilled Cheese Sandwich",
        "Pasta with Tomato Sauce",
        "Vegetable Stir Fry",
        "Egg Fried Rice",
        "Fruit Salad"
    )

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

        btnSuggestions = findViewById(R.id.btnSuggestions)
        listViewRecipes = findViewById(R.id.listViewRecipes)

        if (ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS)
            != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.SEND_SMS), 1)
        }

        dbHelper = FridgeDatabaseHelper(this)
        itemList.addAll(dbHelper.getAllItems())

        checkExpiringItems()

        scheduleDailyExpiryCheck()
        showContactDialog()

        recyclerView = findViewById(R.id.fridgeItemsRecyclerView)
        fridgeAdapter = FridgeAdapter(itemList) { position ->
```

```
    val item = itemList[position]
    AlertDialog.Builder(this)
        .setTitle("Choose Action")
        .setItems(arrayOf("Edit", "Delete")) { _, which ->
            when (which) {
                0 -> showEditDialog(position, item)
                1 -> {
                    if (dbHelper.deleteItem(item.id)) {
                        itemList.removeAt(position)
                        fridgeAdapter.notifyItemRemoved(position)
                        Toast.makeText(this, "Item deleted", Toast.LENGTH_SHORT).show()
                    }
                }
            }
        }.show()
}

recyclerView.layoutManager = LinearLayoutManager(this)
recyclerView.adapter = fridgeAdapter

findViewById<FloatingActionButton>(R.id.addItemFab).setOnClickListener {
    val dialogView = layoutInflater.inflate(R.layout.dialog_add_item, null)
    val editName = dialogView.findViewById<EditText>(R.id.editItemName)
    val editQuantity = dialogView.findViewById<EditText>(R.id.editItemQuantity)
    val editExpiry = dialogView.findViewById<EditText>(R.id.editItemExpiry)
    val btnAdd = dialogView.findViewById<Button>(R.id.btnAddItem)

    val dialog = AlertDialog.Builder(this).setView(dialogView).create()

    btnAdd.setOnClickListener {
        val name = editName.text.toString()
```

```
            val quantity = editQuantity.text.toString()
            val expiry = editExpiry.text.toString()

            if (name.isNotEmpty() && quantity.isNotEmpty() && expiry.isNotEmpty()) {
                val newItem = FridgeItem(name = name, quantity = quantity, expiryDate =
expiry)
                val insertedId = dbHelper.insertItem(newItem)

                if (insertedId != -1L) {
                    itemList.add(FridgeItem(insertedId.toInt(), name, quantity, expiry))
                    fridgeAdapter.notifyItemInserted(itemList.size - 1)
                    dialog.dismiss()
                } else {
                    Toast.makeText(this, "Failed to insert item",
Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(this, "Please fill all fields", Toast.LENGTH_SHORT).show()
            }
        }
        dialog.show()
    }

    btnSuggestions.setOnClickListener {
        val sdf = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())
        val today = Calendar.getInstance().apply {
            set(Calendar.HOUR_OF_DAY, 0)
            set(Calendar.MINUTE, 0)
            set(Calendar.SECOND, 0)
            set(Calendar.MILLISECOND, 0)
        }.time
```

```
        val expiringItems = dbHelper.getAllItems().filter {
            try {
                val expiry = sdf.parse(it.expiryDate)
                val diff = (expiry?.time ?: 0) - today.time
                val daysLeft = diff / (1000 * 60 * 60 * 24)
                daysLeft in 0..1
            } catch (e: Exception) {
                false
            }
        }

        val suggestedRecipes = dbHelper.getSuggestedRecipes(expiringItems)
        val recipeNames = suggestedRecipes.map { it.title } // assuming Recipe has a
'name' field
        Log.d("DEBUG", "Suggested recipes: $recipeNames")
        if (suggestedRecipes.isNotEmpty()) {
            val intent = Intent(this, SuggestionsActivity::class.java)
            intent.putStringArrayListExtra("recipes", ArrayList(recipeNames))
            startActivity(intent)

        } else {
            Toast.makeText(this, "No recipe suggestions available",
Toast.LENGTH_SHORT).show()
        }
    }


}

override fun onResume() {
```

```
    super.onResume()
    checkExpiringItems()
}

private fun showContactDialog() {
    val view = layoutInflater.inflate(R.layout.dialog_user_contact, null)
    val editPhone = view.findViewById<EditText>(R.id.editPhoneNumber)
    val editEmail = view.findViewById<EditText>(R.id.editEmail)

    AlertDialog.Builder(this)
        .setTitle("Enter Contact Info")
        .setView(view)
        .setPositiveButton("Save") { _, _ ->
            userPhone = editPhone.text.toString()
            userEmail = editEmail.text.toString()
            Toast.makeText(this, "Contact info saved", Toast.LENGTH_SHORT).show()
        }
        .setNegativeButton("Cancel", null)
        .show()
}

private fun showEditDialog(position: Int, item: FridgeItem) {
    val dialogView = layoutInflater.inflate(R.layout.dialog_add_item, null)
    val editName = dialogView.findViewById<EditText>(R.id.editItemName)
    val editQuantity = dialogView.findViewById<EditText>(R.id.editItemQuantity)
    val editExpiry = dialogView.findViewById<EditText>(R.id.editItemExpiry)
    val btnAdd = dialogView.findViewById<Button>(R.id.btnAddItem)
    btnAdd.text = "Update"

    editName.setText(item.name)
    editQuantity.setText(item.quantity)
```

```
    editExpiry.setText(item.expiryDate)

    val dialog = AlertDialog.Builder(this).setView(dialogView).create()

    btnAdd.setOnClickListener {
        val name = editName.text.toString()
        val quantity = editQuantity.text.toString()
        val expiry = editExpiry.text.toString()

        if (name.isNotEmpty() && quantity.isNotEmpty() && expiry.isNotEmpty()) {
            val success = dbHelper.updateItem(item.id, name, quantity, expiry)
            if (success) {
                itemList[position] = FridgeItem(item.id, name, quantity, expiry)
                fridgeAdapter.notifyItemChanged(position)
                Toast.makeText(this, "Item updated", Toast.LENGTH_SHORT).show()
                dialog.dismiss()
            } else {
                Toast.makeText(this, "Update failed", Toast.LENGTH_SHORT).show()
            }
        }
    }
    dialog.show()
}

private fun checkExpiringItems() {
    val sdf = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())
    val today = Calendar.getInstance().apply {
        set(Calendar.HOUR_OF_DAY, 0)
        set(Calendar.MINUTE, 0)
        set(Calendar.SECOND, 0)
        set(Calendar.MILLISECOND, 0)
```

```
}

val expiringTomorrow = itemList.filter {
    try {
        val expiryDate = sdf.parse(it.expiryDate)
        val expiryCal = Calendar.getInstance().apply {
            time = expiryDate!!
            set(Calendar.HOUR_OF_DAY, 0)
            set(Calendar.MINUTE, 0)
            set(Calendar.SECOND, 0)
            set(Calendar.MILLISECOND, 0)
        }
        val diff = expiryCal.timeInMillis - today.timeInMillis
        val daysLeft = diff / (1000 * 60 * 60 * 24)
        daysLeft in 0..1
    } catch (e: Exception) {
        false
    }
}

if (expiringTomorrow.isNotEmpty()) {
    val message = expiringTomorrow.joinToString("\n") {
        "${it.name} expires on ${it.expiryDate}"
    }

    AlertDialog.Builder(this)
        .setTitle("Item Expiring Tomorrow!")
        .setMessage(message)
        .setPositiveButton("OK", null)
        .show()
```

```
            val phone = if (userPhone.isNotEmpty()) userPhone else "5554"
            sendSMS(phone, message)

            if (userEmail.isNotEmpty()) sendEmail(userEmail, "Fridge Item Expiry", message)
        }
    }

    private fun sendSMS(phoneNumber: String, message: String) {
        try {
            val smsManager = SmsManager.getDefault()
            smsManager.sendTextMessage(phoneNumber, null, message, null, null)
            Toast.makeText(this, "SMS sent to $phoneNumber",
Toast.LENGTH_SHORT).show()
        } catch (e: Exception) {
            Toast.makeText(this, "Failed to send SMS: ${e.message}",
Toast.LENGTH_SHORT).show()
        }
    }

    private fun sendEmail(email: String, subject: String, body: String) {
        val intent = Intent(Intent.ACTION_SEND).apply {
            type = "message/rfc822"
            putExtra(Intent.EXTRA_EMAIL, arrayOf(email))
            putExtra(Intent.EXTRA_SUBJECT, subject)
            putExtra(Intent.EXTRA_TEXT, body)
        }

        try {
            startActivity(Intent.createChooser(intent, "Send Email"))
        } catch (e: Exception) {
            Toast.makeText(this, "No email clients installed.",
```

```
Toast.LENGTH_SHORT).show()
        }
    }
    private fun showSuggestionsDialog(suggestedRecipes: List<Recipe>) {
        val recipeTitles = suggestedRecipes.map { it.title }.toTypedArray()

        AlertDialog.Builder(this)
            .setTitle("Suggested Recipes")
            .setItems(recipeTitles) { _, which ->
                val selectedRecipe = suggestedRecipes[which]
                showRecipeDetailsDialog(selectedRecipe)
            }
            .setNegativeButton("Close", null)
            .show()
    }

    private fun showRecipeDetailsDialog(recipe: Recipe) {
        AlertDialog.Builder(this)
            .setTitle(recipe.title)
            .setMessage("Ingredients:\n${recipe.ingredients}\n\nSteps:\n${recipe.steps}")
            .setPositiveButton("OK", null)
            .show()
    }


    private fun scheduleDailyExpiryCheck() {
        val alarmManager = getSystemService(Context.ALARM_SERVICE) as
AlarmManager
        val intent = Intent(this, ExpiryAlarmReceiver::class.java)
        val pendingIntent = PendingIntent.getBroadcast(this, 0, intent,
PendingIntent.FLAG_IMMUTABLE)
```

```
        val calendar = Calendar.getInstance().apply {
            set(Calendar.HOUR_OF_DAY, 8)
            set(Calendar.MINUTE, 0)
            set(Calendar.SECOND, 0)
        }

        alarmManager.setInexactRepeating(
            AlarmManager.RTC_WAKEUP,
            calendar.timeInMillis,
            AlarmManager.INTERVAL_DAY,
            pendingIntent
        )
    }

    private fun showSuggestedRecipes() {
        val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, allRecipes)
        listViewRecipes.adapter = adapter
    }
```

7.FridgeAdapter.kt

```
package com.example.fridgemanager1

import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import android.view.LayoutInflater


class FridgeAdapter(private val itemList: MutableList<FridgeItem>,
```

```
            private val onItemClick: (Int) -> Unit):  // For updating) :
    RecyclerView.Adapter<FridgeAdapter.FridgeViewHolder>() {

    class FridgeViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val itemName: TextView = itemView.findViewById(R.id.itemNameText)
        val itemQuantity:TextView=itemView.findViewById(R.id.itemQuantityText)
        val itemExpiry: TextView = itemView.findViewById(R.id.itemDateText)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
FridgeViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_fridge, parent,
false)
        return FridgeViewHolder(view)
    }

    override fun onBindViewHolder(holder: FridgeViewHolder, position: Int) {
        val item = itemList[position]
        holder.itemName.text = item.name
        holder.itemQuantity.text=item.quantity
        holder.itemExpiry.text = item.expiryDate
        holder.itemView.setOnClickListener {
            onItemClick(position)
        }
        holder.itemView.setOnClickListener {
            onItemClick(holder.adapterPosition)
        }

    }

    override fun getItemCount(): Int = itemList.size
```

}

8.Fridge Databasehelper.kt

```
package com.example.fridgemanager1

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

class FridgeDatabaseHelper(context: Context) : SQLiteOpenHelper(context, "FridgeDB",
null, 1) {

   override fun onCreate(db: SQLiteDatabase) {
         db.execSQL(
            "CREATE TABLE FoodItems(id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT, quantity TEXT, expiry TEXT)"
         )
         db.execSQL(
            "CREATE TABLE Recipes(id INTEGER PRIMARY KEY
AUTOINCREMENT, title TEXT, ingredients TEXT, steps TEXT)"
         )
      }


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

```
fun insertItem(item: FridgeItem): Long {
    val db = writableDatabase
    val values = ContentValues().apply {
        put("name", item.name)
        put("quantity", item.quantity)
        put("expiry", item.expiryDate)
    }

    return db.insert("FoodItems", null, values)
}


fun getAllItems(): MutableList<FridgeItem> {
    val itemList = mutableListOf<FridgeItem>()
    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM FoodItems", null)
    while (cursor.moveToNext()) {
        val name = cursor.getString(cursor.getColumnIndexOrThrow("name"))
        val quantity = cursor.getString(cursor.getColumnIndexOrThrow("quantity"))
        val expiry = cursor.getString(cursor.getColumnIndexOrThrow("expiry"))
        val id = cursor.getInt(cursor.getColumnIndexOrThrow("id"))
        itemList.add(FridgeItem(id, name, quantity, expiry))

    }
    cursor.close()
    return itemList
}

fun updateItem(id: Int, name: String, quantity: String, expiry: String): Boolean {
    val db = this.writableDatabase
    val contentValues = ContentValues().apply {
```

```
        put("name", name)
        put("quantity", quantity)
        put("expiry", expiry)
    }
    return db.update("FoodItems", contentValues, "id=?", arrayOf(id.toString())) > 0
}

fun deleteItem(id: Int): Boolean {
    val db = this.writableDatabase
    return db.delete("FoodItems", "id=?", arrayOf(id.toString())) > 0
}
fun getAllRecipes(): List<Recipe> {
    val recipeList = mutableListOf<Recipe>()
    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM Recipes", null)
    while (cursor.moveToNext()) {
        val id = cursor.getInt(cursor.getColumnIndexOrThrow("id"))
        val title = cursor.getString(cursor.getColumnIndexOrThrow("title"))
        val ingredients = cursor.getString(cursor.getColumnIndexOrThrow("ingredients"))
        val steps = cursor.getString(cursor.getColumnIndexOrThrow("steps"))

        recipeList.add(Recipe(id, title, ingredients, steps))
    }
    cursor.close()
    return recipeList
}
fun getSuggestedRecipes(fridgeItems: List<FridgeItem>): List<Recipe> {
    val allRecipes = getAllRecipes()
    val suggestions = mutableListOf<Recipe>()

    val fridgeIngredients = fridgeItems.map { it.name.lowercase().trim() }
```

```
        for (recipe in allRecipes) {
            val recipeIngredients = recipe.ingredients.lowercase().split(",").map { it.trim() }
            if (recipeIngredients.any { it in fridgeIngredients }) {
                suggestions.add(recipe)
            }
        }

        return suggestions
    }


}
```

9.REceipeAdapter.kt

```
package com.example.fridgemanager1

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class RecipeAdapter(private val recipes: List<Recipe>) :
    RecyclerView.Adapter<RecipeAdapter.RecipeViewHolder>() {

    class RecipeViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val titleText: TextView = itemView.findViewById(R.id.recipeTitle)
        val ingredientsText: TextView = itemView.findViewById(R.id.recipeIngredients)
    }
```

```
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
RecipeViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_recipe, parent, false)
        return RecipeViewHolder(view)
    }

    override fun onBindViewHolder(holder: RecipeViewHolder, position: Int) {
        val recipe = recipes[position]
        holder.titleText.text = recipe.title
        holder.ingredientsText.text = "Ingredients: ${recipe.ingredients}"

    }

    override fun getItemCount(): Int = recipes.size
}
```

10.SuggestionActivity.kt

```
package com.example.fridgemanager1

import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class SuggestionsActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_suggestions)
```

```
        val recipeList = intent.getStringArrayListExtra("recipes")
        Log.d("DEBUG", "Received recipes: $recipeList")

        val textView = findViewById<TextView>(R.id.tvSuggestion)

        if (textView != null) {
            textView.text = recipeList?.joinToString("\n") ?: "No suggestions available"
        } else {
            // fallback if the view was not found (to avoid crash)
            println("tvSuggestion TextView not found in layout!")
        }
    }
}
```

5.ADVANTAGES

- Simple and intuitive user interface.
- Helps reduce food wastage.
- Smart reminders via SMS, email, and in-app notifications.
- Recipe suggestions improve meal planning.
- Fully offline supported with SQLite.

LIMITATIONS

- Only works on Android devices.
- Email functionality depends on available email clients.
- SMS may incur carrier charges.
- No cloud synchronization or multi-device support.

FUTURE ENHANCEMENTS

- Cloud backup and sync across multiple devices.

- Barcode scanning for easy item entry.
- AI-based recipe recommendations using more complex logic.
- Voice assistant support for hands-free use.
- Integration with smart fridge devices.

6.RESULT

The application was successfully developed, tested, and executed on Android Studio using Kotlin and XML layouts. It met all defined objectives: storing items, notifying expiry, and recipe suggestions. User information was collected to facilitate SMS/email reminders. The app worked reliably across emulator and real devices.

7.CONCLUSION

The Fridge Manager Android app offers an effective and convenient way to manage food inventory. It minimizes food wastage by notifying users in time and enhances user experience through recipe suggestions. The application successfully integrates various Android components such as SQLite, SMS, and notification services into a cohesive and useful product. It provides a scalable foundation for future enhancements and real-world deployment.