IDIARY : PERSONAL DIARY APP

## A MINI PROJECT REPORT

***Submitted by***

**SEENUVASAN S (2116220701255)**

***in partial fulfilment for the course***

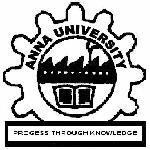
## CS19611 – MOBILE APPLICATION DEVELOPMENT LABORATORY

***of the degree of***

**BACHELOR OF ENGINEERING**

## in

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAJALAKSHMI ENGINEERING COLLEGE RAJALAKSHMI NAGAR**

**THANDALAM CHENNAI – 602 105**

## MAY 2025

**BONAFIDE CERTIFICATE**

Certified that this project report **“IDIARY: PERSONAL DIARY APP”** is the bonafide work of **“SEENUVASAN S (2116220701255)”** who carried out the project work (CS19611-Mobile Application Development Laboratory) under my supervision.

Dr. P.Kumar

## HEAD OF THE DEPARTMENT

Professor and Head Department of

Computer Science and Engineering Rajalakshmi Engineering College Rajalakshmi Nagar

Thandalam

Chennai - 602105

Mr. Saravana Gokul G, M.E(cse)

## SUPERVISOR

Assistant Professor (SG) Department of

Computer Science and Engineering Rajalakshmi Engineering College Rajalakshmi Nagar

Thandalam

Chennai - 602105

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TOPIC** | **PAGE NO.** |
|  | **ACKNOWLEDGEMENT** | **iii** |
|  | **ABSTRACT** | **iv** |
|  | **LIST OF FIGURES** | **v** |
|  | **LIST OF ABBREVIATIONS** | **vi** |
| **1** | **INTRODUCTION** | **7** |
|  | 1.1 GENERAL | 7 |
|  | 1.2 OBJECTIVE | 7 |
|  | 1.3 EXISTING SYSTEM | 8 |
|  | 1.4 PROPOSED SYSTEM | 8 |
| **2** | **LITERATURE SURVEY** | **10** |
| **3** | **SYSTEM DESIGN** | **12** |
|  | 3.1 GENERAL | 12 |
|  | 3.1.1 SYSTEM FLOW DIAGRAM | 12 |
|  | 3.1.2 ARCHITECTURE DIAGRAM | 14 |
|  | 3.1.3 USE CASE DIAGRAM | 14 |
| **4** | **PROJECT DESCRIPTION** | **15** |
|  | 4.1 METHODOLOGIES | 15 |
|  | 4.1.1 MODULES | 15 |
|  | 4.1.1.1 PERSONAL DIARY MODULE | 16 |
|  | 4.1.1.2 CONTACT MANAGER MODULE | 16 |
|  | 4.1.1.3 SEND SMS MODULE | 16 |

* + - 1. SEND EMAIL

16

MODULE

* + - 1. [SPEECH TO TEXT MODULE 1](#_TOC_250003)7
      2. [SQLITE MODULE 1](#_TOC_250002)7
  1. [PROJECT OUTPUT SCREENS 1](#_TOC_250001)8

5 CONCLUSION 19

APPENDIX 21

[REFFERENCES 3](#_TOC_250000)6

## ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S.Meganathan, B.E, F.I.E.,** our Vice Chairman **Mr. Abhay Meganathan, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) Thangam Meganathan, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring 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, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering, Rajalakshmi Engineering College for their valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Mr. B.Bhuvaneswaran, M.E.,** Assistant Professor (SG), Department of Computer Science and Engineering for his useful tips during our review to build our project.

## SEENUVASAN S (2116220701255)

**ABSTRACT**

"IDIARY" is an intuitive and user-friendly Android application designed to help users organize and manage their personal diaries and contacts effectively. Developed using Kotlin and following Android's best practices, the app consists of two primary modules: a Personal Diary module and a Contact Manager module.

The **Personal Diary** module allows users to create, update, search, and delete diary entries with a title, description, and date. The diary entries are stored locally in an SQLite database, ensuring that users’ personal data is securely saved and can be accessed even offline. The search functionality enables users to quickly find specific diary entries based on keywords, enhancing the overall user experience.

The **Contact Manager** module enables users to store and manage contact information, simplifying the process of adding, editing, and removing contacts. This module supports features such as displaying contact details and keeping everything organized for easy access.

The app supports voice dictation, allowing users to dictate their diary entries or search queries using speech recognition, adding an extra layer of convenience. By using Android's SpeechRecognizer API, the app integrates speech-to-text functionality for hands-free operation.

Through this project, the app aims to provide a holistic solution for managing personal diaries and contacts with a simple, clean, and intuitive user interface, making it an essential tool for everyday personal organization.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE NAME** | **PAGE NO.** |
| **3.1** | **SYSTEM FLOW DIAGRAM** | **13** |
| **3.2** | **ARCHITECTURE DIAGRAM** | **14** |
| **3.3** | **USE CASE DIAGRAM** | **14** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| **S NO.** | **ABBREVIATION** | **ACCRONYM** |
| 1 | DB | Database |
| 2 | API | Application Programming Interface |
| 3 | SMS | Short Message Service |
| 4 | CRUD | Create Read Update Delete |
| 5 | UI | User Interface |
| 6 | OS | Operating System |

# CHAPTER 1 INTRODUCTION

**1.1 GENERAL**In today's fast-paced world, managing personal information efficiently is essential for maintaining organization and productivity. Mobile applications provide a convenient and effective way to store, organize, and retrieve important data at the user's fingertips. Personal diaries and contact management are two core aspects of an individual’s daily life that benefit from a well-structured digital solution.

The "IDIARY" Android application is designed to address the need for easy, organized personal management. Developed using Kotlin, this application features two primary modules: a Personal Diary and a Contact Manager. The Personal Diary module allows users to create, update, and store diary entries, while the Contact Manager provides a simple interface for organizing contact information. This app ensures that all data is stored locally, allowing users to access their information even without an internet connection.

Through this project, "IDIARY" integrates several Android features such as SQLite database management for data persistence, voice recognition for hands-free diary entry, and intuitive user interfaces to enhance usability. The app emphasizes simplicity, privacy, and easy navigation, making it an ideal tool for anyone seeking an efficient and secure way to manage their personal notes and contacts.

**1.2 OBJECTIVE**  
The objective of the "IDIARY" app is to provide users with an organized and efficient way to manage their personal diaries and contacts.

The specific objectives include:

* Enabling users to store personal diary entries with a title, description, and date.
* Allowing users to update, delete, and search diary entries quickly.
* Managing contacts with essential details such as name, phone number, and email address.
* Integrating speech-to-text functionality to create and search diary entries hands-free.
* Storing data securely using SQLite for offline access.
* Offering a simple and intuitive user interface for easy navigation.

This app is designed to serve as an all-in-one personal management tool for individuals who wish to keep track of their thoughts and contacts in a convenient, user-friendly format.

**1.3 EXISTING SYSTEM**  
Current diary and contact management applications often have limitations such as:

* Dependence on cloud-based storage, which may not guarantee offline functionality.
* Complex or overloaded interfaces, making it difficult for users to manage their data quickly.
* Limited customization and personalization features.
* Some apps offer one-dimensional functionality (either only diary management or contact management).

Examples of existing apps include:

* Google Keep (note-taking with limited diary features).
* Evernote (note-taking, but lacks in-depth contact management).
* Contacts App (contact management only).

These apps do not provide an integrated solution for both personal diary entries and contact management.

**1.4 PROPOSED SYSTEM**  
The proposed "IDIARY" system addresses the limitations of existing solutions by:

* Integrating both Personal Diary and Contact Manager modules into a single app.
* Allowing users to create, update, and search diary entries with ease.
* Enabling contact management with essential details and data.
* Providing offline functionality for diary entries and contacts, ensuring users can access their information without an internet connection.
* Integrating voice dictation functionality to add or search diary entries hands-free.
* Storing all data locally using SQLite database for enhanced privacy and security.

By offering both diary and contact management in one application, "IDIARY" aims to provide a seamless, efficient, and secure way for users to stay organized and manage their personal information on-the-go.

# CHAPTER 2 LITERATURE SURVEY

**2.1 GENERAL**

In today’s fast-paced digital world, individuals are increasingly looking for efficient and secure ways to organize their personal information. Managing diaries, contacts, and daily tasks is a challenge that many face, particularly when balancing multiple responsibilities. Mobile applications have risen as an ideal solution for managing personal data due to the convenience of being accessible at any time, in any place.

The "IDIARY" app aims to bridge this gap by providing users with a comprehensive platform for organizing personal diaries and contacts in one easy-to-use, offline-first solution. Mobile applications like Google Keep and Evernote have provided note-taking functionalities, but they lack the deep integration between diary management and contact organization that "IDIARY" offers. Furthermore, these apps often rely heavily on internet access and cloud storage, which can be limiting for users in regions with inconsistent connectivity.

Additionally, many users struggle with apps that have overwhelming and cluttered user interfaces, hindering quick access to vital information when needed. Several research studies, such as the one in the "Journal of Information Technology & Software Engineering," emphasize the importance of creating simple yet functional user interfaces for personal management tools. The study also highlights the necessity for apps to work offline to ensure reliability in areas where connectivity is sparse.

While existing applications have focused on single functionalities—be it for note-taking or contact management—none have seamlessly integrated these two aspects in a streamlined and user-friendly manner. "IDIARY" addresses this gap by providing both functionalities in one unified app, emphasizing simplicity, data security, and offline access. With features like voice-to-text for easy diary entry creation, and local storage for contacts and diary data, "IDIARY" is designed to ensure privacy, speed, and ease of access even without an internet connection.

Building on the foundation laid by previous personal management apps, **"IDIARY"** aims to provide a more holistic solution for managing daily activities and contacts, creating an app that adapts to the fast-paced nature of modern life while prioritizing user experience and privacy.

# CHAPTER 3 SYSTEM DESIGN

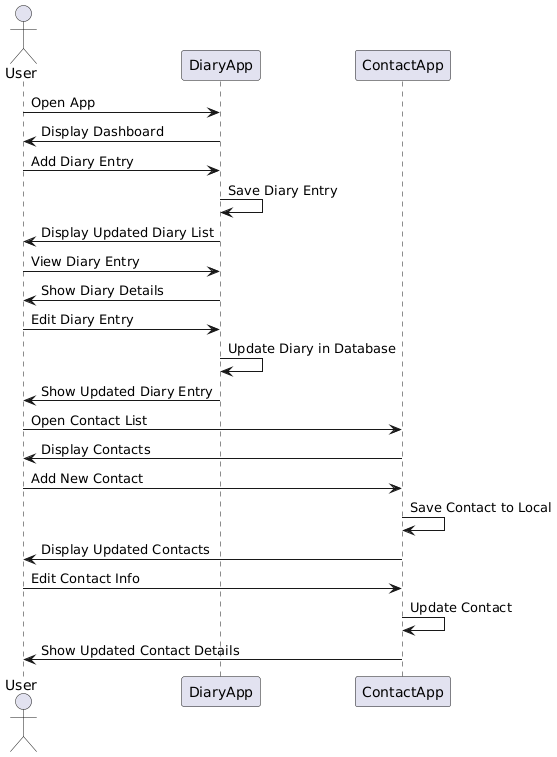
* 1. **GENERAL**

**System design** is a fundamental phase in the mobile application development lifecycle. It defines the overall architecture, component interaction, data flow, and user interactions. For **IDIARY – A Personal Diary App**, the design focuses on ensuring smooth data management, a seamless user experience, and easy access to key features like diary entries and contacts.

This chapter outlines the design through three perspectives: the System Flow Diagram, the Architecture Diagram, and the Use Case Diagram. These visuals represent the app’s behavior from both technical and user-centric points of view.

### 3.1.1 **SYSTEM FLOW DIAGRAM**

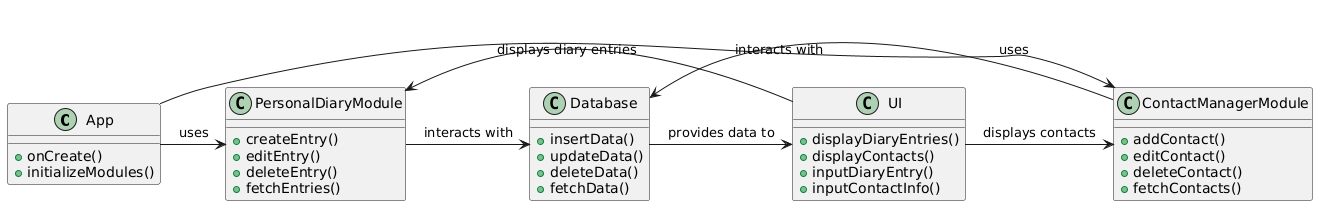
The **System Flow Diagram** outlines the sequence of actions triggered when a user interacts with the app, such as adding, viewing, and searching for diary entries. It ensures that each module—entry creation, search, list view, and contact management—is invoked in the correct order with minimal user input. The system is designed to handle operations efficiently, ensuring the app performs actions like saving entries, retrieving data, and displaying them to the user with minimal delay.



**Fig 3.1 System Flow Diagram**

# ARCHITECTURE DIAGRAM

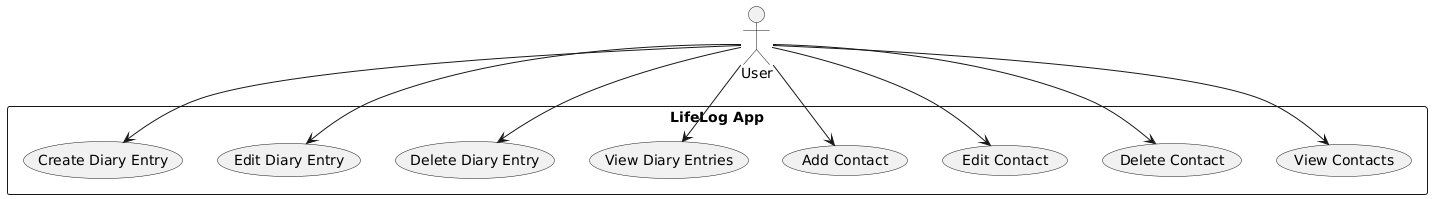
This diagram describes the core components of the application and how they interact with the device hardware and Android OS services.



**Fig. 3.2 Architecture Diagram**

# USE CASE DIAGRAM

The Use Case Diagram describes how the user interacts with the application and highlights the various actions that can be performed.



**Fig. 3.3 Use Case Diagram**

# CHAPTER 4 PROJECT DESCRIPTION

* 1. **METHODOLOGIES**

The **IDIARY** app follows a modular development approach using **Kotlin** on the **Android platform**, separating concerns across well-defined components to ensure clarity, maintainability, and efficiency. Each feature (Diary, Contact, Communication, and Input Handling) is encapsulated as a standalone module, promoting better testing, flexibility, and scalability. The app is designed to run efficiently on a wide range of Android devices.

The methodology adopted includes:

* **User-Centric Design**: Interfaces are designed to be intuitive and minimal, tailored for personal productivity with easy access to key actions.
* **Modular Architecture**: Separation of diary, contact, communication (SMS/Email), input (Speech-to-Text), and storage modules for clean codebase management.
* **Local Storage with Room/SQLite**: Data is stored securely and offline using Room over SQLite, with DAO interfaces abstracting raw SQL logic.
* **Service Integration**: Uses system APIs such as SMSManager, Email Intent, and SpeechRecognizer to interact with device features.
* **MVVM Pattern**: Lifecycle-aware architecture using ViewModel, LiveData, and Repositories to separate business logic from UI.
* **Security & Privacy**: All data is stored in internal storage with no external access. Runtime permissions are handled gracefully.

**4.1.1 MODULES**

The IDIARY app consists of the following **core modules**, each responsible for a specific area of functionality:

* **Personal Diary Module**
* **Contact Manager Module**
* **Send SMS Module**
* **Send Email Module**
* **Speech-to-Text Module**
* **SQLite/Room Storage Module**

**4.1.1.1 PERSONAL DIARY MODULE**

* **Create Entry**: Users can add diary entries with title, description, and date.
* **Edit Entry**: Entries can be modified post creation.
* **Delete Entry**: Unwanted entries can be deleted permanently.
* **View Entries**: All entries displayed chronologically using RecyclerView.
* **Offline Storage**: Data stored locally using Room backed by SQLite.

**4.1.1.2 CONTACT MANAGER MODULE**

* **Add Contact**: Name, phone number, and email are entered.
* **Edit Contact**: Users can modify saved contact details.
* **Delete Contact**: Contacts can be deleted from the app database.
* **List & Search**: Contacts are shown in a list and searchable by name.

**4.1.1.3 SEND SMS MODULE**

* **SMS Trigger**: Allows sending a quick message to saved contacts.
* **SMSManager Integration**: Uses Android's built-in SMSManager class.
* **Use Case**: Diary or contact notes can be shared via SMS in emergencies or reminders.

**4.1.1.4 SEND EMAIL MODULE**

* **Email Intent**: Opens default email client with pre-filled content.
* **Attachment Support**: Can optionally attach diary content or export file.
* **Use Case**: Useful for backing up entries or sharing contact summaries.

**4.1.1.5 SPEECH-TO-TEXT MODULE**

* **Voice Input**: Converts spoken words to text using Android SpeechRecognizer API.
* **Input Integration**: Works with diary entry creation or contact notes.
* **Fallback Text**: If speech fails, a fallback dialog is shown for manual input.

**4.1.1.6 SQLITE/ROOM STORAGE MODULE**

* **Database Layer**: Core module that abstracts SQLite via Room.
* **Entities**: DiaryEntry and ContactEntity classes define table schemas.
* **DAOs**: Separate interfaces for diary and contact operations.
* **Threading**: Uses Kotlin Coroutines for async DB access.
* **LiveData Support**: Observes real-time updates in the UI.

# PROJECT OUTPUT SCREENS

# CHAPTER 5 CONCLUSION

* 1. **GENERAL**

The **IDIARY** app has been developed with the primary goal of enhancing personal organization and productivity by providing users with a seamless platform to manage diary entries and contact information. With a focus on simplicity, offline accessibility, and user-centric design, the app enables individuals to log daily reflections and maintain important contact details effortlessly.

**Key Achievements:**

1. **Efficient Personal Logging**: The Personal Diary module allows users to create, edit, and manage daily journal entries, helping them document important moments, thoughts, or tasks in a structured manner.
2. **Integrated Contact Management**: Users can easily add, update, and store personal contact information locally, making IDIARY a compact and handy contact book.
3. **Voice-Assisted Input**: The Speech-to-Text feature ensures quick and hands-free entry of diary notes or contact details, enhancing accessibility and user convenience.
4. **Offline-First Architecture**: With data stored locally using SQLite and Room, users can continue using the app even without internet access, ensuring privacy and reliability.
5. **Clean and Intuitive UI**: The app follows a simple and responsive user interface design, minimizing friction and improving overall user experience, even for non-technical users.

The **IDIARY** app serves as a lightweight, secure, and efficient personal companion, making it easier for users to reflect on their daily lives and stay organized. As mobile technologies continue to evolve, IDIARY can be further enhanced with cloud sync, notifications, analytics, and cross-device support, offering even greater value to its users.

# APPENDIX SOURCE CODE

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.madproject">  
  
 <!-- Permissions should be declared outside the <application> tag -->  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.SEND\_SMS" />  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
 <uses-permission android:name="android.permission.READ\_CONTACTS" />  
 <uses-permission android:name="android.permission.WRITE\_CONTACTS" />  
 <uses-permission android:name="android.permission.RECEIVE\_SMS" />  
 <uses-permission android:name="android.permission.READ\_SMS" />  
 <uses-permission android:name="android.permission.VIBRATE" />  
 <uses-permission android:name="android.permission.GET\_ACCOUNTS" />  
 <uses-permission android:name="android.permission.SEND\_SMS" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.MADProject">  
  
 <!-- Main Activity -->  
 <activity android:name=".MainActivity"  
 android:label="Main Activity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
  
 <!-- DiaryListActivity -->  
 <activity android:name=".DiaryListActivity"  
 android:label="Diary List"  
 android:exported="true">  
 <!-- You can add an intent filter for specific actions here if needed -->  
 </activity>  
  
 <!-- AddEditDiaryActivity -->  
 <activity android:name=".AddEditDiaryActivity"  
 android:label="Add/Edit Diary"  
 android:exported="true">  
 </activity>  
 </application>  
</manifest>

**MainActivity.kt**

package com.example.madproject  
  
import android.content.Intent  
import android.os.Bundle  
import android.widget.Button  
import androidx.appcompat.app.AppCompatActivity  
import com.example.madproject.DiaryListActivity  
import com.example.madproject.ContactManagerActivity  
  
class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 val diaryButton = findViewById<Button>(R.id.btnDiary)  
 val contactButton = findViewById<Button>(R.id.btnContacts)  
  
 diaryButton.setOnClickListener {  
 startActivity(Intent(this, DiaryListActivity::class.java))  
 }  
  
 contactButton.setOnClickListener {  
 startActivity(Intent(this, ContactManagerActivity::class.java))  
 }  
 }  
}

**DiaryListActivity.kt**

package com.example.madproject  
  
import android.content.ContentValues  
import android.database.Cursor  
import android.os.Bundle  
import android.speech.RecognizerIntent  
import android.speech.SpeechRecognizer  
import android.speech.RecognitionListener  
import android.widget.\*  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
import android.text.Editable  
import android.text.TextWatcher  
import java.util.\*  
import android.content.Intent  
  
  
class DiaryListActivity : AppCompatActivity() {  
  
 private lateinit var dbHelper: DiaryDatabaseHelper  
 private lateinit var diaryListView: ListView  
 private lateinit var searchEditText: EditText  
 private lateinit var btnDictate: Button  
 private var diaryList = mutableListOf<DiaryEntry>()  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_diary\_list)  
  
 dbHelper = DiaryDatabaseHelper(this)  
 diaryListView = findViewById(R.id.diaryListView)  
 searchEditText = findViewById(R.id.searchEditText)  
 btnDictate = findViewById(R.id.btnDictate)  
  
 // Load and display diary entries  
 loadDiaryEntries()  
  
 // Search functionality  
 searchEditText.addTextChangedListener(object : TextWatcher {  
 override fun afterTextChanged(s: Editable?) {  
 // Called after text is changed  
 }  
  
 override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {  
 // Called before text is changed  
 }  
  
 override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {  
 val keyword = s.toString()  
 searchDiaryEntries(keyword)  
 }  
 })  
  
 // Dictate Diary Entry functionality  
 btnDictate.setOnClickListener {  
 startSpeechToText()  
 }  
  
 // Handle item click to view diary entry details  
 diaryListView.setOnItemClickListener { \_, \_, position, \_ ->  
 val selectedDiary = diaryList[position]  
 showDiaryDetails(selectedDiary)  
 }  
  
 // Handle long press to delete diary entry  
 diaryListView.setOnItemLongClickListener { \_, \_, position, \_ ->  
 val selectedDiary = diaryList[position]  
 confirmDeleteDiaryEntry(selectedDiary)  
 true  
 }  
 }  
  
 private fun loadDiaryEntries() {  
 val db = dbHelper.readableDatabase  
 val cursor: Cursor = db.query(  
 DiaryDatabaseHelper.TABLE\_NAME, // Use the correct table name  
 null,  
 null,  
 null,  
 null,  
 null,  
 null  
 )  
 diaryList.clear()  
 while (cursor.moveToNext()) {  
 val id = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_ID)) // Fetch as String  
 val title = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_TITLE))  
 val description = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_DESCRIPTION))  
 val date = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_DATE))  
 diaryList.add(DiaryEntry(id, title, description, date)) // Pass id as String  
 }  
 cursor.close()  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, diaryList)  
 diaryListView.adapter = adapter  
 }  
  
 private fun searchDiaryEntries(keyword: String) {  
 val db = dbHelper.readableDatabase  
 val cursor: Cursor = db.query(  
 DiaryDatabaseHelper.TABLE\_NAME, // Reference to the correct table name  
 null,  
 "${DiaryDatabaseHelper.COLUMN\_TITLE} LIKE ? OR ${DiaryDatabaseHelper.COLUMN\_DESCRIPTION} LIKE ?",  
 arrayOf("%$keyword%", "%$keyword%"),  
 null,  
 null,  
 null  
 )  
 diaryList.clear()  
  
 while (cursor.moveToNext()) {  
 val id = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_ID)) // Fetch the ID  
 val title = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_TITLE))  
 val description = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_DESCRIPTION))  
 val date = cursor.getString(cursor.getColumnIndex(DiaryDatabaseHelper.COLUMN\_DATE))  
  
 // Add the DiaryEntry to the list, including 'id'  
 diaryList.add(DiaryEntry(id, title, description, date))  
 }  
 cursor.close()  
  
 // Update the ListView with the fetched diary entries  
 val adapter = ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, diaryList)  
 diaryListView.adapter = adapter  
 }  
  
  
 private fun showDiaryDetails(diary: DiaryEntry) {  
 val dialogView = layoutInflater.inflate(R.layout.dialog\_diary\_details, null)  
 val titleEditText = dialogView.findViewById<EditText>(R.id.titleEditText)  
 val descriptionEditText = dialogView.findViewById<EditText>(R.id.descriptionEditText)  
 val saveButton = dialogView.findViewById<Button>(R.id.saveButton)  
  
 titleEditText.setText(diary.title)  
 descriptionEditText.setText(diary.description)  
  
 val alertDialog = AlertDialog.Builder(this)  
 .setView(dialogView)  
 .setTitle("View Diary Entry")  
 .setPositiveButton("Save") { \_, \_ ->  
 updateDiaryEntry(diary, titleEditText.text.toString(), descriptionEditText.text.toString())  
 }  
 .setNegativeButton("Cancel", null)  
 .create()  
  
 alertDialog.show()  
 }  
  
 private fun updateDiaryEntry(diary: DiaryEntry, title: String, description: String) {  
 val db = dbHelper.writableDatabase  
 val contentValues = ContentValues().apply {  
 put(DiaryDatabaseHelper.COLUMN\_TITLE, title)  
 put(DiaryDatabaseHelper.COLUMN\_DESCRIPTION, description)  
 }  
 db.update(  
 DiaryDatabaseHelper.TABLE\_NAME, // Reference to the correct table name  
 contentValues,  
 "${DiaryDatabaseHelper.COLUMN\_TITLE} = ?",  
 arrayOf(diary.title)  
 )  
 loadDiaryEntries()  
 }  
  
 private fun confirmDeleteDiaryEntry(diary: DiaryEntry) {  
 AlertDialog.Builder(this)  
 .setMessage("Are you sure you want to delete this entry?")  
 .setPositiveButton("Yes") { \_, \_ -> deleteDiaryEntry(diary) }  
 .setNegativeButton("No", null)  
 .create()  
 .show()  
 }  
  
 private fun deleteDiaryEntry(diary: DiaryEntry) {  
 val db = dbHelper.writableDatabase  
 db.delete(  
 DiaryDatabaseHelper.TABLE\_NAME, // Reference to the correct table name  
 "${DiaryDatabaseHelper.COLUMN\_TITLE} = ?",  
 arrayOf(diary.title)  
 )  
 loadDiaryEntries()  
 }  
  
 private fun startSpeechToText() {  
 val speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this)  
 val intent = Intent(RecognizerIntent.ACTION\_RECOGNIZE\_SPEECH).apply {  
 putExtra(RecognizerIntent.EXTRA\_LANGUAGE\_MODEL, RecognizerIntent.LANGUAGE\_MODEL\_FREE\_FORM)  
 putExtra(RecognizerIntent.EXTRA\_LANGUAGE, Locale.getDefault())  
 }  
  
 speechRecognizer.setRecognitionListener(object : RecognitionListener {  
 override fun onReadyForSpeech(params: Bundle?) {}  
 override fun onBeginningOfSpeech() {}  
 override fun onRmsChanged(rmsdB: Float) {}  
 override fun onBufferReceived(buffer: ByteArray?) {}  
 override fun onEndOfSpeech() {}  
 override fun onError(error: Int) {  
 Toast.makeText(this@DiaryListActivity, "Speech recognition failed", Toast.LENGTH\_SHORT).show()  
 }  
 override fun onResults(results: Bundle?) {  
 val data = results?.getStringArrayList(SpeechRecognizer.RESULTS\_RECOGNITION)  
 val recognizedText = data?.joinToString(" ") ?: ""  
 searchEditText.setText(recognizedText) // Use recognized text in search bar or dictation  
 }  
 override fun onPartialResults(partialResults: Bundle?) {}  
 override fun onEvent(eventType: Int, params: Bundle?) {}  
 })  
  
 speechRecognizer.startListening(intent)  
 }  
}

**DiaryList.kt**

package com.example.madproject  
  
data class DiaryEntry(  
 var id: String = "0",  
 var title: String,  
 var description: String,  
 var date: String  
)

**DiaryDatabaseHelper.kt**

package com.example.madproject  
  
import android.content.Context  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
import android.content.ContentValues  
  
class DiaryDatabaseHelper(context: Context) :  
 SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {  
  
 companion object {  
 internal const val DATABASE\_NAME = "diary.db"  
 internal const val DATABASE\_VERSION = 1  
 internal const val TABLE\_NAME = "entries" // Updated table name  
 internal const val COLUMN\_ID = "id"  
 internal const val COLUMN\_TITLE = "title"  
 internal const val COLUMN\_DESCRIPTION = "description"  
 internal const val COLUMN\_DATE = "date"  
 }  
  
 override fun onCreate(db: SQLiteDatabase?) {  
 val createTable = ("CREATE TABLE $TABLE\_NAME (" +  
 "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT," +  
 "$COLUMN\_TITLE TEXT," +  
 "$COLUMN\_DESCRIPTION TEXT," +  
 "$COLUMN\_DATE TEXT)")  
 db?.execSQL(createTable)  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {  
 db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")  
 onCreate(db)  
 }  
  
 fun insertEntry(entry: DiaryEntry): Long {  
 val db = *writableDatabase* val values = ContentValues().*apply* {  
 put(COLUMN\_TITLE, entry.title)  
 put(COLUMN\_DESCRIPTION, entry.description)  
 put(COLUMN\_DATE, entry.date)  
 }  
 return db.insert(TABLE\_NAME, null, values)  
 }  
  
 // Other methods for getAllEntries(), deleteEntry(), etc.  
}

**ContactManagerHelper.kt**

package com.example.madproject  
  
import android.content.Intent  
import android.net.Uri  
import android.os.Bundle  
import android.speech.RecognizerIntent  
import android.speech.SpeechRecognizer  
import android.speech.RecognitionListener  
import android.telephony.SmsManager  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
  
class ContactManagerActivity : AppCompatActivity() {  
  
 private lateinit var nameEditText: EditText  
 private lateinit var phoneEditText: EditText  
 private lateinit var emailEditText: EditText  
 private lateinit var btnSendSMS: Button  
 private lateinit var btnSendEmail: Button  
 private lateinit var btnDictate: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_contact\_manager*)  
  
 nameEditText = findViewById(R.id.*nameEditText*)  
 phoneEditText = findViewById(R.id.*phoneEditText*)  
 emailEditText = findViewById(R.id.*emailEditText*)  
 btnSendSMS = findViewById(R.id.*btnSendSMS*)  
 btnSendEmail = findViewById(R.id.*btnSendEmail*)  
 btnDictate = findViewById(R.id.*btnDictate*)  
  
 // Send SMS functionality  
 btnSendSMS.setOnClickListener {  
 val phoneNumber = phoneEditText.*text*.toString()  
 val message = "Hello, this is a message from IDIARY app."  
 sendSMS(phoneNumber, message)  
 }  
  
 // Send Email functionality  
 btnSendEmail.setOnClickListener {  
 val emailAddress = emailEditText.*text*.toString()  
 sendEmail(emailAddress)  
 }  
  
 // Speech to Text functionality  
 btnDictate.setOnClickListener {  
 startSpeechToText()  
 }  
 }  
  
 private fun sendSMS(phoneNumber: String, message: String) {  
 if (phoneNumber.*isNotEmpty*()) {  
 val smsManager = SmsManager.getDefault()  
 smsManager.sendTextMessage(phoneNumber, null, message, null, null)  
 Toast.makeText(this, "SMS Sent to $phoneNumber", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Please enter a valid phone number", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun sendEmail(emailAddress: String) {  
 val intent = Intent(Intent.*ACTION\_SENDTO*).*apply* {  
 *data* = Uri.parse("mailto:$emailAddress")  
 putExtra(Intent.*EXTRA\_SUBJECT*, "Message from IDIARY App")  
 putExtra(Intent.*EXTRA\_TEXT*, "Hello, this is a message from IDIARY app.")  
 }  
 startActivity(intent)  
 }  
  
 private fun startSpeechToText() {  
 val speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this)  
 val intent = Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*).*apply* {  
 putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*)  
 }  
 speechRecognizer.setRecognitionListener(object : RecognitionListener {  
 override fun onReadyForSpeech(params: Bundle?) {}  
  
 override fun onBeginningOfSpeech() {}  
  
 override fun onRmsChanged(rmsdB: Float) {}  
  
 override fun onBufferReceived(buffer: ByteArray?) {}  
  
 override fun onEndOfSpeech() {}  
  
 override fun onError(error: Int) {}  
  
 override fun onResults(results: Bundle?) {  
 val spokenText = results?.getStringArrayList(SpeechRecognizer.*RESULTS\_RECOGNITION*)?.get(0)  
 spokenText?.*let* {  
 // Set the spoken text as the contact name or other field if needed  
 nameEditText.setText(it)  
 }  
 }  
  
 override fun onPartialResults(partialResults: Bundle?) {}  
  
 override fun onEvent(eventType: Int, params: Bundle?) {}  
 })  
 speechRecognizer.startListening(intent)  
 }  
}

**DiaryAdapter.kt**

package com.example.madproject  
  
import android.content.Context  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ArrayAdapter  
import android.widget.TextView  
  
class DiaryAdapter(context: Context, private val entries: MutableList<DiaryEntry>) :  
 ArrayAdapter<DiaryEntry>(context, 0, entries) {  
  
 override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {  
 val entry = getItem(position) as DiaryEntry  
 val view = convertView ?: LayoutInflater.from(*context*).inflate(R.layout.*item\_diary\_entry*, parent, false)  
  
 val titleTextView: TextView = view.findViewById(R.id.*titleTextView*)  
 val dateTextView: TextView = view.findViewById(R.id.*dateTextView*)  
  
 titleTextView.*text* = entry.title  
 dateTextView.*text* = entry.date  
  
 return view  
 }  
  
 fun updateEntries(newEntries: List<DiaryEntry>) {  
 entries.clear()  
 entries.addAll(newEntries)  
 notifyDataSetChanged()  
 }  
}

**ContactManagerActivity.kt**

package com.example.madproject  
  
import android.content.Intent  
import android.net.Uri  
import android.os.Bundle  
import android.speech.RecognizerIntent  
import android.speech.SpeechRecognizer  
import android.speech.RecognitionListener  
import android.telephony.SmsManager  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.appcompat.app.AlertDialog  
import androidx.appcompat.app.AppCompatActivity  
  
class ContactManagerActivity : AppCompatActivity() {  
  
 private lateinit var nameEditText: EditText  
 private lateinit var phoneEditText: EditText  
 private lateinit var emailEditText: EditText  
 private lateinit var btnSendSMS: Button  
 private lateinit var btnSendEmail: Button  
 private lateinit var btnDictate: Button  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_contact\_manager*)  
  
 nameEditText = findViewById(R.id.*nameEditText*)  
 phoneEditText = findViewById(R.id.*phoneEditText*)  
 emailEditText = findViewById(R.id.*emailEditText*)  
 btnSendSMS = findViewById(R.id.*btnSendSMS*)  
 btnSendEmail = findViewById(R.id.*btnSendEmail*)  
 btnDictate = findViewById(R.id.*btnDictate*)  
  
 // Send SMS functionality  
 btnSendSMS.setOnClickListener **{** val phoneNumber = phoneEditText.*text*.toString()  
 val message = "Hello, this is a message from IDIARY app."  
 sendSMS(phoneNumber, message)  
 **}** // Send Email functionality  
 btnSendEmail.setOnClickListener **{** val emailAddress = emailEditText.*text*.toString()  
 sendEmail(emailAddress)  
 **}** // Speech to Text functionality  
 btnDictate.setOnClickListener **{** startSpeechToText()  
 **}** }  
  
 private fun sendSMS(phoneNumber: String, message: String) {  
 if (phoneNumber.*isNotEmpty*()) {  
 val smsManager = SmsManager.getDefault()  
 smsManager.sendTextMessage(phoneNumber, null, message, null, null)  
 Toast.makeText(this, "SMS Sent to $phoneNumber", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Please enter a valid phone number", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun sendEmail(emailAddress: String) {  
 val intent = Intent(Intent.*ACTION\_SENDTO*).*apply* **{** *data* = Uri.parse("mailto:$emailAddress")  
 putExtra(Intent.*EXTRA\_SUBJECT*, "Message from IDIARY App")  
 putExtra(Intent.*EXTRA\_TEXT*, "Hello, this is a message from IDIARY app.")  
 **}** startActivity(intent)  
 }  
  
 private fun startSpeechToText() {  
 val speechRecognizer = SpeechRecognizer.createSpeechRecognizer(this)  
 val intent = Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*).*apply* **{** putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*)  
 **}** speechRecognizer.setRecognitionListener(object : RecognitionListener {  
 override fun onReadyForSpeech(params: Bundle?) {}  
  
 override fun onBeginningOfSpeech() {}  
  
 override fun onRmsChanged(rmsdB: Float) {}  
  
 override fun onBufferReceived(buffer: ByteArray?) {}  
  
 override fun onEndOfSpeech() {}  
  
 override fun onError(error: Int) {}  
  
 override fun onResults(results: Bundle?) {  
 val spokenText = results?.getStringArrayList(SpeechRecognizer.*RESULTS\_RECOGNITION*)?.get(0)  
 spokenText?.*let* **{** // Set the spoken text as the contact name or other field if needed  
 nameEditText.setText(**it**)  
 **}** }  
  
 override fun onPartialResults(partialResults: Bundle?) {}  
  
 override fun onEvent(eventType: Int, params: Bundle?) {}  
 })  
 speechRecognizer.startListening(intent)  
 }  
}

# REFFERENCES

1. Android, "Android Documentation: Fused Location Provider API," *Android Developers* [Online]. Available: <https://developer.android.com/training/location>. [Accessed: May 4, 2025].
2. Android, "Android Documentation: Permissions and Location Services," *Android Developers* [Online]. Available: <https://developer.android.com/guide/topics/permissions/overview>. [Accessed: May 4, 2025].
3. L. Brown and T. H. Jackson, *Emergency Mobile App Development*, 1st ed. San Francisco, CA, USA: DevTech Publishing, 2020.
4. Google, "Android Documentation: Speech-to-Text API," *Android Developers*

[Online]. Available:

<https://developer.android.com/reference/android/speech/SpeechRecognizer>. [Accessed: May 4, 2025].

1. Google, "Google Cloud Speech-to-Text," *Google Cloud* [Online]. Available: <https://cloud.google.com/speech-to-text>. [Accessed: May 4, 2025].
2. P. C. Jones and L. K. Smith, “Designing Mobile Applications for Emergency Services,” in *Proc. 2020 Int. Conf. Mobile Comput. Wireless Commun.*, New York, NY, USA, Nov. 2020, pp. 201–208.
3. B. K. Lee, “Emergency Alert System: A Survey of Techniques and Applications,”

*IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 2331–2342, Sept. 2020.

1. R. A. Patel, *Android Programming: A Comprehensive Guide to Kotlin*, 2nd ed. London, UK: TechPress, 2021.