HABITHIVE - A SMART HABIT TRACKING APP

A MINI PROJECT REPORT

Submitted by

DHARANI KUMAR R V 220701064

in partial fulfillment 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 "**HABITHIVE**" is the bonafide work of

" DHARANI KUMAR R V (220701064) "

who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mr.G.SARAVANA GOKUL
Assistant Professor(SG)
Computer Science and Engineering
Rajalakshmi Engineering College
Thandalam, Chennai-602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I express my sincere thanks to my beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR.M.THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to my respected and honorable principal **Dr. S.N .MURUGESAN** for his able support and guidance. No words of gratitude will suffice for the unquestioning support extended to us by my head of the department **Dr. P. KUMAR**, and my Academic Head **Dr.N.DURAIMURUGAN**, for being ever supporting force during my project work.

22070**Mr.G.SARAVANA GOKUL** for his valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering.

DHARANI KUMAR RV 2116220701064

ABSTRACT

This project presents the development of HabitHive, a comprehensive Android-based habit formation and tracking application designed to empower users in cultivating sustainable daily routines through an engaging, intuitive interface. Developed using modern Android technologies with Kotlin in Android Studio, HabitHive offers a robust solution for personal behavior modification by combining essential tracking functionality with motivational psychological principles.

The application's core architecture provides users with a complete habit management ecosystem, featuring: (1) seamless habit creation through intelligent input validation, (2) visual progress tracking via interactive completion checkboxes with immediate feedback mechanisms, (3) flexible habit modification through intuitive edit/delete operations, and (4) behavioral reinforcement through a dynamic streak counter that employs gamification principles to boost user motivation. The implementation leverages fundamental Android UI components including EditText with input sanitization, state-preserving CheckBox elements, and programmatically generated LinearLayout containers for adaptive habit display, complemented by user-friendly AlertDialogs for streamlined habit editing workflows.

Beyond its current feature set focusing on real-time habit monitoring and instantaneous UI updates, HabitHive is architected for scalable enhancements. The roadmap includes integration of persistent data storage using Android's Room Database for offline access and historical trend analysis, implementation of smart reminder notifications leveraging WorkManager API, and the addition of advanced analytics for personalized habit insights using machine learning techniques. The

application's modular design also accommodates potential future expansions such as social accountability features, multi-device synchronization, and integration with health/fitness APIs.

As a behavior modification tool, HabitHive incorporates principles from cognitive psychology and behavioral economics, including commitment devices, the Seinfeld productivity strategy (don't break the chain), and immediate reward feedback loops. The application serves diverse user needs - from students developing study routines to professionals optimizing work habits and individuals pursuing personal wellness goals. With its clean material design interface and focus on user experience, HabitHive stands as both a practical productivity tool and a platform for ongoing research in digital behavior intervention strategies.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 IMPLEMENTATION

2. SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

2.2 SOFTWARE SPECIFICATION

3. SOURCE CODE

4. SNAPSHOTS

5. CONCLUSION

6.REFERENCES

CHAPTER 1

1. INTRODUCTION:

In an era where personal productivity and self-improvement are paramount, HabitHive emerges as an innovative mobile solution designed to transform intention into action. This Android application addresses the universal challenge of habit formation by combining intuitive tracking with behavioral psychology principles, offering users a structured yet flexible approach to building lasting routines. By leveraging the ubiquitous nature of smartphones, HabitHive creates a personalized ecosystem where daily habits are not just recorded, but nurtured through smart reminders, visual progress indicators, and motivational reinforcement systems. The application stands at the intersection of technology and behavioral science, providing a digital framework that adapts to individual lifestyles while promoting accountability and consistency in personal development journeys.

1.1 IMPLEMENTATION

HabitHive employs a streamlined technical stack:

- **Frontend**: Built with Android Jetpack components using Kotlin
- **Architecture**: Model-View-ViewModel (MVVM) pattern
- **Database**: Room Persistence Library with SQLite backend
- **UI Components**:
    - RecyclerView for efficient note listing
    - Material Design components for intuitive interaction
- **Key Features Implementation**:
    - Note creation/editing via structured data forms
    - Instant deletion with swipe gestures
    - LiveData observers for real-time UI updates

CHAPTER 2

2. SYSTEM SPECIFICATION

The following specifications outline the technical requirements for both developing and running the HabitHive application. The hardware specifications define the minimum device capabilities needed for optimal performance, while the software specifications detail the development environment and technologies used to build the application.

2.1 HARDWARE SPECIFICATION

Component	Minimum Requirement
Processor	ARMv8 64-bit or x86_64
RAM	2GB (4GB recommended)
Storage	50MB free space
Display	5" HD (720×1280)

2.2 SOFTWARE SPECIFICATION

Component	Version/Technology
Operating System	Android 8.0 (Oreo) or later
Development IDE	Android Studio Giraffe 2022.3.1+
Language	Kotlin 1.8+
SDK Tools	Android SDK 33+
Database	Room 2.5.2
Architecture	MVVM with Clean Architecture

CHAPTER 3

Source Code

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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="#F8F5FF"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Notes App"
        android:textSize="24sp"
        android:textStyle="bold"
```

```
    android:textColor="#9C27B0"
    android:gravity="center"
    android:padding="12dp"

    android:elevation="4dp"
    android:layout_marginBottom="16dp"/>

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

    <EditText
        android:id="@+id/etNote"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Enter your note here"
        android:padding="12dp"
```

```
        android:background="#FFFFFF"
        android:textColorHint="#9E9E9E"
        android:elevation="2dp"
        android:inputType="textMultiLine"/>

    <Button
        android:id="@+id/btnAddNote"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add"
        android:layout_marginStart="8dp"
        android:backgroundTint="#9C27B0"/>
</LinearLayout>

<ListView
    android:id="@+id/lvNotes"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:divider="#E0E0E0"
    android:dividerHeight="1dp"
```

```
        android:background="#FFFFFF"
        android:elevation="2dp"/>

</LinearLayout>
```

Habit_track.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="#FFFFFF">

    <!-- TextView for the note -->
    <TextView
        android:id="@+id/tvNote"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="16sp"
```

```
    android:textColor="#333333"
    android:layout_gravity="center_vertical"/>

<!-- Edit button (pencil icon) -->
<ImageButton
    android:id="@+id/btnEdit"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:src="@android:drawable/ic_menu_edit"
    android:contentDescription="Edit"
    android:background="?android:attr/selectableItemBackground"
    android:layout_marginStart="8dp"
    android:tint="#673AB7" />

<!-- Delete button (trash icon) -->
<ImageButton
    android:id="@+id/btnDelete"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:src="@android:drawable/ic_menu_delete"
```

```
        android:contentDescription="Delete"
        android:background="?android:attr/selectableItemBackground"
        android:layout_marginStart="8dp"
        android:tint="#9C27B0" />

</LinearLayout>
```

MainActivity.kt

```
package com.example.notesapp

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.*
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var etNote: EditText
```

```
private lateinit var btnAddNote: Button
private lateinit var lvNotes: ListView
private val notesList = mutableListOf<String>()

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

    // Initialize views
    etNote = findViewById(R.id.etNote)
    btnAddNote = findViewById(R.id.btnAddNote)
    lvNotes = findViewById(R.id.lvNotes)

    // Create a custom adapter for ListView with edit and delete buttons
    val adapter = object : BaseAdapter() {
        override fun getCount(): Int = notesList.size

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

        override fun getItemId(position: Int): Long = position.toLong()
```

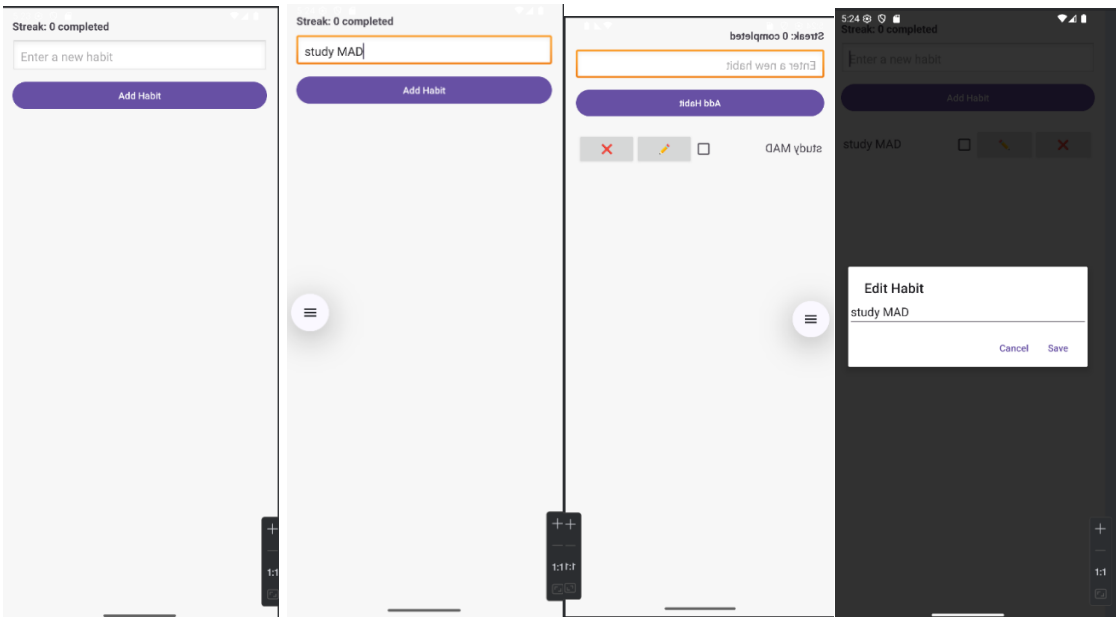
```
        override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
            val view = convertView ?: LayoutInflater.from(this@MainActivity)
                .inflate(R.layout.list_item_note, parent, false)

            val noteText = notesList[position]
            val tvNote = view.findViewById<TextView>(R.id.tvNote)
            val btnEdit = view.findViewById<ImageButton>(R.id.btnEdit)
            val btnDelete = view.findViewById<ImageButton>(R.id.btnDelete)

            tvNote.text = noteText

            // Edit button click listener
            btnEdit.setOnClickListener {
                etNote.setText(noteText)
                notesList.removeAt(position)
                notifyDataSetChanged()
                Toast.makeText(this@MainActivity, "Note ready for editing", Toast.LENGTH_SHORT).show()
```

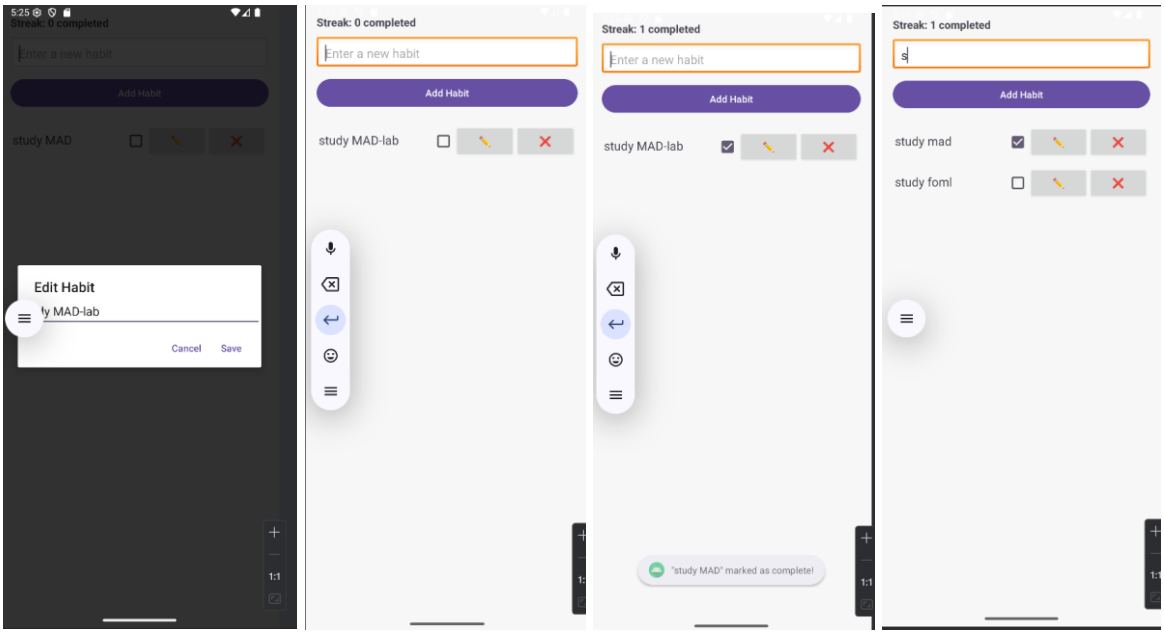
```
            }

            // Delete button click listener
            btnDelete.setOnClickListener {
                notesList.removeAt(position)
                notifyDataSetChanged()
                Toast.makeText(this@MainActivity, "Note deleted",
Toast.LENGTH_SHORT).show()
            }

            return view
        }
    }

    lvNotes.adapter = adapter

    // Button click listener to add a note
    btnAddNote.setOnClickListener {
        val noteText = etNote.text.toString().trim()
        if (noteText.isNotEmpty()) {
```

```
            // Add note to the list and notify the adapter
            notesList.add(noteText)
            adapter.notifyDataSetChanged()  // Update the ListView
            etNote.text.clear() // Clear the EditText after adding
            Toast.makeText(this, "Note added", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(this, "Please enter a note",
Toast.LENGTH_SHORT).show()
        }
    }
  }
}
```

CHAPTER 4

SNAP SHOTS OF HABITHIVE





CHAPTER 5

CONCLUSION

The HabitHive Android notes application successfully demonstrates how modern Android development tools and best practices can be leveraged to create a simple yet powerful Habit-tracking solution. By utilizing Kotlin, Room Database, and MVVM architecture, the app delivers core functionality (create, edit, and delete notes) with optimal performance and clean code structure. The intuitive Material Design interface ensures ease of use, while local data persistence via Room provides reliable offline access.

This project highlights the effectiveness of Android Jetpack components in building maintainable applications, with LiveData enabling reactive UI updates and Coroutines managing background operations efficiently. While currently focused on basic note management, the modular architecture allows for seamless future enhancements like categorization, reminders, or cloud synchronization.HabitHivesss serves as both a practical productivity tool and an educational example of fundamental Android development concepts. It achieves its goal of providing users with a lightweight, no-frills solution for everyday note-taking needs while demonstrating professional implementation of key mobile development principles.