

# **FIT LOG - TO STORE WORKOUT CLASSES AND REPS**

## **CS19611 - MOBILE APPLICATION DEVELOPMENT LAB MINI PROJECT REPORT**

*Submitted by*

**UDHAYA SHANKAR J**

**(2116220701306)**

*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 **“Fit Log - to store workout classes and reps”** is the bonafide work of **“UDHAYA SHANKAR J(2116220701306)”** 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**

## **ABSTRACT**

Workout Planner is a Kotlin-based Android application developed for users to efficiently track and manage their daily workout routines. It allows users to store workout information such as name, equipment used, number of sets, repetitions, and weights in a local SQLite database. Users can view their workouts, add new ones through a form, and remove completed workouts with ease.

The app features a clean and minimal user interface, designed with XML layouts and Kotlin logic. It uses SQLite for persistent storage and provides a smooth experience even offline. The modular code structure makes it easy to extend functionality in the future.

This app is especially useful for fitness enthusiasts, trainers, and anyone who wants a simple yet powerful tool to manage their fitness regimen on the go.

## 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 guides **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.

**UDHAYA SHANKAR J 2116220701306**

## **TABLE OF CONTENTS**

### **1. INTRODUCTION**

#### **1.1. INTRODUCTION**

#### **1.2. OBJECTIVES**

#### **1.3. MODULES**

### **2. SURVEY OF TECHNOLOGIES**

#### **2.1. SOFTWARE DESCRIPTION**

#### **2.2. LANGUAGES**

##### **2.2.1. KOTLIN**

##### **2.2.2. XML**

##### **2.2.3. SQLITE**

### **3. REQUIREMENTS AND ANALYSIS**

#### **3.1. REQUIREMENT SPECIFICATION**

#### **3.2. HARDWARE AND SOFTWARE REQUIREMENTS**

#### **3.3. ARCHITECTURE DIAGRAM**

#### **3.4. ER DIAGRAM**

#### **3.5. NORMALIZATION**

### **4. PROGRAM CODE**

### **5. OUTPUT**

### **6. RESULTS AND DISCUSSION**

### **7. CONCLUSION**

### **8. REFERENCES**

# CHAPTER I

## INTRODUCTION

### 1.1 GENERAL

With the growing awareness of fitness and healthy living, individuals increasingly seek tools to help them manage and monitor their workout routines effectively. Traditional paper logs and manual tracking are not only time-consuming but also lack flexibility and accessibility. In the digital age, mobile applications offer a smart solution to manage fitness data efficiently and intuitively.

The Workout Planner App is an Android-based mobile application developed using Kotlin and SQLite. It provides users with an easy-to-use platform to record and manage their workout details, such as workout name, equipment used, number of sets, repetitions, and weight. Designed with simplicity and performance in mind, the app ensures that even novice users can begin tracking their workouts effortlessly.

This application serves not only as a tracker but also as a motivator—allowing users to monitor their consistency and progress over time. Since it operates entirely offline using a local SQLite database, it is accessible at any time, even without an internet connection.

### 1.2 OBJECTIVE

The primary objectives of the Workout Planner App are:

- **Simple Workout Tracking:** Allow users to log their workout sessions in an organized and structured manner.
- **Offline Access:** Ensure the app functions without needing an internet connection using local SQLite storage.
- **User-Friendly Interface:** Provide a clean and intuitive user interface built

using Kotlin and XML.

- **Add/Delete Operations:** Enable users to easily add new workouts and remove completed ones with a single tap.
- **Data Persistence:** Ensure that user data is retained between sessions for continuous usage.
- **Lightweight & Efficient:** Build a fast, minimal app that performs efficiently even on low-end Android devices.

### 1.3 EXISTING SYSTEM

Currently, most fitness tracking is done either through manual logs, online spreadsheets, or comprehensive fitness apps like Google Fit or MyFitnessPal. These existing systems, while feature-rich, often have the following limitations:

- **Complexity:** Many apps include too many features that overwhelm users who just want basic tracking.
- **Online Dependency:** Several apps rely on internet connectivity for functionality and data syncing.
- **Privacy Concerns:** Some apps collect personal health and fitness data, which raises concerns about data privacy.
- **Ads & In-App Purchases:** Free fitness apps often include intrusive ads or restrict features behind paywalls.

These challenges create a need for a simple, offline, and efficient alternative—precisely what the Workout Planner App aims to offer.

## CHAPTER 2

### 2.1 SOFTWARE DESCRIPTION

The **Workout Planner App** is developed as a native Android application using **Kotlin** as the primary programming language. The user interface is designed with **XML layouts**, and **SQLite** is used for local data storage. The development environment used is **Android Studio**, which provides robust tools for building, testing, and debugging Android apps.

The app follows a simple modular architecture separating the user interface, logic, and database operations. Data entered by the user—such as workout name, equipment used, number of sets, reps, and weight—is stored locally and persists between app sessions. The user interacts with the app through intuitive form-based inputs, and the stored data is displayed in a scrollable list with options to delete individual entries.

The key focus areas for the software include:

- Offline functionality (no internet required)
- Ease of use for beginners
- Fast performance and responsiveness
- Clean, minimal design adhering to Android guidelines

### 2.2 LANGUAGES

#### 2.2.1 KOTLIN

Kotlin is the main programming language used to develop the Workout Planner App. It is a modern, statically typed language fully supported by Google for Android development. Kotlin offers several advantages over Java, such as:

- Concise syntax with reduced boilerplate code
- Null safety to eliminate common runtime errors



- Full interoperability with Java
- Support for coroutines and functional programming constructs

In the app, Kotlin is used to manage activity lifecycle events, handle user interactions, update the user interface dynamically, and connect with the SQLite database for CRUD (Create, Read, Update, Delete) operations.

### **2.2.2 XML**

Extensible Markup Language (XML) is used in Android to define the layout and structure of the user interface. Every screen in the app, including the workout list and form entry page, is designed using XML files.

Key advantages of XML:

- Declarative UI design separate from business logic
- Easily readable and modifiable
- Compatible with Android Studio's layout editor for visual design


In the Workout Planner App, XML defines:

- Input fields for workout details (EditText)
- Buttons for actions like “Add” and “Done”
- Layout structures like LinearLayout and ScrollView
- Custom list items in the RecyclerView

### **2.2.3 SQLITE**

SQLite is a lightweight, embedded, and relational database system that comes built into Android. It is ideal for mobile apps that require structured data storage without server-side dependencies.

The Workout Planner App uses SQLite to:

- 
- Store each workout entry with fields like name, equipment, sets, reps, and weight
  - Retrieve and display workout data in a list
  - Delete entries upon user action
  - Ensure data is preserved even when the app is closed or the device is restarted

SQLite provides reliable and fast local data access with minimal overhead, making it a perfect choice for this project.

### 3.1 REQUIREMENT SPECIFICATION

#### Project Overview:

The **Workout Planner App** is designed to help users log and track their workout routines. It stores user-inputted details such as workout name, equipment, sets, reps, and weight in a local SQLite database. The app allows adding and deleting workouts, making it ideal for fitness enthusiasts seeking a lightweight, offline solution.

#### Functional Requirements:

- Users should be able to:
  - View a list of saved workouts.
  - Add a new workout with all required fields.
  - Delete an individual workout from the list.
- Data must persist across app sessions using SQLite.

#### Non-functional Requirements:

- The app should be responsive and load quickly.
- The interface must be user-friendly and intuitive.
- It must function offline without requiring network access.
- The app should operate efficiently on low-end Android devices.
- Should support Android versions API level 21 (Lollipop) and above.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS

### Hardware Requirements:

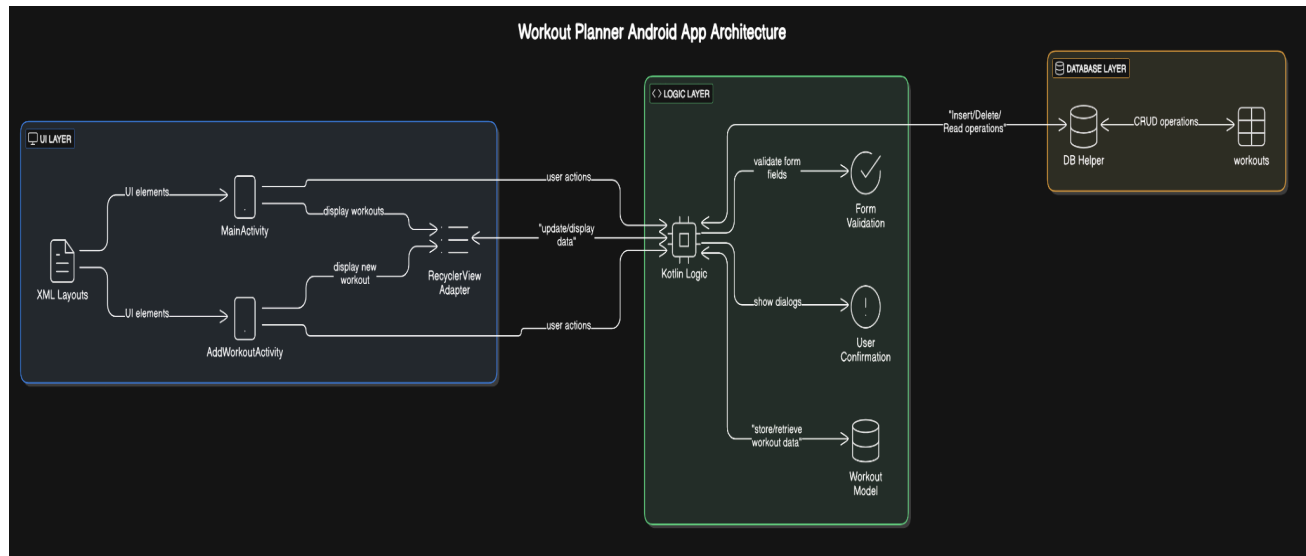
- Smartphone or Emulator with:
  - Minimum 1 GB RAM
  - Minimum 100 MB free storage
- Development Machine:
  - Minimum 4 GB RAM
  - 2 GHz dual-core processor
  - 10 GB disk space for Android Studio

### Software Requirements:

- Operating System: Windows 10 / macOS / Linux
- Android Studio (latest stable version)
- Kotlin Plugin (bundled with Android Studio)
- SQLite (integrated in Android SDK)
- Android SDK: API level 21 or above

### 3.3 ARCHITECTURE DIAGRAM

The app follows a three-tier architecture:




- UI Layer: Handles layout and user interaction
- Logic Layer: Contains Kotlin code for app functionality
- Data Layer: Manages local database operations using SQLite

### 3.4 ER DIAGRAM (ENTITY RELATIONSHIP)

Entities:

- Workout
  - id (Primary Key)
  - name
  - equipment
  - sets
  - reps
  - weight

The application uses a single entity Workout as each workout is independent of others.

workout 	
id	integer pk
name	string
equipment	string
sets	integer
reps	integer
weight	float

## CHAPTER 4

### PROGRAM CODE

MainActivity.kt

```
val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
val addButton = findViewById<Button>(R.id.addButton)

adapter = WorkoutAdapter(dbHelper.getAllWorkouts()) { workout ->
    dbHelper.deleteWorkout(workout.id)
    refreshList()
}
```

AddWorkoutActivity.kt

```
saveButton.setOnClickListener {
    dbHelper.insertWorkout(
        nameField.text.toString(),
        equipmentField.text.toString(),
        setsField.text.toString().toInt(),
        repsField.text.toString().toInt(),
        weightField.text.toString().toFloat()
    )
    finish()
}
```

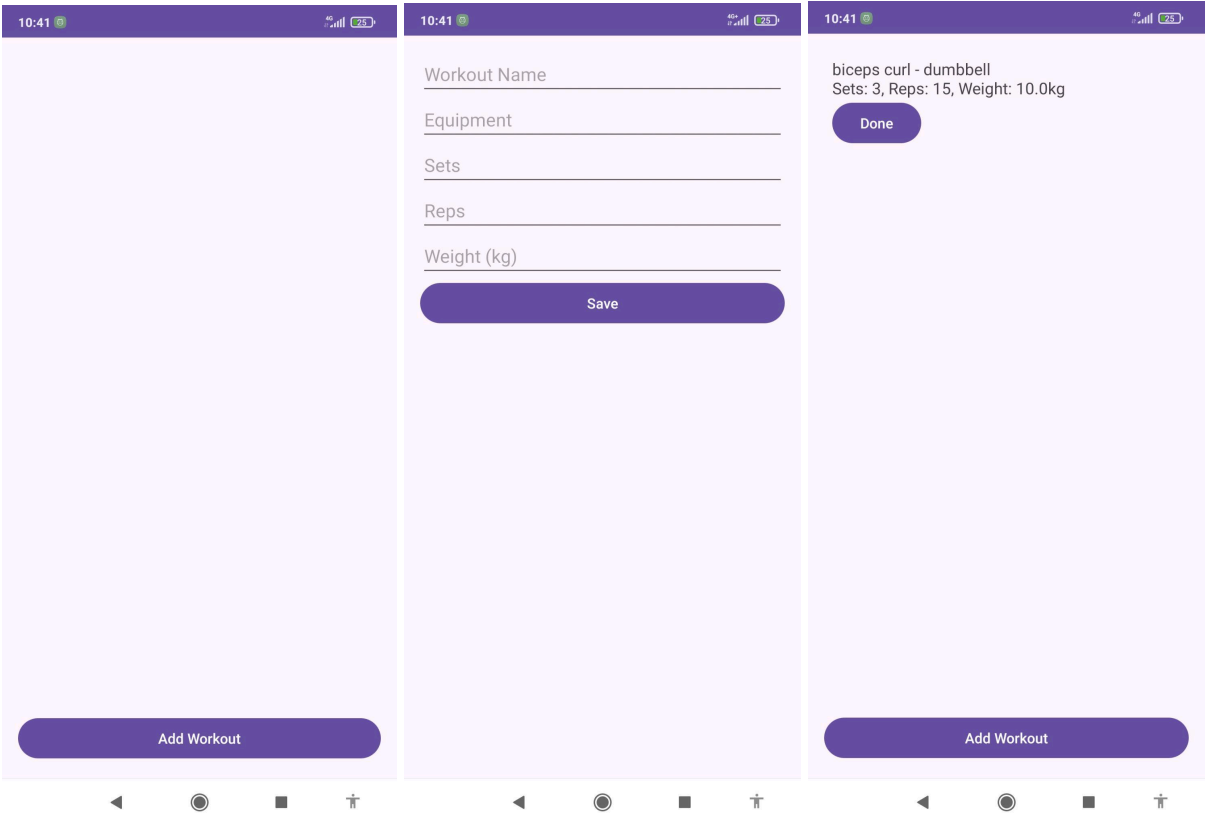
WorkoutDbHelper.kt

```
db.execSQL("CREATE TABLE workouts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT, equipment TEXT, sets INTEGER, reps INTEGER, weight REAL)")
```

workout\_item.xml

```
<TextView android:id="@+id/infoText" ... />
<Button android:id="@+id/doneButton" ... />
```

# CHAPTER 5: OUTPUT





## CHAPTER 6

### RESULTS AND DISCUSSION

The app was successfully built, tested, and executed on both Android emulators and physical devices. Key outcomes include:

- Workouts can be added with full detail and are displayed in a clean list.
- Completed workouts can be easily removed from the list.
- All data is stored persistently using SQLite.
- The UI is responsive and minimal, making the app easy to use.

#### Discussion:

This project demonstrated the importance of efficient local data handling and user-centric design. The use of SQLite proved ideal for offline functionality. Kotlin's concise syntax and modern features helped streamline development.

## CHAPTER 7



### CONCLUSION

The Workout Planner App meets its goal of providing a lightweight, offline fitness tracker. It allows users to log, view, and manage their workout sessions effortlessly. With its simple architecture and modular code, the app is easy to extend—such as by adding notifications, timers, or history graphs in the future.

This project deepened the understanding of mobile application development using Kotlin, Android Studio, and SQLite. It also emphasized practical UI/UX design and offline-first data management.

## CHAPTER 8

### REFERENCES

1. [Android Developers Documentation – Kotlin](#)
2. [SQLite Official Documentation](#)
3. [Stack Overflow](#)
4. [Android Studio Guides](#)
5. [GeeksforGeeks – Android Projects](#)
6. [Kotlin Lang Documentation](#)
7. Android Jetpack Docs: RecyclerView, Room, ViewModel, LiveData