

**A**  
**MINOR PROJECT REPORT**  
**On**  
**Building A Fitness App In Android**  
**Environment**  
**Using MVVM**

Submitted in partial fulfillment of the requirement for the award of the  
degree of

**B.TECH**  
**in**  
**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**  
**NAME OF THE STUDENT : Jathiswar B**

**Reg. No : RA1811003040088**



**SRM**  
**INSTITUTE OF SCIENCE & TECHNOLOGY**  
*Deemed to be University u/s 3 of UGC Act, 1956*

**DEPT. OF COMPUTER SCIENCE & ENGINEERING**  
**SRM Institute of Science & Technology**  
**Vadapalani Campus, Chennai**

**November 2021**

## **BONAFIDE CERTIFICATE**

Certified that this project report **“Building A Fitness App In Android Environment Using MVVM”** is the bonafide work of **“Jathiswar B”** who carried out the project work under my supervision.

### **SIGNATURE OF THE GUIDE**

C.Sabarinathan

Assistant Professor  
Department of Computer Science and  
Engineering  
SRM Institute of Science & Technology  
Vadapalani Campus

### **SIGNATURE OF THE HOD**

Dr.S.Prasanna Devi, B.E.,M.E., Ph.D.,  
PGDHRM.,PDF(IISc)

Professor  
Department of Computer Science and  
Engineering  
SRM Institute of Science & Technology  
Vadapalani Campus

## ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, **Mr.C.Sabarinathan** for his valuable inputs, able guidance,encouragement,whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Dr .S.Prasanna Devi, for encouraging and allowing us to present the project on the topic “**Building A Fitness App In Android Environment Using MVVM**“ at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

We take this opportunity to thank all our faculty members, Dean, Dr.C.V.Jayakumar, and Management who have directly or indirectly helped our project. Last but not the least we express our thanks to our friends for their cooperation and support.

## TABLE OF CONTENTS

<b>Chapter</b>	<b>Title</b>	<b>Page No</b>
	<b>Cover Page</b>	<b>1</b>
	<b>Title Page</b>	<b>1</b>
	<b>Bonafide Certificate</b>	<b>2</b>
	<b>Acknowledgement</b>	<b>3</b>
	<b>Table of Contents</b>	<b>4</b>
	<b>Abstract</b>	<b>7</b>
<b>1</b>	<b>Introduction</b>	
1.1	Overview of Project	8
1.2	Aim of the Project	9
<b>2</b>	<b>Project Work</b>	
2.1	System Requirements	10
2.2	Android Studio	10
2.3	Firebase Realtime Database	11
2.4	Firebase Authentication	14
2.5	SQLite Local Database	15
2.6	RESTful Services	16
2.7	Motion Sensors	17
2.8	Software Architecture Diagram	19
2.9	Modules	20
2.10	Model View View Model	21

2.11	Volley Library	23
3	<b>Source Code and Screenshots</b>	
3.1	Splash Screen	24
3.2	Login Activity	26
3.3	Signup Activity	32
3.4	Home Fragment	38
3.5	Diet Fragment	42
3.6	Step Tracker	48
3.7	SSL Certification	52
3.8	Home Activity	53
3.9	Videos Activity	57
3.10	Videos Model	63
3.11	Interface	65
3.12	Full Screen Activity	67
3.13	Admin Activity	72
3.14	Dependencies	73
3.15	Manifests	73
3.16	Firebase Realtime Database	74
3.17	Firebase Authentication	74
4	<b>Conculsion and Future Scopes</b>	
4.1	Conclusion	75
4.2	Future Scopes	76
5	<b>References and Certification</b>	
5.1	References	77

5.2	Certifications	80
-----	----------------	----

# ABSTRACT

Development of mobile applications especially android development is increasing day by day. It is one of the dominant mobile operating systems in the current scenario with an 80 percent market share. There is a need to develop and deliver high quality mobile applications. Selecting a right architectural pattern for the development purpose is essential because it does not only make things easier but it also makes testing more efficient and increases the quality of an application. There are different architectures for mobile development such as MVC, MVP and MVVM, clean architecture and VIPER. Apple (iOS) uses MVP to develop its applications but android developers do not have a specific architecture to develop their applications it rather depends on the developer's experience that which pattern he chooses for the development. This project aims to create an application based on MVVM architectural pattern. The aim of this project is to test and prove the smoothness and reliability of such architectural models in handling the views and data.

# CHAPTER 1

## INTRODUCTION

### 1.1. Overview of the Project

The demand of mobile application is increasing day by day especially for Android and iOS. Android operating system took 80% of the market share according to statistics from the research company Gartner. It can be assumed that the growing importance of software architectural principles is a natural progression of software-engineering practices in any IT field. Therefore, it is necessary to understand the drivers behind the quick adoption of software architectures in mobile development and the nature of the adopted architectural concepts and principles, and how their utilization has improved and how their utilization has improved and brought advancement to the development of mobile software. Software architecture is the model for structure of software and define how the software will work. This makes it possible to reuse the modules for the other software. Every structure consists software elements and relationships between them. Software architecture is used making important structural decisions that are very costly to modify after implementation. The architecture options include structural options specific to the possibilities in the software design. It has been observed that the application developed without architecture are not maintainable cannot be further developed and it can also drastically. There is no fixed architecture for the development of the android application. Because the design pattern is abstract and its implementation depends on specific requirements. Developers use different established architectural patterns to design the interactive applications such as MVC, MVP and MVVM, viper and clean architecture.



## **1.2. Aim of the Project**

In this project we are going to analyze the working of MVVM architecture. We are designing a fitness application which has all the solutions for our fitness needs getting the data and views using MVVM. We store the data for our project in Firebase Database.

The project consists of three modules – Fitness practice guide done using MVVM , Step Tracker and a Diet information displayer done using REST API. Overall you can see the entire working of the architecture in this project.

## **CHAPTER 2**

### **PROJECT WORK**

#### **2.1 System Requirements**

##### **Software Requirements**

- Windows 7,8 or 10 / Mac / Linux
- Android Studio
- Firebase Cloud Database

##### **Hardware Requirements**

- 8 GB DDR3 RAM
- Intel i5 processor and above

##### **Technology Requirements**

- Android Studio
- Firebase Realtime Database
- Firebase Authentication
- SQLite Local Database
- RESTful Services
- Motion Sensors

#### **2.2 Android Studio**

Android Studio is the official Integrated Development Environment for Android app Development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance productivity

when building Android apps. Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV and Android Auto.

### Why Android Studio?

- A Flexible Gradle Based Build System
- A Fast and feature-rich emulator
- A unified environment where you can build for all Android devices.
- Apply changes to push code and resource changes to your app without restarting your app.
- Code templates and Github integration to help you build common app features and import Sample code.
- Extensive Testing Tools and Frameworks.
- C++ and SDK support.
- Can be used for Java or Kotlin or even for Cross platform like Flutter.
- Lint tools to catch performance, usability, version compatibility and other problems.
- Easy connection to Firebase using plugins.

## 2.3 Firebase Realtime Database

Firebase is a Backend as a Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base and earn profit. It is built on Google's infrastructure.

Firebase evolved from Envolv, a prior start-up founded by James Templin and Andrew Lee in 2011. Envolv provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Templin and Lee found that it was being used to pass application data that were not chat messages. Templin and Lee found that it was being used to pass application data that were not chat messages. Developers were using Envolv to sync application data such as game state in real time across their users. Templin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in 2011 and it launched to the public in April 2012. Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with our Apple platforms, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

## 2.4 Firebase Authentication

Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices.

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.

To sign a user into your app, you first get authentication credentials from the user. These credentials can be the user's email address and password, or an OAuth token from a federated identity provider. Then, you pass these credentials to the Firebase Authentication SDK. Our backend services will then verify those credentials and return a response to the client.

After a successful sign in, you can access the user's basic profile information, and you can control the user's access to data stored in other Firebase products. You can also use the provided authentication token to verify the identity of users in your own backend services.

## 2.5 SQLite Local Database

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is one of the fastest-growing database engines around, but that's growth in terms of popularity, not anything to do with its size. The source code for SQLite is in the public domain.

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

### Why SQLite?

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
- SQLite supports most of the query language features found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and provides simple and easy-to-use API.

## 2.6 RESTful Services

RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

The following principles encourage RESTful applications to be simple, lightweight, and fast:

- **Resource identification through URI:** A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery. See The `@Path` Annotation and URI Path Templates for more information.
- **Uniform interface:** Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource. See Responding to HTTP Methods and Requests for more information.
- **Self-descriptive messages:** Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.



- **Stateful interactions through hyperlinks:** Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields.

## 2.7 Motion Sensors

The Android platform provides several sensors that let you monitor the motion of a device.

The sensors' possible architectures vary by sensor type:

- The gravity, linear acceleration, rotation vector, significant motion, step counter, and step detector sensors are either hardware-based or software-based.
- The accelerometer and gyroscope sensors are always hardware-based.

Most Android-powered devices have an accelerometer, and many now include a gyroscope. The availability of the software-based sensors is more variable because they often rely on one or more hardware sensors to derive their data. Depending on the device, these software-based sensors can derive their data either from the accelerometer and magnetometer or from the gyroscope.

Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. The movement is usually a reflection of direct user input (for example, a user steering a car in a game or a user controlling a ball in a game), but it can also be a reflection of the physical environment in which the device is sitting (for example, moving with you while you drive your car). In the first case, you are monitoring motion relative to the device's frame of reference or your application's frame of reference; in the second case you are monitoring motion relative to the world's frame of reference.

Motion sensors by themselves are not typically used to monitor device position, but they can be used with other sensors, such as the geomagnetic field sensor, to determine a device's position relative to the world's frame of reference (see Position Sensors for more information).

All of the motion sensors return multi-dimensional arrays of sensor values for each `SensorEvent`. For example, during a single sensor event the accelerometer returns acceleration force data for the three coordinate axes, and the gyroscope returns rate of rotation data for the three coordinate axes. These data values are returned in a float array (values) along with other `SensorEvent` parameters. Table 1 summarizes the motion sensors that are available on the Android platform.

## 2.8 Software Architecture Diagram

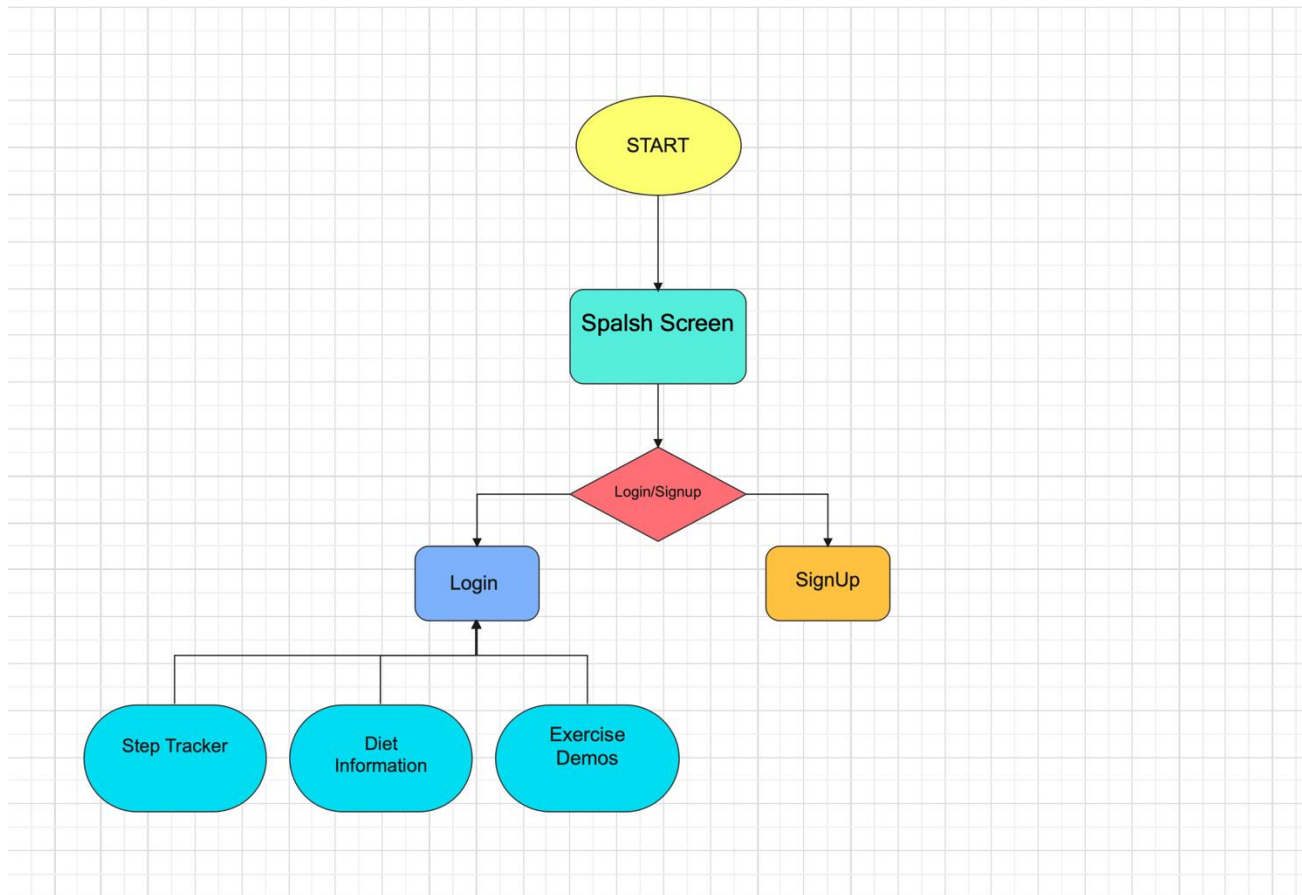


Fig. 2.1: Architecture Diagram Of Fitness App

## 2.9 Modules

- MainActivity – Which contain a bottom navigation bar to show different fragments.
- Step Tracker – Detection of steps based on motion sensor.
- Exercise Demos – Information regarding exercises developed using Firebase (MVVM)
- Diet Information – REST API diet information.
- Login – Login Using Firebase Authentication
- Sign Up – New User Signup
- SplashScreen

## 2.10 Model View View-Model

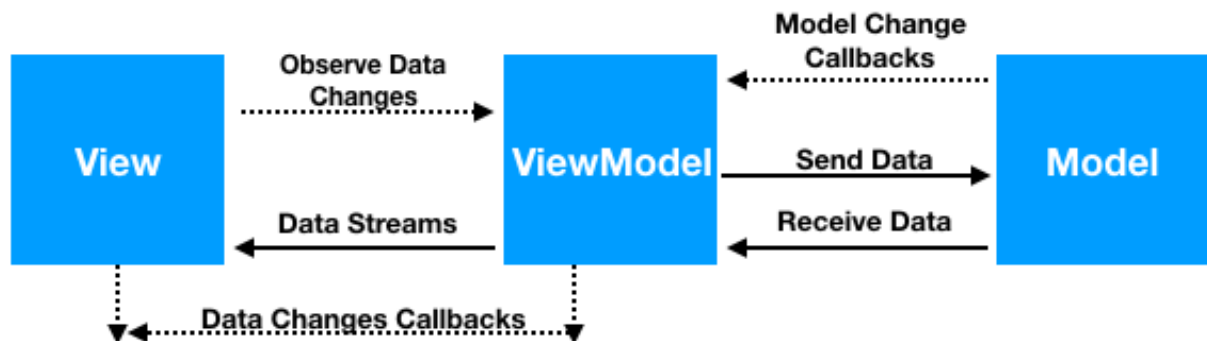


Fig. 2.2: Design Pattern of MVVM

The main players in the MVVM pattern are:

- The View — that informs the View Model about the user's actions
- The View Model — exposes streams of data relevant to the View
- The Data Model — abstracts the data source. The View Model works with the Data Model to get and save the data.

At a first glance, MVVM seems very similar to the Model-View-Presenter pattern, because both of them do a great job in abstracting the view's state and behaviour. The Presentation Model abstracts a View independent from a specific user-interface platform, whereas the MVVM pattern was created to simplify the **event driven** programming of user interfaces.

If the MVP pattern meant that the Presenter was telling the View directly what to display, in MVVM, **View Model exposes streams of events** to which the Views can bind to. Like this, the View Model does not need to hold a reference to the View anymore, like the Presenter is. This also means that all the interfaces that the MVP pattern requires, are now dropped.

The Views also notify the View Model about different actions. Thus, the MVVM pattern supports two-way data binding between the View and View Model and there is a many-to-one relationship between View and View Model. View has a reference to View Model but **View Model has no information about the View**. The consumer of the data should know about the producer, but the producer the View Model doesn't know, and doesn't care, who consumes the data.

## **Model**

Model represents the data and business logic of the app. One of the recommended implementation strategies of this layer, is to expose its data through observables to be decoupled completely from ViewModel or any other observer/consumer (This will be illustrated in our MVVM sample app below).

## **ViewModel**

This interacts with model and also prepares observable(s) that can be observed by a View.

ViewModel can optionally provide hooks for the view to pass events to the model.

One of the important implementation strategies of this layer is to decouple it from the View, i.e., ViewModel should not be aware about the view who is interacting with.

## **View**

Finally, the view role in this pattern is to observe (or subscribe to) a ViewModel observable to get data in order to update UI elements accordingly.

## 2.11 Volley Library

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. Volley is available on GitHub.

Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP cache coherence.
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- Ease of customization, for example, for retry and backoff.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

Volley excels at RPC-type operations used to populate a UI, such as fetching a page of search results as structured data. It integrates easily with any protocol and comes out of the box with support for raw strings, images, and JSON. By providing built-in support for the features you need, Volley frees you from writing boilerplate code and allows you to concentrate on the logic that is specific to your app.

Volley is not suitable for large download or streaming operations, since Volley holds all responses in memory during parsing. For large download operations, consider using an alternative like DownloadManager.

The core Volley library is developed on GitHub and contains the main request dispatch pipeline as well as a set of commonly applicable utilities, available in the Volley "toolbox."

## CHAPTER 3

### SOURCE CODE & SCREENSHOTS

#### 3.1 Splash Screen

##### Java

```
package com.jathiswarbhaskar.myfitpal;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;

public class SplashScreenActivity extends AppCompatActivity {

    private int SPLASH_SCREEN_LENGTH = 5000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new
Intent(SplashScreenActivity.this, MainActivity.class);
                startActivity(intent);
            }
        }, SPLASH_SCREEN_LENGTH);
    }
}
```

##### XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:alpha="0.7"
android:background="@drawable/fit_cover"
```



```
tools:context=".SplashScreenActivity">
```

```
<ImageView
```

```
    android:id="@+id/imageView"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
```

```
    app:layout_constraintEnd_toEndOf="parent"
```

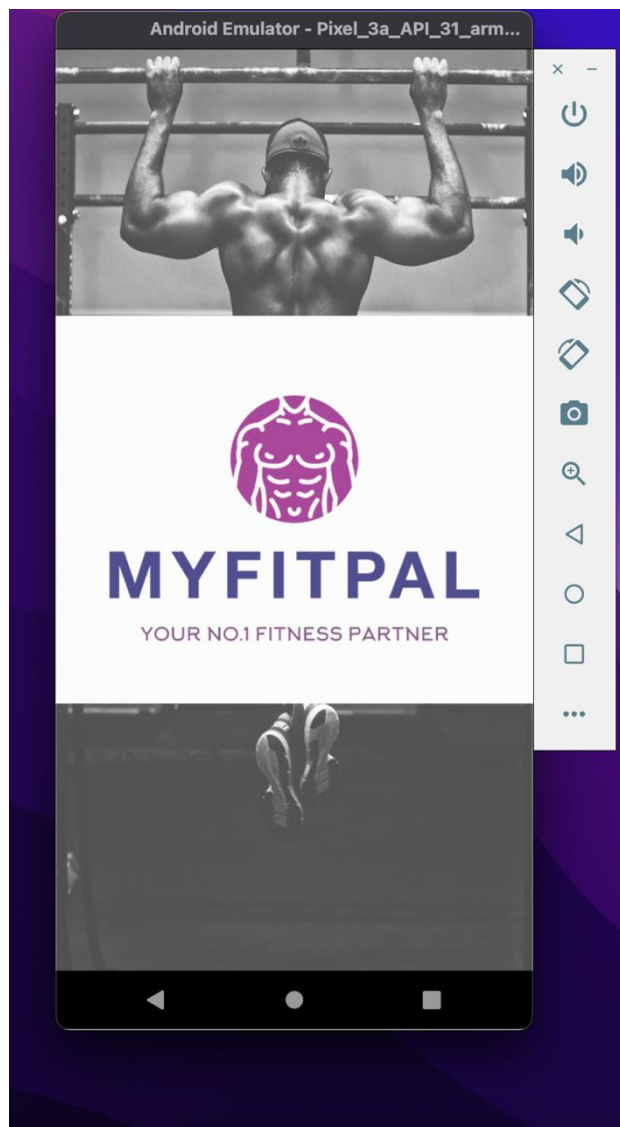
```
    app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintTop_toTopOf="parent"
```

```
    app:srcCompat="@drawable/logo" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

## Screenshot



## 3.2 Login Activity

### Java

```
package com.jathiswarbhaskar.myfitpal;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {

    private Button loginbtn, signupbtn;
    private EditText memail, mpassword;
    private FirebaseAuth mAuth;
    private ProgressDialog loadingBar;
    private TextView admintext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);

        memail = (EditText) findViewById(R.id.login_email_input);
        signupbtn = (Button) findViewById(R.id.signup_button);
        mpassword = (EditText) findViewById(R.id.login_password_input);
        loginbtn = (Button) findViewById(R.id.login_button);
        admintext = (TextView) findViewById(R.id.textView3);
    }
}
```

```

mAuth = FirebaseAuth.getInstance();
loadingBar = new ProgressDialog(this);
Login();
Signup();
textviewtouch();

}

public void Login() {
    loginbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String email = memail.getText().toString().trim();
            String password = mpassword.getText().toString().trim();

            if (TextUtils.isEmpty(email)) {
                memail.setError("Email is required");
                return;
            } else if (TextUtils.isEmpty(password)) {
                mpassword.setError("Password is required");
                return;
            } else if (TextUtils.isEmpty(password) && TextUtils.isEmpty(email)) {
                memail.setError("Email is required");
                mpassword.setError("Password is required");
                return;
            } else {
                loadingBar.setTitle("Logging In");
                loadingBar.setMessage("Please Wait while we are checking the credentials...");
                loadingBar.setCanceledOnTouchOutside(false);
                loadingBar.show();

                mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new
                OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            Intent intent = new Intent(MainActivity.this, HomeActivity.class);
                            startActivity(intent);
                        } else {
                            loadingBar.dismiss();
                            AlertDialog.Builder builder1 = new AlertDialog.Builder(MainActivity.this);
                            builder1.setMessage("Please Check Your User Id and Password.");
                            builder1.setCancelable(true);

                            builder1.setPositiveButton(
                                "Ok",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int id) {
                                        dialog.cancel();

```

```

        }
    });
    AlertDialog alert11 = builder1.create();
    alert11.show();

    mpassword.getText().clear();
    memail.getText().clear();

    }
}

});

}

});

}

}

public void Signup(){

    signupbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this,SignUpActivity.class);
            startActivity(intent);
        }
    });

}

}

public void textviewtouch(){
    admintext.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this,VideosActivity.class);
            startActivity(intent);
        }
    });
}
}

```

```
}
```

## XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:background="#faf9e1"
tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="414dp"
        android:layout_height="216dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/auth_image" />

    <EditText
        android:id="@+id/login_password_input"
        android:layout_width="357dp"
        android:layout_height="62dp"
        android:layout_below="@+id/login_email_input"
        android:background="@drawable/input_design"
        android:hint="Password"
        android:inputType="textPassword"
        android:padding="20dp"
        android:textColor="@color/grey"
        android:textColorHint="@color/grey"
        android:textSize="19sp"
        app:layout_constraintBottom_toTopOf="@+id/login_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/login_email_input" />

    <EditText
        android:id="@+id/login_email_input"
        android:layout_width="352dp"
        android:layout_height="63dp"
        android:background="@drawable/input_design"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:padding="20dp"
```

```

        android:textColor="@color/grey"
        android:textColorHint="@color/grey"
        android:textSize="19sp"
        app:layout_constraintBottom_toTopOf="@+id/login_password_input"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

```

```

<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/login_button"
    android:layout_width="352dp"
    android:layout_height="92dp"
    android:background="@drawable/buttons"
    android:fontFamily="@font/crete_round"
    android:text="Login"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textColor="@color/black"
    android:textSize="22dp"
    android:typeface="normal"
    app:layout_constraintBottom_toTopOf="@+id/textView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/login_password_input"
    tools:ignore="MissingConstraints" />

```

```

<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/signup_button"
    android:layout_width="340dp"
    android:layout_height="86dp"
    android:background="@drawable/input_design"
    android:backgroundTint="@color/black"
    android:fontFamily="@font/crete_round"
    android:text="SignUp"
    android:textAllCaps="false"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textColor="@color/white"
    android:textSize="22dp"
    android:typeface="normal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.542"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    tools:ignore="MissingConstraints" />

```

```

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/carter_one"
    android:text="or"
    android:textColor="#191818"

```

```

        android:textSize="21sp"
        app:layout_constraintBottom_toTopOf="@+id/signup_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/login_button" />

```

### <TextView

```

        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Admin?"
        android:textColor="#142385"
        app:layout_constraintBottom_toTopOf="@+id/login_button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.892"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/login_password_input" />

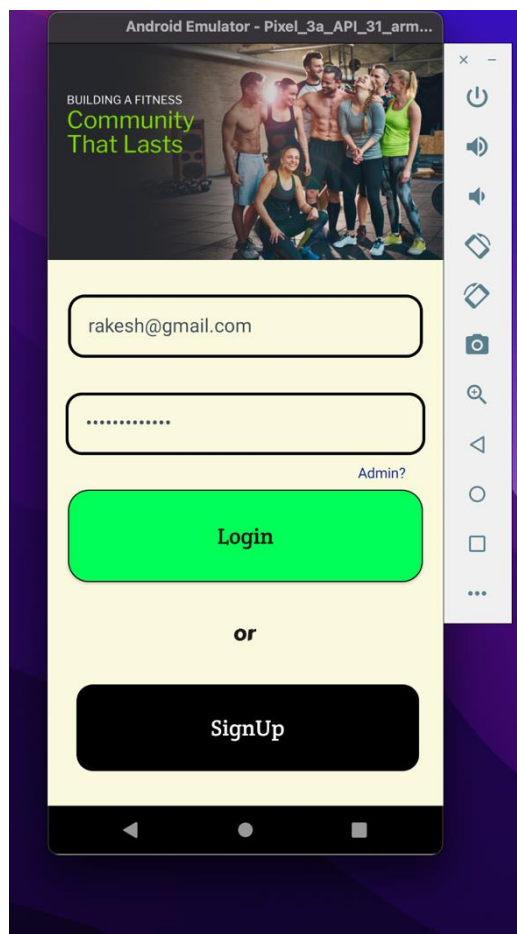
```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

## Screenshot



## 3.3 SignUp Activity

### Java

```
package com.jathiswarbhaskar.myfitpal;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.text.TextUtils;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class SignUpActivity extends AppCompatActivity {
    private Button signupbtn;
    private EditText memail,mpassword,mrecheck;
    private ProgressDialog loadingBar;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);

        signupbtn = (Button) findViewById(R.id.register_btn);
        memail = (EditText) findViewById(R.id.register_email_input);
        mpassword=(EditText) findViewById(R.id.register_password_input);
        mrecheck = (EditText) findViewById(R.id.register_password_input_1);
        mAuth = FirebaseAuth.getInstance();
        loadingBar = new ProgressDialog(this);
        Register();
    }
}
```



```

}

public void Register(){

    signupbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String email = memail.getText().toString().trim();
            String pwd = mpassword.getText().toString().trim();
            String recheck = mrecheck.getText().toString().trim();

            if (TextUtils.isEmpty(email)) {
                memail.setError("Email is required");
                return;
            }
            else if (TextUtils.isEmpty(pwd)) {
                mpassword.setError("Password is required");
                return;
            }

            else if (TextUtils.isEmpty(recheck)){
                mrecheck.setError("Retype Password is missing");
            }

            else{

                if(!pwd.equals(recheck)){
                    AlertDialog.Builder recheckbuilder = new AlertDialog.Builder(SignUpActivity.this);
                    recheckbuilder.setMessage("Password and Retype Password not same! Please try
again!");
                    recheckbuilder.setCancelable(true);

                    recheckbuilder.setPositiveButton(
                        "Ok",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });
                    AlertDialog alert11 = recheckbuilder.create();
                    alert11.show();
                }

                else{
                    loadingBar.setTitle("Creating Account");
                    loadingBar.setMessage("Please Wait while we are checking the credentials...");
                    loadingBar.setCanceledOnTouchOutside(false);
                    loadingBar.show();
                }
            }
        }
    });
}

```



```

});

}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:background="@drawable/signupbg"
tools:context=".MainActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/constraintLayout"
        android:layout_width="0dp"
        android:layout_height="457dp"
        android:padding="10dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:background="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <EditText
            android:id="@+id/register_email_input"
            android:layout_width="347dp"
            android:layout_height="69dp"
            android:background="@drawable/input_design"
            android:backgroundTint="#322C2C"
            android:hint="Email"
            android:inputType="textEmailAddress"
            android:padding="20dp"

```

```

        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:textSize="19sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/register_password_input"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/signuptextview" />

```

#### <EditText

```

        android:id="@+id/register_password_input"
        android:layout_width="350dp"
        android:layout_height="71dp"
        android:layout_below="@+id/register_email_input"
        android:background="@drawable/input_design"
        android:backgroundTint="#322C2C"
        android:hint="Password"
        android:inputType="textPassword"
        android:padding="20dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:textSize="19sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/register_password_input_1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/register_email_input" />

```

#### <EditText

```

        android:id="@+id/register_password_input_1"
        android:layout_width="353dp"
        android:layout_height="73dp"
        android:layout_below="@+id/register_password_input"
        android:background="@drawable/input_design"
        android:backgroundTint="#322C2C"
        android:hint="Retype Password"
        android:inputType="textPassword"
        android:padding="20dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:textSize="19sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/register_btn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/register_password_input" />

```

#### <androidx.appcompat.widget.AppCompatButton

```

        android:id="@+id/register_btn"
        android:layout_width="351dp"
        android:layout_height="81dp"
        android:background="@color/green"

```

```

        android:padding="17dp"
        android:layout_marginTop="5dp"
        android:text="Create Account"
        android:textAllCaps="false"
        android:textColor="@android:color/white"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/register_password_input_1" />

```

### <TextView

```

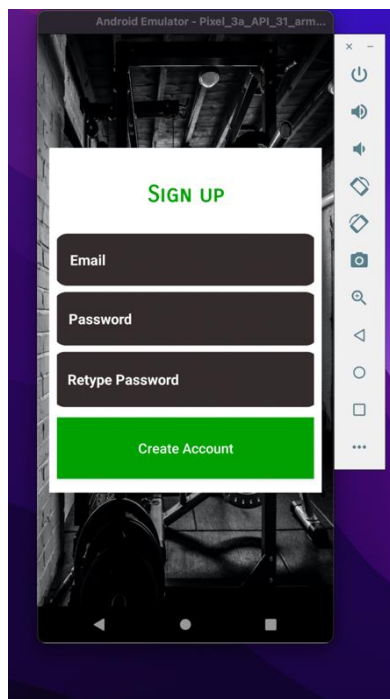
        android:id="@+id/signuptextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-smallcaps"
        android:padding="25dp"
        android:text="Sign up"
        android:textAppearance="@style/TextAppearance.AppCompat.Body2"
        android:textColor="@color/green"
        android:textSize="30dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/register_email_input"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

## Screenshot



## 3.4 Home Fragment

### Java

```
package com.jathiswarbhaskar.myfitpal;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.YouTubePlayer;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.listeners.AbstractYouTubePlayerListener;

public class HomeFragment extends Fragment {
    DatabaseReference VideosRef;
    RecyclerView recyclerView;
    String videoId;
    ProgressDialog loadingBar;
    Button mybutton,viewmore;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup view,
    @Nullable Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_home, view, false);
        mybutton = (Button) rootView.findViewById(R.id.layout_watch_btn);
        viewmore = (Button) rootView.findViewById(R.id.view_more);
        mybutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```

        Intent intent = new Intent(getActivity(),FullScreenActivity.class);
        intent.putExtra("link","kL_NJAcQBg");
        startActivity(intent);
    }
});

viewmore.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(),VideosActivity.class);
        startActivity(intent);
    }
});

return rootView;

}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/white"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <ScrollView
        android:id="@+id/recycler_menu"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical">

        <androidx.cardview.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:layout_marginTop="20dp"
            android:layout_marginEnd="15dp"
            app:cardBackgroundColor="#8E8787"
            app:cardElevation="15dp">

            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@color/white">

```

```
<com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView
    android:id="@+id/layout_videoplayer"
    android:layout_width="366dp"
    android:layout_height="244dp"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:padding="15dp"
    app:autoPlay="false"
    app:showFullScreenButton="false"
    app:showYouTubeButton="false"
    app:videoId="@string/link" />
```

#### <TextView

```
    android:id="@+id/layout_video_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/layout_videoplayer"
    android:layout_marginStart="20dp"
    android:layout_marginTop="22dp"
    android:layout_marginEnd="20dp"
    android:layout_marginBottom="10dp"
    android:fontFamily="@font/biryani"
    android:text="@string/title"
    android:textColor="#000000"
    android:textSize="25sp" />
```

#### <TextView

```
    android:id="@+id/layout_video_description"
    android:layout_width="343dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/layout_video_title"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="15dp"
    android:layout_marginBottom="20dp"
    android:fontFamily="@font/biryani"
    android:text="@string/description"
    android:textColor="#020202"
    android:textSize="10sp" />
```

#### <androidx.appcompat.widget.AppCompatButton

```
    android:id="@+id/layout_watch_btn"
    android:layout_width="336dp"
    android:layout_height="69dp"
    android:layout_below="@id/layout_video_description"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="35dp"
```



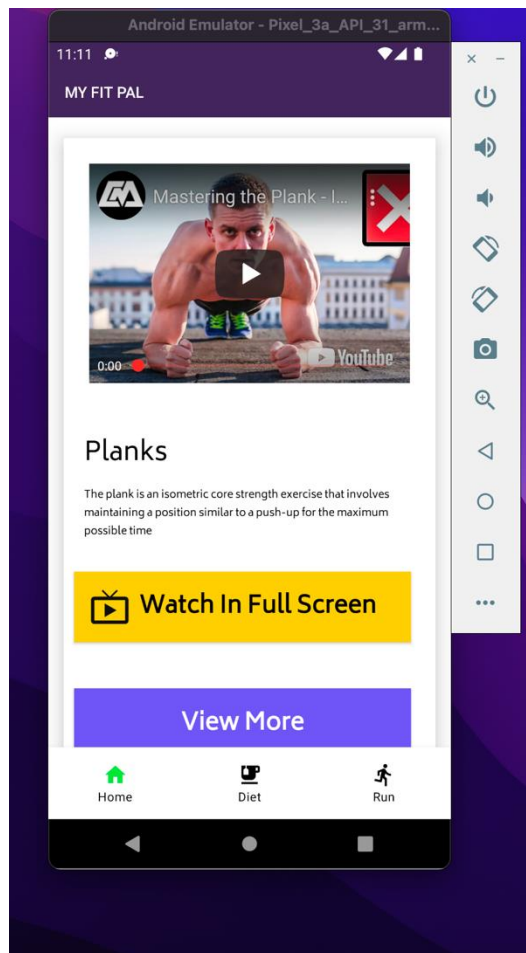
```

        android:background="@color/yellow"
        android:drawableLeft="@drawable/ic_watch"
        android:drawablePadding="-10dp"
        android:fontFamily="@font/biryani"
        android:gravity="center"
        android:padding="15dp"
        android:text="Watch In Full Screen"
        android:textAlignment="center"
        android:textAllCaps="false"
        android:textColor="@color/black"
        android:textSize="23dp"
        android:textStyle="bold" />
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/view_more"
    android:layout_width="336dp"
    android:layout_height="69dp"
    android:layout_below="@id/layout_watch_btn"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="35dp"
    android:background="@color/blue"
    android:drawablePadding="-10dp"
    android:fontFamily="@font/biryani"
    android:gravity="center"
    android:padding="15dp"
    android:text="View More"
    android:textAlignment="center"
    android:textAllCaps="false"
    android:textColor="@color/white"
    android:textSize="23dp"
    android:textStyle="bold" />
</RelativeLayout>
</androidx.cardview.widget.CardView>

</ScrollView>
</RelativeLayout>

```

## Screenshot



### 3.5 Diet Information Fragment

#### Java

```
package com.jathiswarbhaskar.myfitpal;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.StrictMode;
import android.text.method.ScrollingMovementMethod;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import org.jsoup.select.*;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
```

```

import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.jsoup.Jsoup;

import org.w3c.dom.Text;

import java.util.Queue;

public class FoodFragment extends Fragment {
    private View foodFragmentView;
    private TextView displayText;
    private Button button;
    private RequestQueue Queue;
    private EditText getsearch;
    private int count = 0;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
        @Nullable Bundle savedInstanceState) {

        foodFragmentView = inflater.inflate(R.layout.fragment_food,container,false);
        Queue = Volley.newRequestQueue(container.getContext());
        displayText = (TextView) foodFragmentView.findViewById(R.id.food_textview);
        displayText.setMovementMethod(new ScrollingMovementMethod());
        Queue = Volley.newRequestQueue(container.getContext());
        button = (Button) foodFragmentView.findViewById(R.id.searchbtn);
        getsearch = (EditText) foodFragmentView.findViewById(R.id.search_input);
    }

```

```

new NukeSSLCerts().nuke();

button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        displayText.setText(" ");
        jsonParse();
    }
});

return foodFragmentView;
}
private void jsonParse() {

    String url = "https://myjson.dit.upm.es/api/bins/wkg";
    String search_item = getsearch.getText().toString().toLowerCase();
    String ua = "Mozilla/5.0 (Windows; U; WindowsNT 5.1; en-US; rv1.8.1.6) Gecko/20070725
Firefox/2.0.0.6";

    JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try {
                    JSONArray jsonArray = response.getJSONArray("FoodItems");

                    for (int i = 0; i < jsonArray.length(); i++) {
                        JSONObject employee = jsonArray.getJSONObject(i);

                        String food = employee.getString("Food");
                        String searchfood = food.toLowerCase();
                        String measure = employee.getString("Measure");
                        String grams = employee.getString("Grams");
                        String calories = employee.getString("Calories");
                        String protein = employee.getString("Protein");
                        String fat = employee.getString("Fat");
                        String sat_fat = employee.getString("Sat.Fat");
                        String fiber = employee.getString("Fiber");
                        String carbs = employee.getString("Carbs");

                        if(searchfood.equals(search_item)||searchfood.contains(search_item)) {

```

```

        displayText.append("Item name: " + food + "\n Item measure: " + measure
            + "\n Item grams: " + grams+ "\n Item calories: " + calories
            + "\n Item protein: " + protein + "\n Item fat: " + fat
            + "\n Item saturated fat: " + sat_fat + "\n Item fiber: " + fiber
            + "\n Item carbohydrates: " + carbs+"\n\n");
        count=1;
        //String res = getImage(search_item, ua);
        // displayText.append(res);
    }

}

    } catch (JSONException e) {
        e.printStackTrace();
    }
}
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
    error.printStackTrace();
}
});

Queue.add(request);
}

public static String getImage(String question, String ua) {
    String finRes = "";
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

    try {
        String googleUrl = "https://www.google.com/search?tbm=isch&q=" + question.replace(" ",
        "");
        Document doc1 = Jsoup.connect(googleUrl).userAgent(ua).timeout(10 * 1000).get();
        Element media = doc1.select("[data-src]").first();
        String finUrl = media.attr("abs:data-src");

        finRes= "<a href=\"http://images.google.com/search?tbm=isch&q=" + question + "\"><img
src=\"\" + finUrl.replace("&quot;", "") + "\" border=1/></a>";

    } catch (Exception e) {
        System.out.println(e);
    }
}

```

```

    return finRes;
}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:background="@color/off_white">

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/searchbtn"
        android:layout_width="238dp"
        android:layout_height="52dp"
        android:background="@drawable/search_button"
        android:text="Search"
        android:textColor="@color/white"
        app:layout_constraintBottom_toTopOf="@+id/relativeLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/search_input" />

    <EditText
        android:id="@+id/search_input"
        android:layout_width="0dp"
        android:layout_height="61dp"
        android:layout_marginStart="25dp"
        android:layout_marginEnd="25dp"
        android:background="@drawable/input_design"
        android:hint="Search..."
        android:padding="20dp"
        android:textColor="@color/black"
        android:textColorHint="@color/black"
        android:textSize="19sp"
        app:layout_constraintBottom_toTopOf="@+id/searchbtn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <RelativeLayout

```

```

android:id="@+id/relativeLayout"
android:layout_width="330dp"
android:layout_height="408dp"
android:background="#6b0a65"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.5"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/searchbtn"
app:layout_constraintVertical_bias="0.372">

```

#### <TextView

```

android:id="@+id/food_textview"
android:layout_width="311dp"
android:layout_height="389dp"
android:layout_alignParentStart="true"
android:layout_alignParentTop="true"
android:layout_alignParentEnd="true"
android:layout_alignParentBottom="true"
android:layout_marginStart="10dp"
android:layout_marginTop="10dp"
android:layout_marginEnd="10dp"
android:layout_marginBottom="10dp"
android:background="@color/white"
android:fontFamily="@font/crete_round"
android:gravity="top|left"
android:importantForAutofill="no"
android:lines="400"
android:overScrollMode="always"
android:padding="5dp"
android:scrollbarStyle="insideInset"
android:scrollbars="vertical"
android:scrollHorizontally="false"
android:textAppearance="?android:attr/textAppearanceMedium"
android:textColor="@color/black"
android:textSize="18sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent" />

```

#### </RelativeLayout>

#### <TextView

```

android:id="@+id/textView2"
android:layout_width="214dp"
android:layout_height="28dp"
android:text="Enter Your Food Name"
android:textColor="#322727"
android:textSize="20sp"
app:layout_constraintBottom_toTopOf="@+id/search_input"
app:layout_constraintEnd_toEndOf="parent"

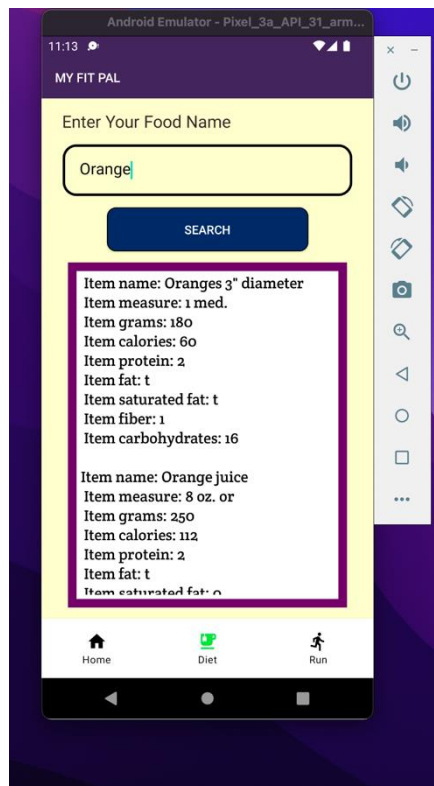
```

```

app:layout_constraintHorizontal_bias="0.137"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

## Screenshot



## 3.6 Step Tracker Fragment

### Java

```

package com.jathiswarbhaskar.myfitpal;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

```



```

import org.w3c.dom.Text;

import static androidx.core.content.ContextCompat.getCodeCacheDir;
import static androidx.core.content.ContextCompat.getSystemService;

public class RunFragment extends Fragment implements SensorEventListener {
    private View runFragmentView;
    private boolean isSensorPresent = false;
    private TextView mytextView;
    private static final String LOG_TAG = "STATS";
    private Sensor stepSensor;
    private Integer stepCount = 0;
    private double MagnitudePrevious = 0;
    private SensorManager sensorManager;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
        @Nullable Bundle savedInstanceState) {
        runFragmentView = inflater.inflate(R.layout.fragment_run, container, false);
        mytextView = (TextView) runFragmentView.findViewById(R.id.run_steps);

        sensorManager = (SensorManager)
            getActivity().getSystemService(getActivity().SENSOR_SERVICE);
        stepSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        return runFragmentView;
    }

    @Override
    public void onResume() {
        super.onResume();
        if(stepSensor != null) {
            sensorManager.registerListener(this, stepSensor, SensorManager.SENSOR_DELAY_UI);
        }
        else {
            Toast.makeText(getActivity(), "Sensor not available!", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        if(isSensorPresent)
        {
            sensorManager.unregisterListener(this);
        }
    }
}

```

```

    }

    @Override
    public void onSensorChanged(SensorEvent sensorevent) {
        if (sensorevent != null) {
            float x_acceleration = sensorevent.values[0];
            float y_acceleration = sensorevent.values[1];
            float z_acceleration = sensorevent.values[2];
            double Magnitude = Math.sqrt(x_acceleration*x_acceleration +
y_acceleration*y_acceleration
            + z_acceleration*z_acceleration);
            double MagnitudeDelta = Magnitude - MagnitudePrevious;
            MagnitudePrevious = Magnitude;

            if (sensorevent.values[0] > 6){
                stepCount++;
            }
            mytextView.setText(stepCount.toString());
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="#f7f7f7"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_marginTop="50dp"
        android:gravity="center"
        android:padding="16dp"
        >

        <ImageView
            android:layout_width="356dp"
            android:layout_height="369dp"

```

```
android:src="@drawable/step_counter" />
```

```
<TextView
```

```
    android:id="@+id/run_info"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:fontFamily="@font/roboto_medium"  
    android:text="Steps"  
    android:textAllCaps="true"  
    android:textAppearance="@style/TextAppearance.AppCompat.Body1"  
    android:textColor="@color/black"  
    android:textSize="40dp"  
    android:textStyle="bold" />
```

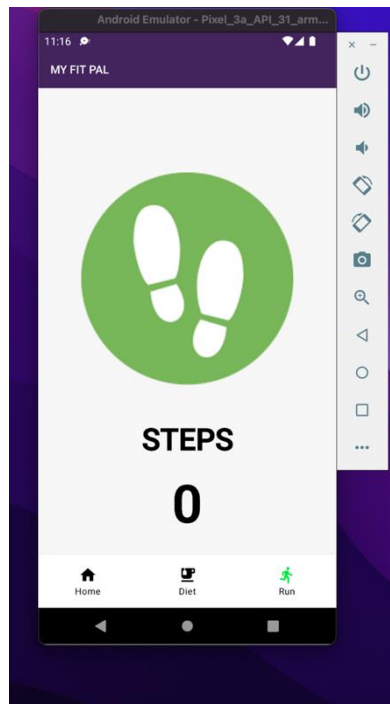
```
<TextView
```

```
    android:id="@+id/run_steps"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="10dp"  
    android:text="0"  
    android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"  
    android:textColor="@color/black"  
    android:textSize="70dp"  
    android:textStyle="bold" />
```

```
</LinearLayout>
```

```
</ScrollView>
```

## Screenshot



## 3.7 SSL Certification for API

### Java

```
package com.jathiswarbhaskar.myfitpal;

import java.security.SecureRandom;
import java.security.cert.X509Certificate;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

public class NukeSSLCerts {
    protected static final String TAG = "NukeSSLCerts";

    public static void nuke() {
        try {
            TrustManager[] trustAllCerts = new TrustManager[] {
                new X509TrustManager() {
                    public X509Certificate[] getAcceptedIssuers() {
```

```

        /* Create a new array with room for an additional trusted certificate. */
        X509Certificate[] myTrustedAnchors = new X509Certificate[0];
        return myTrustedAnchors;
    }

    @Override
    public void checkClientTrusted(X509Certificate[] certs, String authType) {}

    @Override
    public void checkServerTrusted(X509Certificate[] certs, String authType) {}
}

SSLContext sc = SSLContext.getInstance("SSL");
sc.init(null, trustAllCerts, new SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
    @Override
    public boolean verify(String arg0, SSLSession arg1) {
        return true;
    }
});
} catch (Exception e) {
    // pass
}
}
}

```

## 3.8 Home Activity

### Java

```

package com.jathiswarbhaskar.myfitpal;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;

import com.google.android.material.bottomnavigation.BottomNavigationView;

public class HomeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        BottomNavigationView bottomNav = findViewById(R.id.bottom_navigation);
        bottomNav.setOnNavigationItemSelectedListener(navlistener);
    }
}

```

```

        if(savedInstanceState==null){
            bottomNav.setSelectedItemId(R.id.nav_home);
        }

    }

    private BottomNavigationView.OnNavigationItemSelectedListener navlistener = new
    BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Fragment selectedFragment = null;
            switch (item.getItemId()){
                case R.id.nav_home:
                    selectedFragment = new HomeFragment();
                    break;
                case R.id.nav_food:
                    selectedFragment = new FoodFragment();
                    break;
                case R.id.nav_run:
                    selectedFragment = new RunFragment();
                    break;
            }

            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,selectedFragme
            nt).commit();
            return true;
        }
    };
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
tools:context=".MainActivity">
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="@color/purple_500">
    <TextView
        android:layout_width="wrap_content"
        android:text="My Fit Pal"
        android:textAllCaps="true"
        android:textAppearance="@style/TextAppearance.AppCompat.Body2"
        android:textColor="@color/white"
        android:textSize="15dp"
        android:layout_height="wrap_content"/>

</androidx.appcompat.widget.Toolbar>

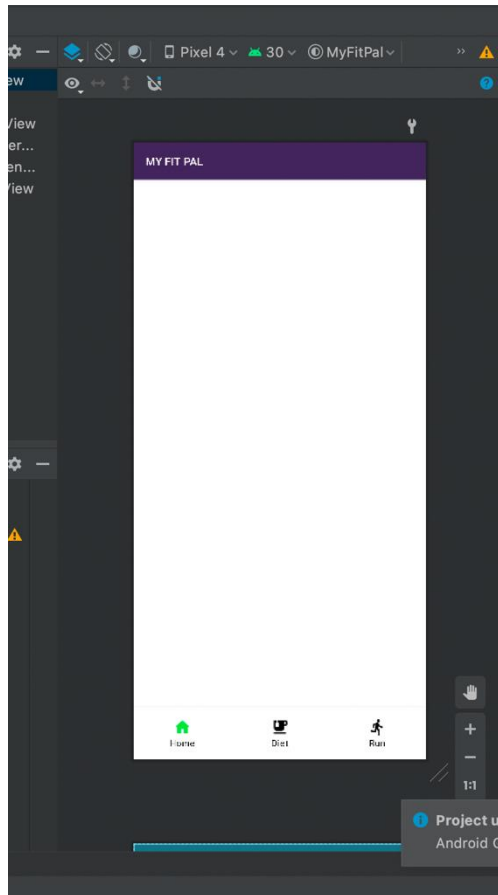
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@id/bottom_navigation"
    android:layout_below="@+id/toolbar" />

<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentBottom="true"
    android:background="@color/white"
    app:itemIconTint="@drawable/selector"
    app:itemTextColor="@color/black"
    app:menu="@menu/bottom_navigation" />

</RelativeLayout>

```

## Screenshot





## 3.9 Videos Activity (Admin Side)

### Java

```
package com.jathiswarbhaskar.myfitpal;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.YouTubePlayer;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.listeners.AbstractYouTubePlayerListener;

public class VideosActivity extends AppCompatActivity {
    private DatabaseReference VideosRef, SearchRef;
    private RecyclerView recyclerView;
    RecyclerView.LayoutManager layoutManager;
    private EditText searchText;
    private Button searchbtn;
    private String searchinput;
    private String type = "";
    private ProgressDialog loadingBar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_videos);
        VideosRef = FirebaseDatabase.getInstance().getReference().child("Videos");
        searchText = (EditText) findViewById(R.id.search_product);
        searchbtn = (Button) findViewById(R.id.search_btn);
        // getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        recyclerView = findViewById(R.id.recycler_menu);
        recyclerView.setHasFixedSize(true);
```

```

layoutManager = new LinearLayoutManager(this);
loadingBar = new ProgressDialog(this);
recyclerView.setLayoutManager(layoutManager);

loadingBar.setTitle("Please Wait");
loadingBar.setMessage("Your videos are loading...");
loadingBar.setCanceledOnTouchOutside(false);
loadingBar.show();

searchbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        searchinput = searchText.getText().toString();
        SearchRef = FirebaseDatabase.getInstance().getReference().child("Videos");

        FirebaseRecyclerOptions<VideosModel> options =
            new FirebaseRecyclerOptions.Builder<VideosModel>()
                .setQuery(SearchRef.orderByChild("title").startAt(searchinput),
VideosModel.class)
                .build();

        if (searchinput.isEmpty()) {
            options =
                new FirebaseRecyclerOptions.Builder<VideosModel>()
                    .setQuery(SearchRef, VideosModel.class)
                    .build();
        }

        FirebaseRecyclerAdapter<VideosModel, VideosViewHolder> adapter =
            new FirebaseRecyclerAdapter<VideosModel, VideosViewHolder>(options) {
                @Override
                protected void onBindViewHolder(@NonNull VideosViewHolder holder, int
position, @NonNull final VideosModel model) {
                    holder.title.setText(model.getTitle());
                    holder.description.setText(model.getDescription());
                    holder.player.addYouTubePlayerListener(new
AbstractYouTubePlayerListener() {
                        @Override
                        public void onReady(YouTubePlayer youTubePlayer) {
                            String videoId = model.getLink();
                            youTubePlayer.loadVideo(videoId, 0);
                        }
                    });
                }
            });
    }
}

```

```

        @NonNull
        @Override
        public VideosViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
            View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.exercise_layout, parent, false);
            VideosViewHolder holder = new VideosViewHolder(view);
            return holder;
        }
    };

    recyclerView.setAdapter(adapter);
    adapter.startListening();

}
});

}

@Override
protected void onStart() {
    super.onStart();

    FirebaseRecyclerOptions<VideosModel> options =
        new FirebaseRecyclerOptions.Builder<VideosModel>()
            .setQuery(VideosRef, VideosModel.class)
            .build();

    FirebaseRecyclerAdapter<VideosModel, VideosViewHolder> adapter =
        new FirebaseRecyclerAdapter<VideosModel, VideosViewHolder>(options) {
            @Override
            protected void onBindViewHolder(@NonNull final VideosViewHolder holder, int
position, @NonNull final VideosModel model) {
                holder.title.setText(model.getTitle());
                holder.description.setText(model.getDescription());
                holder.player.addYouTubePlayerListener(new AbstractYouTubePlayerListener() {
                    @Override
                    public void onReady(YouTubePlayer youTubePlayer) {
                        String videoId = model.getLink();
                        youTubePlayer.loadVideo(videoId, 0);
                    }
                });

                holder.watchButton.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intent = new Intent(VideosActivity.this, FullScreenActivity.class);

```

```

        intent.putExtra("link",model.getLink());
        loadingBar.dismiss();
        startActivity(intent);
    }
});

loadingBar.dismiss();

}

@NonNull
@Override
public VideosViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.exercise_layout, parent, false);
    VideosViewHolder holder = new VideosViewHolder(view);
    return holder;
}
};

recyclerView.setAdapter(adapter);
adapter.startListening();
}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context=".VideosActivity">

```

### <RelativeLayout

```
android:id="@+id/main_relav"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
app:layout_behavior="@string/appbar_scrolling_view_behavior"  
tools:context=".HomeActivity"  
tools:showIn="@layout/app_bar_home">
```

### <RelativeLayout

```
android:id="@+id/relav_search"  
android:layout_width="match_parent"  
android:layout_height="50dp"  
android:layout_alignParentTop="true"  
android:layout_centerHorizontal="true"  
android:background="#d55aa9">
```

### <ImageView

```
android:id="@+id/search"  
android:layout_width="40dp"  
android:layout_height="match_parent"  
android:layout_centerVertical="true"  
android:layout_marginStart="10dp"  
android:src="@drawable/ic_baseline_search_24" />
```

### <EditText

```
android:id="@+id/search_product"  
android:layout_width="280dp"  
android:layout_height="wrap_content"  
android:layout_centerVertical="true"  
android:layout_toRightOf="@+id/search"  
android:hint="Search"  
android:textColor="@color/white" />
```

### <androidx.appcompat.widget.AppCompatButton

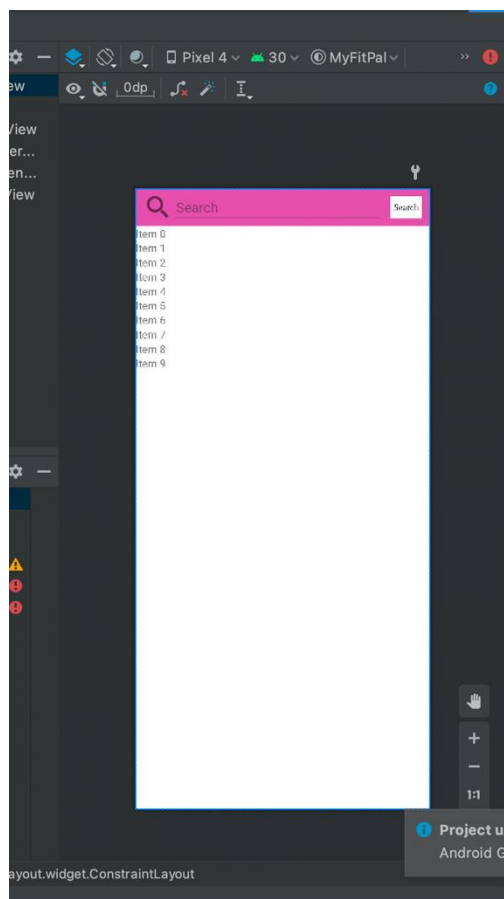
```
android:id="@+id/search_btn"  
android:textSize="10dp"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_centerVertical="true"  
android:layout_marginStart="10dp"  
android:layout_marginTop="10dp"  
android:layout_marginEnd="10dp"  
android:layout_marginBottom="10dp"  
android:layout_toRightOf="@+id/search_product"  
android:background="@color/white"  
android:fontFamily="@font/adamina"  
android:text="Search"  
android:textAllCaps="false" />
```

```
</RelativeLayout>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler_menu"
    android:layout_below="@id/relav_search"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical">
</androidx.recyclerview.widget.RecyclerView>

</RelativeLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

## Screenshot



### 3.10 Videos Model

```
package com.jathiswarbhaskar.myfitpal;

class VideosModel {
    private String title, description, link, Id;

    public VideosModel() {
    }

    public VideosModel(String title, String description, String link, String id) {
        this.title = title;
        this.description = description;
        this.link = link;
        Id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getLink() {
        return link;
    }

    public void setLink(String link) {
        this.link = link;
    }

    public String getId() {
        return Id;
    }

    public void setId(String id) {
        Id = id;
    }
}
```

### 3.11 Videos View Holder

```
package com.jathiswarbhaskar.myfitpal;

import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView;

class VideosViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    public YouTubePlayerView player;
    public TextView title,description;
    public Button watchButton;
    public ItemClickListener listener;

    public VideosViewHolder(@NonNull View itemView) {
        super(itemView);
        player = (YouTubePlayerView) itemView.findViewById(R.id.layout_videoplayer);
        title = (TextView)itemView.findViewById(R.id.layout_video_title);
        description = (TextView)itemView.findViewById(R.id.layout_video_description);
        watchButton = (Button)itemView.findViewById(R.id.layout_watch_btn);
    }
    public void setItemClickListener(ItemClickListener listener)
    {
        this.listener = listener;
    }

    @Override
    public void onClick(View view)
    {
        listener.onClick(view, getAdapterPosition(), false);
    }
}
```

### 3.12 Interface ItemClickListener

```
package com.jathiswarbhaskar.myfitpal;

import android.view.View;

public interface ItemClickListener {
    void onClick(View view , int position, boolean isLongClick);
}
```



### 3.13 Full Screen Activity

#### Java

```
package com.jathiswarbhaskar.myfitpal;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.WindowManager;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.PlayerConstants;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.YouTubePlayer;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.listeners.YouTubePlayerListener;
import com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView;

public class FullScreenActivity extends AppCompatActivity {
    private YouTubePlayerView youTubePlayerView;
    private DatabaseReference videoref;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_full_screen);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        Intent intent = getIntent();
        final String videolink = intent.getStringExtra("link");
        YouTubePlayerView youTubePlayerView = (YouTubePlayerView)
        findViewById(R.id.full_screen_player);
        youTubePlayerView.addYouTubePlayerListener(new YouTubePlayerListener() {
            @Override
            public void onReady(YouTubePlayer youTubePlayer) {
                youTubePlayer.loadVideo(videolink, 0);
            }
        })

        @Override
        public void onStateChange(YouTubePlayer youTubePlayer, PlayerConstants.PlayerState
        playerState) {
```

```

    }

    @Override
    public void onPlaybackQualityChange(YouTubePlayer youTubePlayer,
    PlayerConstants.PlaybackQuality playbackQuality) {

    }

    @Override
    public void onPlaybackRateChange(YouTubePlayer youTubePlayer,
    PlayerConstants.PlaybackRate playbackRate) {

    }

    @Override
    public void onError(YouTubePlayer youTubePlayer, PlayerConstants.PlayerError
    playerError) {

    }

    @Override
    public void onCurrentSecond(YouTubePlayer youTubePlayer, float v) {

    }

    @Override
    public void onVideoDuration(YouTubePlayer youTubePlayer, float v) {

    }

    @Override
    public void onVideoLoadedFraction(YouTubePlayer youTubePlayer, float v) {

    }

    @Override
    public void onVideoId(YouTubePlayer youTubePlayer, String s) {

    }

    @Override
    public void onApiChange(YouTubePlayer youTubePlayer) {

    }
});

}

public void onConfigurationChanged(Configuration newConfig) {

```

```

        super.onConfigurationChanged(new Config);
    }
}

```

## 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:background="@color/black"
    android:layout_height="match_parent"
    tools:context="com.jathiswarbhaskar.myfitpal.FullScreenActivity">

    <com.pierfrancescosoffritti.androidyoutubeplayer.core.player.views.YouTubePlayerView
        android:layout_width="match_parent"
        android:id="@+id/full_screen_player"
        app:showFullScreenButton="false"
        android:layout_margin="20dp"
        app:showYouTubeButton="false"
        android:layout_height="match_parent"/>
</LinearLayout>

```

### 3.14 Admin Activity

#### Java

```

package com.jathiswarbhaskar.myfitpal;

import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class AdminActivity extends AppCompatActivity {

```

```

private DatabaseReference videosRef;
private EditText mtitle,mdescription,mlink;
private String title,description,link,Id;
private ProgressDialog loadingBar;
private Button addButton;
private VideosModel newVideo;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_admin);
    mtitle = (EditText) findViewById(R.id.video_title);
    mdescription = (EditText) findViewById(R.id.video_description);
    mlink = (EditText) findViewById(R.id.video_link);
    videosRef = FirebaseDatabase.getInstance().getReference().child("Videos");
    addButton = (Button) findViewById(R.id.add_new_video_btn);

    newVideo = new VideosModel();

    addButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ValidateVideoData();
        }
    });
}

protected void ValidateVideoData(){

    title = mtitle.getText().toString().trim();
    description = mdescription.getText().toString().trim();
    link = mlink.getText().toString().trim();

    if (TextUtils.isEmpty(title))
    {
        Toast.makeText(this, "Please write video title...", Toast.LENGTH_SHORT).show();
    }
    else if (TextUtils.isEmpty(description))
    {
        Toast.makeText(this, "Please write video description...", Toast.LENGTH_SHORT).show();
    }
    else if (TextUtils.isEmpty(link))
    {
        Toast.makeText(this, "Please write video link...", Toast.LENGTH_SHORT).show();
    }
    else
    {
        SaveInfoToDatabase();
    }
}

```

```

Log.d("path",videosRef.toString());
    }

}

protected void SaveInfoToDatabase(){
    newVideo.setTitle(title);
    newVideo.setDescription(description);
    newVideo.setLink(link);
    String key = videosRef.push().getKey();
    Log.d("key",key);
    newVideo.setId(key);
    videosRef.child(key).setValue(newVideo);
    Toast.makeText(AdminActivity.this,"Video added
successfully",Toast.LENGTH_SHORT).show();

    Intent intent = new Intent(AdminActivity.this,VideosActivity.class);
    startActivity(intent);
}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:background="@drawable/details_bg"
    android:layout_height="match_parent"
    tools:context=".AdminActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="401dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"

```

```
android:layout_alignParentTop="true"
android:layout_marginStart="15dp"
android:layout_marginTop="132dp"
android:fontFamily="@font/amaranth_bold"
android:text="Please Add Your New Video Details"
android:textColor="#630C0C"
android:textSize="24sp" />
```

#### <EditText

```
android:id="@+id/video_title"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/textView"
android:layout_centerHorizontal="true"
android:layout_marginLeft="45dp"
android:layout_marginTop="48dp"
android:layout_marginRight="45dp"
android:background="@drawable/input_design"
android:hint="Video Title..."
android:inputType="textMultiLine"
android:padding="20dp" />
```

#### <EditText

```
android:id="@+id/video_description"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/video_title"
android:layout_centerHorizontal="true"
android:layout_marginLeft="45dp"
android:layout_marginTop="48dp"
android:layout_marginRight="45dp"
android:background="@drawable/input_design"
android:hint="Video Description..."
android:inputType="textMultiLine"
android:padding="20dp" />
```

#### <EditText

```
android:id="@+id/video_link"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/video_description"
android:layout_centerHorizontal="true"
android:layout_marginLeft="45dp"
android:layout_marginTop="48dp"
android:layout_marginRight="45dp"
android:background="@drawable/input_design"
android:hint="Video Link..."
android:inputType="textMultiLine"
android:padding="20dp" />
```

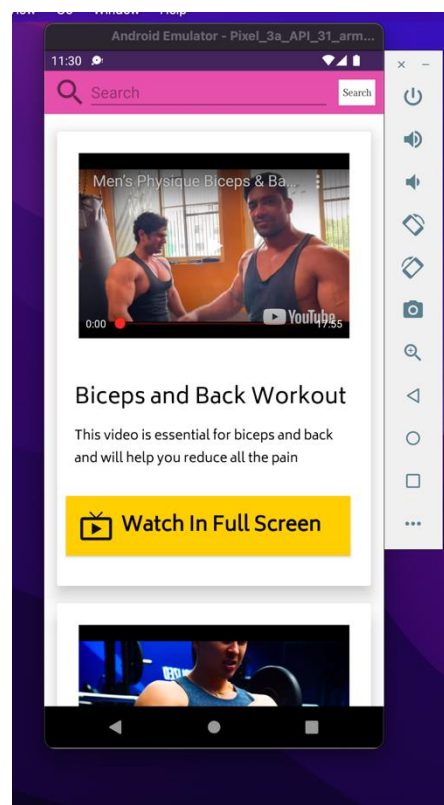
#### <Button

```
android:id="@+id/add_new_video_btn"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_below="@id/video_link"
android:layout_centerHorizontal="true"
android:layout_marginLeft="45dp"
android:layout_marginTop="53dp"
android:layout_marginRight="45dp"
android:background="@drawable/buttons"
android:padding="17dp"
android:text="Add"
android:textAllCaps="false"
android:textColor="@color/white"
android:textSize="18sp" />
```

</RelativeLayout>

## Screenshot



## 3.15 Dependencies

```
dependencies {  
    implementation 'androidx.appcompat:appcompat:1.3.0'  
    implementation 'org.jsoup:jsoup:1.14.2'  
    implementation 'com.google.android.material:material:1.1.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
    implementation 'com.google.firebase:firebase-database:19.6.0'  
    implementation 'com.google.firebase:firebase-auth:20.0.3'  
    implementation 'com.android.volley:volley:1.1.0'  
    testImplementation 'junit:junit:4.+'  
    implementation 'com.squareup.picasso:picasso:2.71828'  
    implementation 'com.google.firebase:firebase-auth:17.0.0'  
    implementation 'com.google.android.material:material:1.2.1'  
    implementation 'com.google.firebase:firebase-core:16.0.4'  
    implementation 'com.firebaseui:firebase-ui-database:6.3.0'  
    implementation 'com.pierfrancescosoffritti.androidyoutubeplayer:core:10.0.5'  
    implementation 'com.pierfrancescosoffritti.androidyoutubeplayer:chromecast-sender:0.23'  
}
```

Project update recommended  
Android Gradle Plugin can be [upgraded](#).



## 3.15 Manifests

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.jathiswarbhaskar.myfitpal">

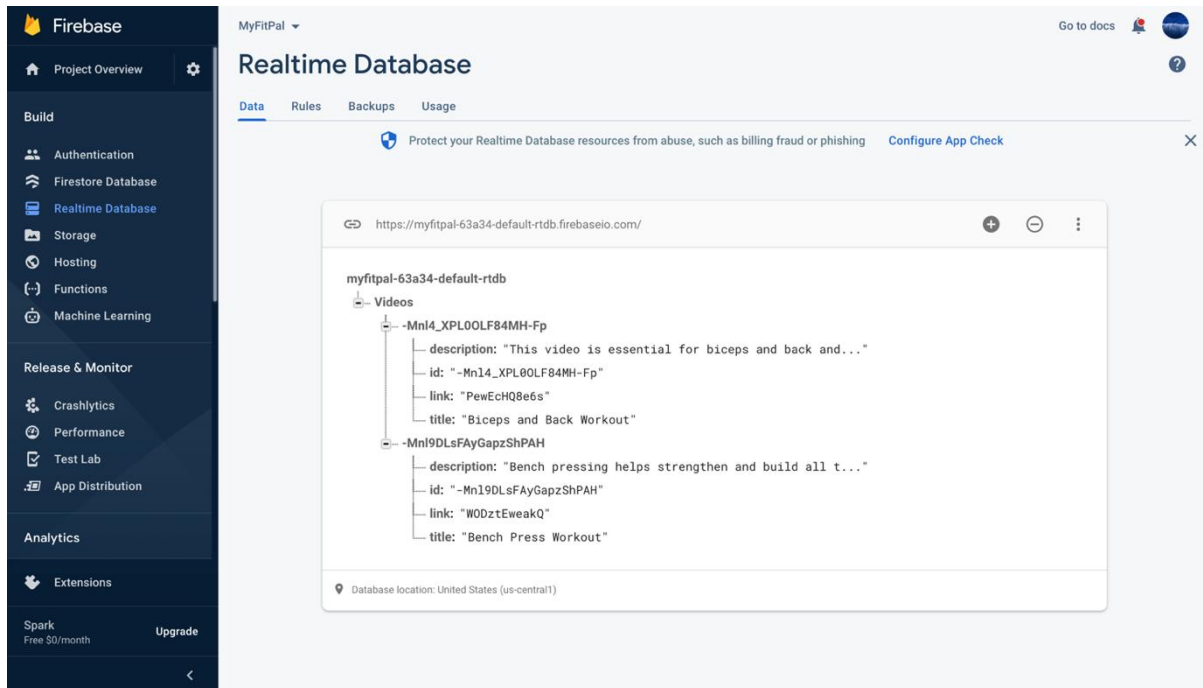
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:roundIcon="@drawable/icon"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyFitPal"
        android:usesCleartextTraffic="true"
        android:windowSoftInputMode="adjustPan|adjustResize">
        <activity
            android:name=".VideosActivity"
            android:exported="true" />
        <activity
            android:name=".AdminActivity"
            android:exported="true" />
        <activity
            android:name=".FullScreenActivity"
            android:exported="true" />
        <activity android:name=".SignUpActivity" />
        <activity
            android:name=".HomeActivity"
            android:windowSoftInputMode="adjustPan|adjustResize" />
        <activity android:name=".MainActivity" />
        <activity android:name=".SplashScreenActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>
</manifest>
```

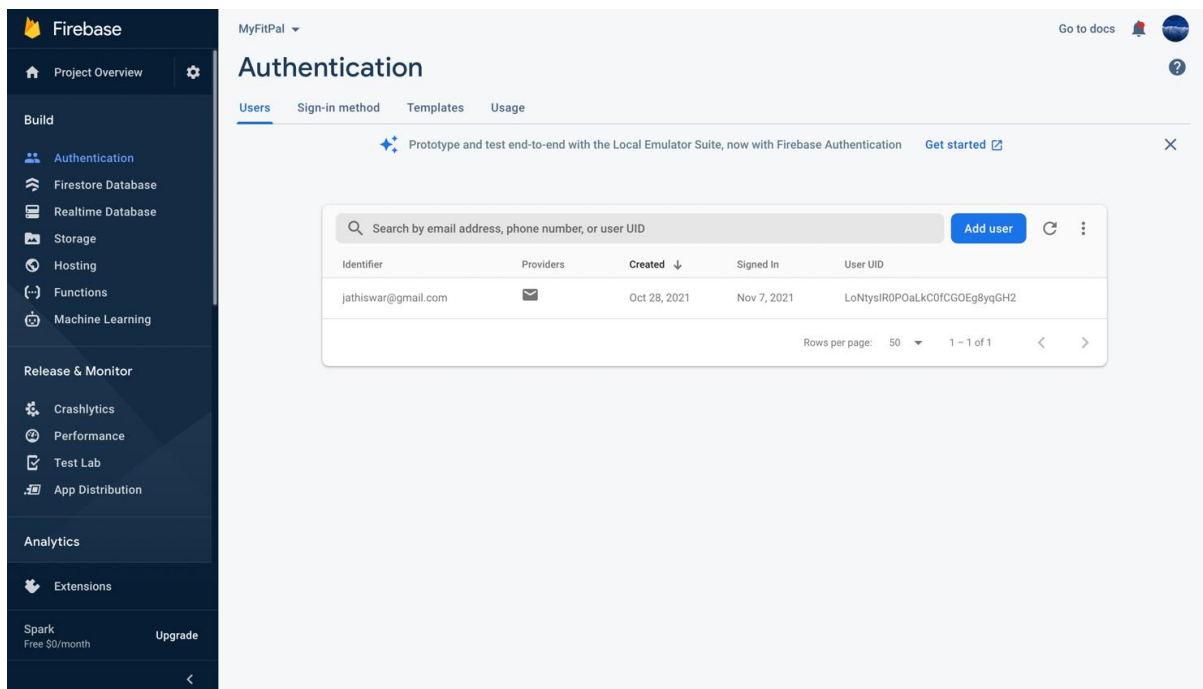
## 3.16 Firebase Relatime Database



The screenshot shows the Firebase Realtime Database console for the project 'MyFitPal'. The left sidebar contains navigation options: Project Overview, Build (Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning), Release & Monitor (Crashlytics, Performance, Test Lab, App Distribution), Analytics, and Extensions. The main panel displays the 'Realtime Database' interface with tabs for Data, Rules, Backups, and Usage. A notification banner at the top states: 'Protect your Realtime Database resources from abuse, such as billing fraud or phishing. Configure App Check'. The 'Data' tab shows a JSON tree structure for the database at 'https://myfitpal-63a34-default-rtdb.firebaseio.com/'. The root node is 'myfitpal-63a34-default-rtdb', which contains a 'Videos' collection. This collection has two nodes: '-MnI4\_XPL0OLF84MH-Fp' and '-MnI9DLsFayGapzShPAH'. Each node contains a 'description', 'id', 'link', and 'title' field. The database location is noted as 'United States (us-central1)'.

```
myfitpal-63a34-default-rtdb
├── Videos
│   ├── -MnI4_XPL0OLF84MH-Fp
│   │   ├── description: "This video is essential for biceps and back and..."
│   │   ├── id: "-MnI4_XPL0OLF84MH-Fp"
│   │   ├── link: "PewEcHQ8e6s"
│   │   └── title: "Biceps and Back Workout"
│   └── -MnI9DLsFayGapzShPAH
│       ├── description: "Bench pressing helps strengthen and build all t..."
│       ├── id: "-MnI9DLsFayGapzShPAH"
│       ├── link: "W0DztEweakQ"
│       └── title: "Bench Press Workout"
```

## 3.17 Firebase Authentication



The screenshot shows the Firebase Authentication console for the project 'MyFitPal'. The left sidebar is identical to the previous screenshot. The main panel displays the 'Authentication' interface with tabs for Users, Sign-in method, Templates, and Usage. A notification banner at the top states: 'Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication. Get started'. The 'Users' tab shows a search bar with the placeholder 'Search by email address, phone number, or user UID' and an 'Add user' button. Below the search bar is a table listing users.

Identifier	Providers	Created ↓	Signed In	User UID
jathiswar@gmail.com		Oct 28, 2021	Nov 7, 2021	LoNtysiR0POaLKC0fCG0Eg8yqGH2

At the bottom of the table, it indicates 'Rows per page: 50' and '1 - 1 of 1'.

## **CHAPTER 4 – CONCLUSION AND FUTURE SCOPES**

### **4.1 Conclusion**

In this work, the proposed work is introducing an android design pattern onto a fitness application. Specifically, we have designed a fitness videos guide fetching data from firebase database backend . Furthermore, there are also features for users that involve RESTful API services and Android based Sensors for optimal tracking. Compared with the spaghetti coding styles, this is a very reliable and robust architecture.

## 4.2 Future Scope

Architecture plays a great role in the development of application. MVVM architecture is viable for the development of the android. We performed analysis on the basis of the properties such as testability, cohesion, maintainability and coupling.

All the architectures support different properties but we have chosen the combination of MVVM with clean architecture because it supports all the attributes and clean architecture overcomes the problems of MVVM. But still we cannot say that MVVM is the best architecture for android development in all the situations. Every project has different nature, so architecture must be chosen according to the nature of the project and needs. Future work can be done by performing analysis and experiment on other attributes like comparing the performance in different architectural patterns. Every year different patterns are appearing so in future new architectures can be compared with traditional ones.

## CHAPTER 5 - REFERENCES

- [1] Sokolova, K. and Lemercier, M., 2013. Android Passive MVC: A Novel Architecture Model for Android Application Development. [Online] Academia.edu. Available at: <[http://www.academia.edu/download/51333721/Android\\_Passive\\_MVC\\_a\\_Novel\\_Architecture\\_Model.pdf](http://www.academia.edu/download/51333721/Android_Passive_MVC_a_Novel_Architecture_Model.pdf)> [Accessed 18 June 2020].
- [2] Daoud, A. and Moha, N., 2019. An Exploratory Study of MVC-Based Architectural Patterns in Android Apps | Proceedings of the 34Th ACM/SIGAPP Symposium on Applied Computing. [Online] Dl.acm.org. Available at: <<https://dl.acm.org/doi/abs/10.1145/3297280.3297447>> [Accessed 18 June 2020].
- [3] Bagheri, H., Garcia, J., Sadeghi, A., Malek, S. and Medvidovic, N., 2014. Towards High Quality Mobile Applications: Android Passive MVC Architecture. [Online] Available at: <[https://www.researchgate.net/publication/286936519\\_Towards\\_High\\_Quality\\_Mobile\\_Applications\\_Android\\_Passive\\_MVC\\_Architecture](https://www.researchgate.net/publication/286936519_Towards_High_Quality_Mobile_Applications_Android_Passive_MVC_Architecture)> [Accessed 18 June 2020].
- [4] H. Bagheri, J. Garcia, A. Sadeghi, S. Malek and N. Medvidovic, "Software architectural principles in contemporary mobile software: from conception to practice", 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0164121216300607>. [Accessed: 18-Jun- 2020].
- [5] M. Caporuscio, P. Inverardi and P. Pelliccione, "Formal Analysis of Architectural Patterns", 2004. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-3-540-24769-2\\_2](https://link.springer.com/chapter/10.1007/978-3-540-24769-2_2). [Accessed: 18- Jun- 2020].
- [6] T. Lou, "A comparison of Android Native App Architecture", Eindhoven University of Technology research portal, 2016. [Online]. Available: <https://research.tue.nl/en/studentTheses/a-comparison-of-android-native-app-architecture>. [Accessed: 18- Jun- 2020].

7] A. Bansal, A. Rana, A. Bansod and P. Baviskar<sup>4</sup>, "Mobile Based Campus Information Retrieval Android Applicatio", 2015. [Online]. Available:

<https://www.semanticscholar.org/paper/Mobile- Based-Campus-Information-Retrieval-Android- Bansal- Rana/32f244e6e0de20165880ae0e07133940765a239> 4. [Accessed: 18-Jun- 2020].

[8] H. Källström, "Increasing Maintainability for Android Applications", DIVA, 2016. [Online]. Available: <http://www.diva-portal.org/smash/record.jsf?pid=diva2:1052648>. [Accessed: 18-Jun- 2020].

[9] D. Heger, "Mobile Devices-An Introduction to the Android Operating Environment-Design, Architecture, and Performance Implications", Semanticscholar.org, 2012. [Online]. Available: <https://www.semanticscholar.org/paper/Mobile- Devices-An-Introduction-to-the-Android-%2C-%2C- Heger/5dfe58d7bf587570b776e44faa5e889ea73d098> f. [Accessed: 18-Jun- 2020].

[10] J. FAN, Y. ZENG and M. GAO, "Mobile Data Monitoring System Based on MVP Mode", 2018. [Online]. Available: <http://dpi-proceedings.com/index.php/dtcse/article/view/24726>. [Accessed: 18-Jun- 2020].

[11] V. Humeniuk, "Android Architecture Comparison: MVP vs. VIPER", DIVA, 2019. [Online]. Available: <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1291671&dswid=-7213>. [Accessed: 18-Jun- 2020].

[12] "Analysis of the Android Architecture", It.iitb.ac.in, 2010. [Online]. Available: [https://www.it.iitb.ac.in/frg/wiki/images/2/20/2010\\_braehler-stefan\\_android\\_architecture.pdf](https://www.it.iitb.ac.in/frg/wiki/images/2/20/2010_braehler-stefan_android_architecture.pdf). [Accessed: 18-Jun- 2020].

[13] T. Duy, "Reactive Programming and Clean Architecture in Android Development", Theseus.fi, 2017. [Online]. Available: <https://www.theseus.fi/bitstream/handle/10024/126982/Thesis2016-Sunshine.pdf?sequence=1>. [Accessed: 18-Jun- 2020].

[14] M. Diez, "CLEAN ARCHITECTURE Y RXJAVA EN ANDROID", Repositorio.ucundinamarca.edu.co, 2016. [Online]. Available:

[http://repositorio.ucundinamarca.edu.co/bitstream/handle/20.500.12558/1811/documento%20de%20grado.pdf? Sequence=1&isAllowed=y](http://repositorio.ucundinamarca.edu.co/bitstream/handle/20.500.12558/1811/documento%20de%20grado.pdf?Sequence=1&isAllowed=y).  
[Accessed: 18- Jun- 2020].

## COURSE CERTIFICATIONS

### 1. The Complete Android Oreo Developer Course – Build 23 Apps Certification (Issued on 31<sup>st</sup> October 2021).

