

# **Procedure to create the app for chat connect- A real time chat and communication app**

**Team ID:Team-591094**

Chat Connect is a sample project built using the Android Compose UI toolkit. It demonstrates how to create a simple chat app using the Compose libraries. The app allows users to send and receive text messages. The project showcases the use of Compose's declarative UI and state management capabilities. It also includes examples of how to handle input and navigation using composable functions and how to use data from a firebase to populate the UI.

Learning Outcomes:

By end of this project:

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.

Project Workflow:

- Users register into the application.
- After registration, user logins into the application.
- User enters into the main page

Tasks:

- 1.Required initial steps
- 2.Creating a new project
- 3.Integrating Firebase and Authentication
- 4.Creating UI files
- 5.Running the application.

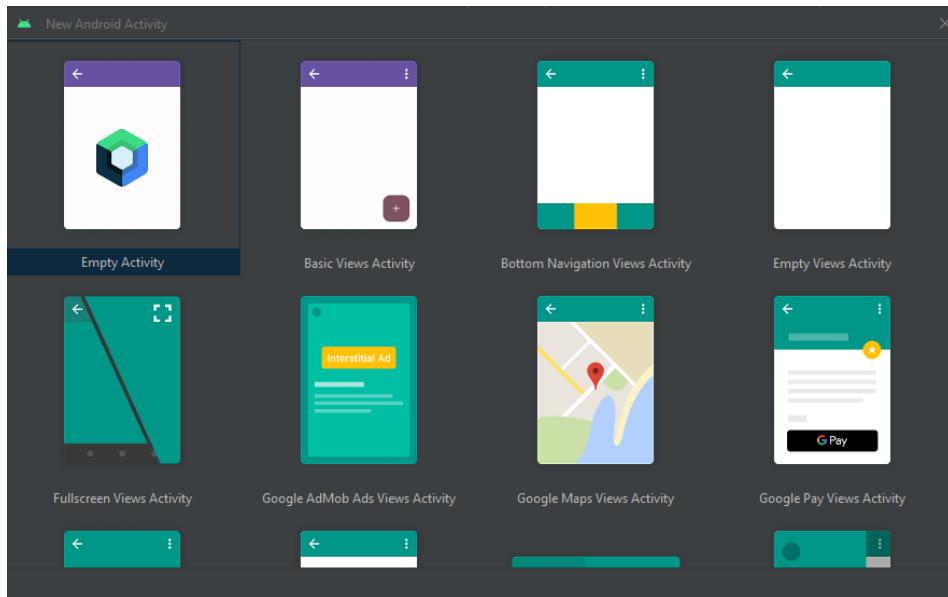
Task 1:

Required initial steps: <https://developer.android.com/studio/install>

Task 2: Creating a new project.

Step 1: Android studio > File > New > New Project > Empty Compose Activity

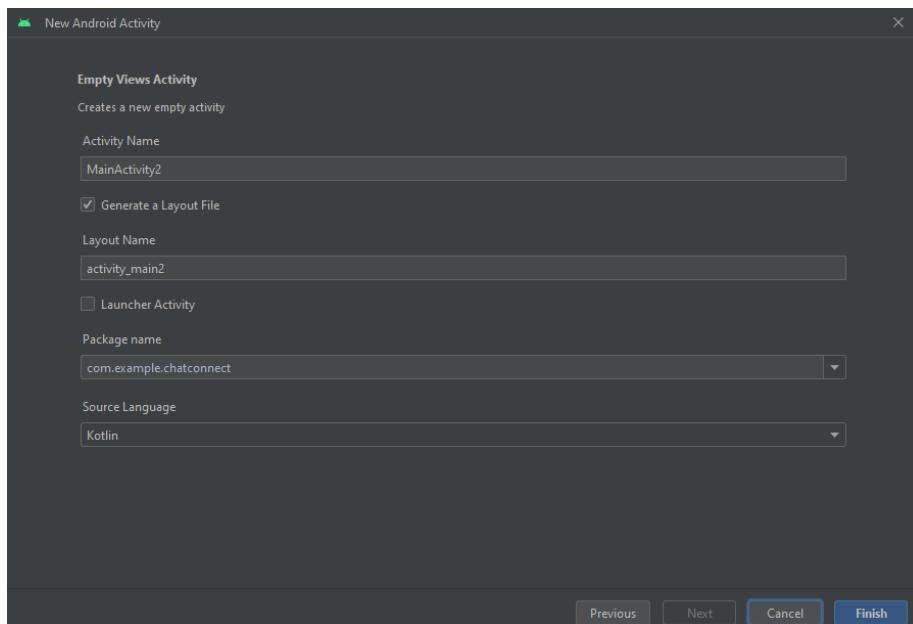
Step 2: Click on Next button.



Step 3: Give name to the new project.

Step 4: Give the Minimum SDK value

Step 5: Click Finish



## Main Activity file

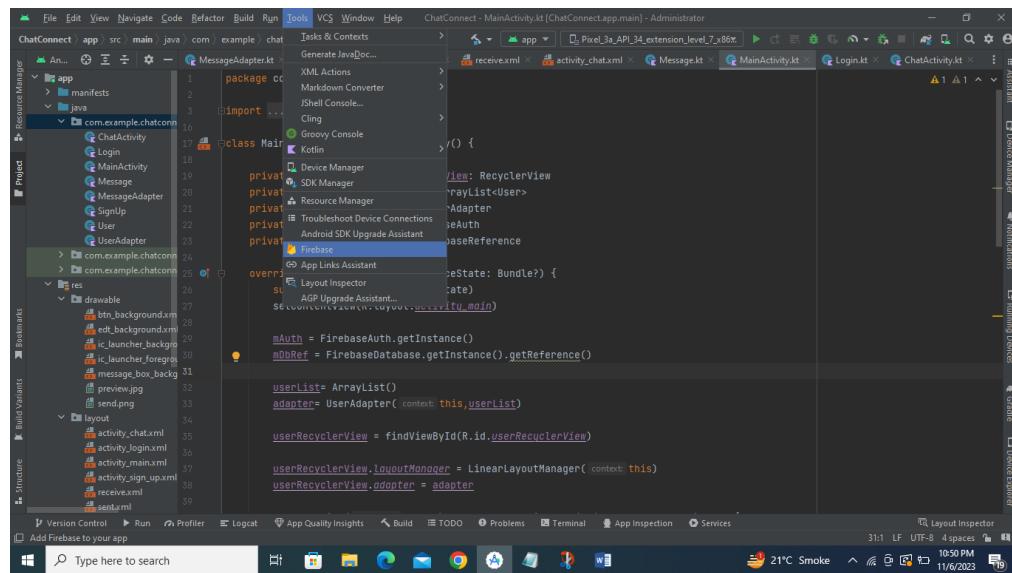
The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "ChatConnect". It contains an "app" module with "java" and "res" directories. The "java" directory includes packages for com.example.chatconnect and com.example.chatconnect, containing classes like ChatActivity, Login, MainActivity, Message, MessageAdapter, SignUp, User, and UserAdapter.
- MainActivity.kt Code:** The code for MainActivity is displayed, showing the implementation of an AppCompatActivity. It initializes variables for user RecyclerView, userList, adapter, mAuth, and mDbRef. It overrides onCreate to set the content view and initialize Firebase Auth and Database. It also initializes the userList, adapter, and RecyclerView.
- Toolbar:** The toolbar at the top shows the file path "ChatConnect > app > src > main > java > com > example > chatconnect" and the current file "MessageAdapter.kt".
- Bottom Navigation:** The bottom navigation bar includes tabs for Version Control, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, App Inspection, and Services.
- Bottom Bar:** The bottom bar displays system icons for battery, signal, and network, along with the date and time "10:49 PM 11/6/2023".

### Task 3:

#### Integrating Firebase

- Menu bar > Tools > Firebase.



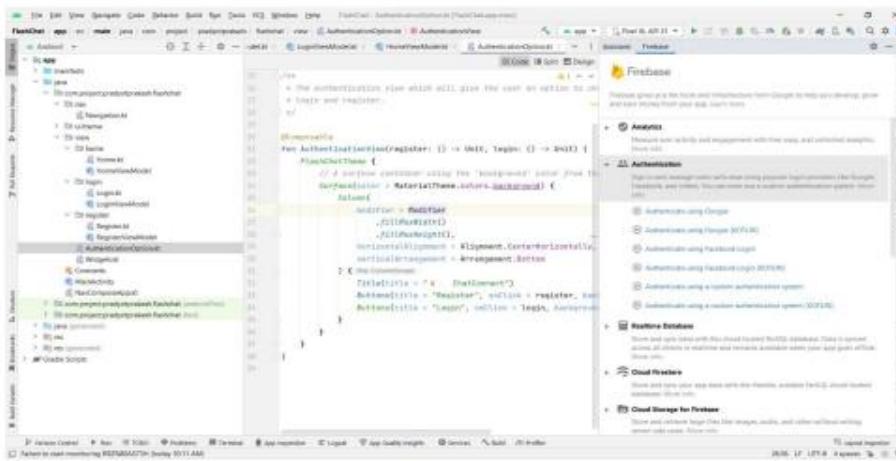
After clicking the fire base tab will open on the side

The open authentication

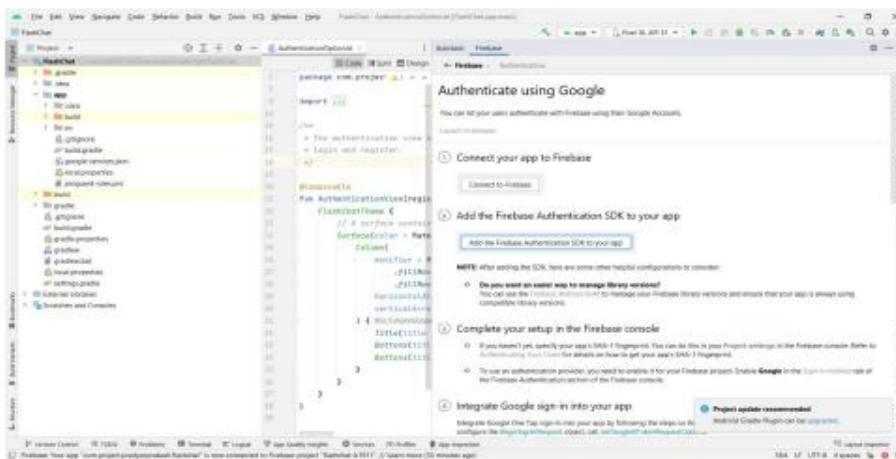
The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "ChatConnect". The "app" module contains Java files like Adapter.kt, MainActivity.kt, and Message.kt, along with XML files for layouts (activity\_main.xml, activity\_login.xml, activity\_main.xml, activity\_chat.xml) and drawables (btn\_background.xml, edit\_background.xml, ic\_launcher\_foreground.xml, message\_box\_backg.xml, preview.jpg, send.png).
- MainActivity.kt Code:** The code initializes Firebase components (mAuth, mDbRef), sets up a RecyclerView, and creates an adapter.
- Firebase Assistant Panel:** This panel provides quick access to various Firebase services:
  - Analytics:** Measure user activity and engagement.
  - Authentication:** Sign in and manage users.
  - Realtime Database:** Store and sync data with a cloud-hosted NoSQL database.
  - Cloud Firestore:** Store and sync your app data with this flexible, scalable NoSQL cloud-hosted database.
  - Cloud Storage for Firebase:** Store and retrieve large files like images, audio, and video.
  - Cloud Functions for Firebase:** Automatically run backend code in response to events triggered by Firebase features and HTTPS requests.
  - Firebase ML:** Machine learning models for vision, speech, and natural language.
- Bottom Bar:** Includes icons for Version Control, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, App Inspection, Services, and Layout Inspector.
- System Status:** Shows the system temperature (21°C), smoke detector status, and system time (10:50 PM, 11/6/2023).

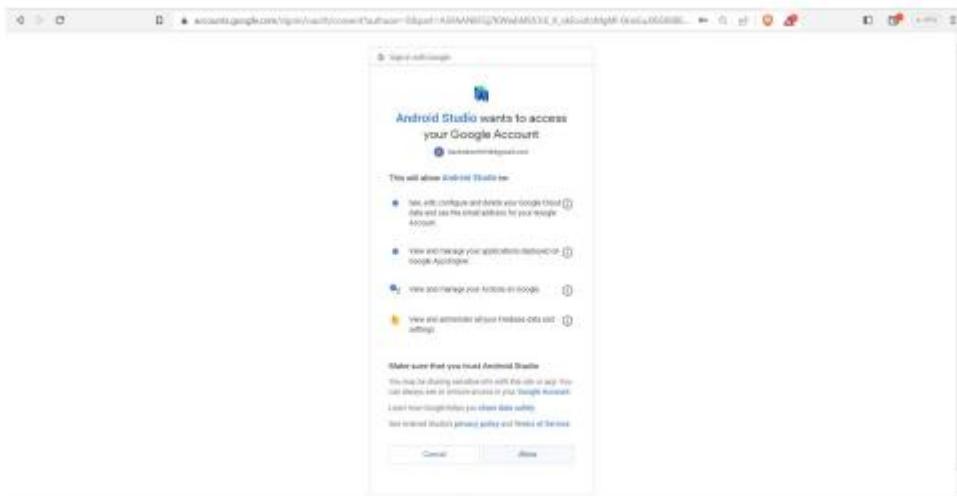
You got to follow some of these procedures



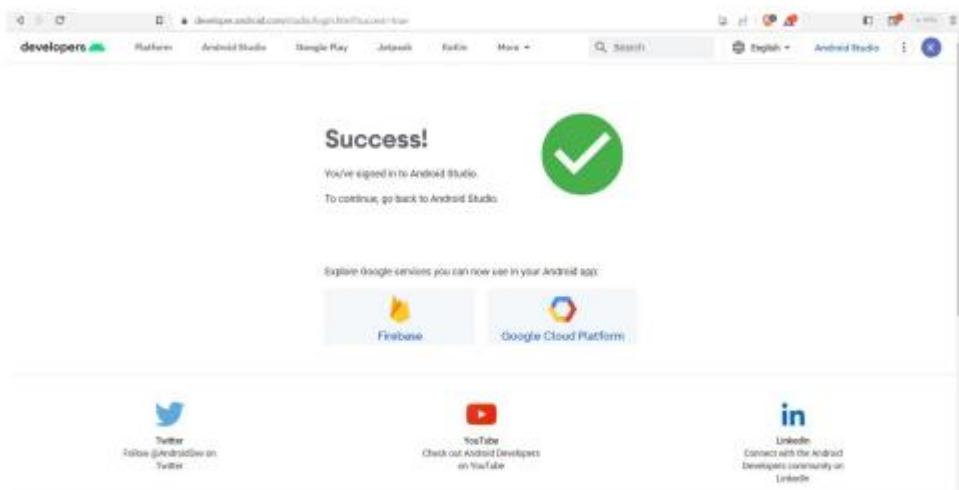
- After that Authenticate using Google will open



- In that Click on point one Connect To Firebase
- Now you will be redirected to Google sign-in page in your respective browser.
- After signing into your google account, below shown interface will appear



- Click on **Allow**



- Now, your Google account is connected to Android
- Now Click on **Firebase**.

- Now the Firebase page will open.



- Click on **continue**

The screenshot shows the 'Create a project (Step 2 of 3)' screen for a Google Analytics setup. The title is 'Google Analytics for your Firebase project'. A brief description explains that Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-app messaging, Remote Config, A/B Testing and Cloud Functions. Below this, there's a section titled 'Google Analytics services' with icons for AdMob, App Measurement, User segmentation and targeting across Firebase products, Crash-free assets, Event-based Cloud Functions trigger, and Free website reporting. A blue button labeled 'Enable Google Analytics for this project (Recommended)' is present. At the bottom, there are 'Previous' and 'Continue' buttons.

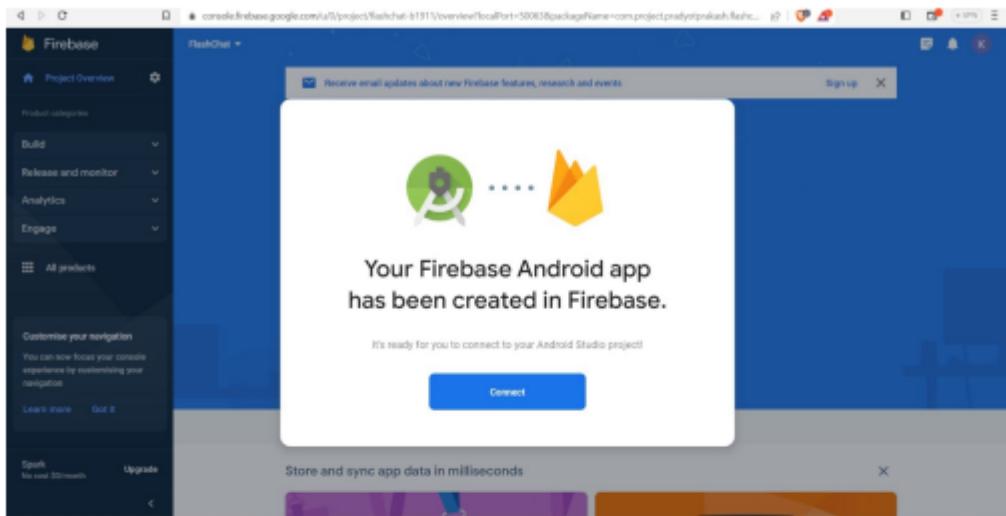
- Now enable the Google Analytics for this project and
- Click on **continue**

The screenshot shows the 'Create a project (Step 3 of 3)' screen for configuring Google Analytics. The title is 'Configure Google Analytics'. It asks if you want to create a Google Analytics account or select an existing one. A dropdown menu shows 'India' selected. A note states that Google Analytics is a business tool. There are checkboxes for 'Use the default settings for sharing Google Analytics data' (which includes sharing with Google products and linking with Google Analytics data from other Google accounts) and 'I accept the Google Analytics terms'. A note below explains that accepting terms means allowing Google to collect and use data from the project. At the bottom, there are 'Previous' and 'Create project' buttons.

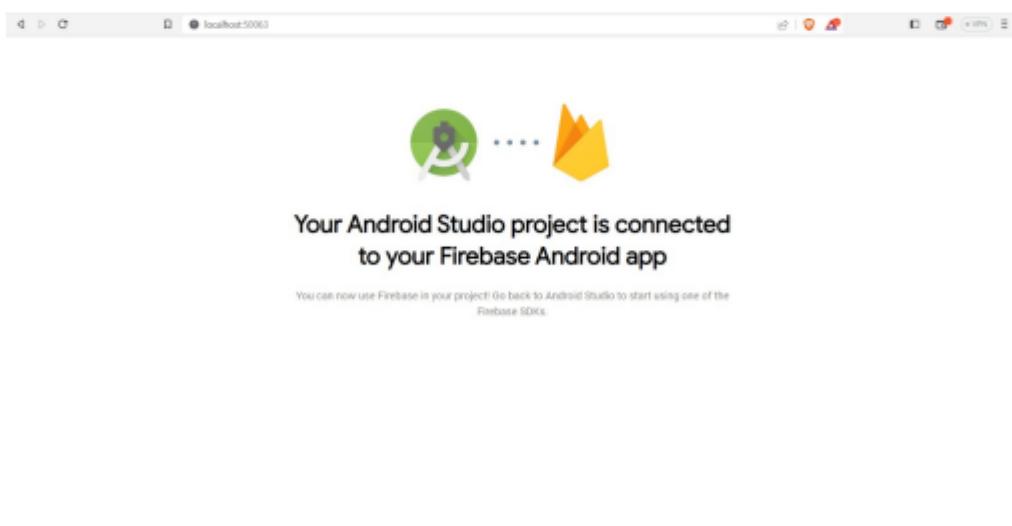
- Set Analytics location to India and check the boxes.
- Now Click on **Create project**

The screenshot shows the final step of creating a project. It displays a circular logo for 'FlashChild' and a message stating 'Your new project is ready.' A blue 'Continue' button is at the bottom. An illustration of a person sitting on a large blue gear is visible on the right side of the page.

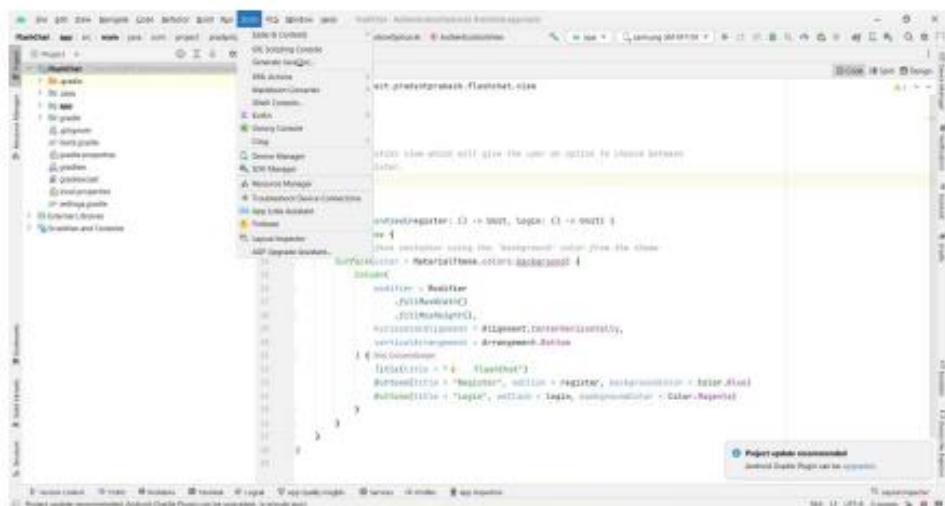
- Click on **Continue**



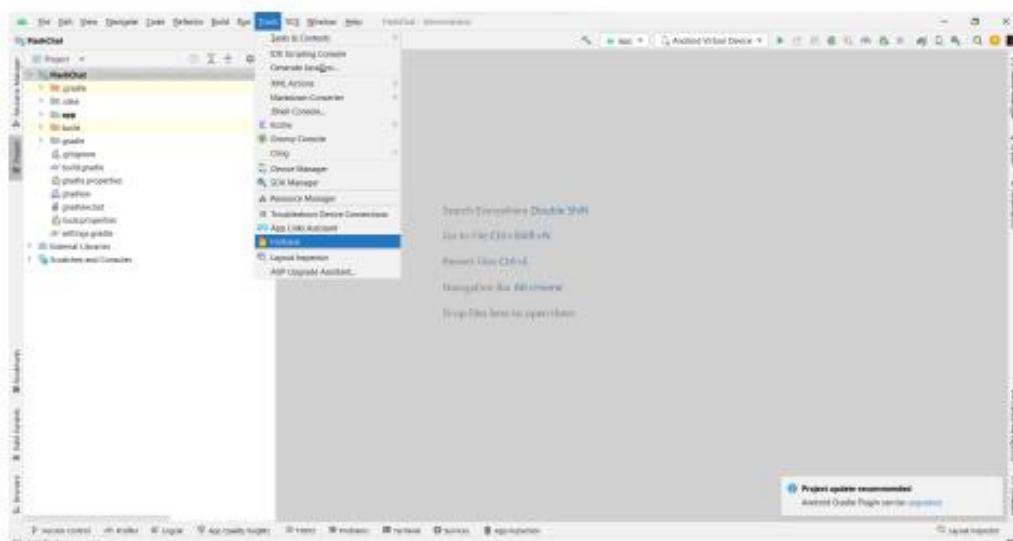
- Click on **Connect**



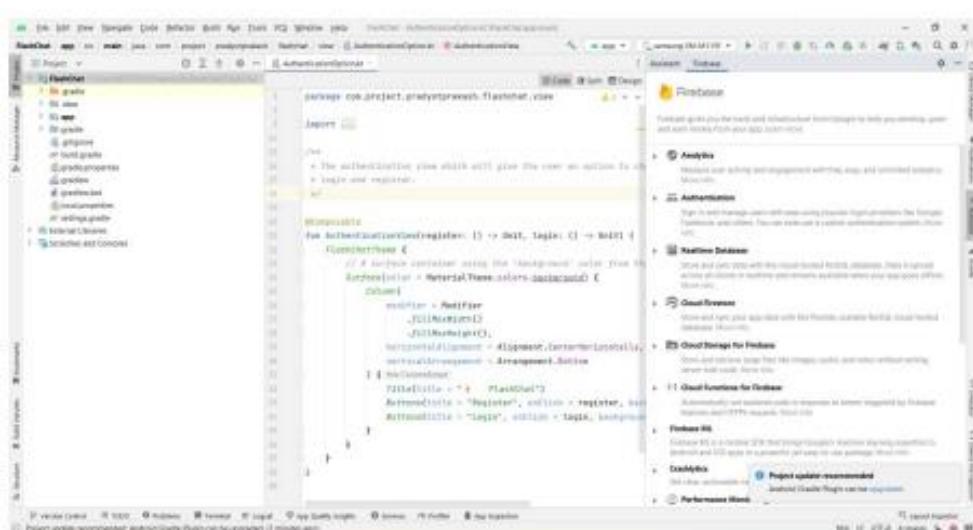
- This shows that your project is connected to Firebase.



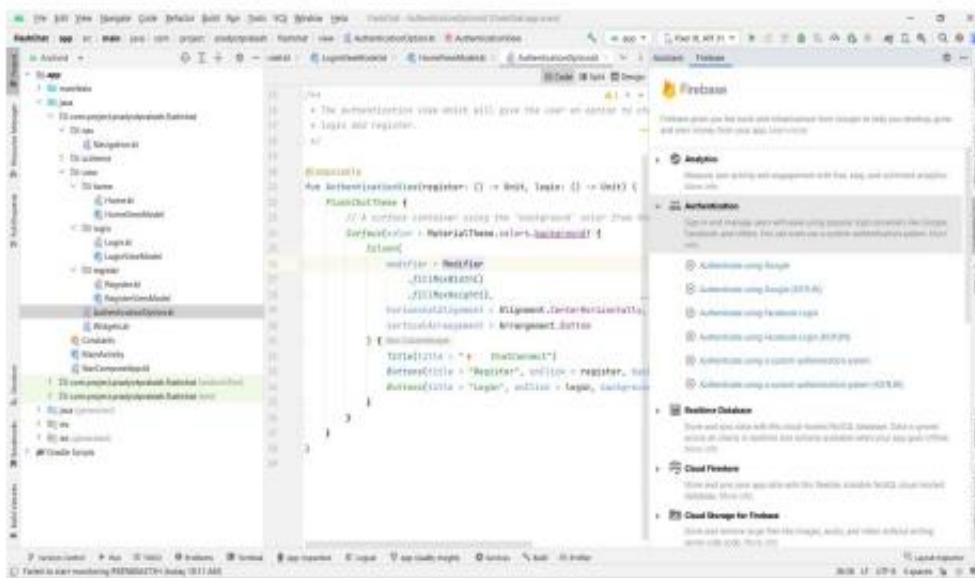
- Menu bar > Tools > Firebase.



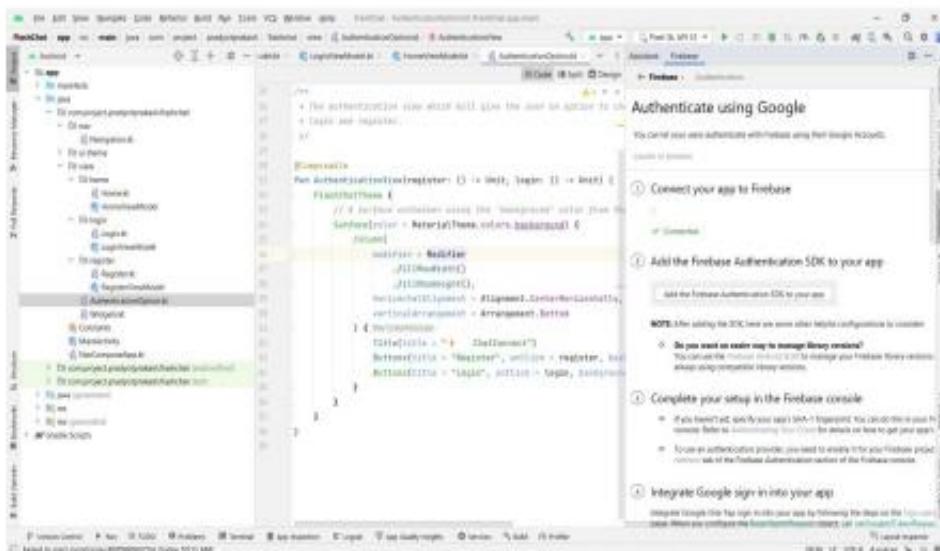
- After clicking the Firebase , the Firebase tab will open.



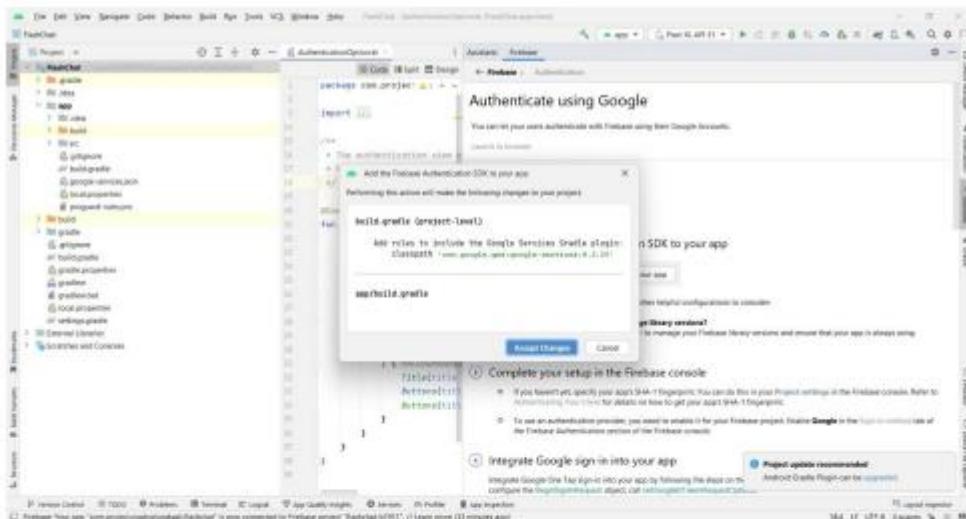
- In Authentication > Go to Authenticate using Google.



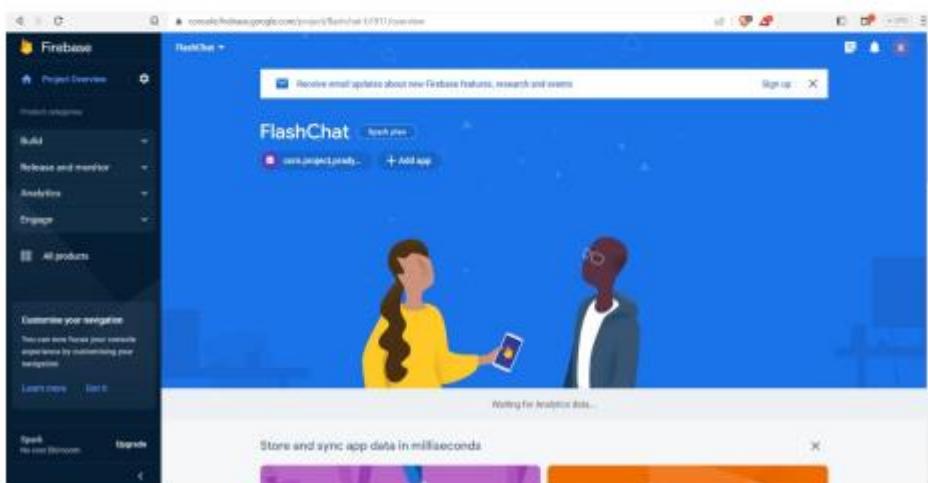
- After that, Authentication using Google will open.



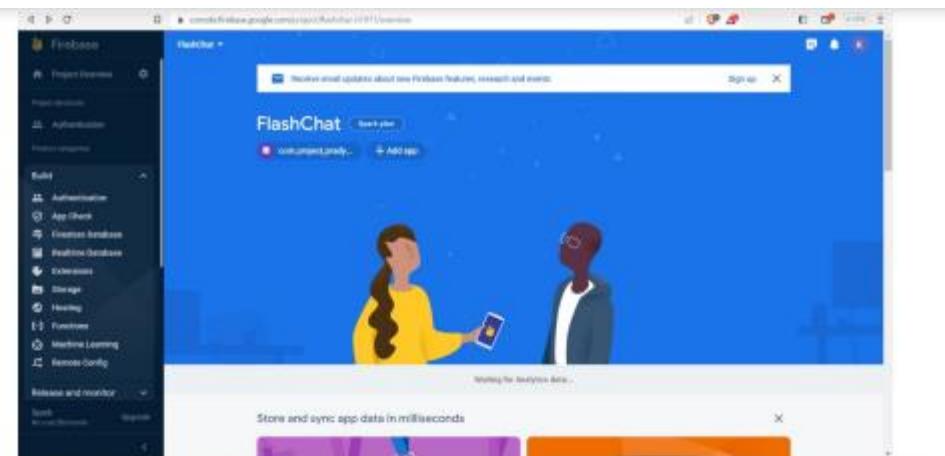
- Now you can see that it shows **Connected**.
  - Now click on the second point "Add the Firebase Authentication SDK to your app".



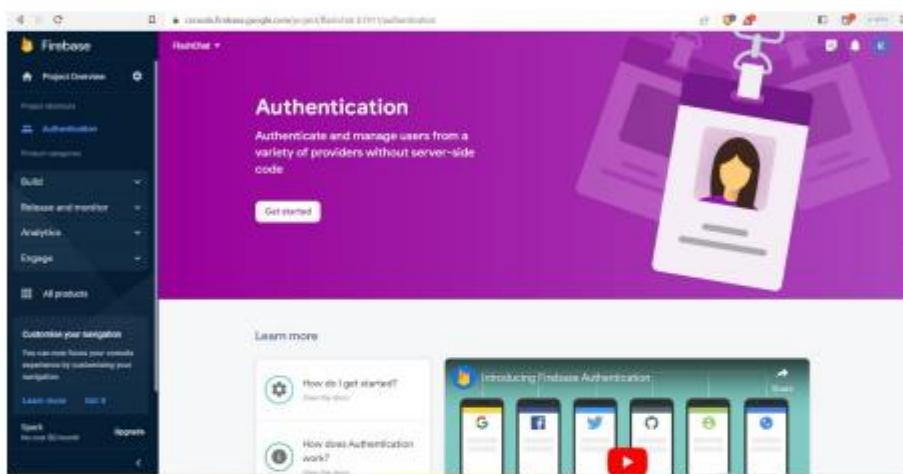
- Now a small tab will open and Click on **Accept Changes**.



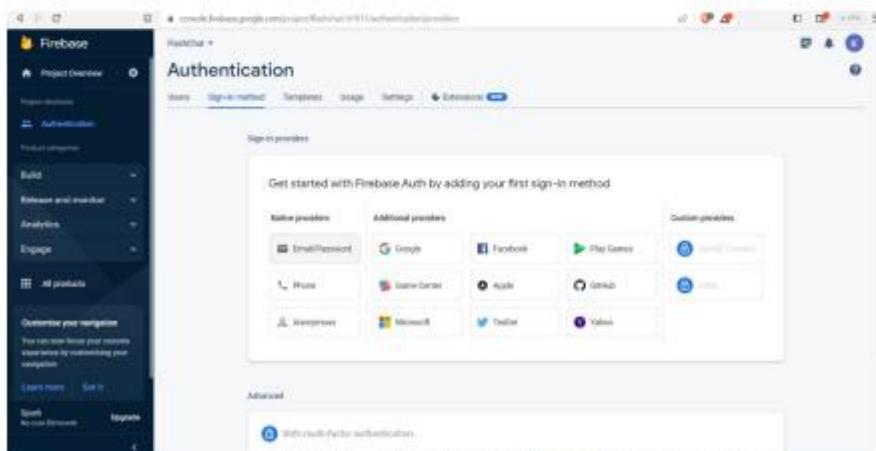
- Now go to your respective browser and search firebase or <https://firebase.google.com/>
- Open Firebase official website



- In the right side panel Click on **Build** < Click on **Authentication**.

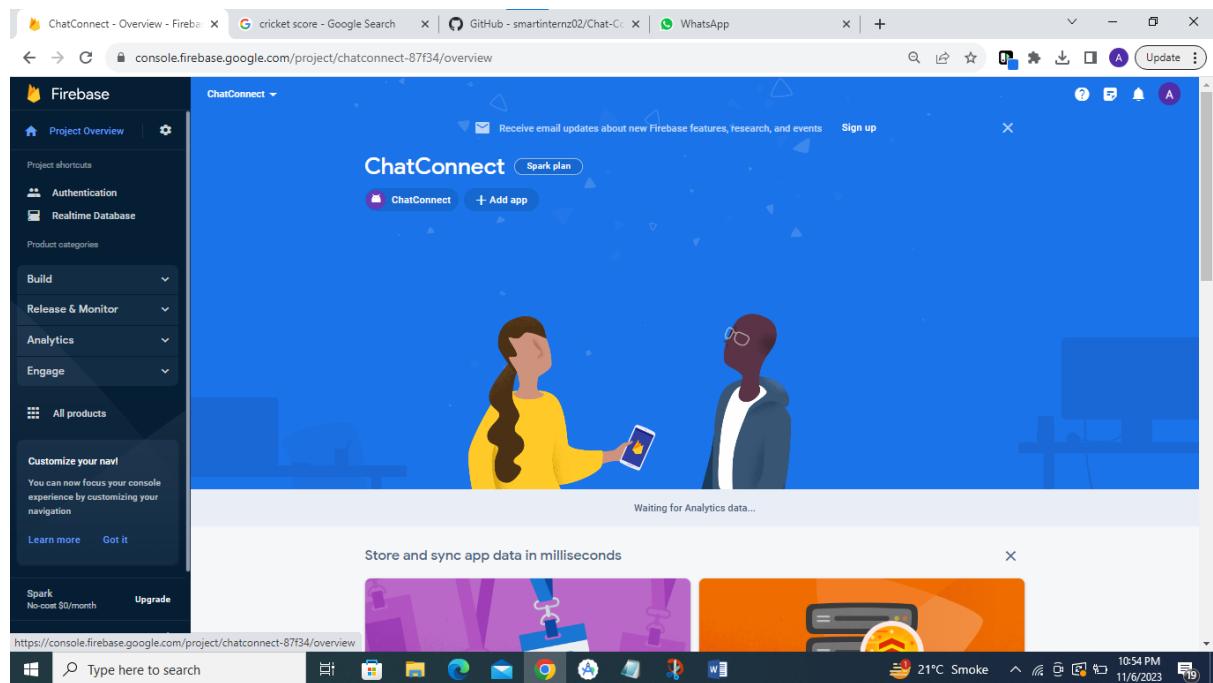


- Click on **Get started**.

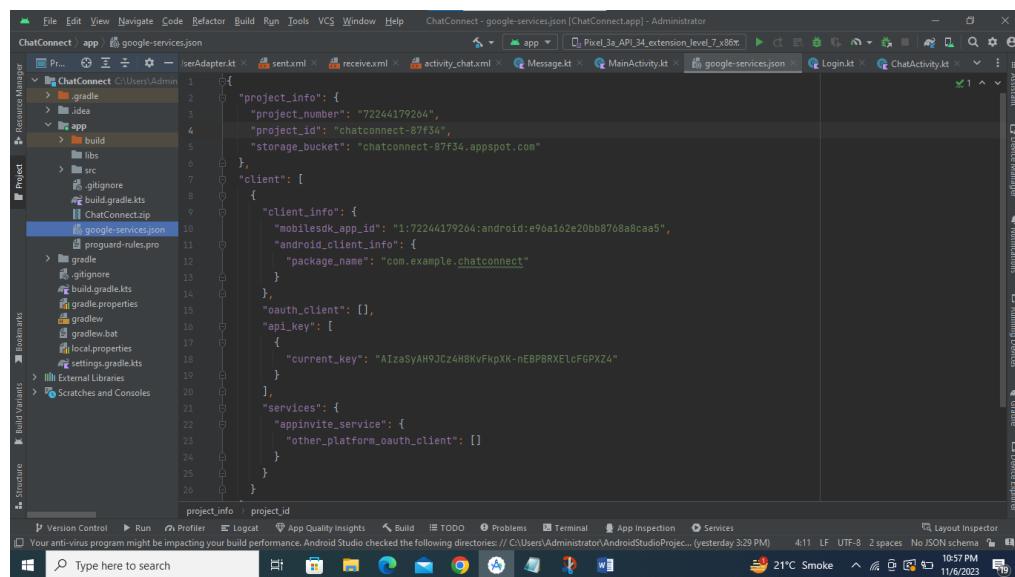


- Now the above page will open for **Authentication**.
- Go to **Sign-in-method**.
- Click on **Email/Password**.

After following those procedures we have firebase ready for our app

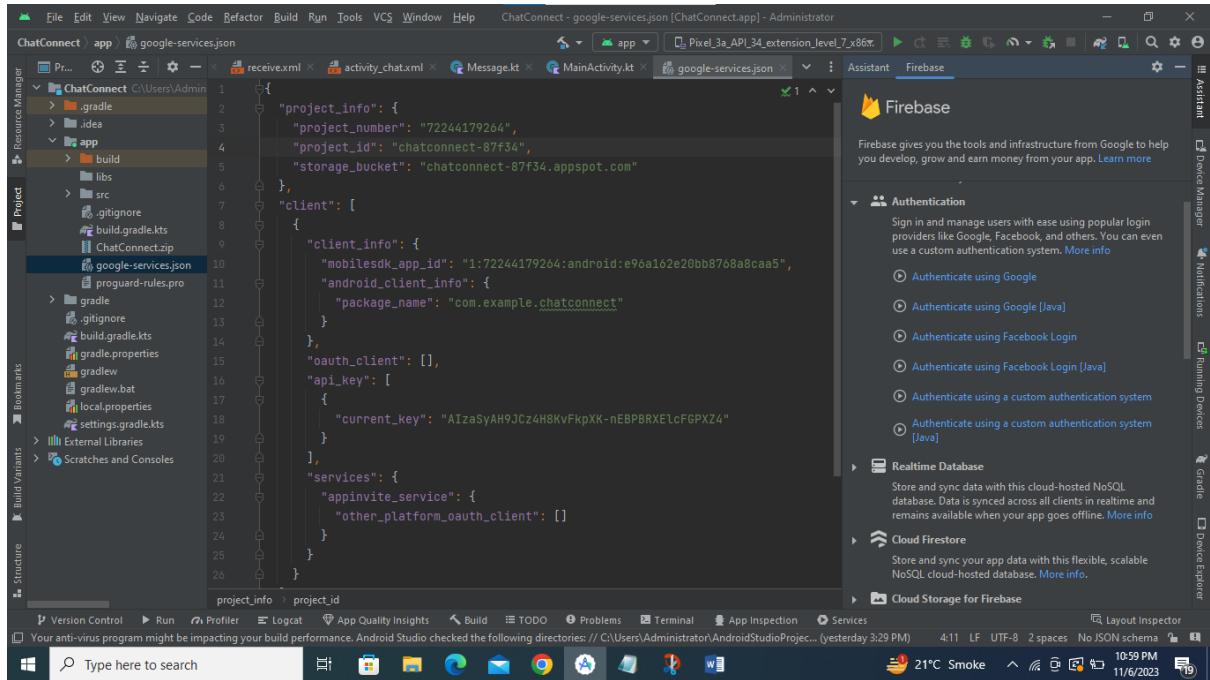


Before this we gotta add some files into our app that is like  
One is google.services.json file to our project we show it below

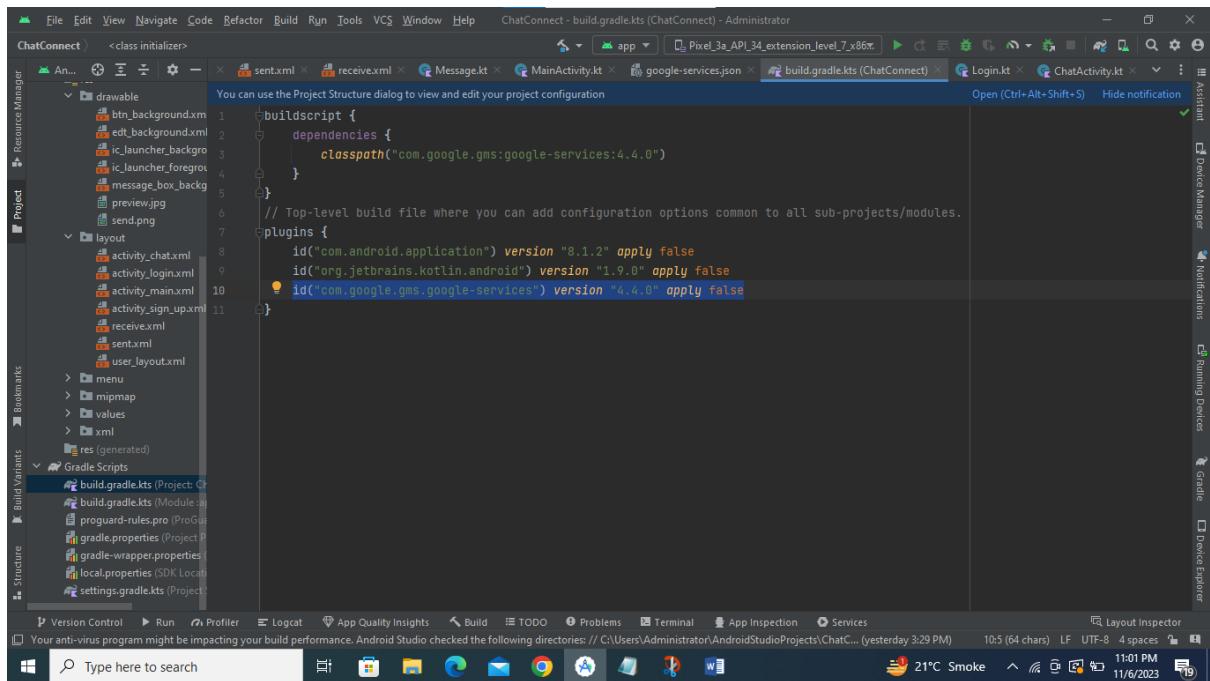


This is like one of the main file we got to copy-paste In our app.

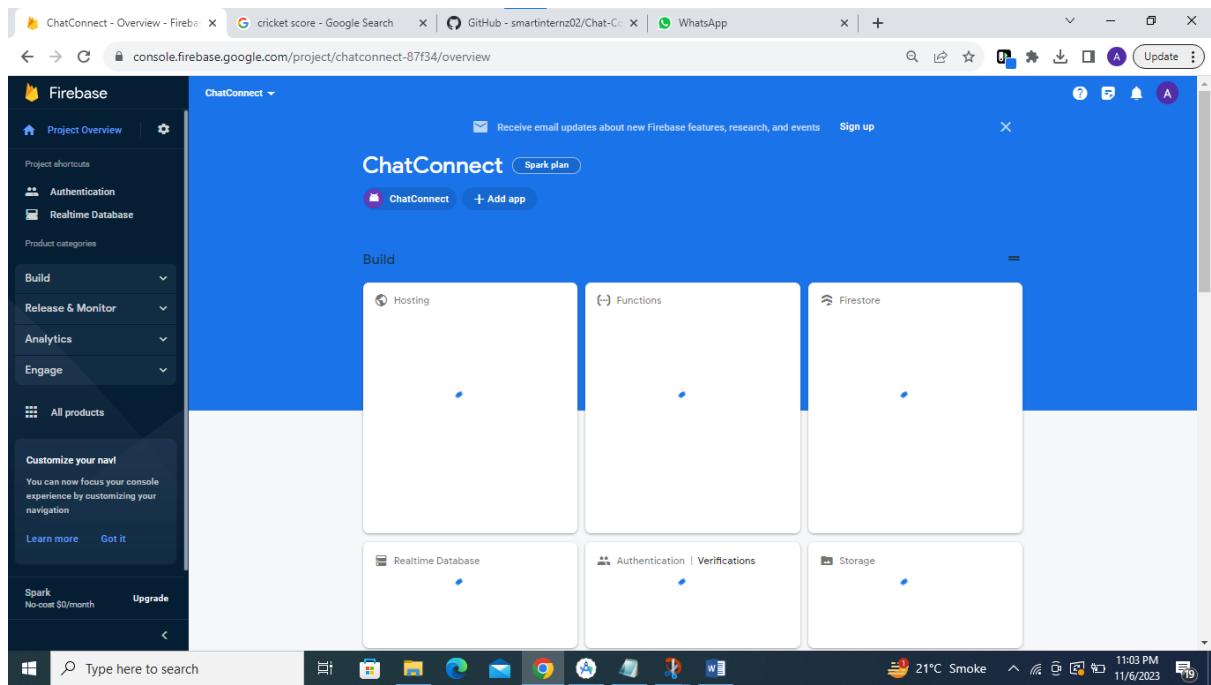
**After this we got to paste gradle things to our app**



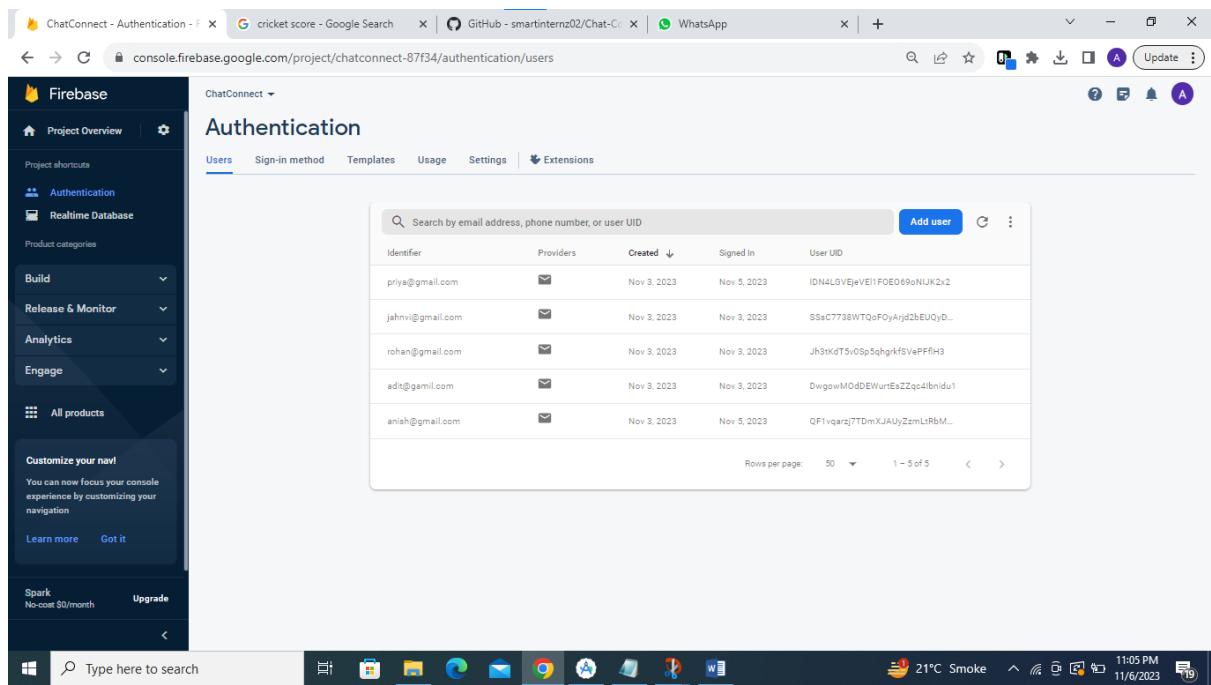
**We pasted this in our gradle the on highlighted**

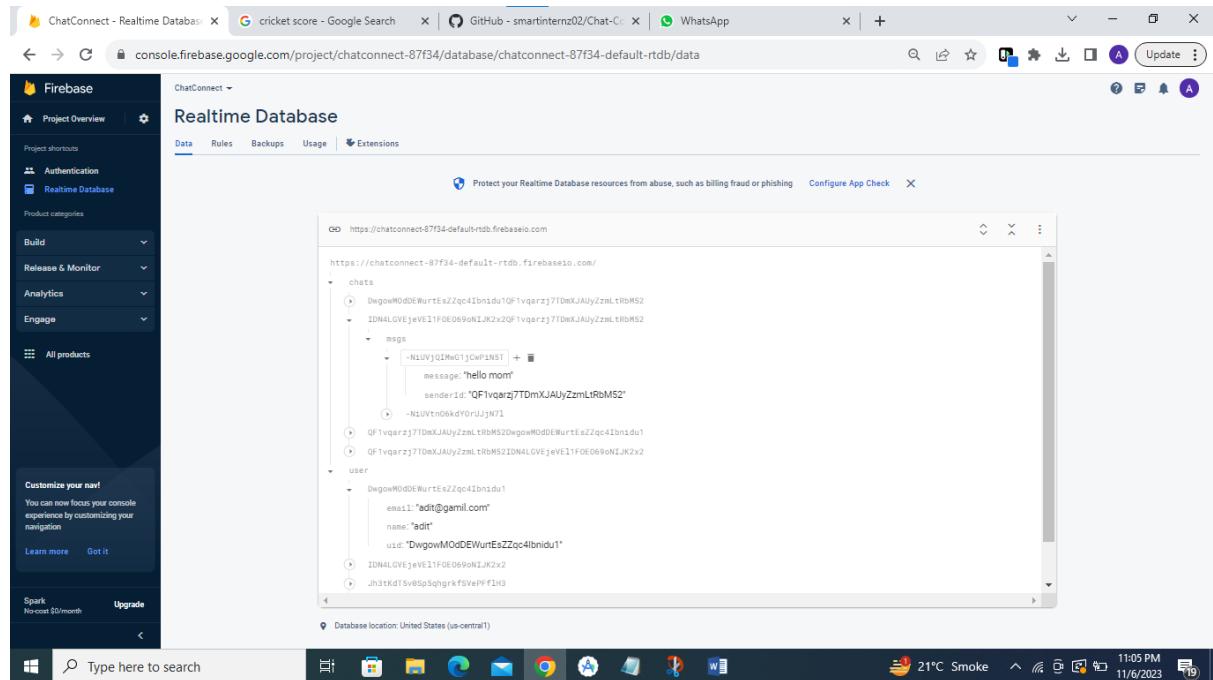


**This our app's firebase overview**



We personally have created authentication and Realtime database





**As you can see above we have made uid for each user and each message they send**

## Task 4:

## MainActivity.kt file

The screenshot shows the Android Studio interface with the following details:

- File menu:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project Structure:** Shows the project tree under "ChatConnect".
- MainActivity.kt:** The current file is `MainActivity.kt`, which contains the following code:

```
package com.example.chatconnect

import ...

class MainActivity : AppCompatActivity() {

    private lateinit var userRecyclerView: RecyclerView
    private lateinit var userList: ArrayList<User>
    private lateinit var adapter: UserAdapter
    private lateinit var mAuth: FirebaseAuth
    private lateinit var mDbRef: DatabaseReference

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

        mAuth = FirebaseAuth.getInstance()
        mDbRef = FirebaseDatabase.getInstance().getReference()

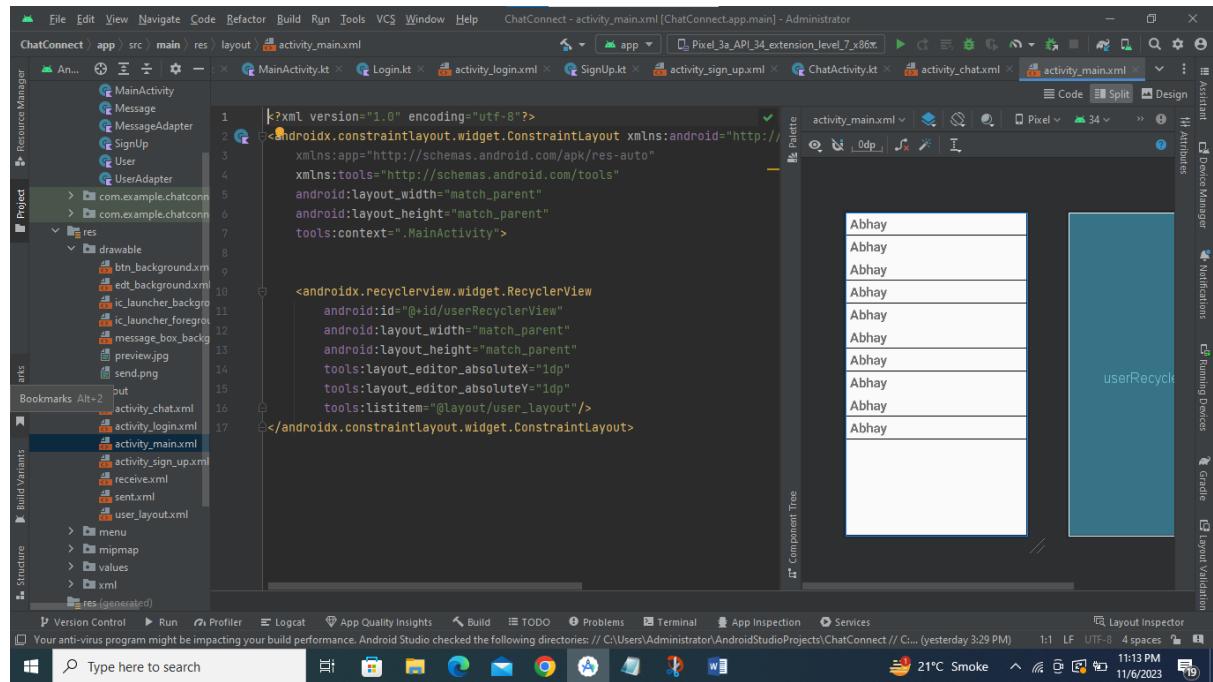
        userList = ArrayList()
        adapter = UserAdapter( context: this, userList)

        userRecyclerView = findViewById(R.id.userRecyclerView)

        userRecyclerView.layoutManager = LinearLayoutManager( context: this)
        userRecyclerView.adapter = adapter
    }
}
```

- Bottom navigation bar:** Version Control, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, App Inspection, Services.
- Bottom status bar:** Your anti-virus program might be impacting your build performance. Android Studio checked the following directories: ... (yesterday 3:29 PM), 80.2 LF, UTB-8, 4 spaces, 21°C Smoke, 11:07 PM, 11/6/2023.

## Xml file for main activity

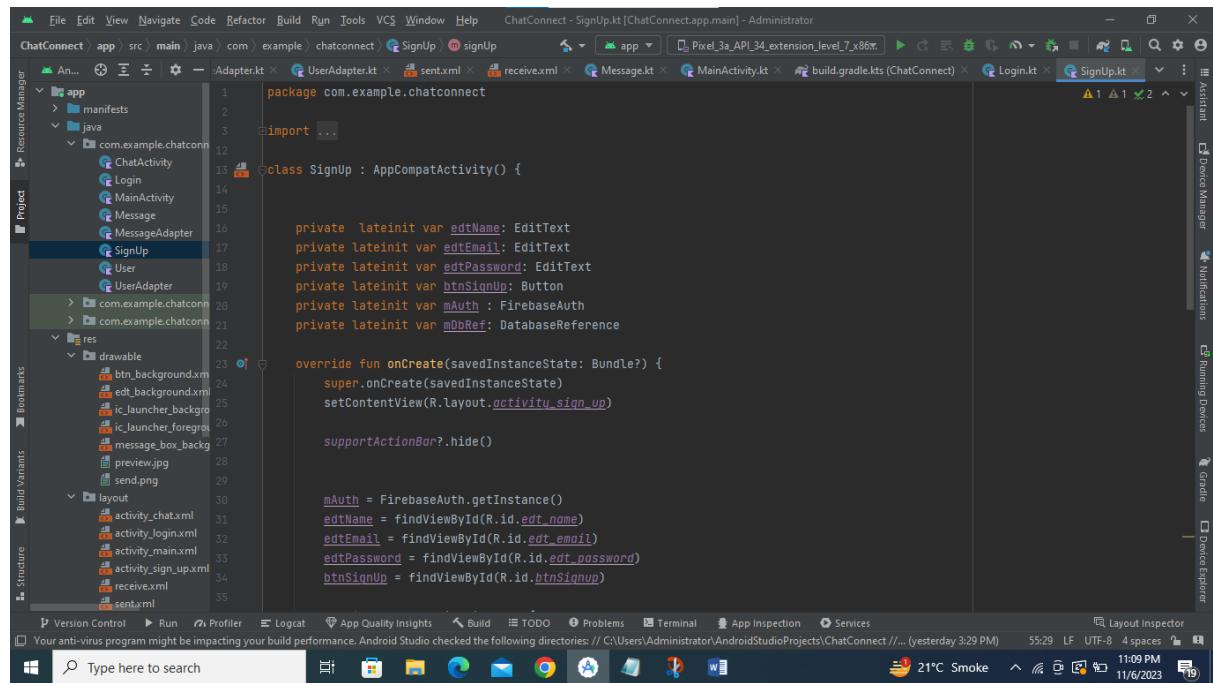


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/userRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:layout_editor_absoluteX="1dp"
        tools:layout_editor_absoluteY="1dp"
        tools:listitem="@layout/user_layout"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

After main activity we made login and signup page for our app

## Signup.kt



```
package com.example.chatconnect

import ...

class SignUp : AppCompatActivity() {

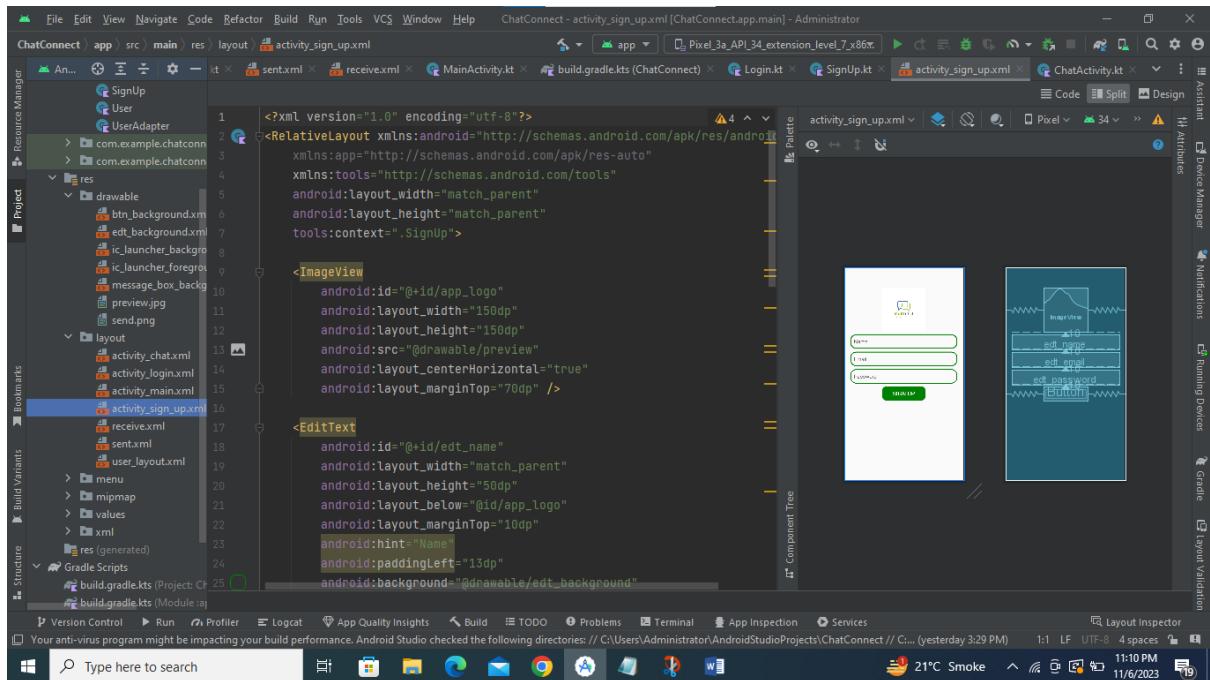
    private lateinit var edtName: EditText
    private lateinit var edtEmail: EditText
    private lateinit var edtPassword: EditText
    private lateinit var btnSignUp: Button
    private lateinit var mAuth : FirebaseAuth
    private lateinit var mDbRef : DatabaseReference

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

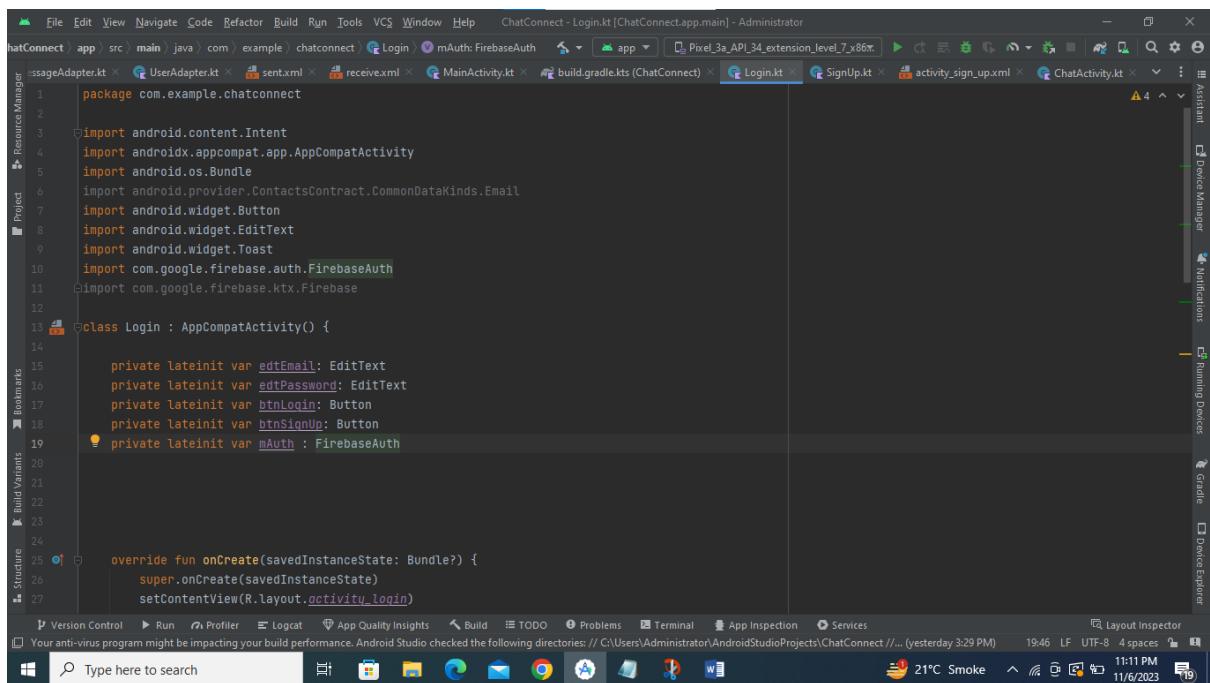
        supportActionBar?.hide()

        mAuth = FirebaseAuth.getInstance()
        edtName = findViewById(R.id.edt_name)
        edtEmail = findViewById(R.id.edt_email)
        edtPassword = findViewById(R.id.edt_password)
        btnSignUp = findViewById(R.id.btnSignup)
    }
}
```

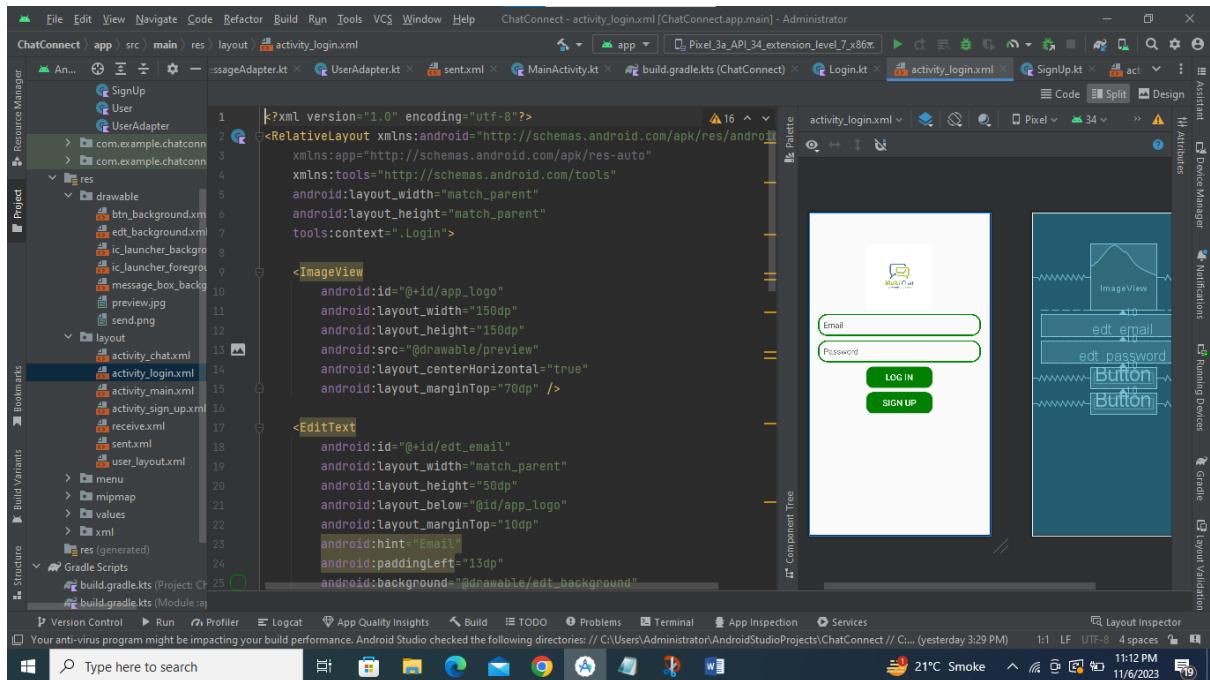
Our xml file for it



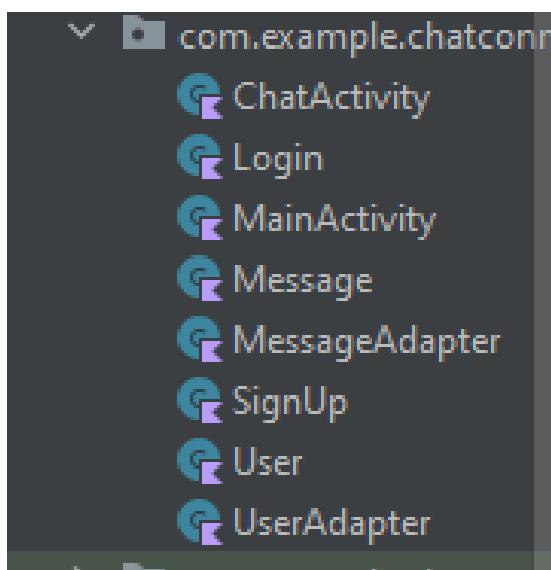
## For log in page now



## Xml file foe login page



After this we made some more of our files like these:



We will put screen shot of all with their kotlin and xml file below

## Message.kt

The screenshot shows the Android Studio interface with the 'Message.kt' file open in the main editor. The code defines a class 'Message' with two constructors: one taking no parameters and another taking a 'message' and a 'senderId'. The project structure on the left shows files like LoginActivity, SignUpActivity, MainActivity, and various adapter classes. The bottom status bar indicates it's 11:15 PM, 21°C, and the device is a Smoke.

```
package com.example.chatconnect

class Message {
    var message: String? = null
    var senderId: String? = null

    constructor()

    constructor(message: String?, senderId: String?) {
        this.message = message
        this.senderId = senderId
    }
}
```

## Chat activity

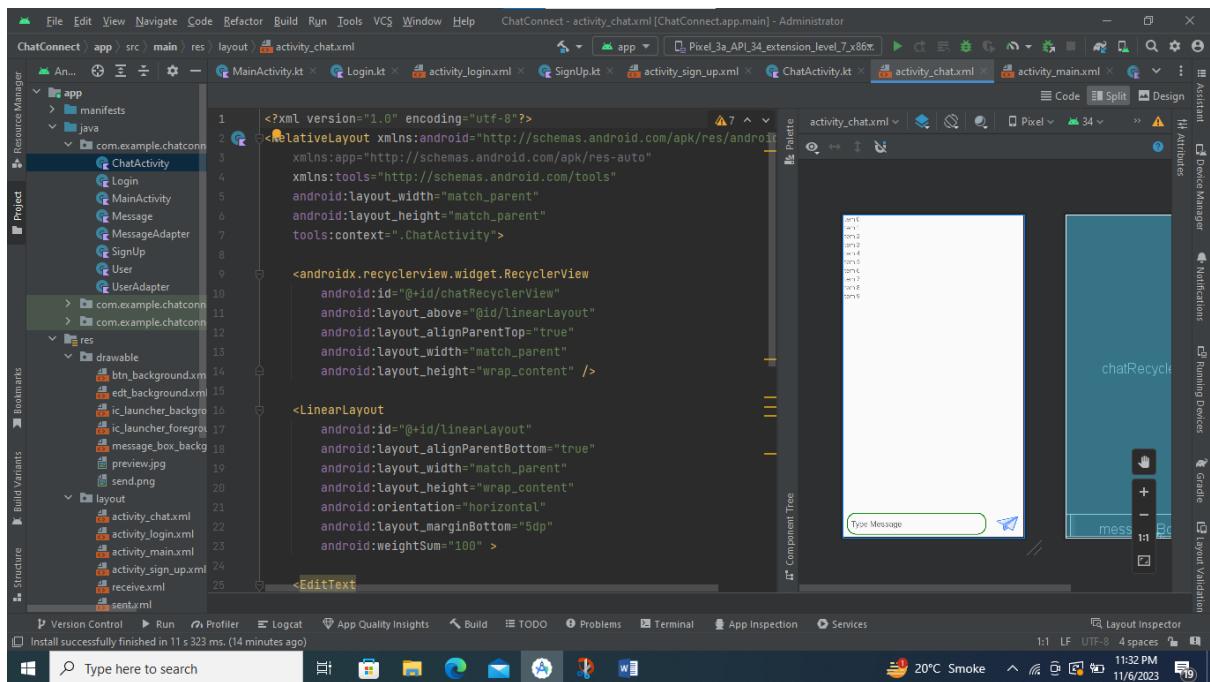
The screenshot shows the Android Studio interface with the 'ChatActivity.kt' file open in the main editor. The code defines a class 'ChatActivity' that extends 'AppCompatActivity'. It imports various Android and Firebase components. The project structure on the left shows files like LoginActivity, SignUpActivity, MainActivity, and various adapter classes. The bottom status bar indicates it's 21:46, 20°C, and the device is a Smoke.

```
package com.example.chatconnect

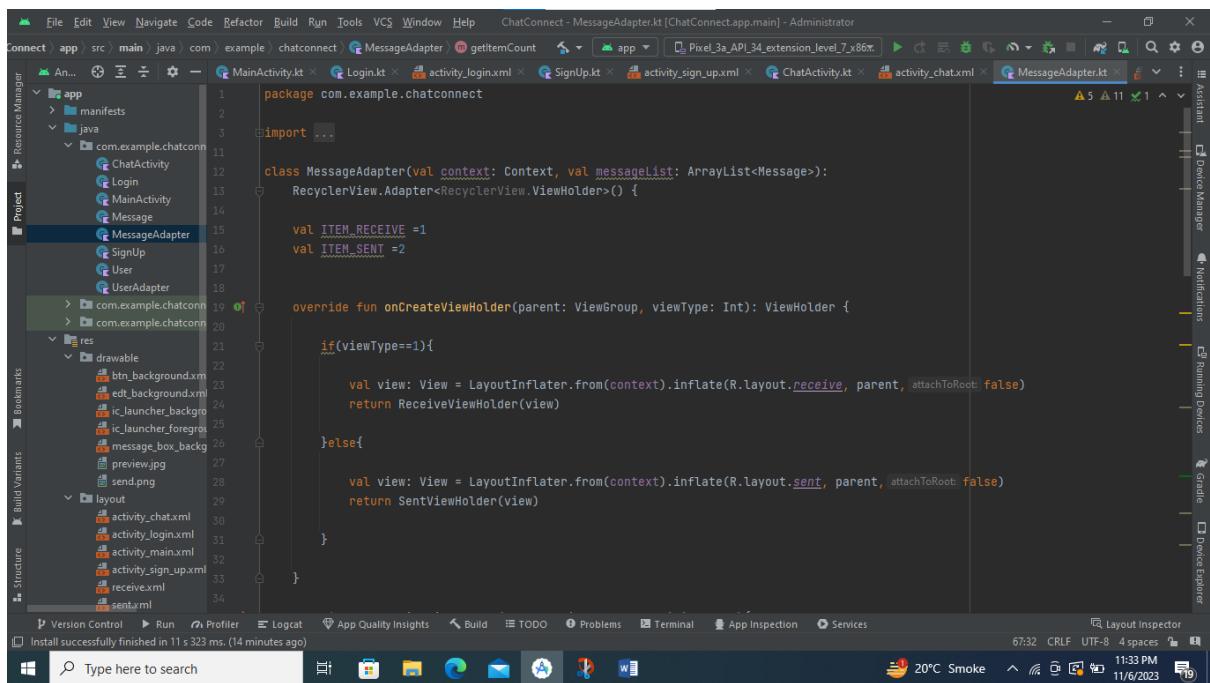
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.EditText
import android.widget.ImageView
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class ChatActivity : AppCompatActivity() {

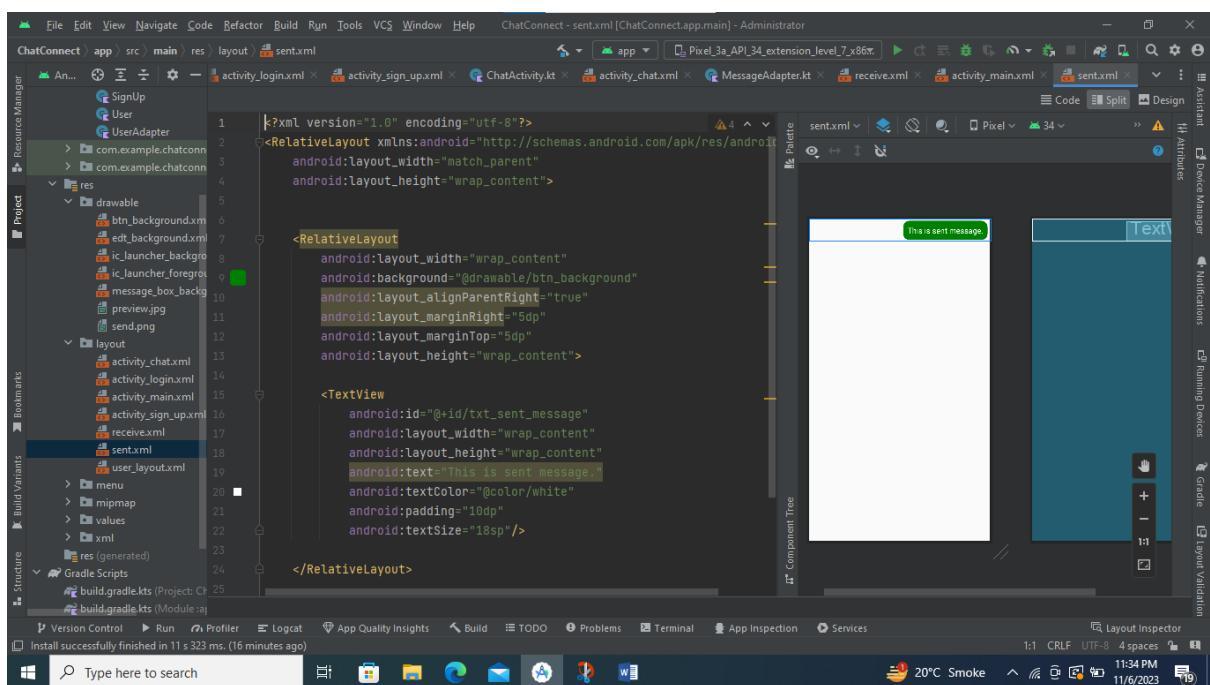
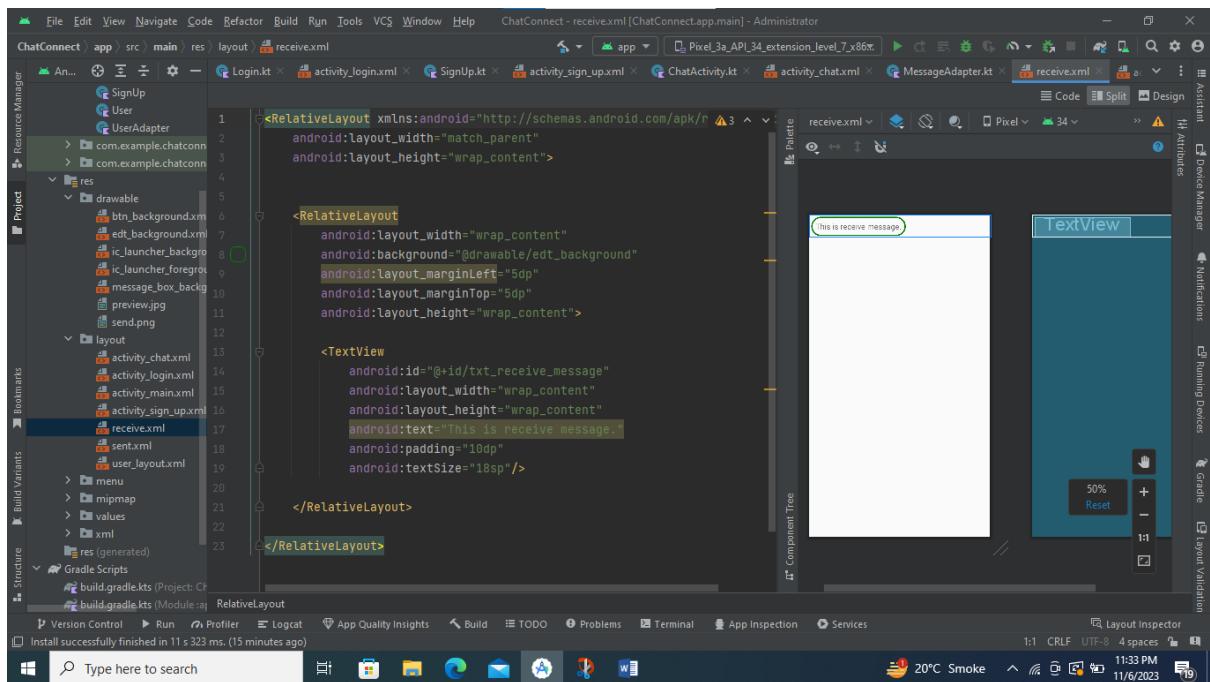
    private lateinit var chatRecyclerView: RecyclerView
    private lateinit var messageBox: EditText
    private lateinit var sendButton: ImageView
    private lateinit var messageAdapter: MessageAdapter
    private lateinit var messageList: ArrayList<com.example.chatconnect.Message>
    private lateinit var mDbRef : DatabaseReference
}
```



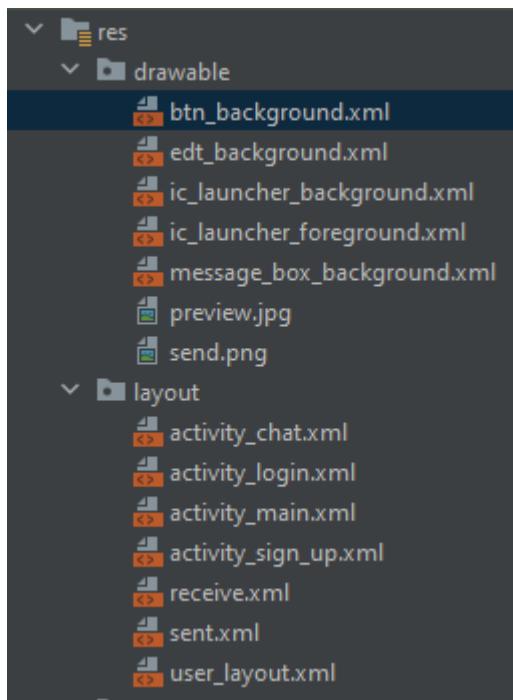
## Message adapter



## Receive and sent xml file



**These are some layout and drawable we use**



## This is our firebase project settings

A screenshot of the Firebase Project Settings page for 'Your project'. The page shows basic project information: Project name (ChatConnect), Project ID (chatconnect-87f34), Project number (72244179264), Default GCP resource location (Not yet selected), and Web API Key (AlzaSyAH9JCz4H8KvFkpXK-nEBPBRXEIcFGPXZ4). Below this, there's an 'Environment' section with a note about customizing for app lifecycle stages, and an 'Environment type' set to 'Unspecified'.

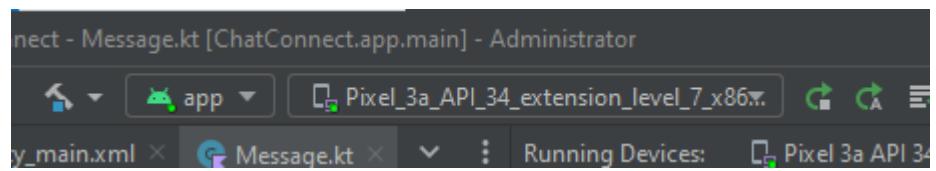
For concluding that we have made this project.

Running the application.

Step 1: Run apps on a hardware device

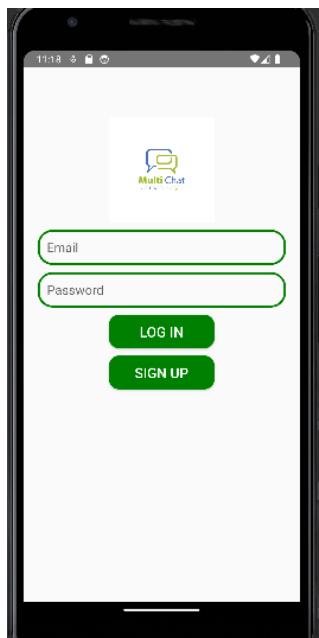
<https://developer.android.com/studio/run/device>

Step 2: Run the application in Mobile

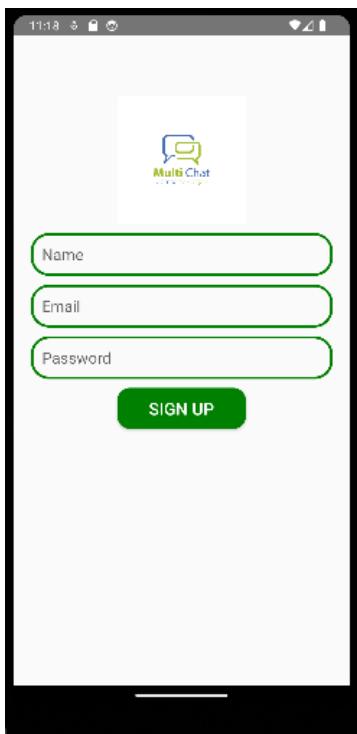


Final Output of the Application:

Login page



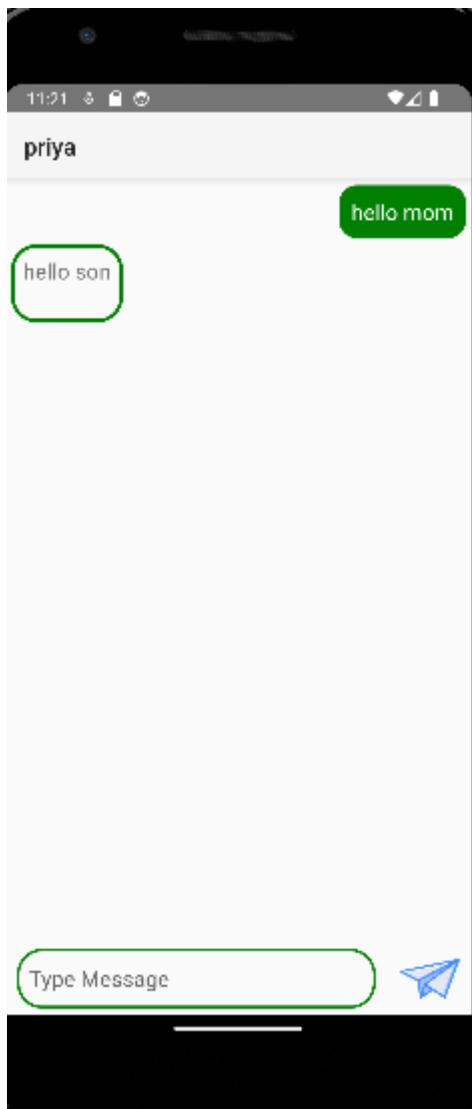
Sign up page



**Main page**



**Sender and receiver chat**



## Our successful compiler ss

```
Build: Build Output × Build Analyzer ×
Build ChatConnect: finished At 11/6/2023 11:18 PM 1 min, 0 sec, 540 ms
  ↴ Download info

> Task :app:mergeDebugJavaResource UP-TO-DATE
> Task :app:checkDebugDuplicateClasses UP-TO-DATE
> Task :app:desugarDebugFileDependencies UP-TO-DATE
> Task :app:mergeExtDexDebug UP-TO-DATE
> Task :app:mergeLibDexDebug UP-TO-DATE
> Task :app:dexBuilderDebug UP-TO-DATE
> Task :app:mergeProjectDexDebug UP-TO-DATE
> Task :app:mergeDebugJniLibFolders UP-TO-DATE
> Task :app:mergeDebugNativeLibs NO-SOURCE
> Task :app:stripDebugDebugSymbols NO-SOURCE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:writeDebugAppMetadata UP-TO-DATE
> Task :app:writeDebugSigningConfigVersions UP-TO-DATE
> Task :app:packageDebug UP-TO-DATE
> Task :app:createDebugApkListingFileRedirect UP-TO-DATE
> Task :app:assembleDebug UP-TO-DATE

BUILD SUCCESSFUL in 59s
33 actionable tasks: 33 up-to-date

Build Analyzer results available
```

A screenshot of the Android Studio Build Output window. It shows a successful build for the 'Build ChatConnect' task. The log output is as follows:  
> Task :app:mergeDebugJavaResource UP-TO-DATE  
> Task :app:checkDebugDuplicateClasses UP-TO-DATE  
> Task :app:desugarDebugFileDependencies UP-TO-DATE  
> Task :app:mergeExtDexDebug UP-TO-DATE  
> Task :app:mergeLibDexDebug UP-TO-DATE  
> Task :app:dexBuilderDebug UP-TO-DATE  
> Task :app:mergeProjectDexDebug UP-TO-DATE  
> Task :app:mergeDebugJniLibFolders UP-TO-DATE  
> Task :app:mergeDebugNativeLibs NO-SOURCE  
> Task :app:stripDebugDebugSymbols NO-SOURCE  
> Task :app:validateSigningDebug UP-TO-DATE  
> Task :app:writeDebugAppMetadata UP-TO-DATE  
> Task :app:writeDebugSigningConfigVersions UP-TO-DATE  
> Task :app:packageDebug UP-TO-DATE  
> Task :app:createDebugApkListingFileRedirect UP-TO-DATE  
> Task :app:assembleDebug UP-TO-DATE  
  
BUILD SUCCESSFUL in 59s  
33 actionable tasks: 33 up-to-date  
  
Build Analyzer results available

