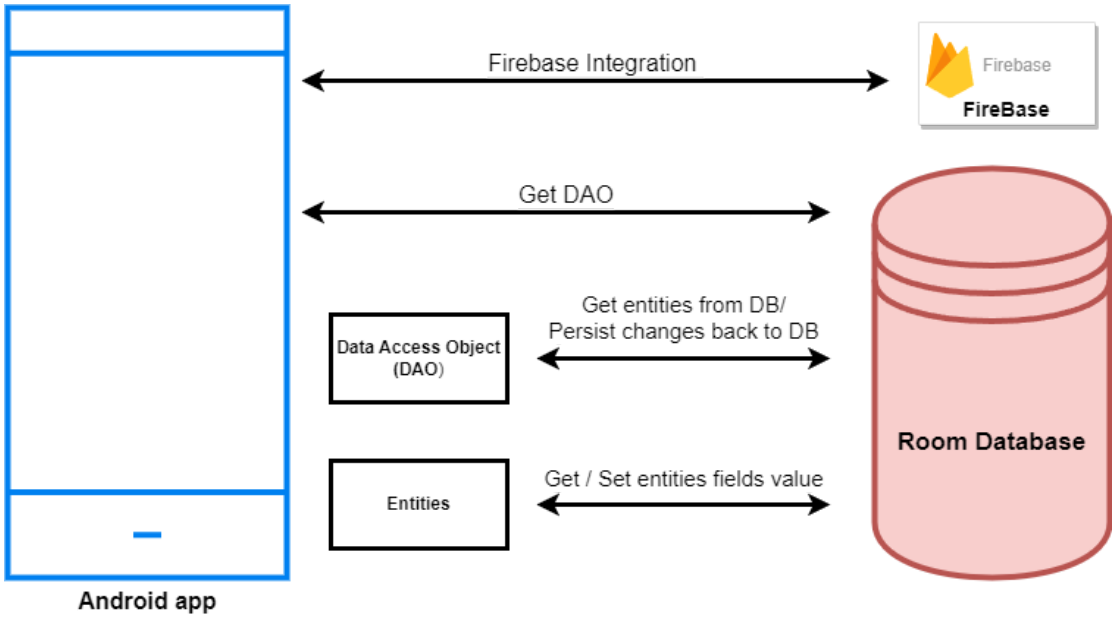
ChatConnect - A Real-Time Chat and Communication App

Project Based Experiential Learning Program

ChatConnect - A Real-Time Chat and Communication App

ChatConnect 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.

Architecture



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:

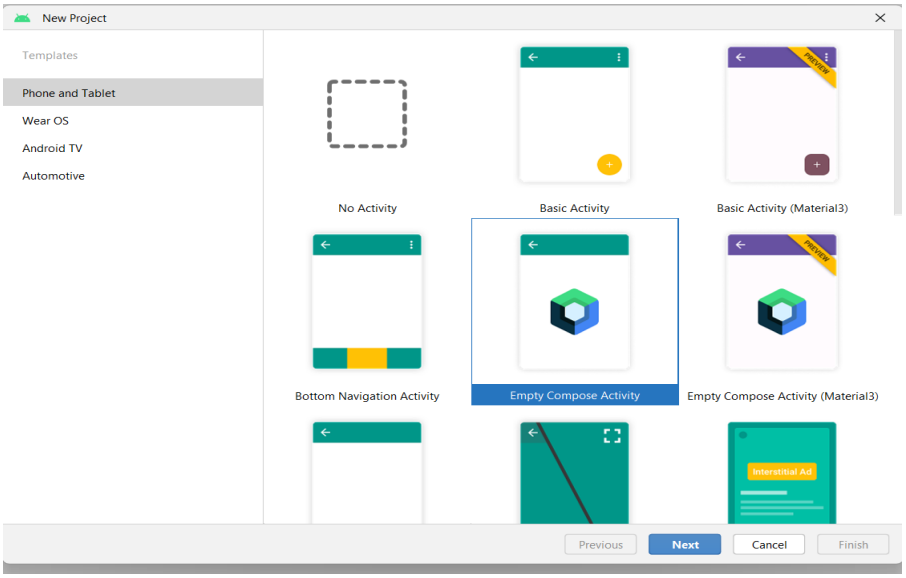
1.Required initial steps
2.Creating a new project
3.Integrating Firebase and Authentication
4.Creating UI files
5.Running the application.

Task 1:

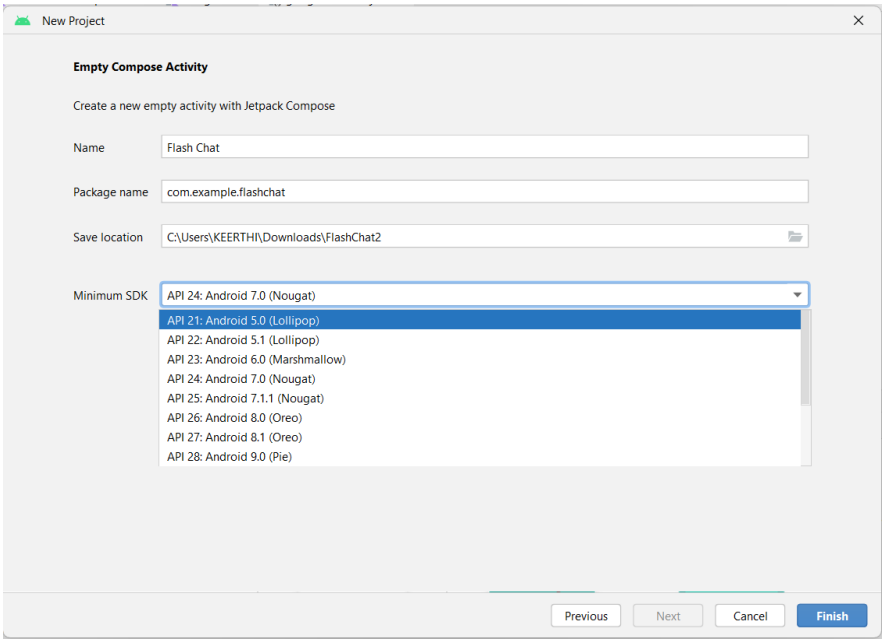
Required initial steps :
https://developer.android.com/studio/install

Task 2 :

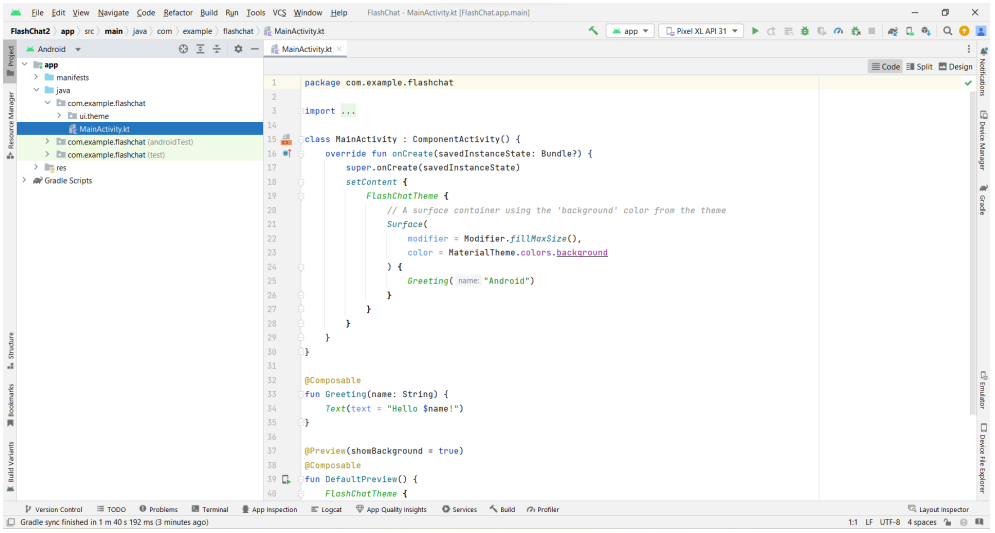
Creating a new project.
Step 1 : Android studio > File > New > New Project > Empty Compose Activity
Step 2 : Click on **Next** button.



Step 3 : Give name to the new project.
Step 4 : Give the Minimum SDK value
Step 5 : Click Finish



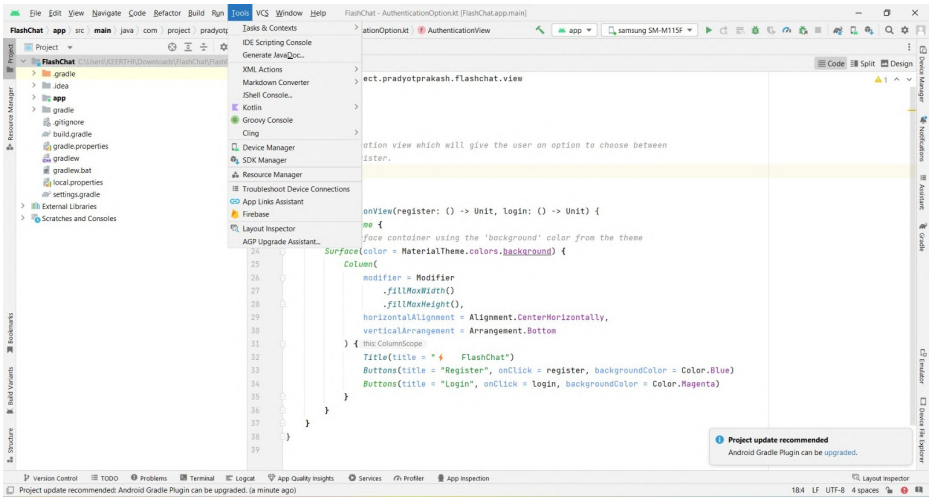
Main activity file

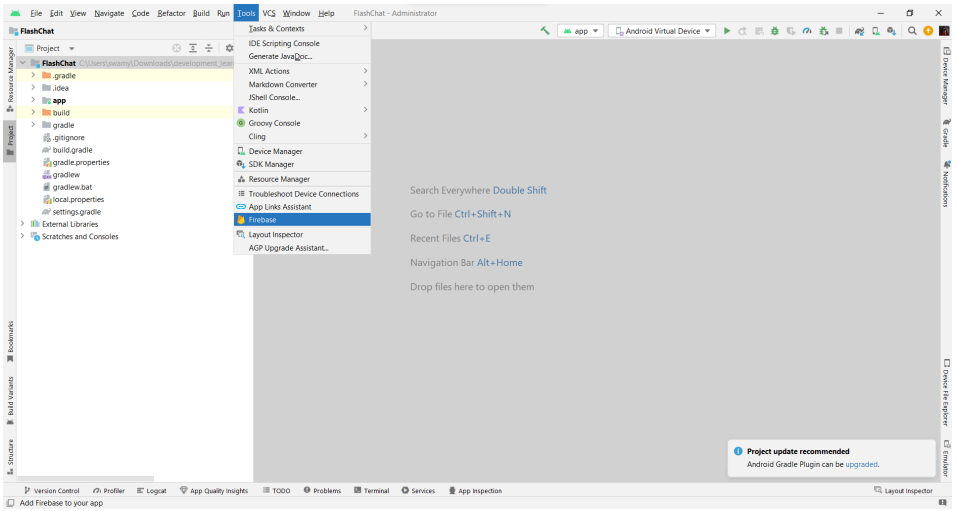


Task 3 :

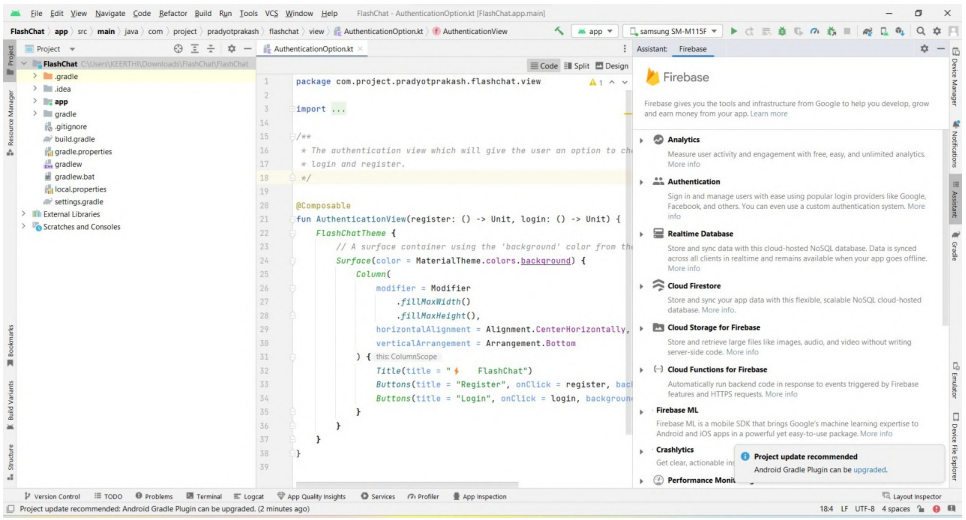
Integrating Firebase

- Menu bar > Tools > Firebase.

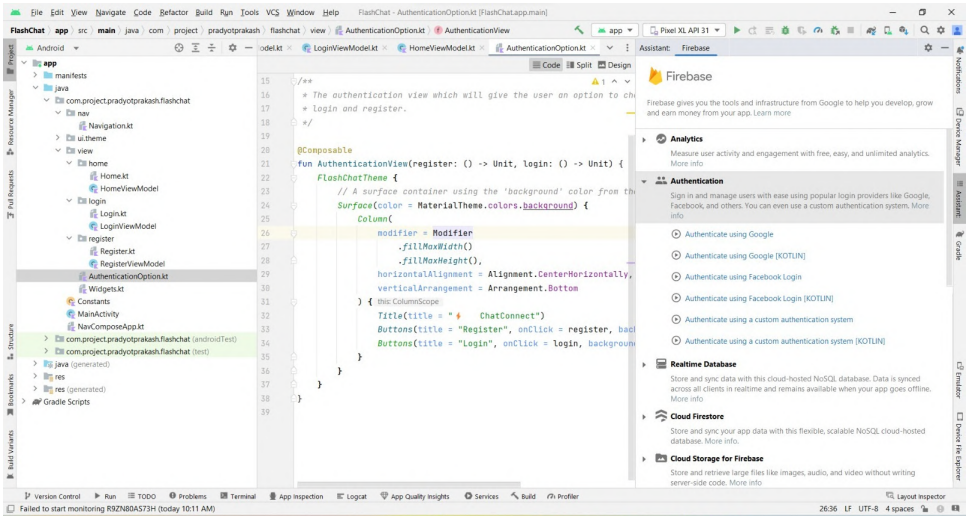




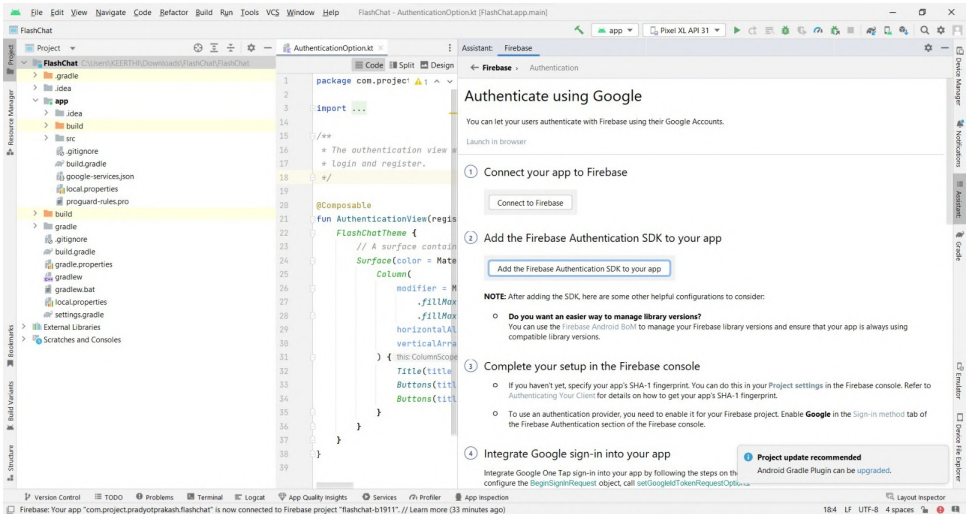
- After clicking the Firebase , Firebase tab will open.



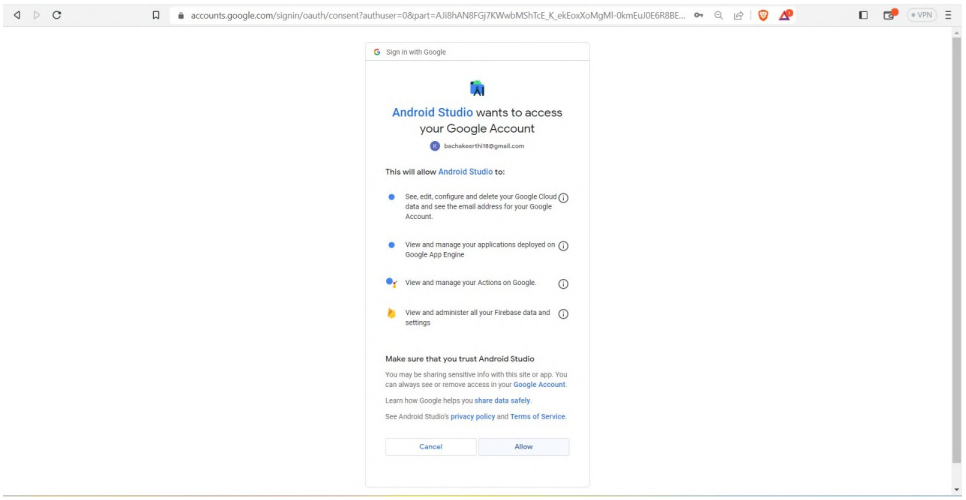
- In Authentication > Go to Authenticate using Google.



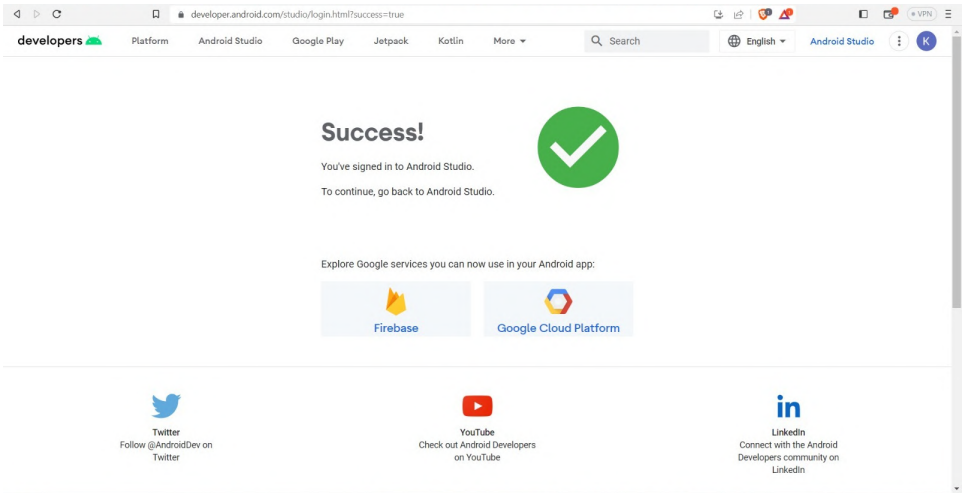
- After that Authenticate using Google will open



- 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

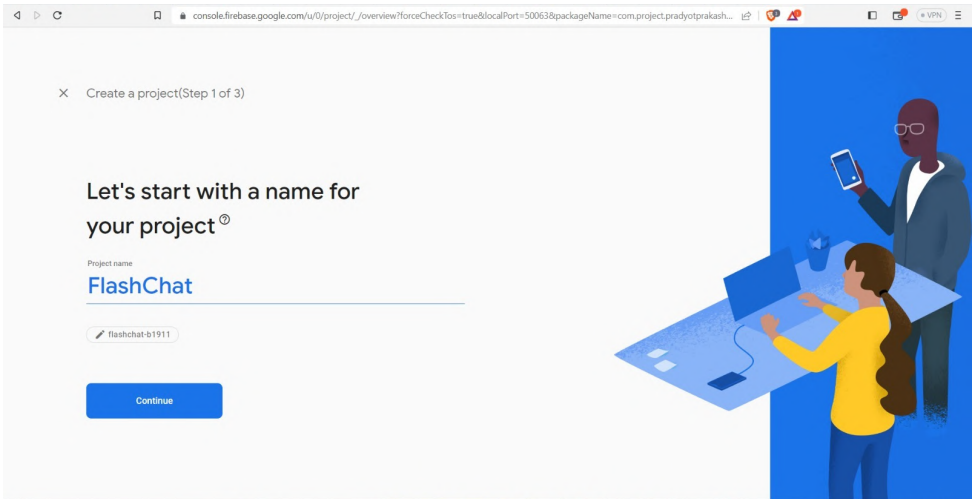


- Click on **Allow**

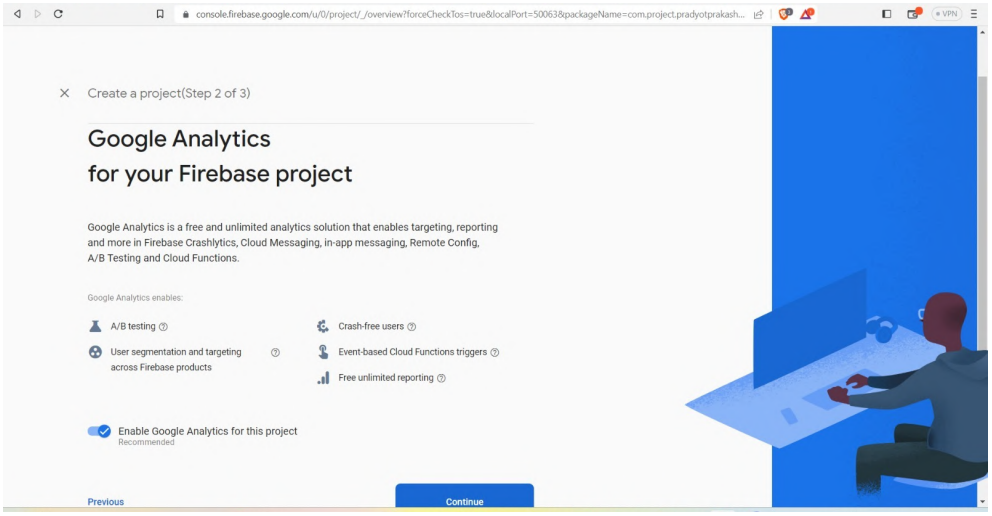


- Now, your Google account is connected to Android
- Now Click on **Firebase**.

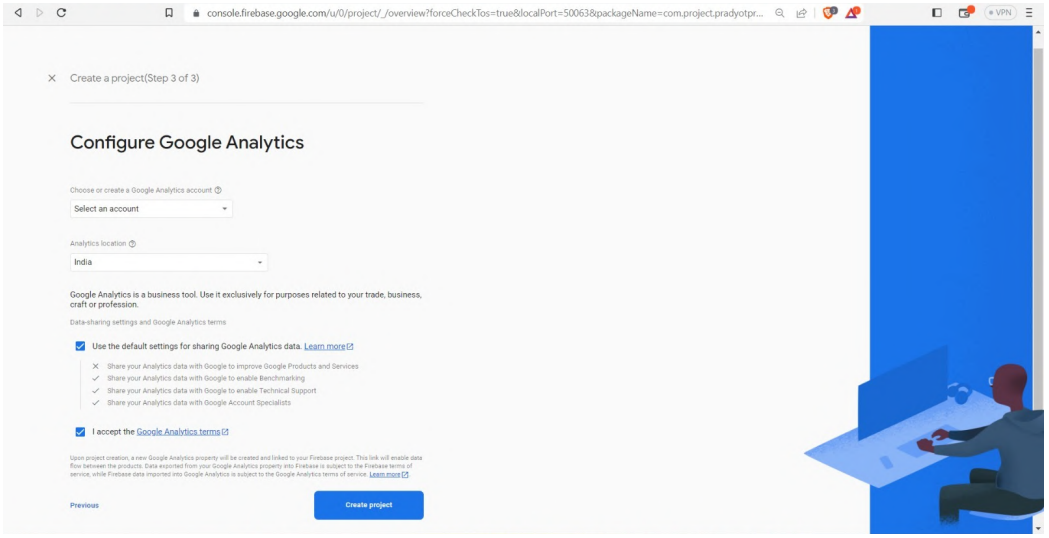
- Now the Firebase page will open.



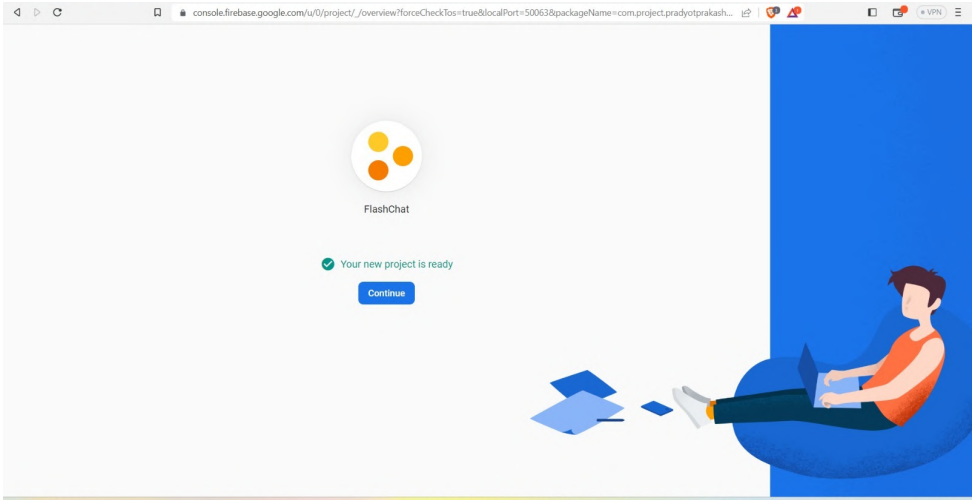
- Click on **continue**



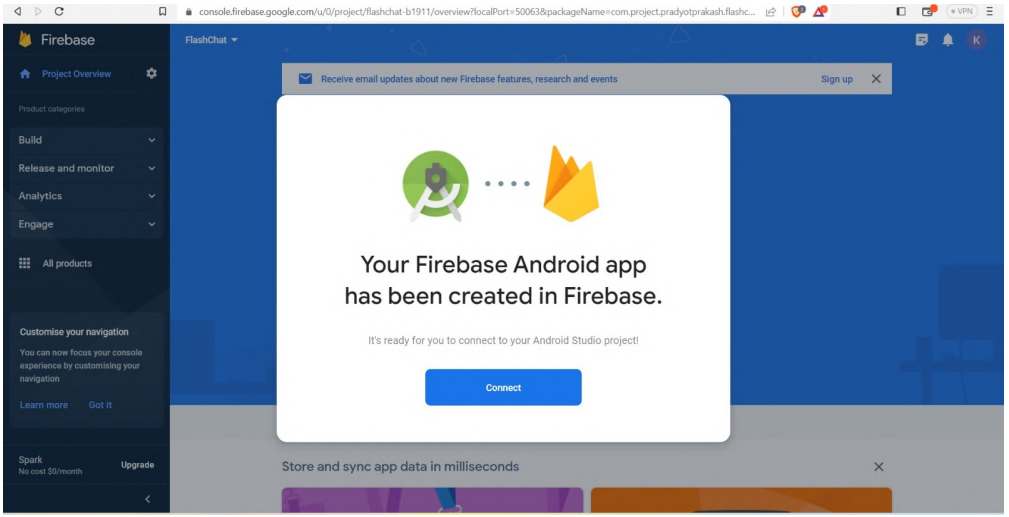
- Now enable the Google Analytics for this project and
- Click on **continue**



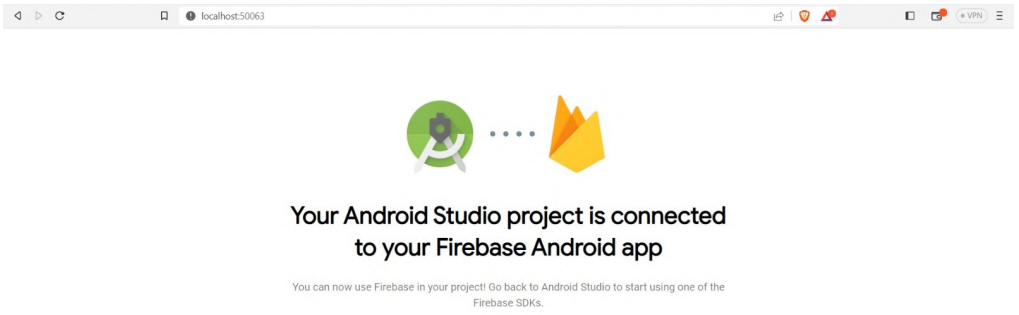
- Set Analytics location to India and check the boxes.
- Now Click on **Create project**



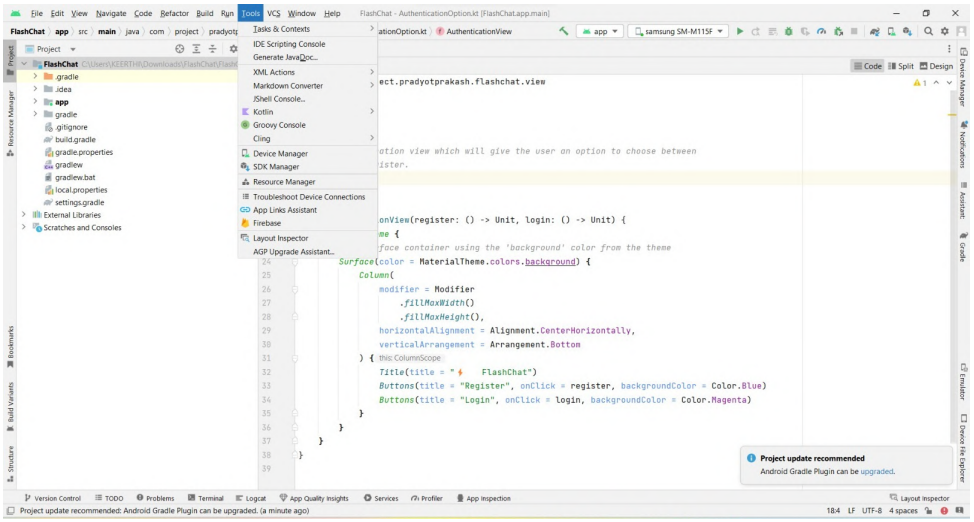
- Click on **Continue**



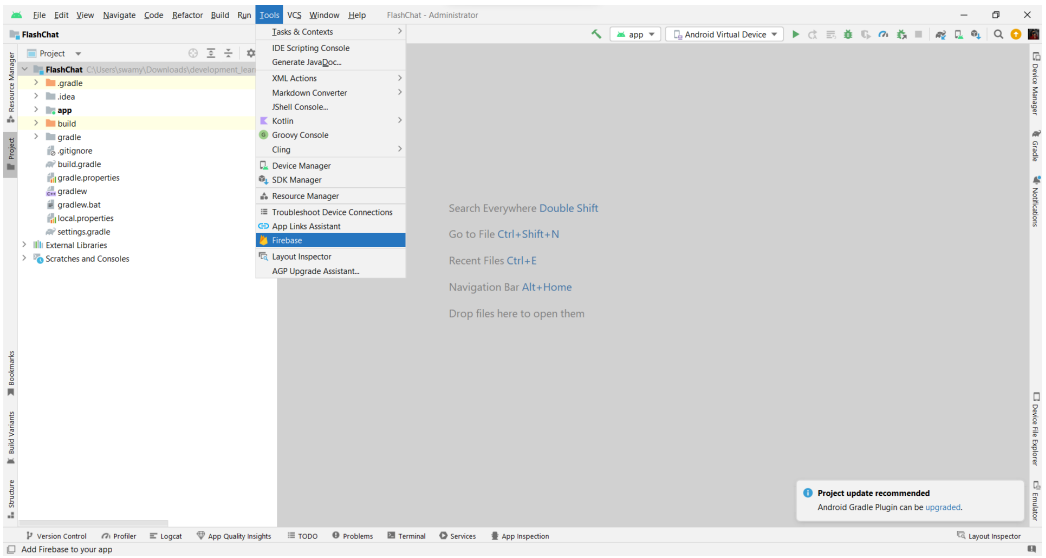
- Click on **Connect**



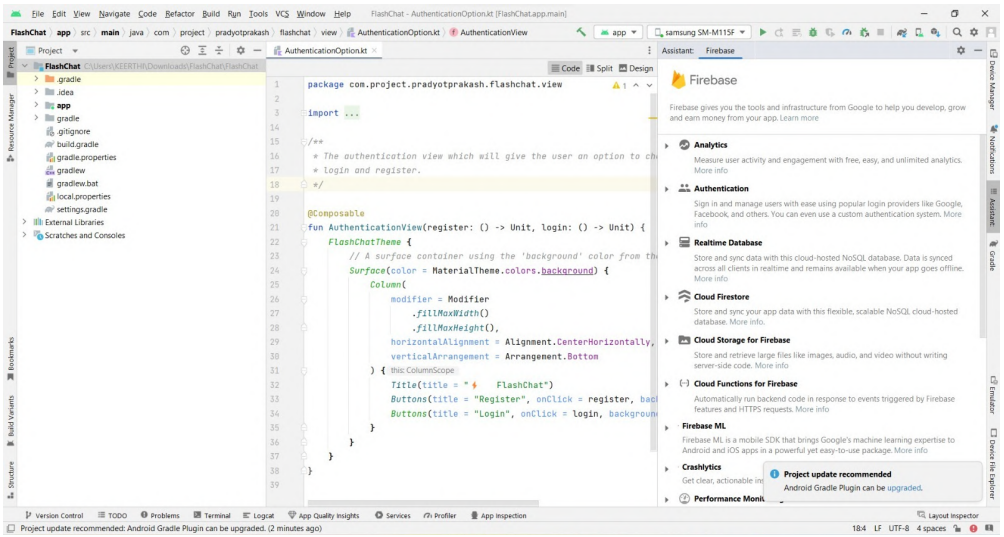
- This shows that your project is connected to Firebase.



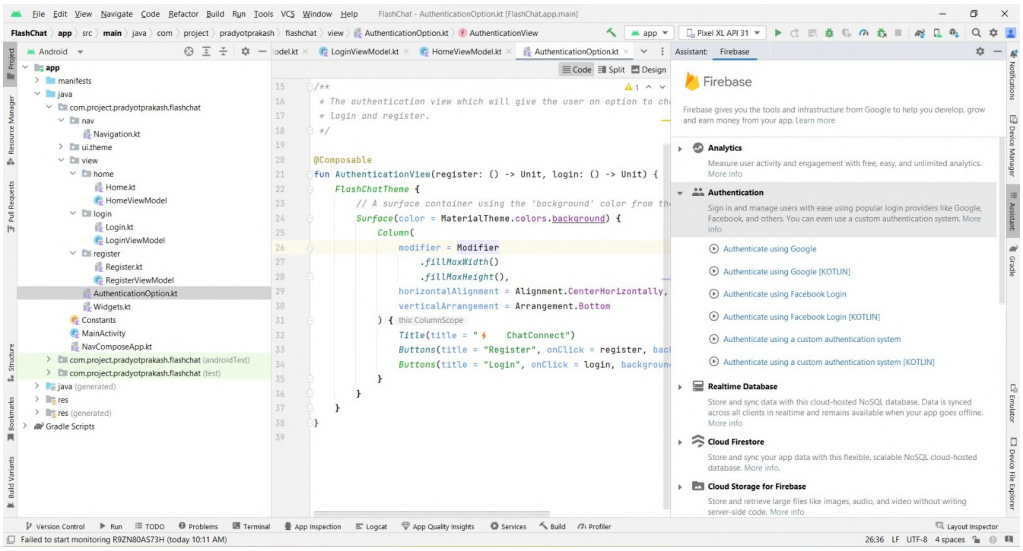
- Menu bar > Tools > Firebase.



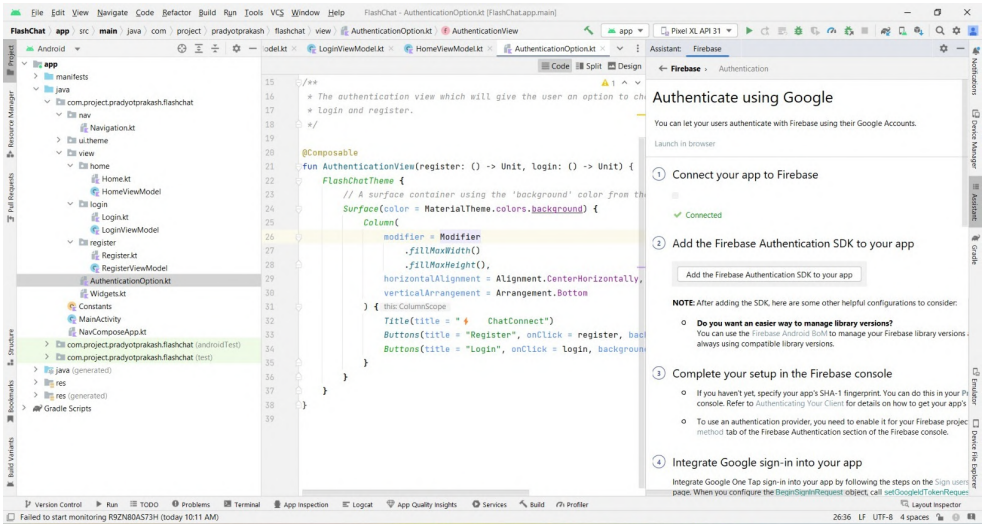
- After clicking the Firebase , the Firebase tab will open.



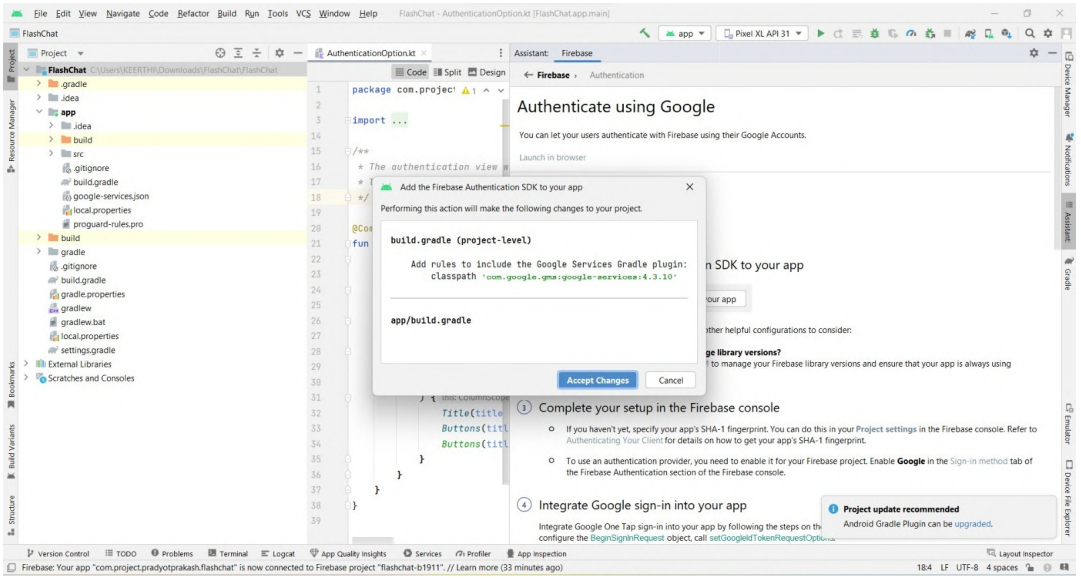
- In Authentication > Go to Authenticate using Google.



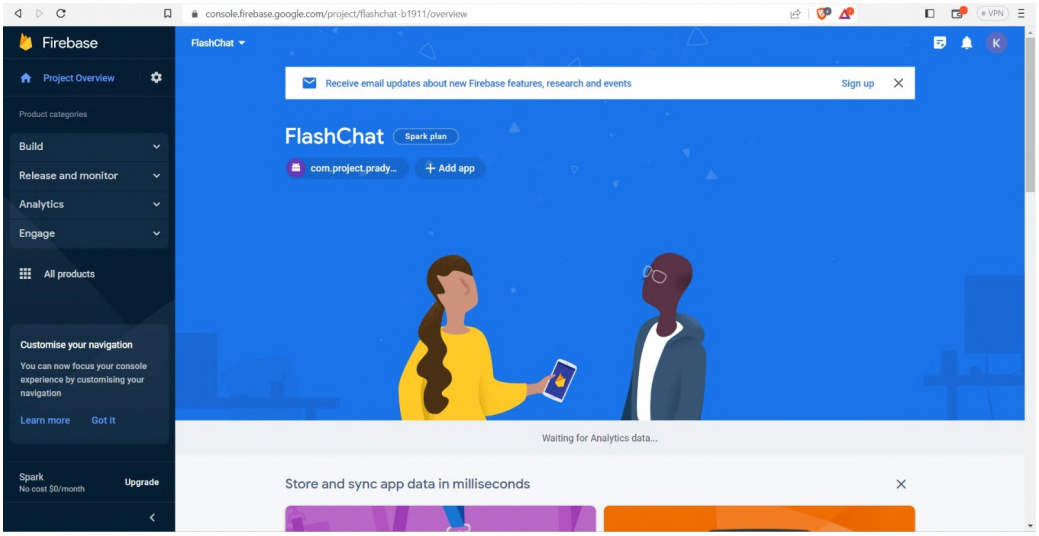
- After that, Authentication using Google will open.



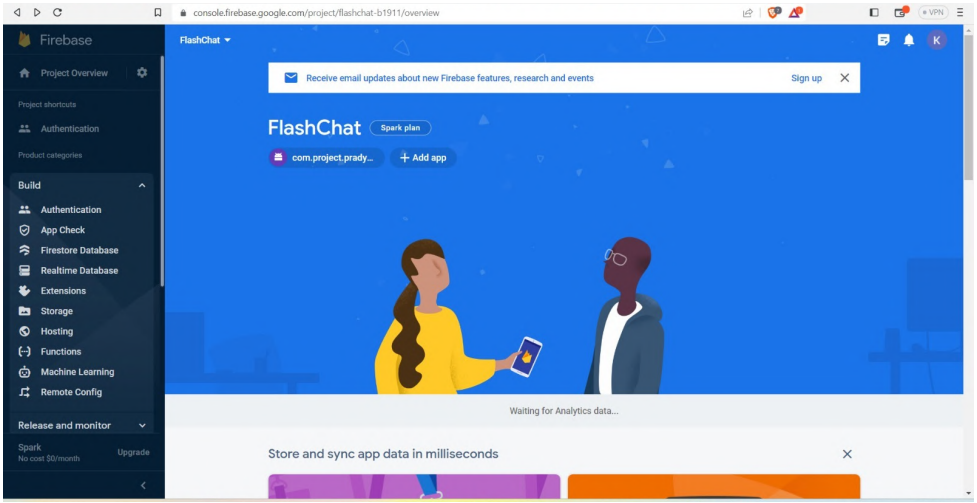
- Now you can see that it shows **Connected.**
- Now click on the second point “Add the Firebase Authentication SDK to your app”.



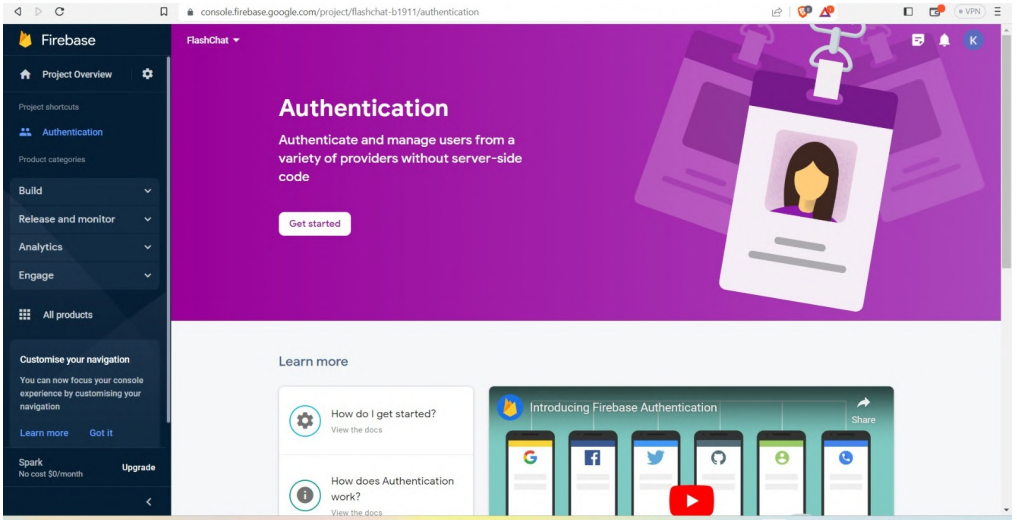
- Now a small tab will open and Click on **Accept Changes**.



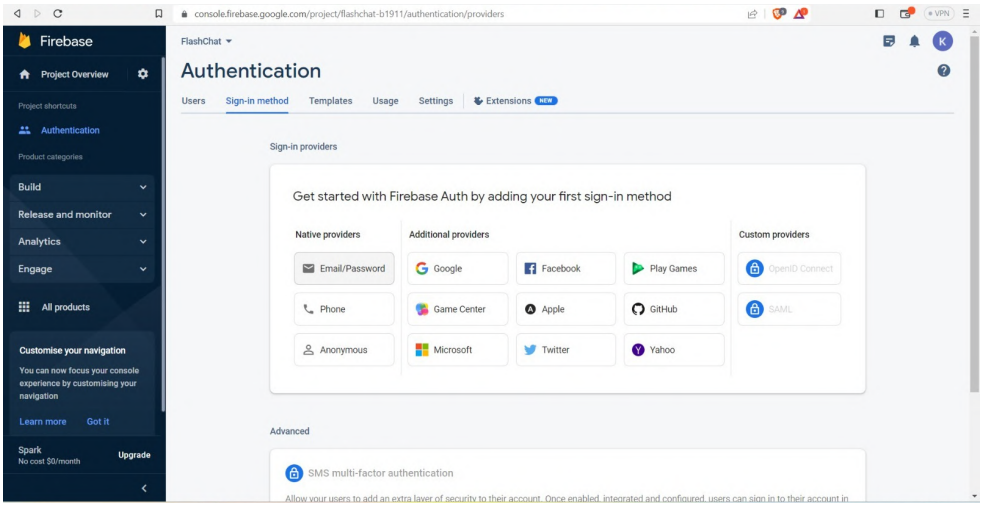
- Now go to your respective browser and Search **Firebase** Or https://firebase.google.com/
- Open Firebase official website.



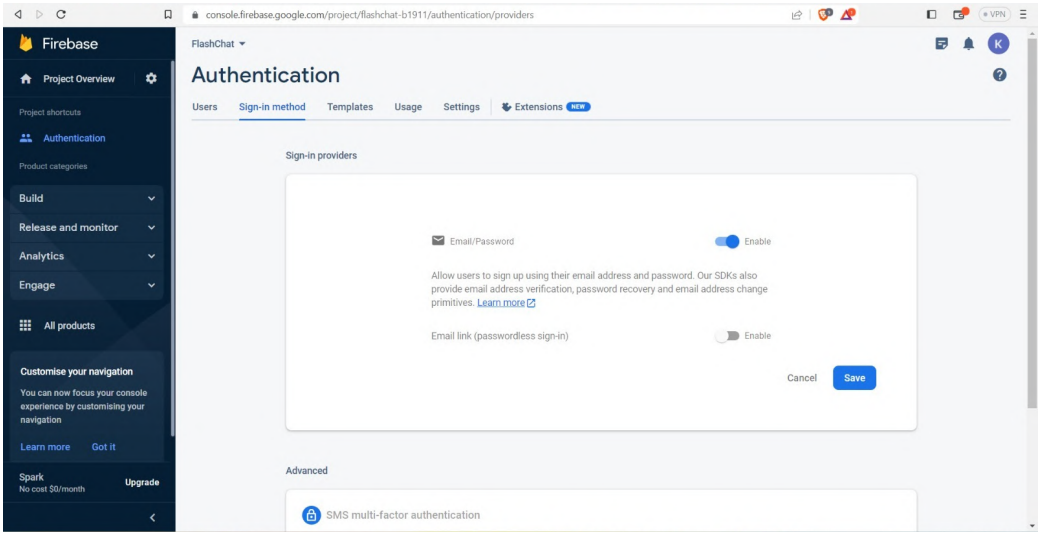
- In the right side panel Click on **Build** < Click on **Authentication**.



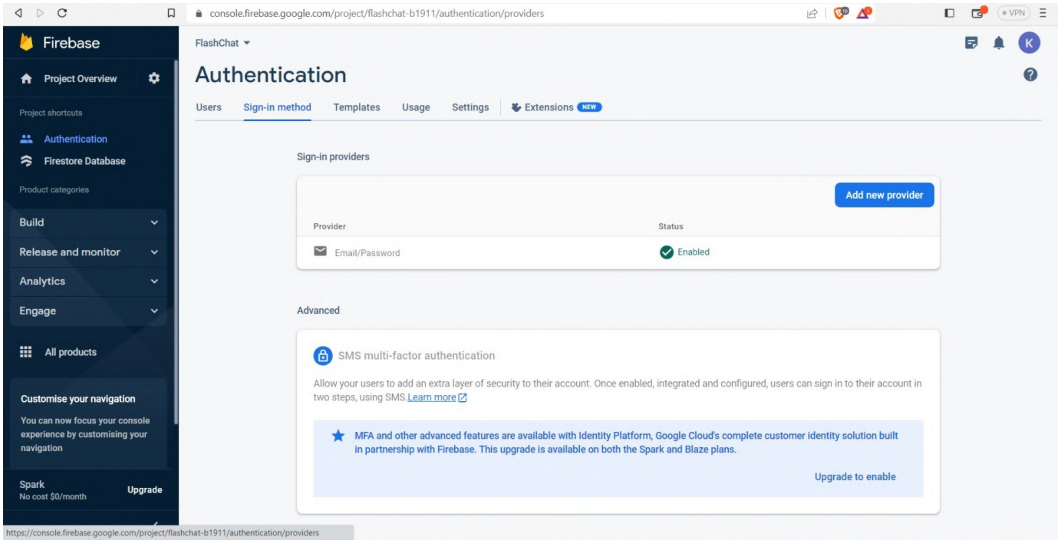
- Click on **Get started**.



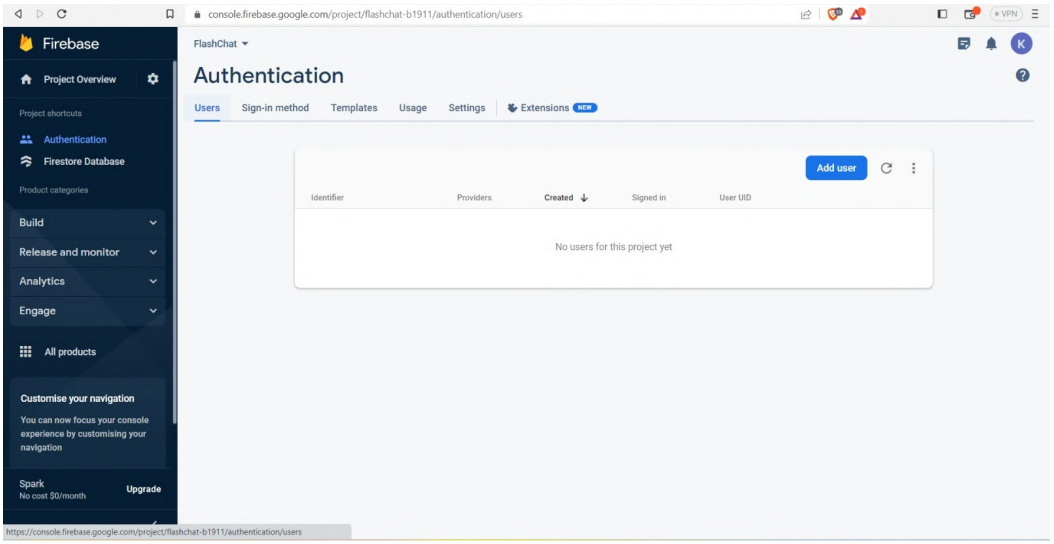
- Now the above page will open for **Authentication**.
- Go to Sign-in-method.
- Click on **Email/Password**.



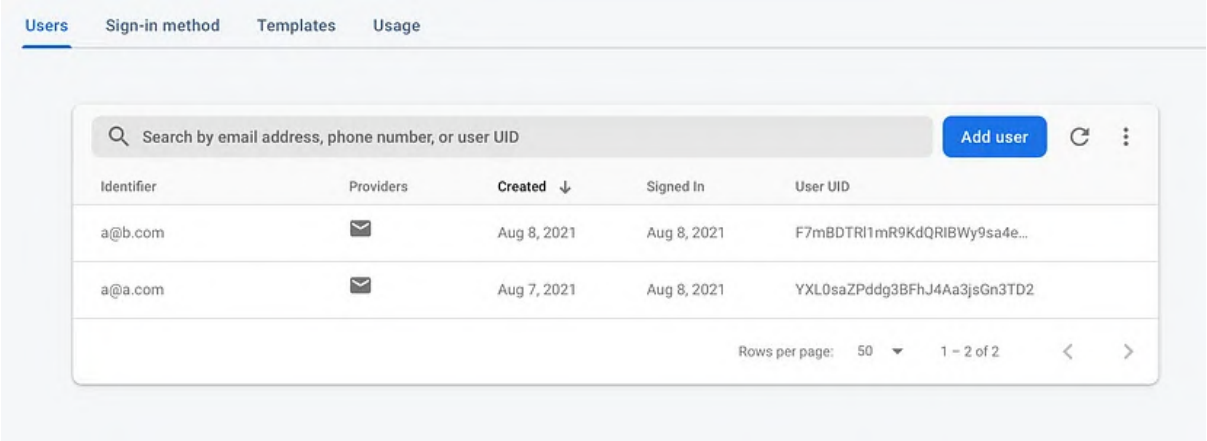
- Enable Email/Password and Click on **Save**.



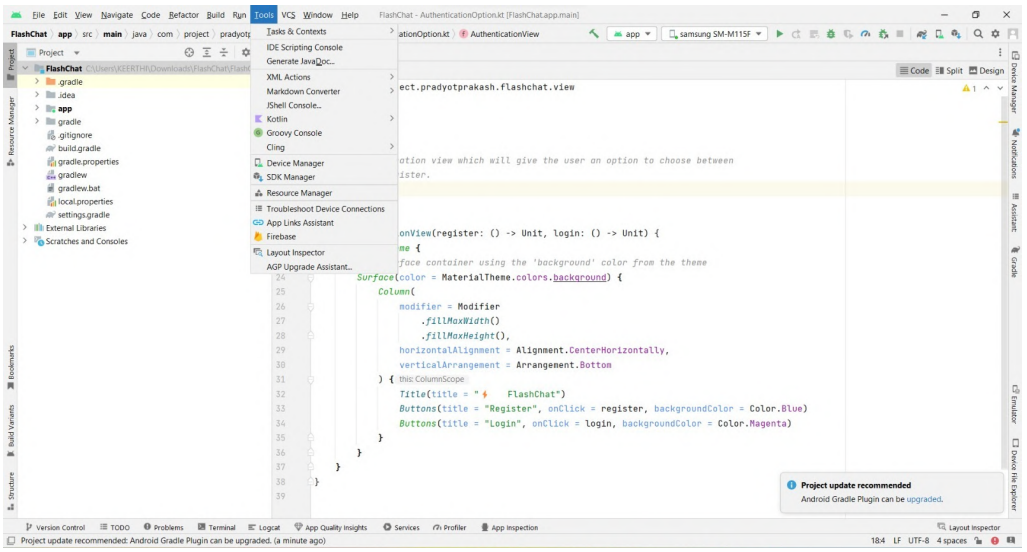
- Now you can see that Email/Password is enabled.



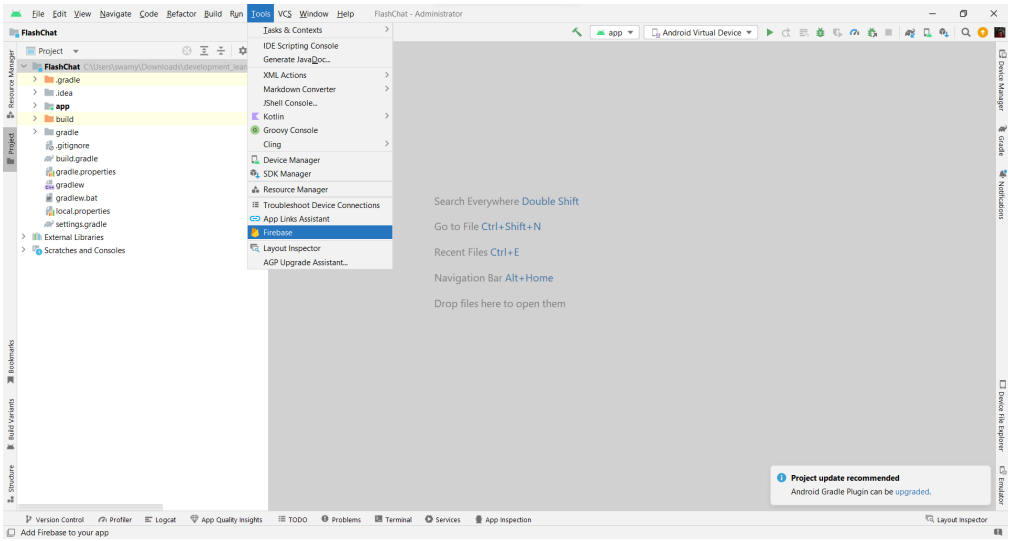
- Now go to Users.



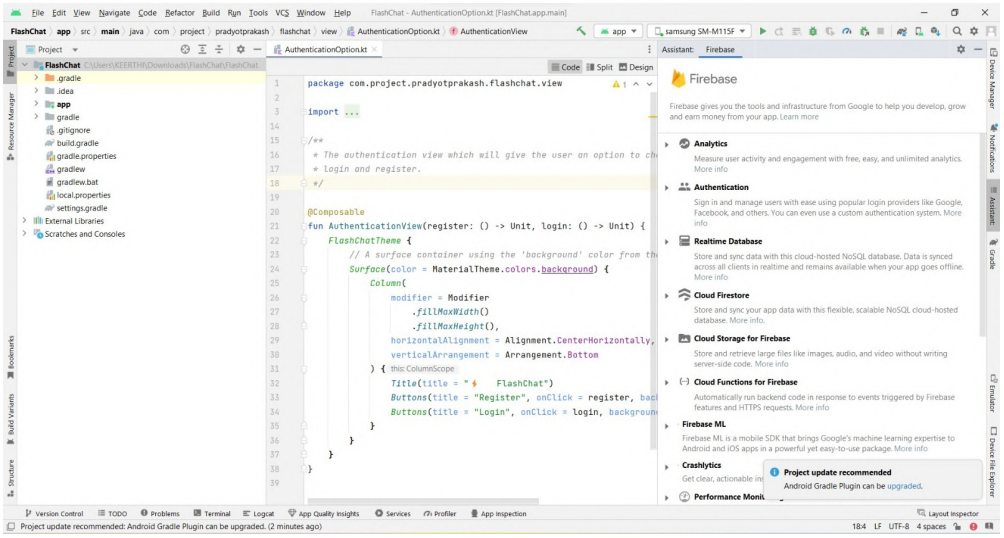
- In the Above image we can see the Emails and details , which will be shown after registering in the Register page of the project , which we are going to create in further steps.



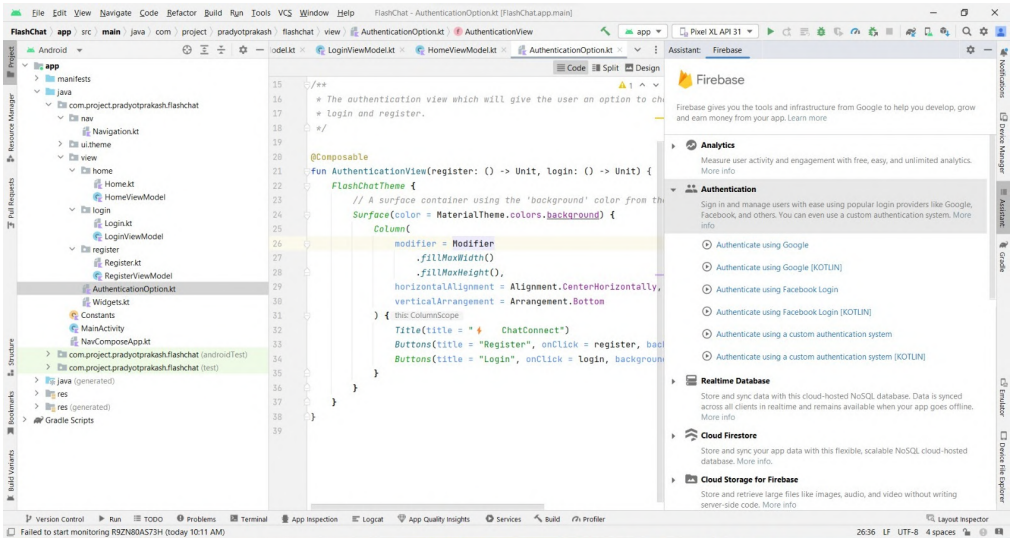
- Menu bar > Tools > Firebase.



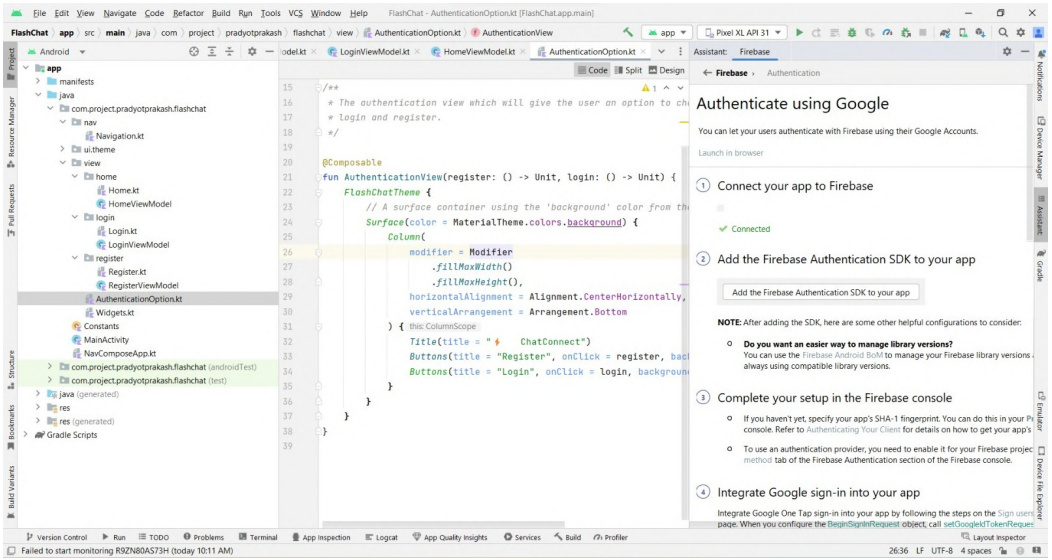
- After clicking the Firebase , the Firebase tab will open.



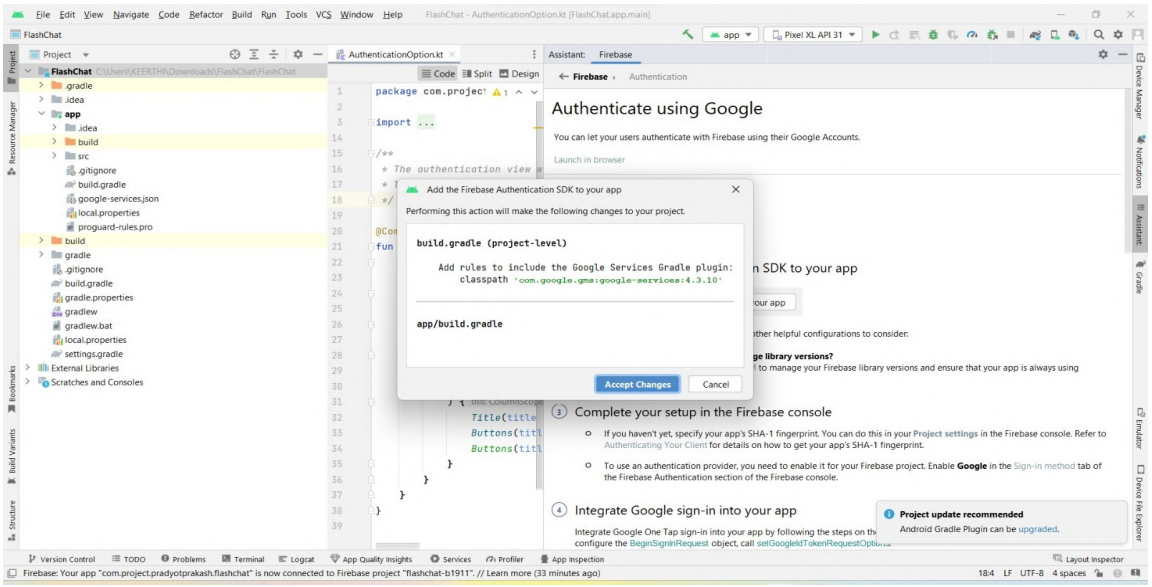
- In Authentication > Go to Authenticate using Google.



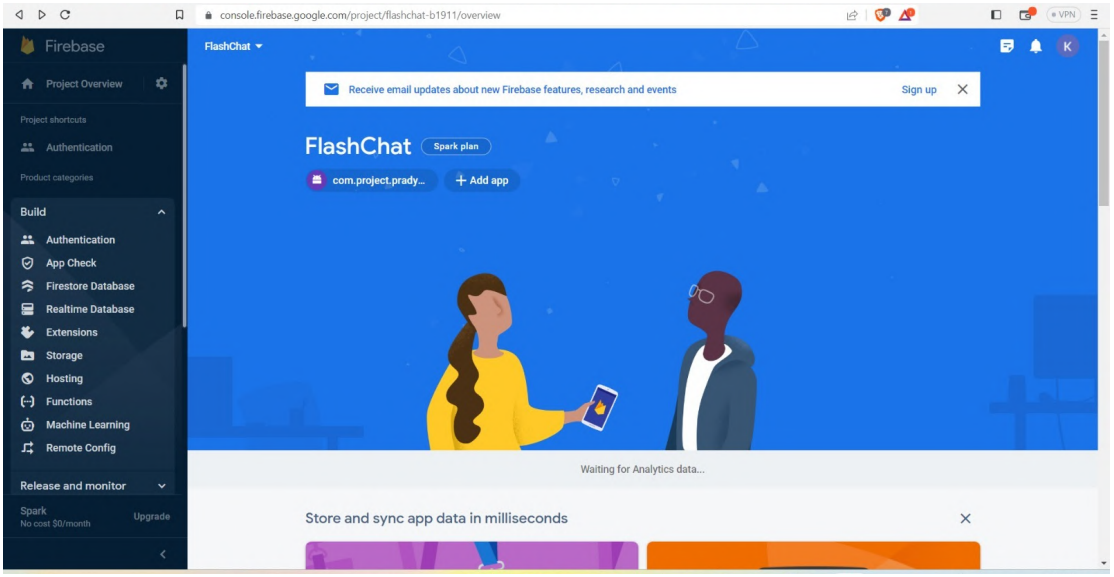
- After that, Authentication using Google will open.



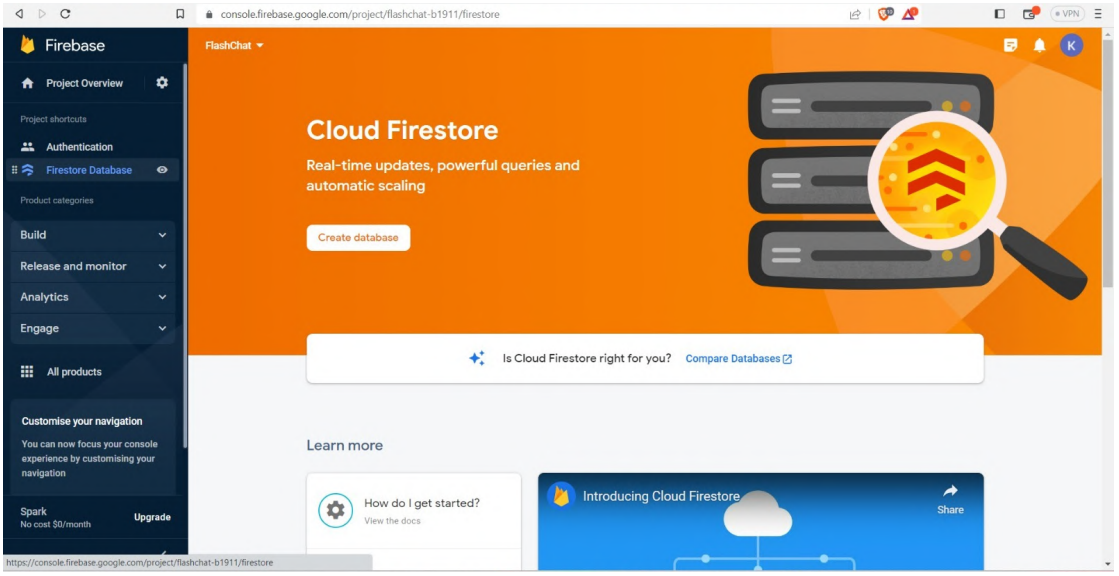
- Now you can see that it shows **Connected.**
- Now click on the second point “Add the Firebase Authentication SDK to your app”.



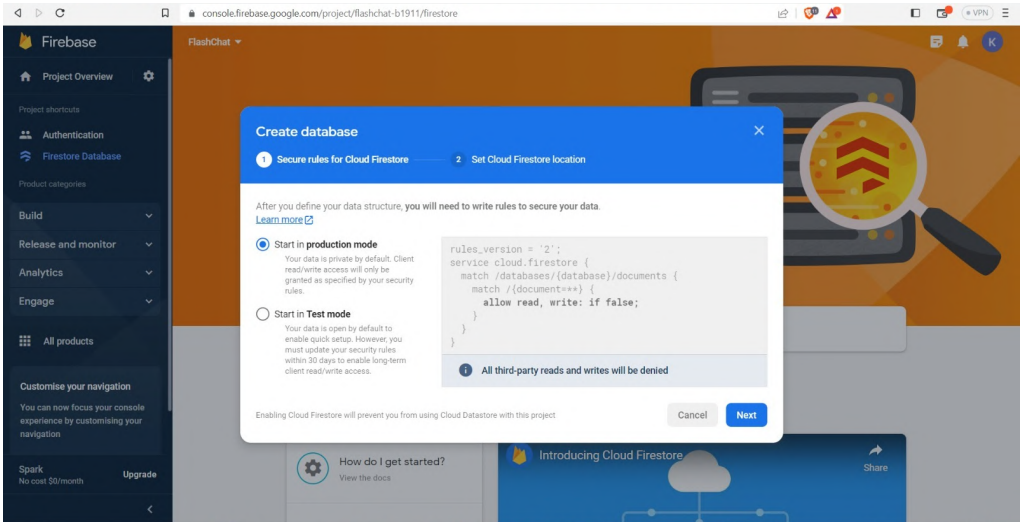
- Now a small tab will open and Click on **Accept Changes**.
- Now go to the Firebase website in the respective browser.



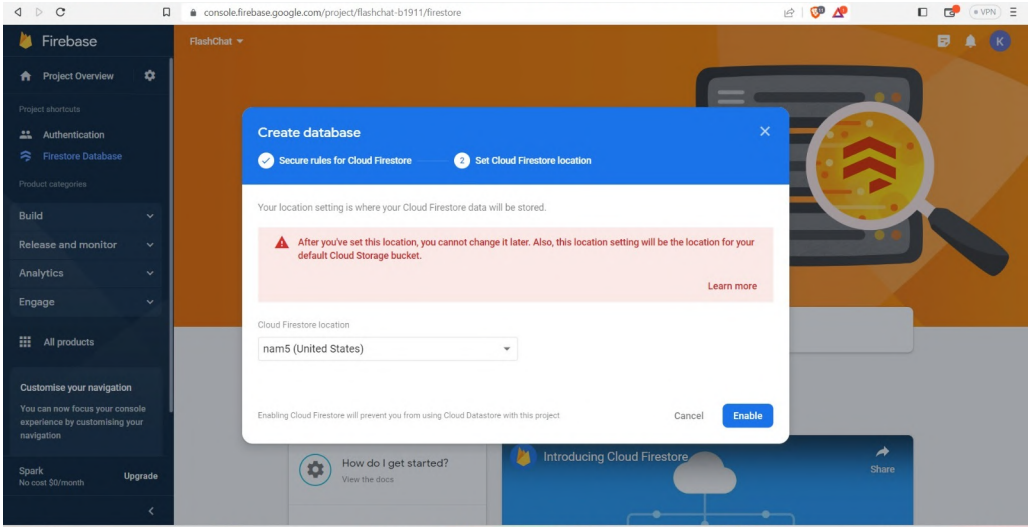
- In the right side panel Click on **Build** < Click on **Firestore Database**.



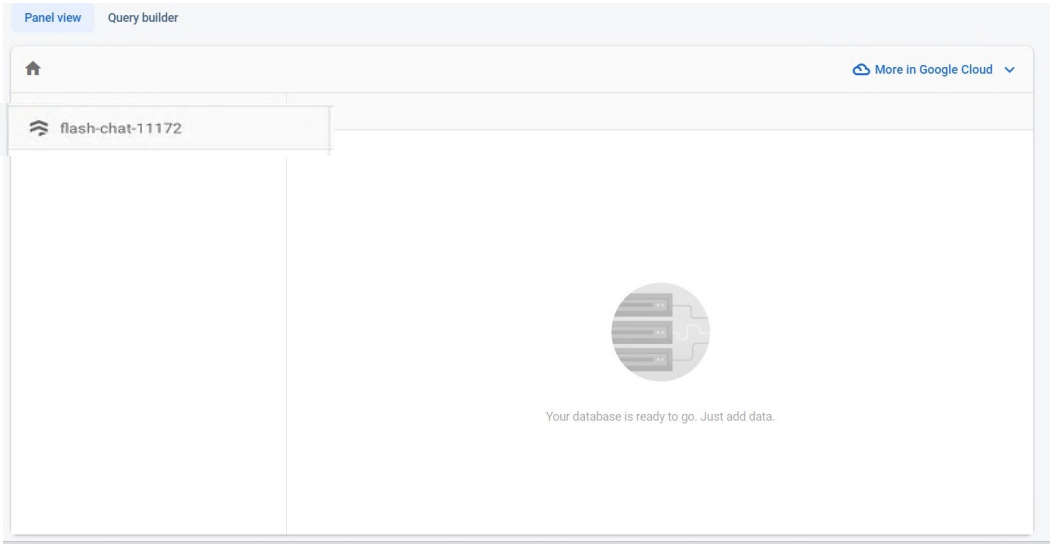
- Click on **Create database**.



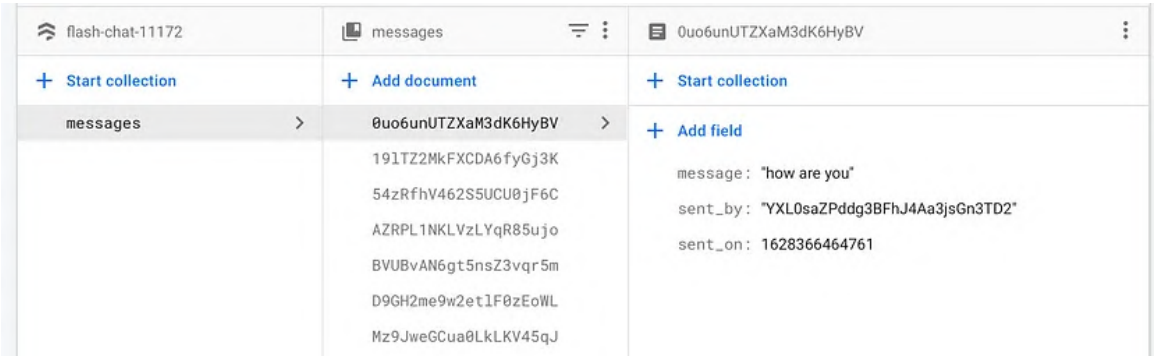
- Click on **Next.**



- Click on **Enable.**



- In ‘Start collection’ we can see the chat messages which are being sent and received in the Activity pages in UI which we are going to create in further steps.
- Like below it will be shown.



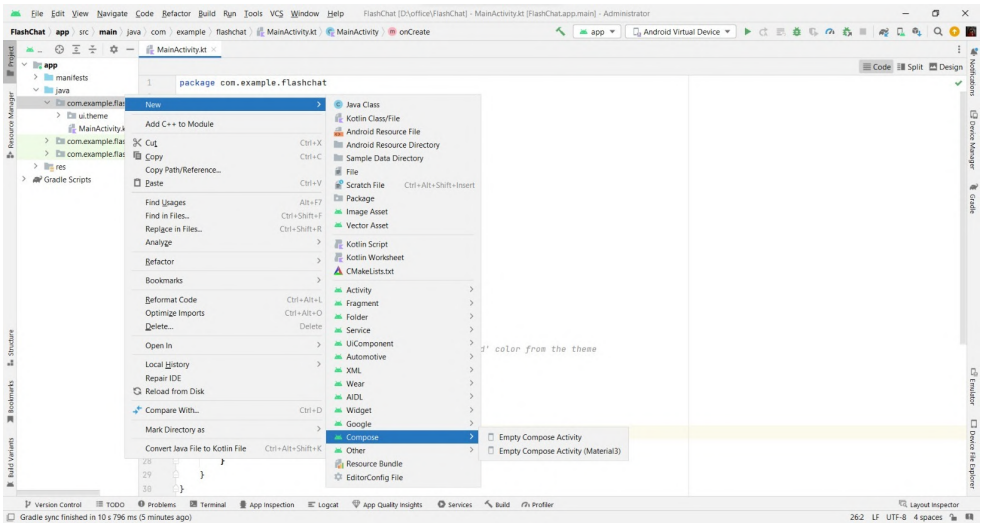
Task 4

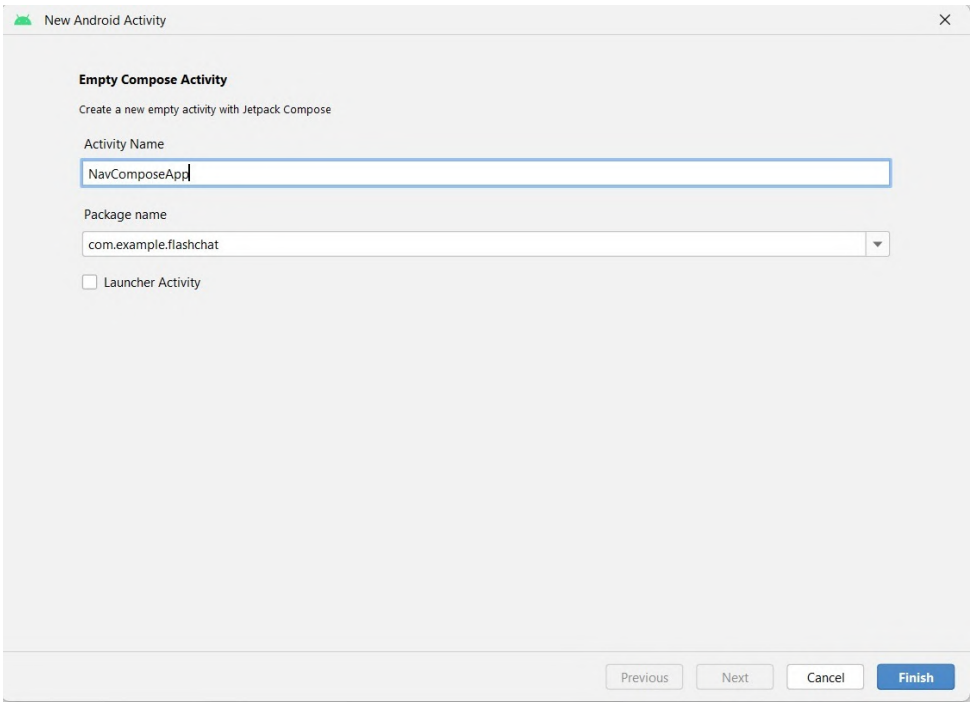
MainActivity.kt file

```
package com.project.pradyotprakash.flashchat

import ...

/**
 * The initial point of the application from where it gets started.
 *
 * Here we do all the initialization and other things which will be required
 * thought out the application.
 */
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        FirebaseApp.initializeApp( context: this)
        setContent {
            NavComposeApp()
        }
    }
}
```

Complete Code for MainActivity.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/MainActivity.kt

Creating NavComposeApp.kt file

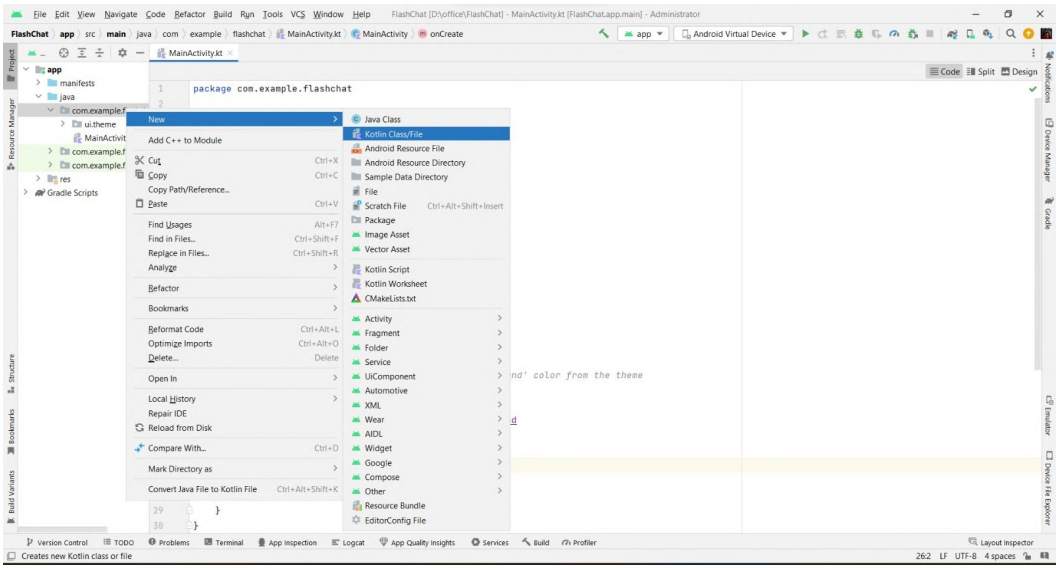


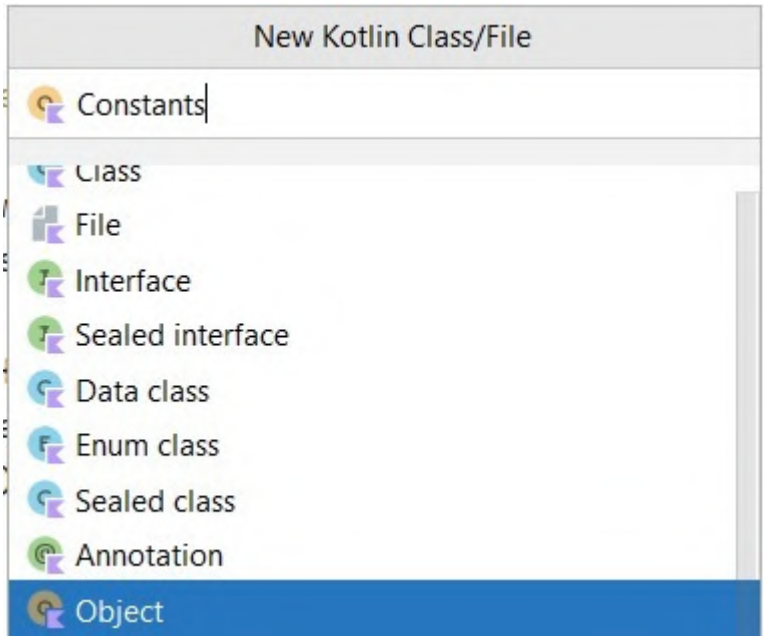


```kotlin
package com.project.pradyotprakash.flashchat

import ...

/**
 * The main Navigation composable which will handle all the navigation stack.
 */

@Composable
fun NavComposeApp() {
    val navController = rememberNavController()
    val actions = remember(navController) { Action(navController) }
    FlashChatTheme {
        NavHost(
            navController = navController,
            startDestination =
            if (FirebaseAuth.getInstance().currentUser != null)
                Home
            else
                AuthenticationOption
        ) { this: NavGraphBuilder
            composable(AuthenticationOption) { it: NavBackStackEntry
                AuthenticationView(
                    register = actions.register,
                    login = actions.login
                )
            }
            composable(Register) { it: NavBackStackEntry
                RegisterView(
                    home = actions.home,
```

Complete Code for NavComposeApp.kt file : https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/NavComposeApp.kt

Creating Constants object

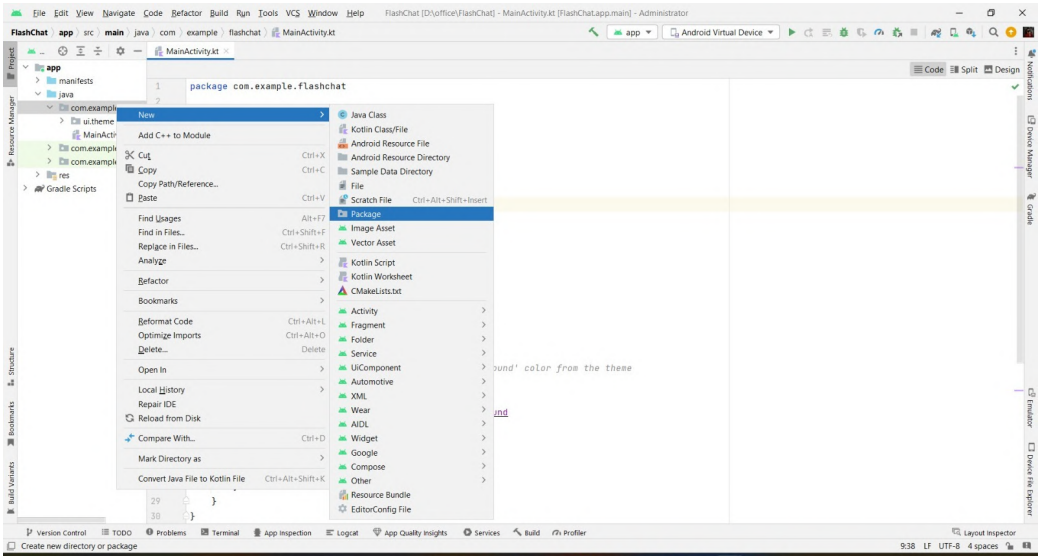


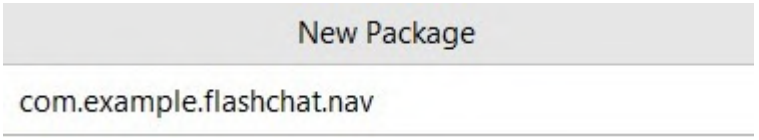



```kotlin
package com.project.pradyotprakash.flashchat

object Constants {
    const val TAG = "flash-chat"

    const val MESSAGES = "messages"
    const val MESSAGE = "message"
    const val SENT_BY = "sent_by"
    const val SENT_ON = "sent_on"
    const val IS_CURRENT_USER = "is_current_user"
}
```

Complete Code for Constants.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/Constants.kt

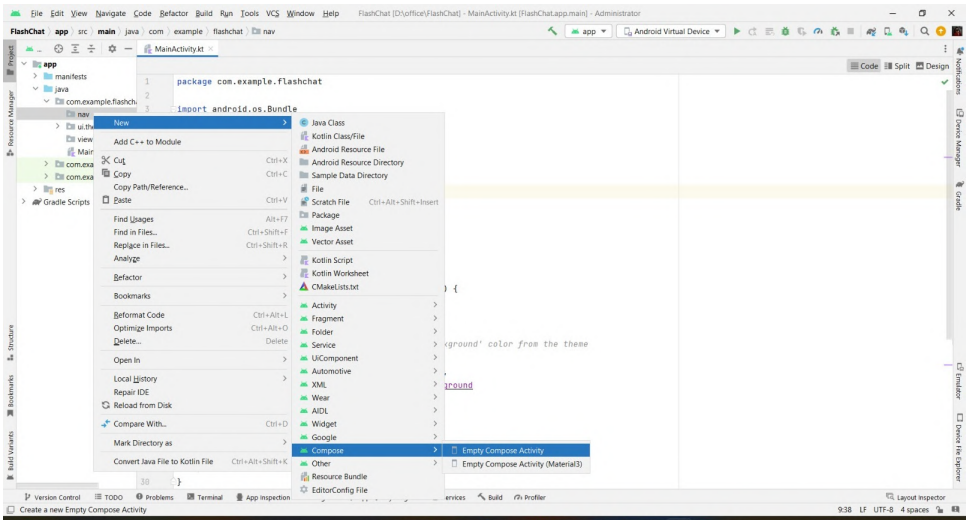
Creating nav package



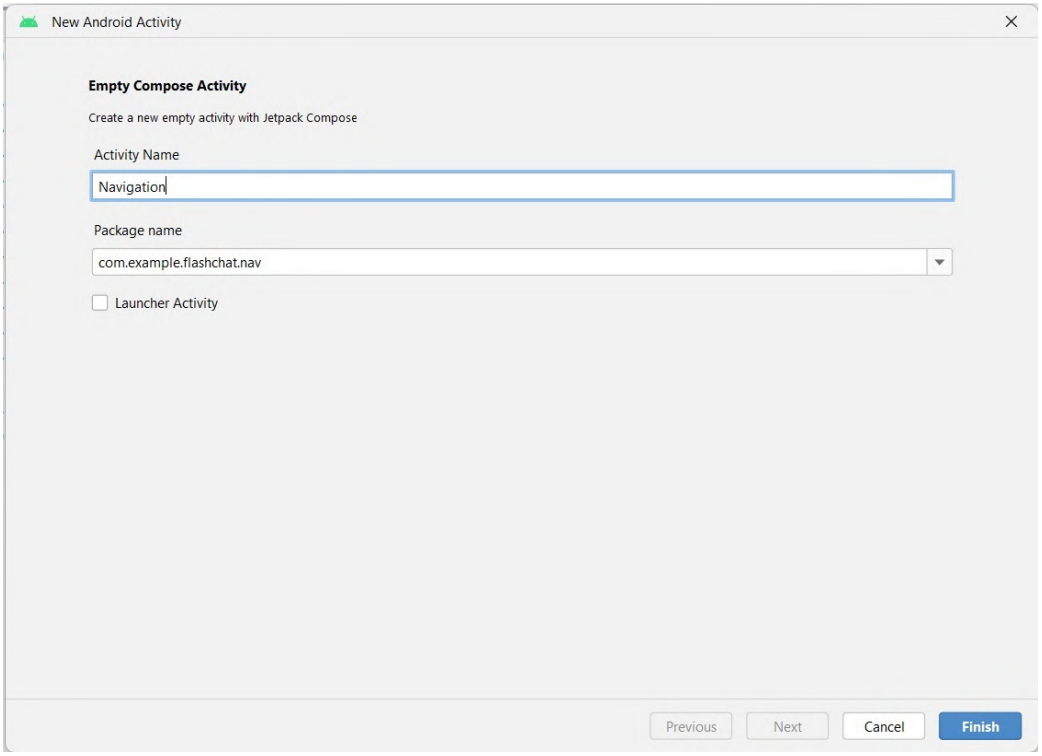


Creating Navigation.kt in nav package

Nav package>right click>new>compose>empty compose activity



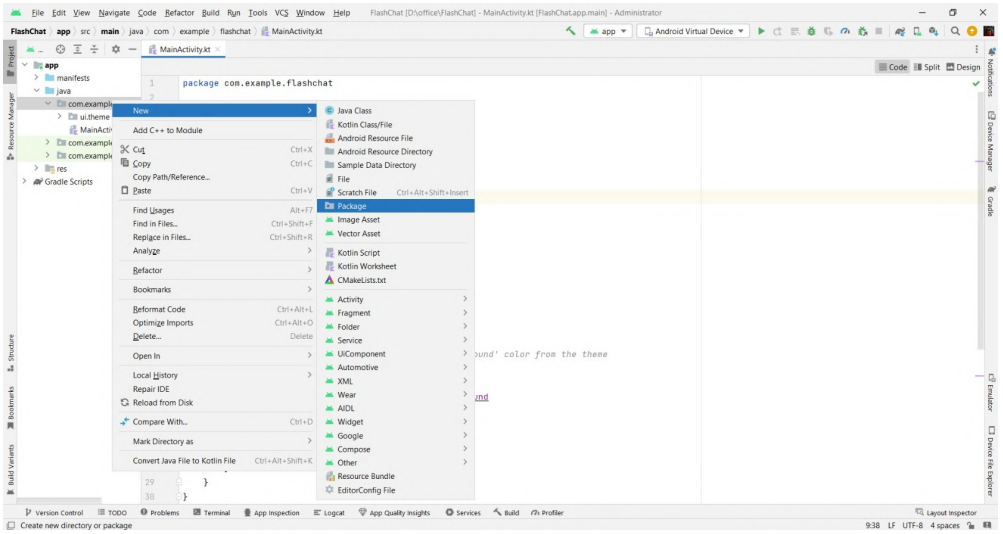
Navigation.kt file

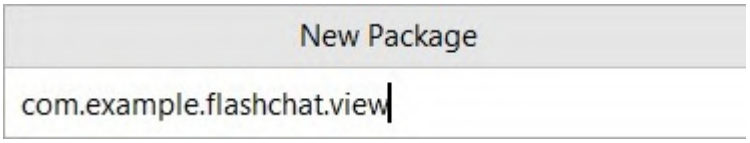



```kotlin
package com.project.pradyotprakash.flashchat.nav

import ...

/**
 * A set of destination used in the whole application
 */
object Destination {
    const val AuthenticationOption = "authenticationOption"
    const val Register = "register"
    const val Login = "login"
    const val Home = "home"
}

/**
 * Set of routes which will be passed to different composable so that
 * the routes which are required can be taken.
 */
class Action(navController: NavHostController) {
    val home: () -> Unit = {
        navController.navigate(Home) { this: NavOptionsBuilder
            popUpTo(Login) { this: PopUpToBuilder
                inclusive = true
            }
            popUpTo(Register) { this: PopUpToBuilder
                inclusive = true
            }
        }
    }
    val login: () -> Unit = { navController.navigate(Login) }
```

Complete Code for Navigation.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/nav/Navigation.kt

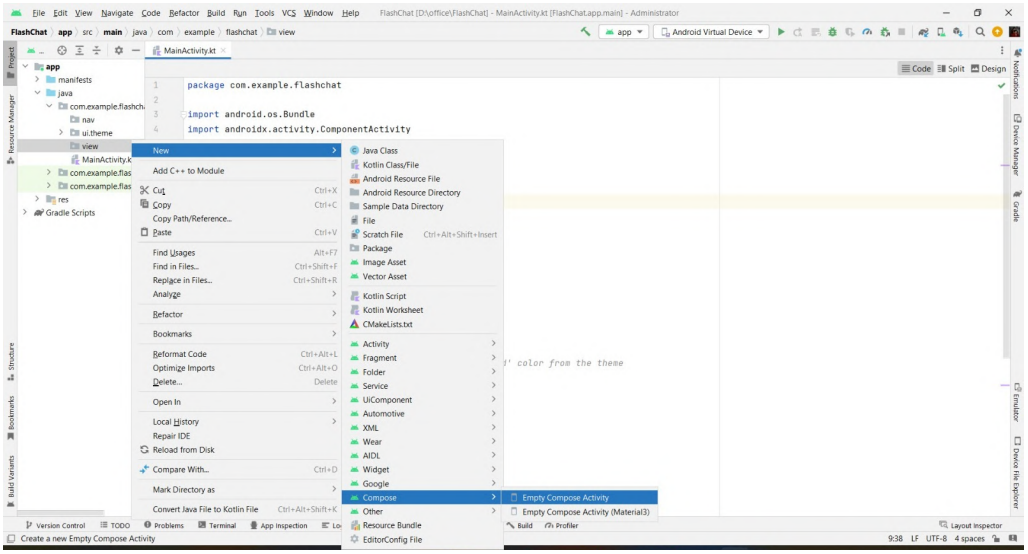
Creating view package



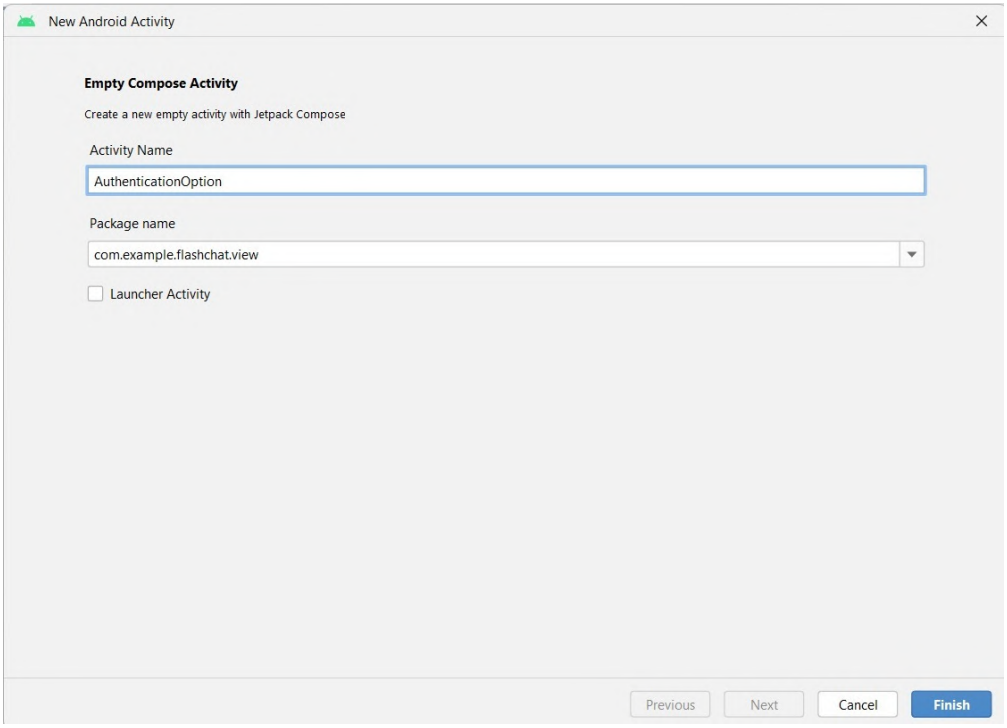


Creating Navigation.kt in nav package

view package>right click>new>compose>empty compose activity



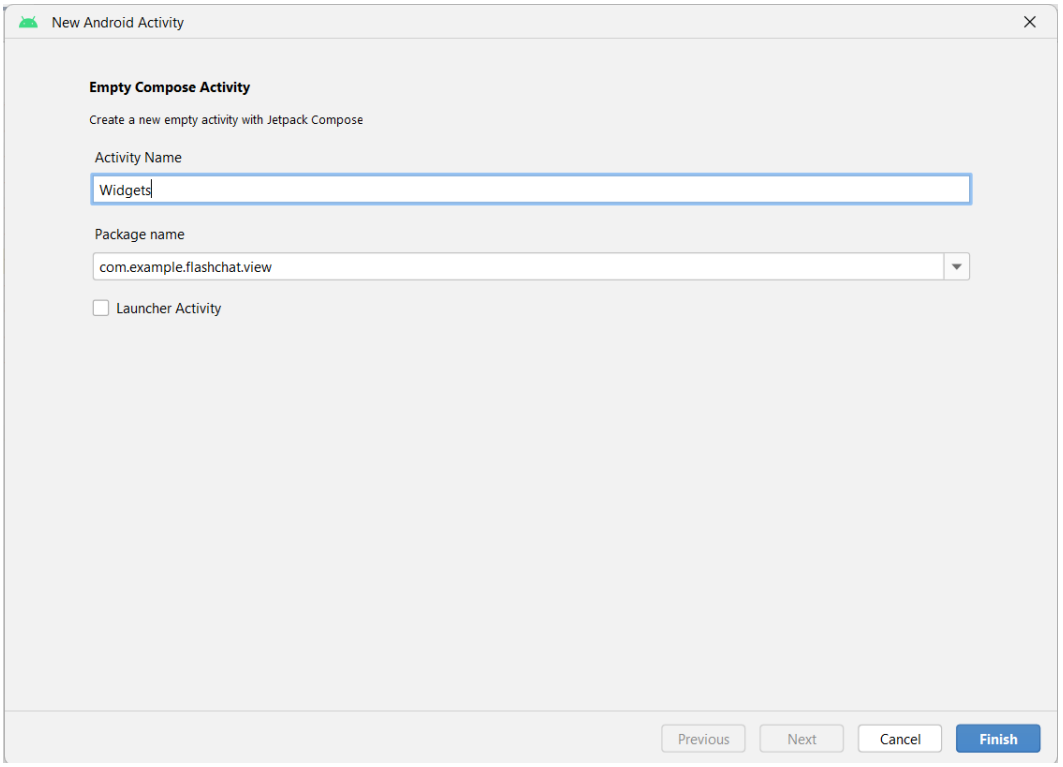
AuthenticationOption.kt file




```kotlin
package com.project.pradyotprakash.flashchat.view

import ...

/**
 * The authentication view which will give the user an option to choose between
 * login and register.
 */

@Composable
fun AuthenticationView(register: () -> Unit, login: () -> Unit) {
    FlashChatTheme {
        // A surface container using the 'background' color from the theme
        Surface(color = MaterialTheme.colors.background) {
            Column(
                modifier = Modifier
                    .fillMaxWidth()
                    .fillMaxHeight(),
                horizontalAlignment = Alignment.CenterHorizontally,
                verticalArrangement = Arrangement.Bottom
            ) { this: ColumnScope
                Title(title = "    Chat Connect")
                Buttons(title = "Register", onClick = register, backgroundColor = Color.Blue)
                Buttons(title = "Login", onClick = login, backgroundColor = Color.Magenta)
            }
        }
    }
}
```

Complete Code for AuthenticationOption.kt file : https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/AuthenticationOption.kt

Widgets.kt



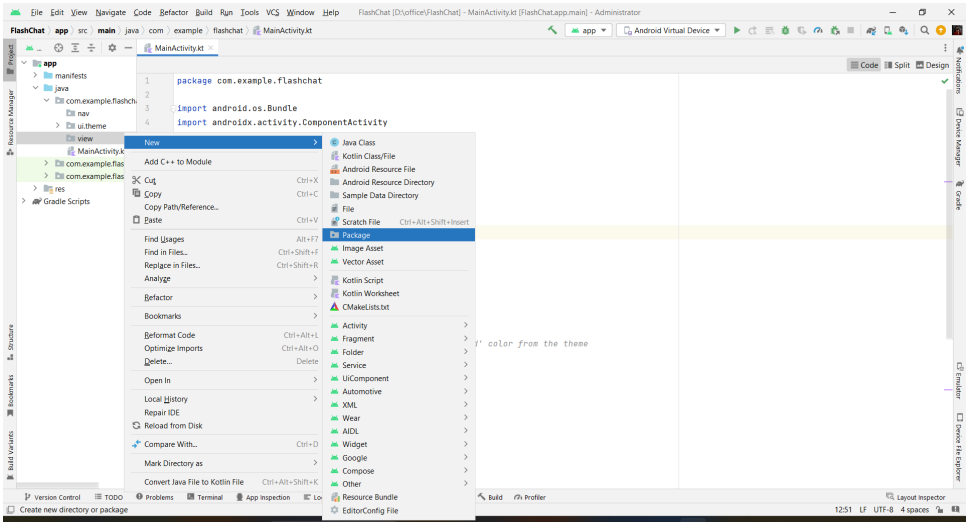

```kotlin
package com.project.pradyotprakash.flashchat.view

import ...

/**
 * Set of widgets/views which will be used throughout the application.
 * This is used to increase the code usability.
 */

@Composable
fun Title(title: String) {
    Text(
        text = title,
        fontSize = 30.sp,
        fontWeight = FontWeight.Bold,
        modifier = Modifier.fillMaxHeight( fraction: 0.5f)
    )
}

// Different set of buttons in this page
@Composable
fun Buttons(title: String, onClick: () -> Unit, backgroundColor: Color) {
    Button(
        onClick = onClick,
        colors = ButtonDefaults.buttonColors(
            backgroundColor = backgroundColor,
            contentColor = Color.White
        ),
        modifier = Modifier.fillMaxWidth(),
        shape = RoundedCornerShape( percent: 0),
```

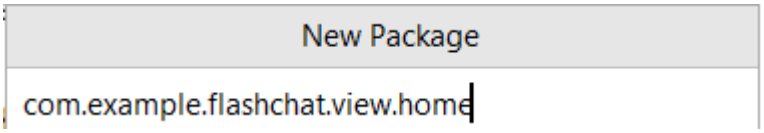
Complete Code for Widgets.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/Widgets.kt

In View package we are having three packages

1. **Creating home package in view package**

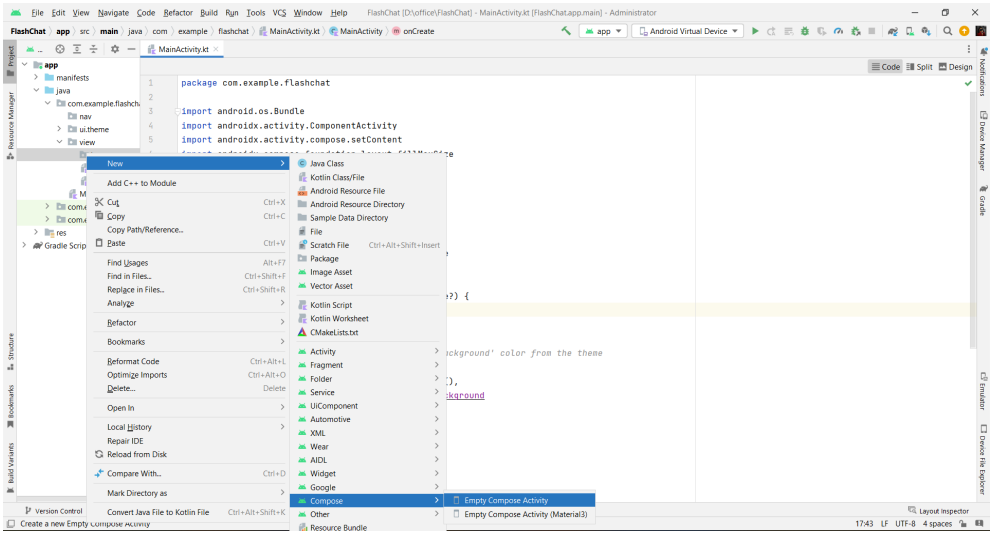
view package>right click>new>package

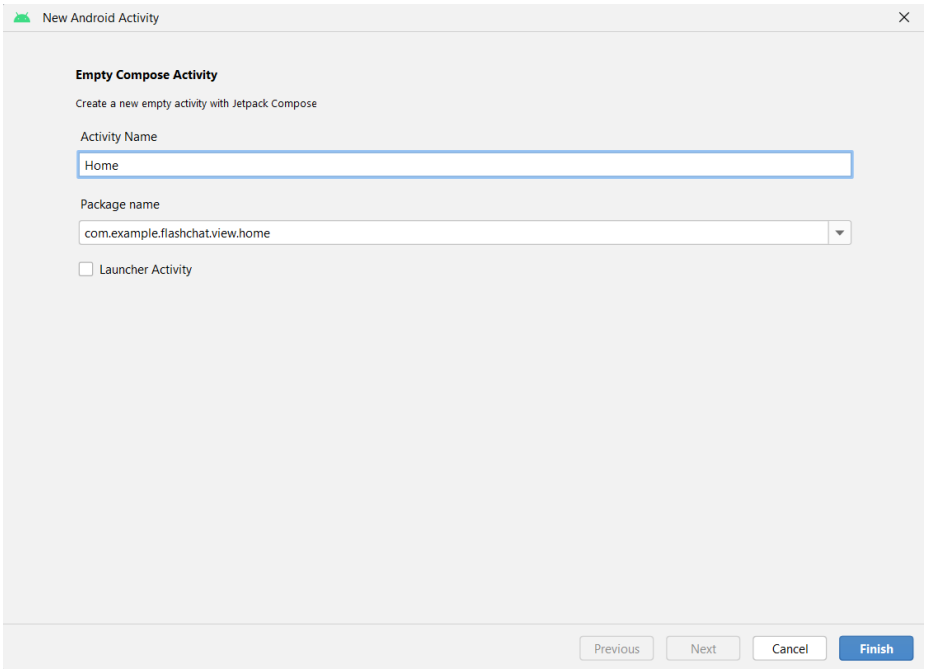




Creating Home.kt file

view package>home package>right click>new>compose>empty compose activity



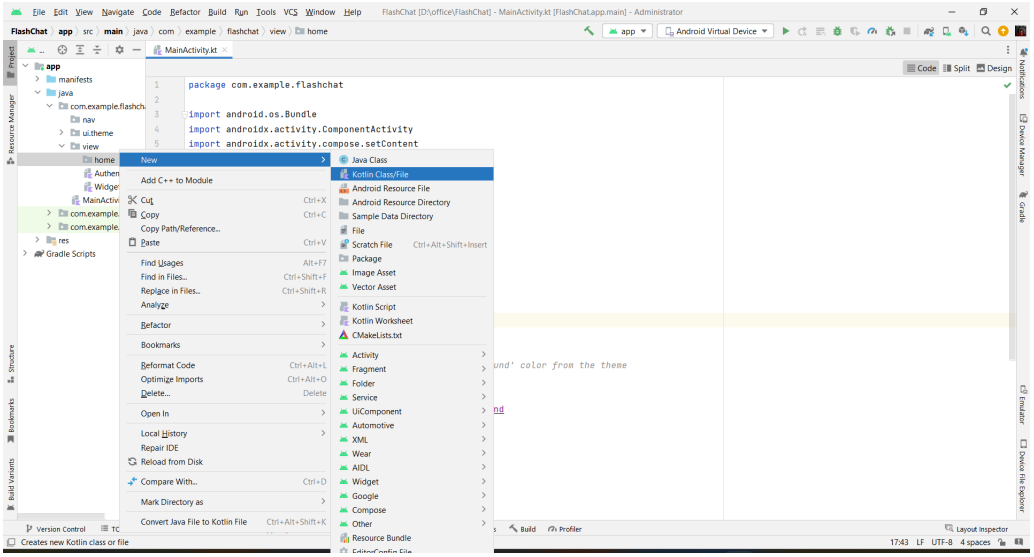


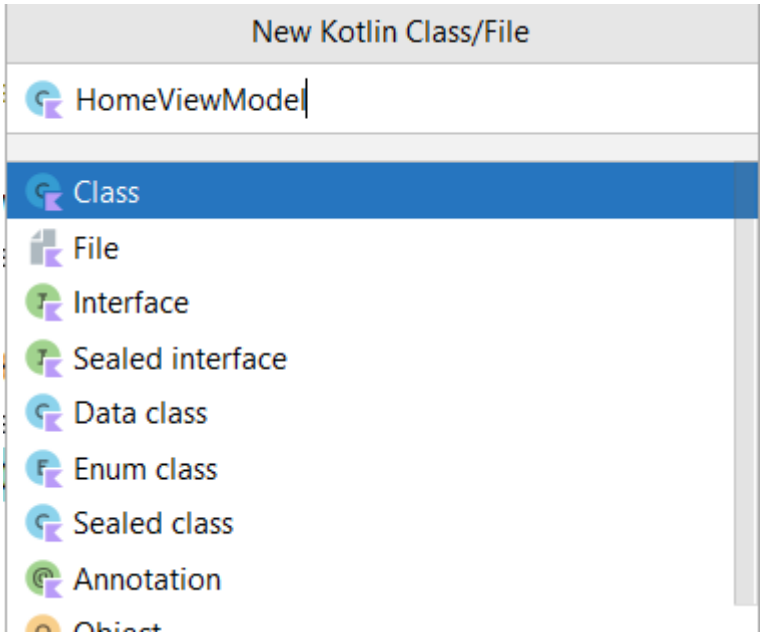

```kotlin
package com.project.pradyotprakash.flashchat.view.home

import ...

/**
 * The home view which will contain all the code related to the view for HOME.
 *
 * Here we will show the list of chat messages sent by user.
 * And also give an option to send a message and logout.
 */

@Composable
fun HomeView(
    homeViewModel: HomeViewModel = viewModel()
) {
    val message: String by homeViewModel.message.ob
    val messages: List<Map<String, Any>> by homeVie
        initial = emptyList<Map<String, Any>>().toMutableList()
    )

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Bottom
    ) { this: ColumnScope
        LazyColumn(
            modifier = Modifier
                .fillMaxWidth()
                .weight(weight = 0.85f, fill = true),
            contentPadding = PaddingValues(horizontal = 16.dp, vertical = 8.dp),
```

val message: String

Complete Code for Home.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/home/Home.kt

Creating HomeViewModel class
view package>home package>right click>new>kotlin class/file



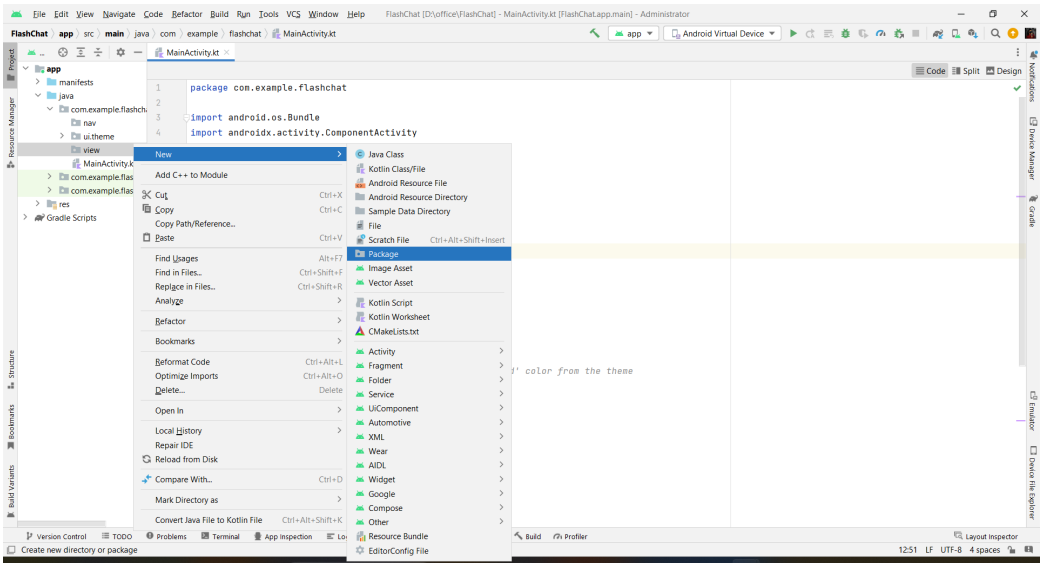


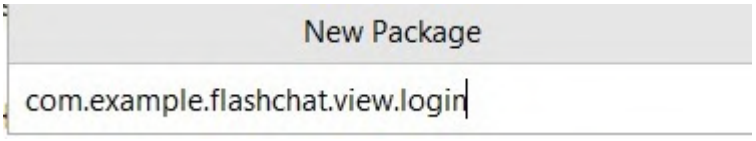
```kotlin
package com.project.pradyotprakash.flashchat.view.home

import ...

/**
 * Home view model which will handle all the logic related to HomeView
 */
class HomeViewModel : ViewModel() {
    init {
        getMessages()
    }

    private val _message = MutableLiveData( value: "")
    val message: LiveData<String> = _message

    private var _messages = MutableLiveData(emptyList<Map<String, Any>>().toMutableList())
    val messages: LiveData<MutableList<Map<String, Any>>> = _messages

    /**
     * Update the message value as user types
     */
    fun updateMessage(message: String) {
        _message.value = message
    }

    /**
     * Send message
     */
    fun addMessage() {
        val message: String = _message.value ?: throw IllegalArgumentException("message empty")
```

Complete Code for HomeViewModel.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/home/HomeViewModel.kt

2. Creating login package in view package

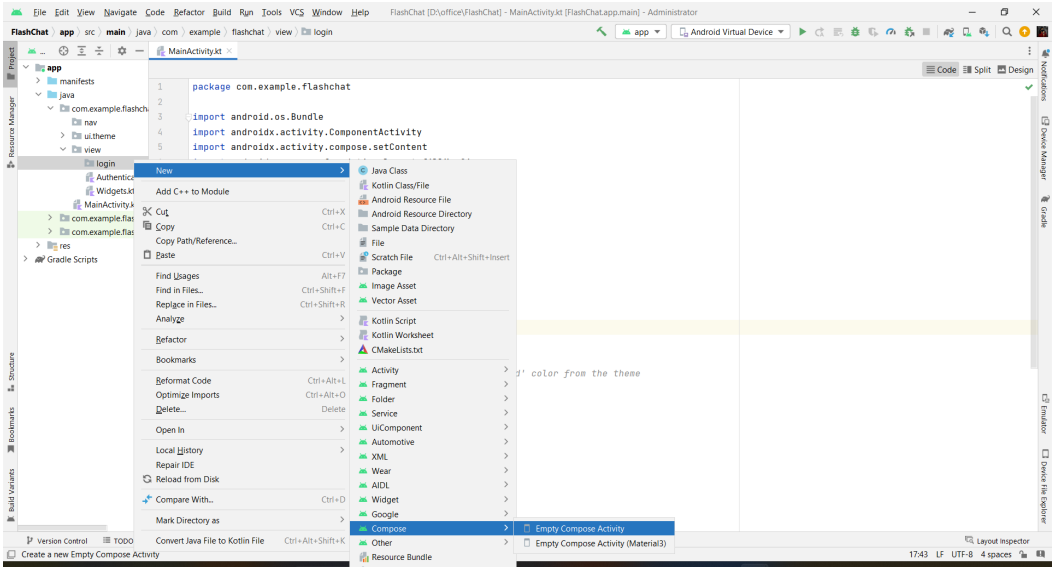
view package>right click>new>package

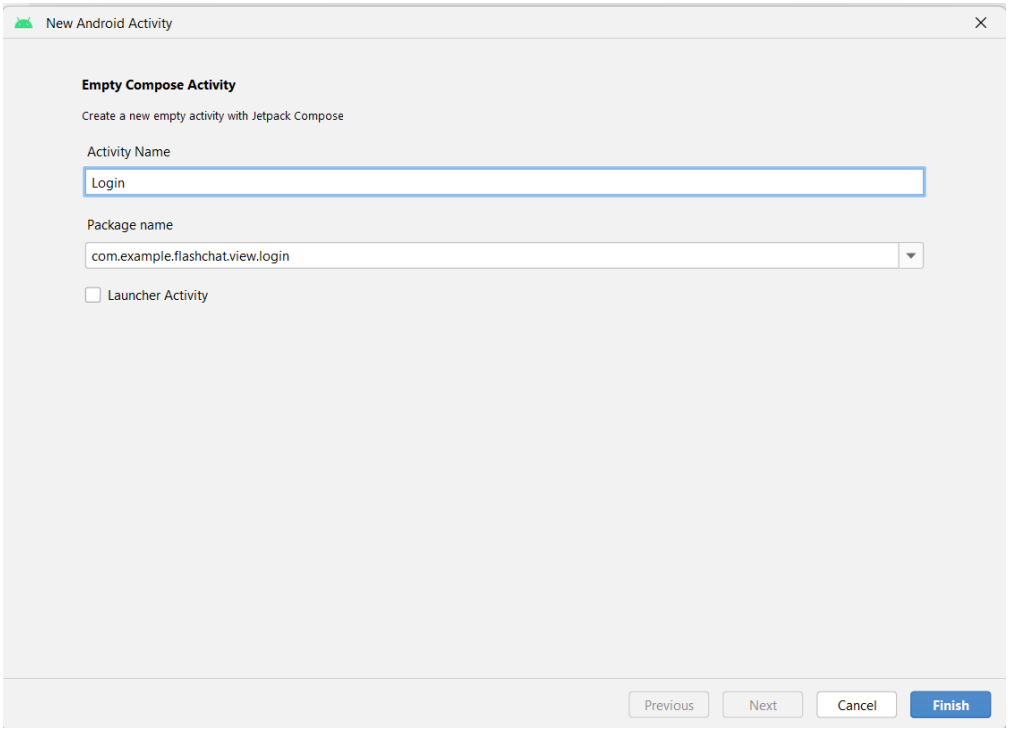




Creating Login.kt file

view package>login package>right click>new>compose>empty compose activity

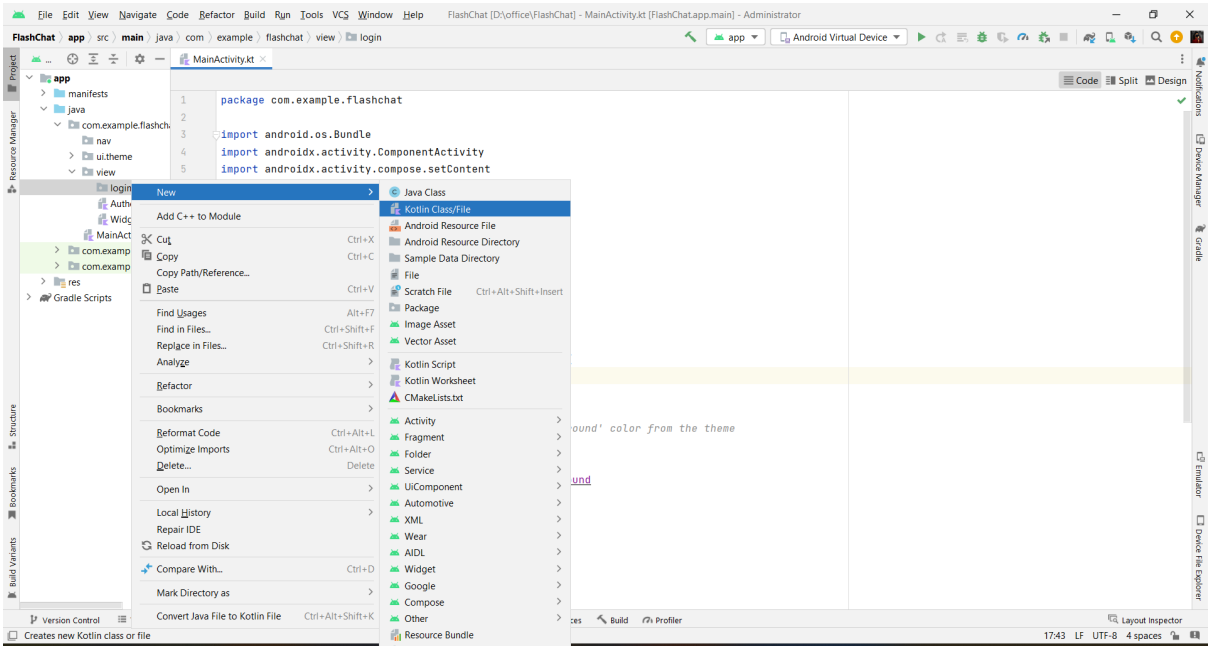


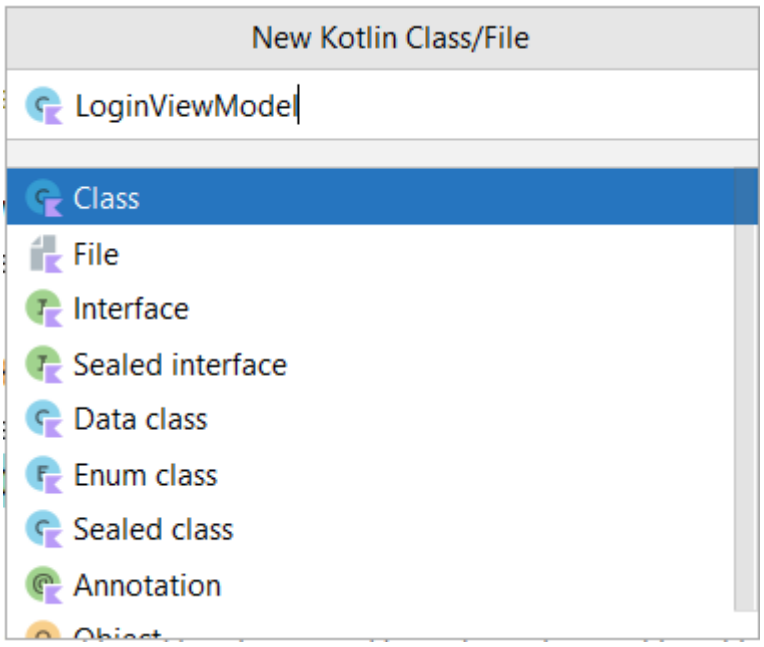



```kotlin
package com.project.pradyotprakash.flashchat.view.login

import ...

/**
 * The login view which will help the user to authenticate themselves and go to the
 * home screen to show and send messages to others.
 */

@Composable
fun LoginView(
    home: () -> Unit,
    back: () -> Unit,
    loginViewModel: LoginViewModel = viewModel()
) {
    val email: String by loginViewModel.email.observeAsState( initial: "")
    val password: String by loginViewModel.password.observeAsState( initial: "")
    val loading: Boolean by loginViewModel.loading.observeAsState(initial = false)

    Box(
        contentAlignment = Alignment.Center,
        modifier = Modifier.fillMaxSize()
    ) { this: BoxScope
        if (loading) {
            CircularProgressIndicator()
        }
        Column(
            modifier = Modifier.fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Top
```

Complete Code for Login.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/login/Login.kt

Creating LoginViewModel class
view package>login package>right click>new>kotlin class/file


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
FlashChat [D:\office\FlashChat] - MainActivity.kt [FlashChat.app.main] - Administrator
MainActivity.kt
Code Split Design
package com.example.flashchat
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
Image Asset
Vector Asset
Kotlin Script
Kotlin Worksheet
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
AIDL
Widget
Google
Compose
Other
Resource Bundle
Creates new Kotlin class or file

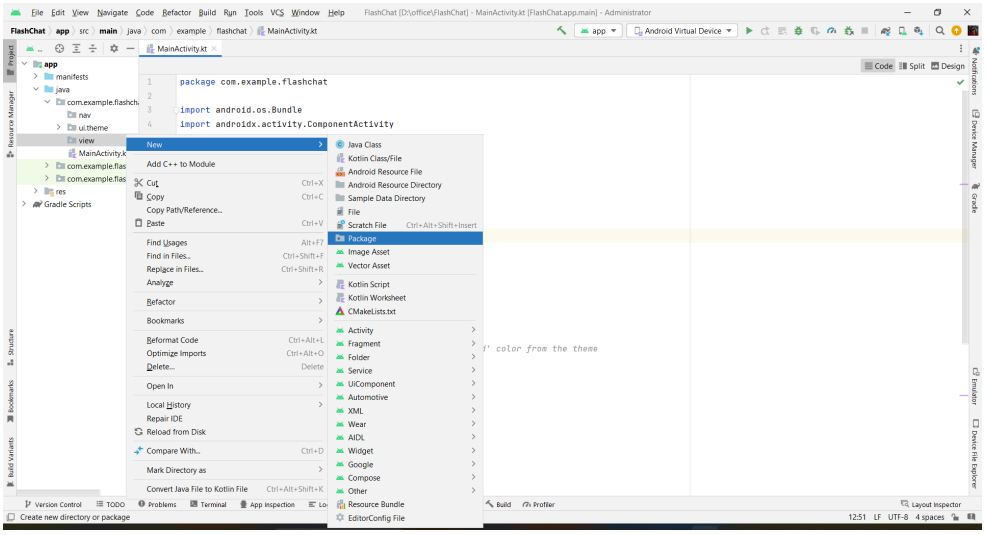

New Kotlin Class/File
LoginViewModel
Class
File
Interface
Sealed interface
Data class
Enum class
Sealed class
Annotation

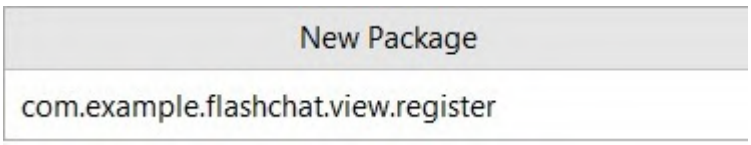
```kotlin
package com.project.pradyotprakash.flashchat.view.login

import ...

/**
 * View model for the login view.
 */
class LoginViewModel : ViewModel() {
    private val auth: FirebaseAuth = Firebase.auth

    private val _email = MutableLiveData( value: "")
    val email: LiveData<String> = _email

    private val _password = MutableLiveData( value: "")
    val password: LiveData<String> = _password

    private val _loading = MutableLiveData( value: false)
    val loading: LiveData<Boolean> = _loading

    // Update email
    fun updateEmail(newEmail: String) {
        _email.value = newEmail
    }

    // Update password
    fun updatePassword(newPassword: String) {
        _password.value = newPassword
    }

    // Register user
```

Complete Code for LoginViewModel.kt file : https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/login/LoginViewModel.kt

3.Creating register package in view package

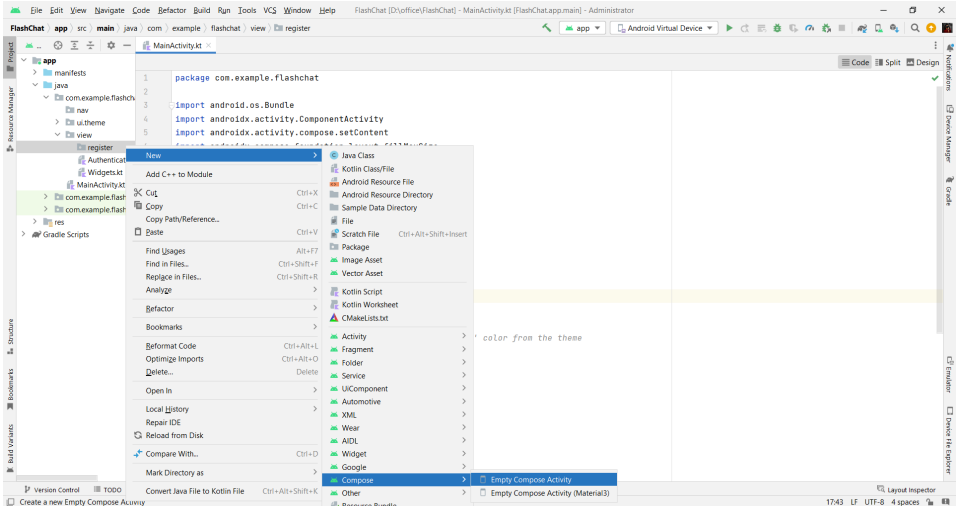
view package>right click>new>package

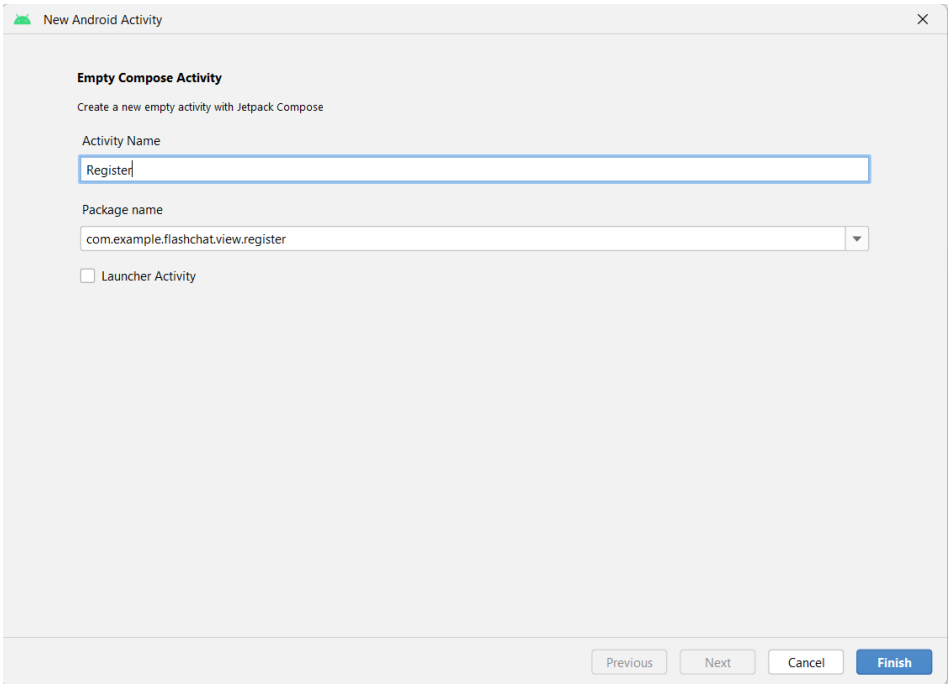




Creating Register.kt file

view package>register package>right click>new>compose>empty compose activity

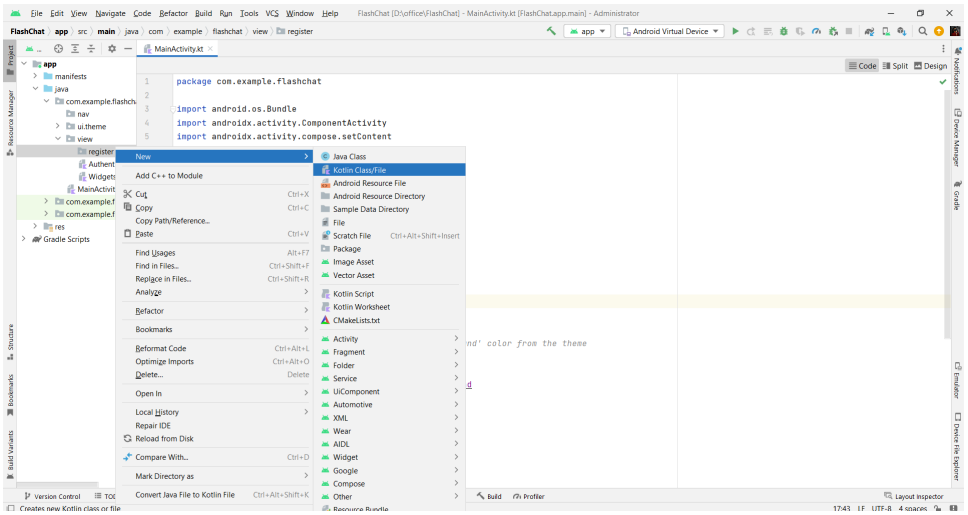


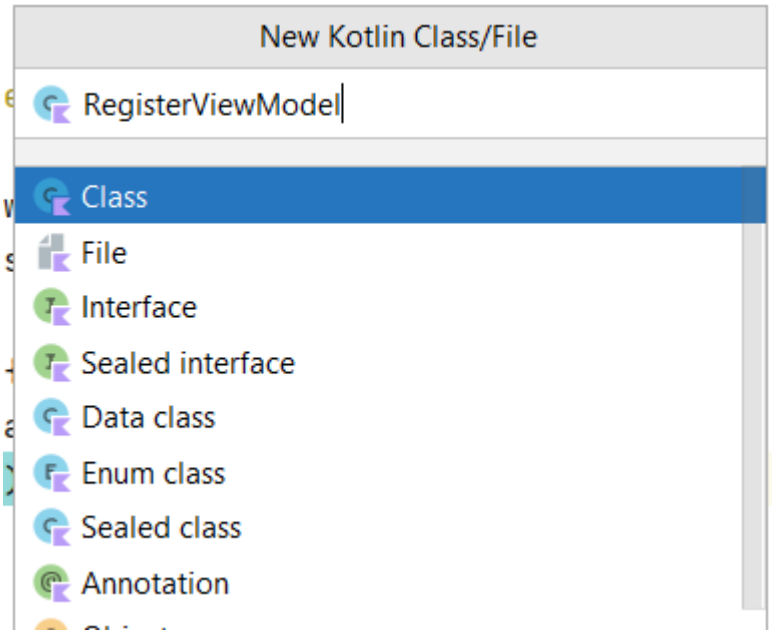


```kotlin
package com.project.pradyotprakash.flashchat.view.register

import ...

/**
 * The Register view which will be helpful for the user to register themselves into
 * our database and go to the home screen to see and send messages.
 */

@Composable
fun RegisterView(
    home: () -> Unit,
    back: () -> Unit,
    registerViewModel: RegisterViewModel = viewModel()
) {
    val email: String by registerViewModel.email.observeAsState( initial: "")
    val password: String by registerViewModel.password.observeAsState( initial: "")
    val loading: Boolean by registerViewModel.loading.observeAsState(initial = false)

    Box(
        contentAlignment = Alignment.Center,
        modifier = Modifier.fillMaxSize()
    ) { this: BoxScope
        if (loading) {
            CircularProgressIndicator()
        }
        Column(
            modifier = Modifier.fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Top
```

Complete Code for Register.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/register/Register.kt

Creating RegisterViewModel class
view package>register package>right click>new>kotlin class/file





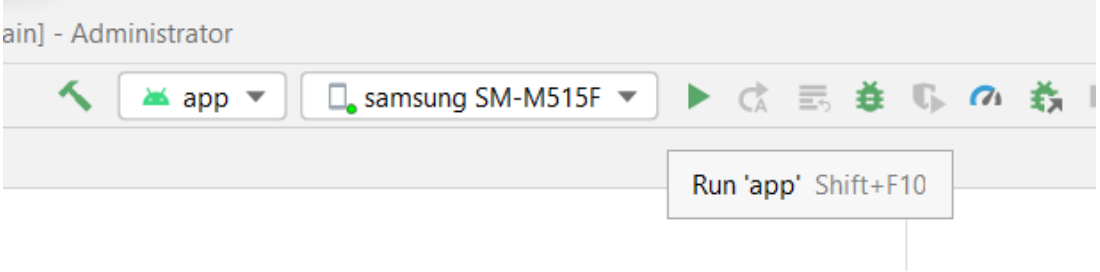

```kotlin
package com.project.pradyotprakash.flashchat.view.register

import ...

/**
 * View model for the login view.
 */
class RegisterViewModel : ViewModel() {
    private val auth: FirebaseAuth = Firebase.auth

    private val _email = MutableLiveData( value: "")
    val email: LiveData<String> = _email

    private val _password = MutableLiveData( value: "")
    val password: LiveData<String> = _password

    private val _loading = MutableLiveData( value: false)
    val loading: LiveData<Boolean> = _loading

    // Update email
    fun updateEmail(newEmail: String) {
        _email.value = newEmail
    }

    // Update password
    fun updatePassword(newPassword: String) {
        _password.value = newPassword
    }

    // Register user
```

Complete Code for RegisterViewModel.kt file :
https://github.com/smartinternz02/Chat-Connect-App/blob/master/app/src/main/java/com/project/pradyotprakash/flashchat/view/register/RegisterViewModel.kt

Task 5 :

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

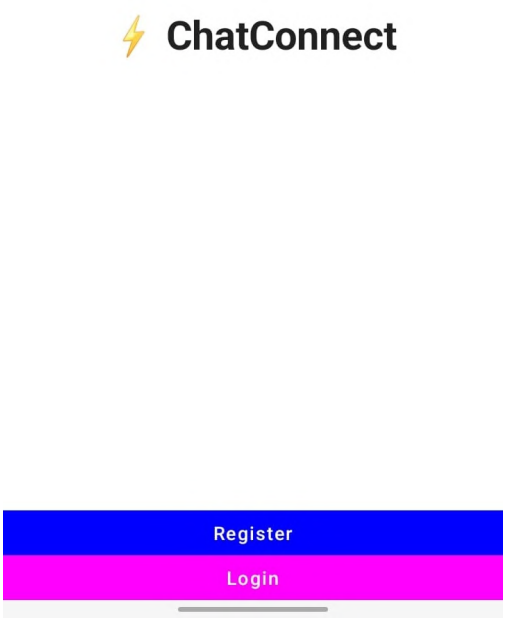


Complete Project Link:
https://github.com/smartinternz02/Chat-Connect-App

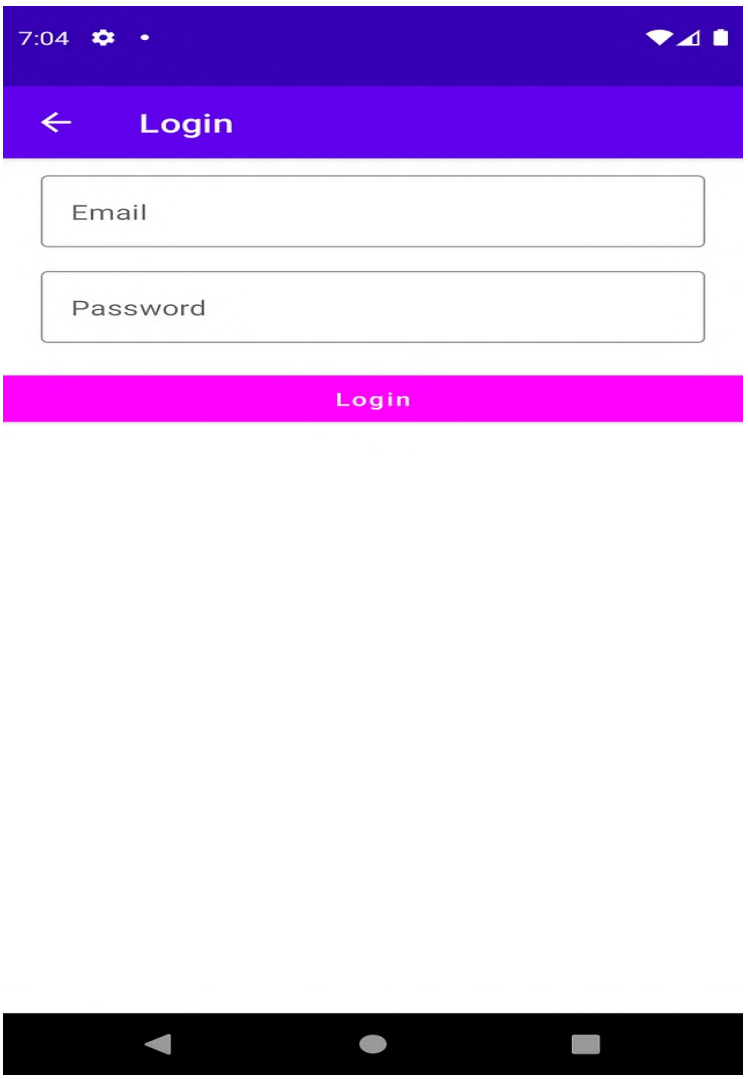
Final Output of the Application :

This App looks better in Mobile Light theme.

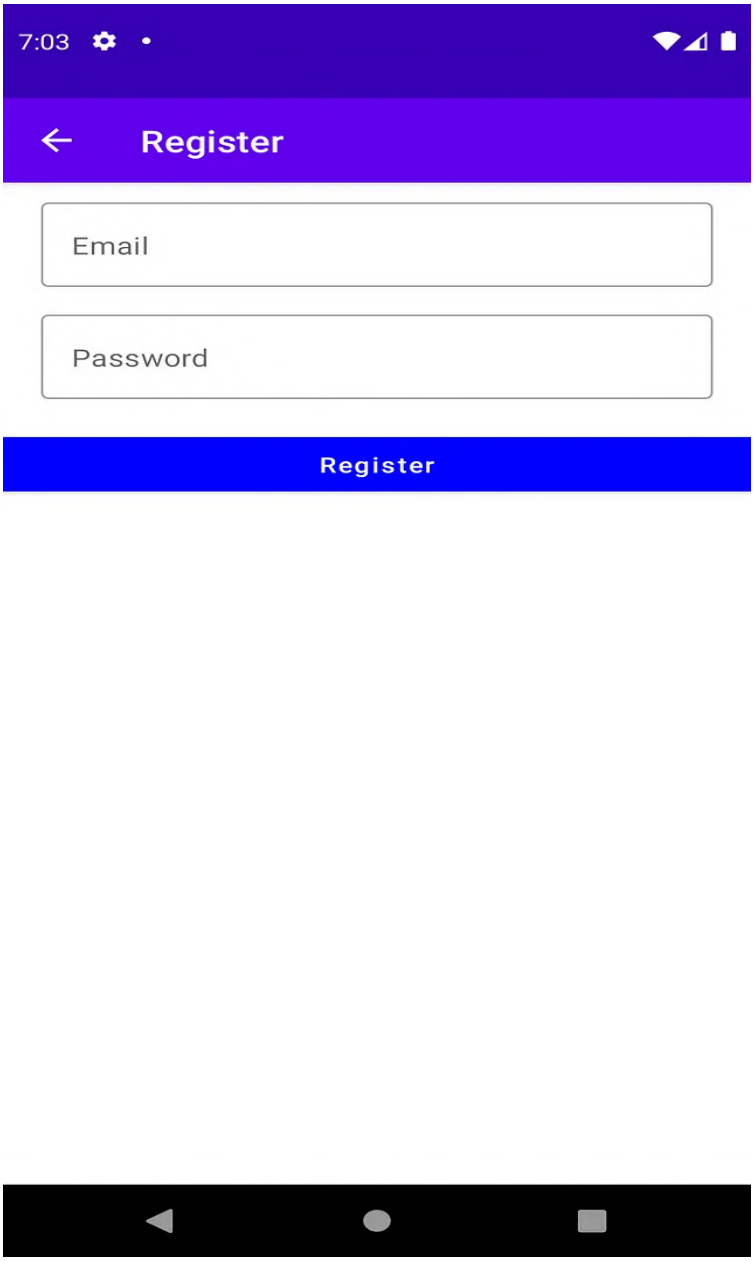
First Screen:



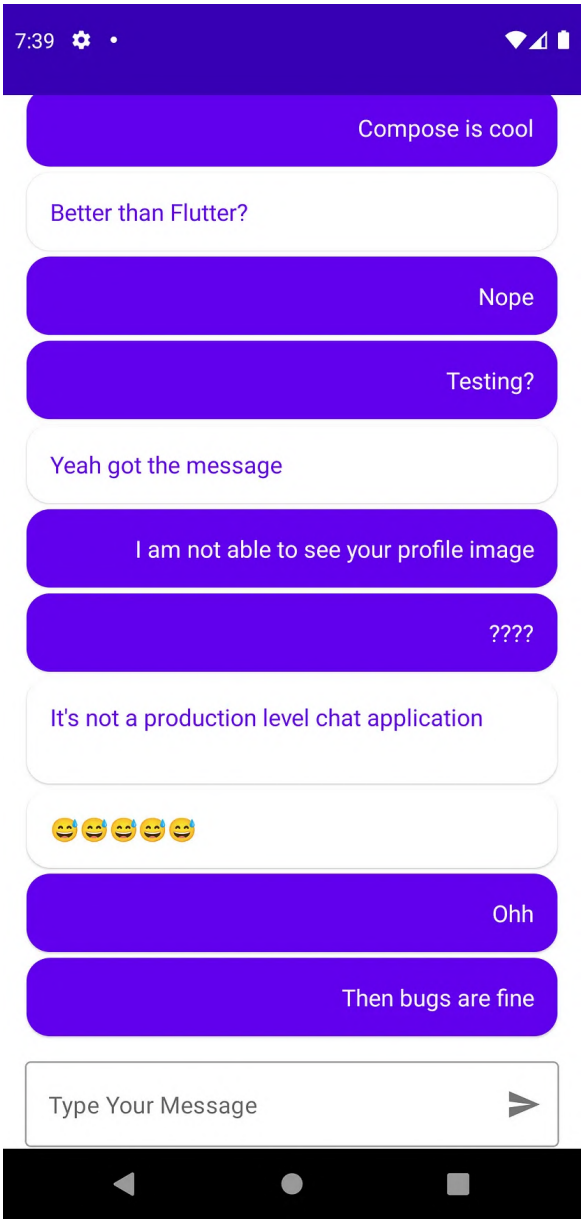
Login Page :



RegisterPage :



Home Screen:


7:39
Compose is cool
Better than Flutter?
Nope
Testing?
Yeah got the message
I am not able to see your profile image
????
It's not a production level chat application
😅😅😅😅😅
Ohh
Then bugs are fine
Type Your Message