Code An Android Chat App With Kotlin

Learn more about Kotlin programming language with The Assembly. Workshop conducted by Shaham Kampala & Fayiza.

Software Used:

1. Android Studio

Workshop Skill Break-up

0% Hardware and 100% Software

Workshop Overview

By using Firebase & Kotlin, you will be able to create a chat app. Users can enter their username, email and password to sign up and chat with the other users who are registered within the specific Firebase database.

References and Tutorials:

1. https://www.youtube.com/playlist?list=PL0dzCUj1L5JE-jiBHjxlmXEkQkum_M3R-

---

Things you need to install before the workshop:

1. Android Studio (preferably 3.5.3 version)
2. Make sure 2 simulators are installed by going to Android Studio and from the menu bar choose Tools > AVD Manager. Add any 2 devices.

Potential Errors:

- During download, path saved should not contain whitespace
- If in the Android view, 'build.gradle (Module: app)' does not show up:

  Go to 'View' > 'Tool Windows' > 'Project'

On the right, where your project name is with (root) next to it > Right Click > 'Ignore Gradle project' > 'Unignore Gradle project' .

Check your Android view again. It should show 'build.gradle (Module:app)

- For any missing packages, Android Studio will prompt you to install the missing packages in the error message. Simply click 'Install the package' and accept the agreement.
- If you cannot find the 'hint' bar, scroll down to 'All Attributes'.
- If you have trouble connecting Firebase make sure the code below is added to build.gradle(Module.app) at the last line.

```
apply plugin: 'com.google.gms.google-services'
```

Part 1: Introduction to Kotlin

What is Kotlin?

Kotlin is an open-source, statically typed programming language based on the Java virtual machine and it works across different platforms. It was first introduced in 2016 and hence continued to be used after that. Both Java and Kotlin go hand in hand for Android development. Although Java is almost ruling the enterprise world, Kotlin is emerging as the most loved programming language among Android developers.

Why Kotlin?

Kotlin has improvised the quality of Java in a few ways. Firstly, it is more concise than Java, meaning that a class which consumes more than 50 lines of code in Java can actually be brought down to 2-3 lines of code in Kotlin. This could be because there would be no need to specify the getters and setters, hash codes and more, as they can be easily generated by Kotlin itself . Secondly, the central focus of Kotlin is to enable mixed language projects and it also introduces improved syntax as well as concise expressions and abstractions. It is an added advantage as the developers who know Java can easily learn Kotlin. Therefore, switching from Java to Kotlin is easy.

Part 2: A sample Kotlin app

We will demonstrate how to develop your first Kotlin app. It is going to a simple inout / output application.

Firstly we will be working on the front-end which is the designing of what the end user sees. The layout is developed in the xml where you can work on the automatically generated layout type which is the ConstraintLayout. In the design tab you can drag and drop to create your desired user interface. I will be showing you how to do it on a Relative Layout . Then we add the other Views inside the layout.

1. We will use the default ConstraintLayout
2. Adjust the width and the height of the layout

```
android:layout_width="match_parent"
android:layout_height="match_parent"
```

3. Add the Title using a TextView and make sure to set the text

```
<TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:padding="5dp"
        android:text="SAMPLE KOTLIN APPLLICATION"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

4. To receive inputs from user we will need an EditText

5. Add a button and another text which will be for displaying the data which you just entered.

6. In the Main Activity we will be managing the logic and make it functioning

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


        addBtn.setOnClickListener(object: View.OnClickListener {
            override fun onClick(v: View?) {
                val msg = editText.text.toString()
                txtResult.setText(msg)
                v?.setBackgroundColor(Color.MAGENTA)
            }
        })
    }
}
```

Part 3: A Kotlin Chat app

Step 1: Sign up & sign in

- Open Android Studio IDE
- From the tabs above, go to File' > 'New Project' > 'Empty Activity. Name your app and the package name as desired. Set the minimum API level to API 16. Click 'Finish'.
- In the Android view on the left, under 'Gradle scripts' go to 'build.gradle (Module:app)'
- In the code, make sure to change:

```
compileSdkversion to 28
```

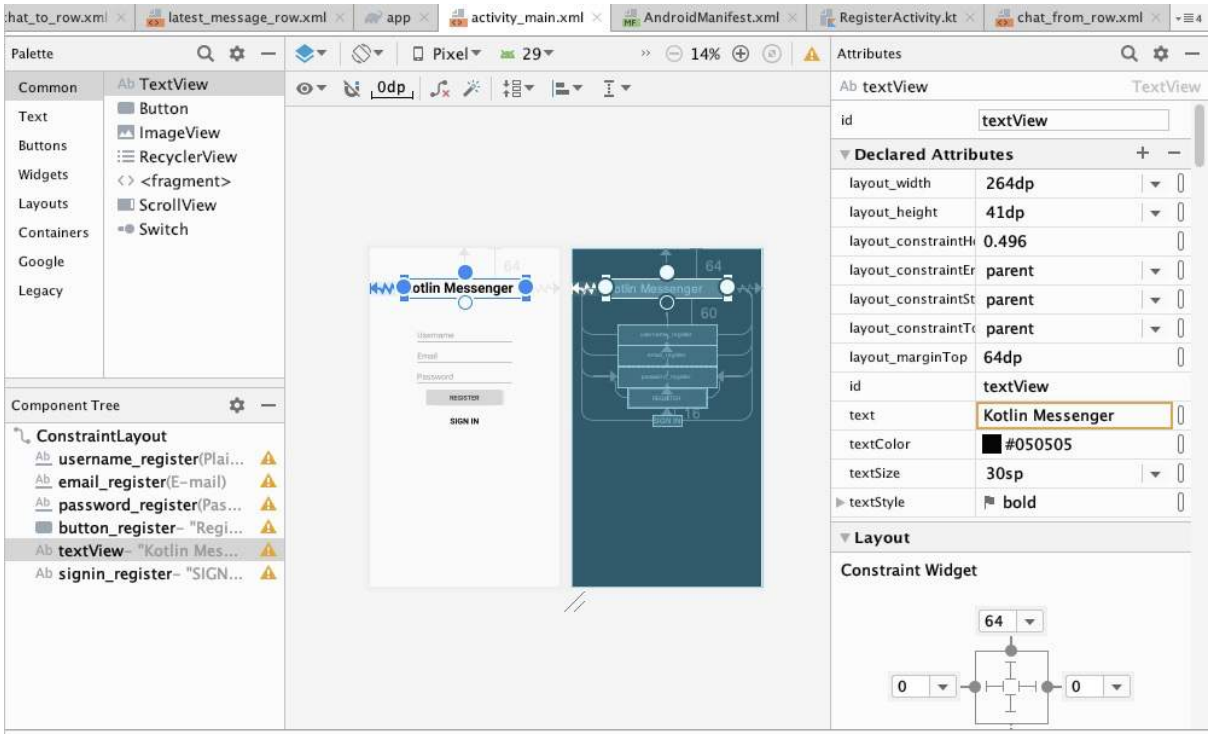
```
targetSdkversion to 28
```

```
android {
    compileSdkVersion 28
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.skapps12345"
        minSdkVersion 16
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

- Click 'Sync Project with Gradle Files'
- Make sure all the dependencies are installed:

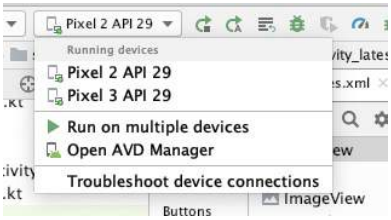
```
implementation 'com.google.firebase:firebase-analytics:17.2.0'
implementation 'com.google.firebase:firebase-auth:19.2.0'
implementation 'com.google.firebase:firebase-database:19.2.0'
implementation 'com.google.firebase:firebase-crash:16.2.1'
implementation "com.xwray:groupie:2.7.0"
implementation "com.xwray:groupie-kotlin-android-extensions:2.7.0"

implementation 'com.android.support:recyclerview-v7:28.0.0-alpha3'
```

- On the left go to 'res' > 'layout' > 'activity_main.xml'. This is the front screen of your app. Design it accordingly as shown below:



- Under 'Palette' use the search bar to search for elements. Drag and drop the elements on the left to the middle of your screen. Add a 'Sign Up' button, and add a 'Sign in' button.
- On the right, go to 'Declared Attributes' > 'text'. Make this field completely empty for the username, email and password only.

- Find the 'hint' bar and change that to give the user a hint on what to type. This will show in greyed-out text for the user.
- Find the 'id' bar to give your component a unique variable that you can use in your code.
- Go to 'Layout' on the right. In your 'Constraint Widget', set the height for the topmost component.
- You must link the components that are below by dragging and dropping.
- To run your program go to 'No devices' > 'Open AVD Manager' > 'Create new device'



- Choose any device from the Pixel devices
- Choose any system image desired. Download it
- Give your AVD a name and click 'Finish'
- Under 'java' > 'com.example.ko' > 'Main Activity'. Right-click it to go to 'Refactor' > 'Rename'. Rename it as RegisterActivity.kt

---

- Go to 'layout' > create a new resource page. This is to create our login page
- Name your file activity_login.xml
- Under your file, right click to create a new Kotlin File/Class. Name it LoginActivity. We are creating a new class for code in our login page.
- onCreate function (replace any 'main' variables with 'login')

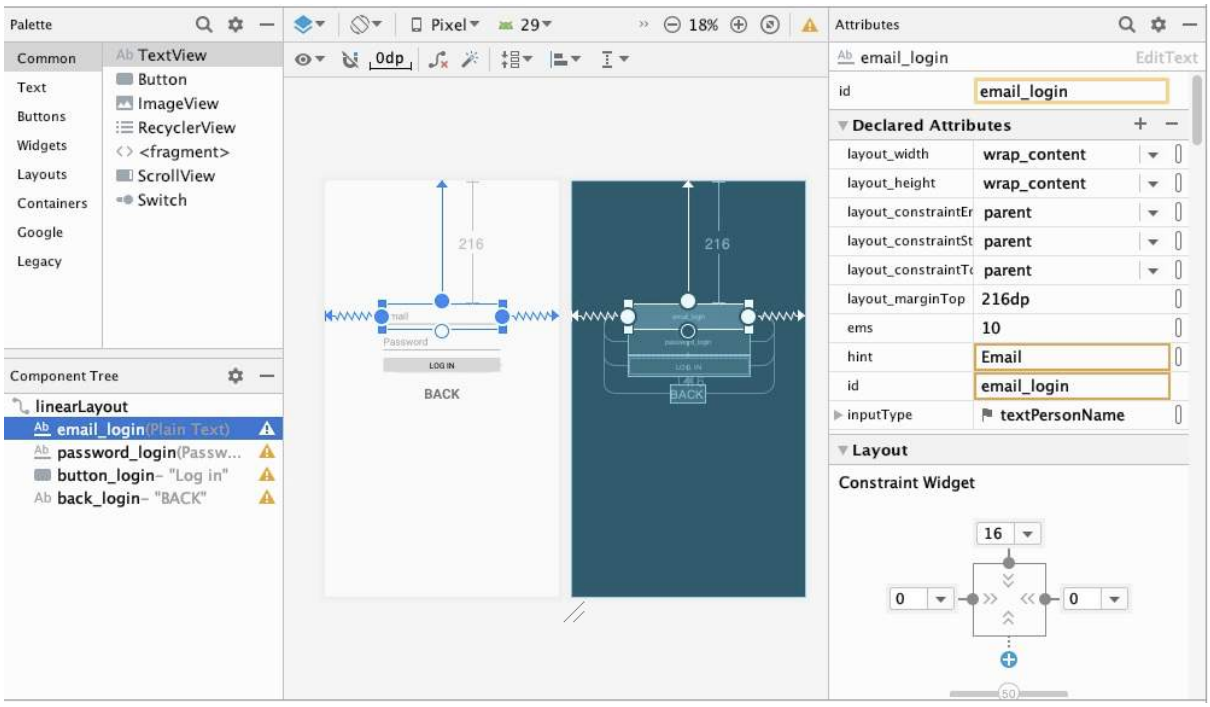
```
class RegisterActivity : AppCompatActivity() {

}
```

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

- Go to Manifest file and add just before /application

```
<activity android:name=".LoginActivity">

</activity>
```

- Go back to your activity_login.xml to edit it and add the components as shown below:



- Under 'Component Tree', make sure when you hover over it is 'Constraint layout'. If not, change its view manually into 'Constant Layout'
- We are now creating the login screen. Add widgets for a username, password, and log in button. Repeat previous steps for design.
- Add a 'Back' button to go back to the main signup page.
- Then run the app to see your results, check your virtual device on Pixel API.
- In LoginActivity.kt, this will be the final code:

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



        button_login.setOnClickListener {

            val email = email_login.text.toString()
            val password = password_login.text.toString()

            if(email.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Please enter text in email/pw", Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            Log.d("LoginActivity", "Attempt to login with email/pw $email/***")

            FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password).addOnCompleteListener {
                if (!it.isSuccessful)   return@addOnCompleteListener

                //else
                Log.d("LoginActivity", "Successfully logged in! ${it.result?.user?.uid}")
                val intent = Intent(this, LatestMessages::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or((Intent.FLAG_ACTIVITY_NEW_TASK)) //clears the stack or when y
                startActivity(intent)


            }.addOnFailureListener {
                Log.d("LoginActivity", "Failed to login :( ${it.message}")
            }


        }

        back_login.setOnClickListener {
            finish()
```

```
        }
    }
```

Step 2: Firebase Authentication

- Navigate to the Firebase Page https://firebase.google.com
- When you reach the page click on add project
- Give a name of your project
- Create project
- Click on Android icon
- Enter the same package name from Android manifest file
- Add app nick name
- Register app
- Download google service.JSON file and drag and drop to android studio
- In android studio tab choose Project mode, and drag and drop the JSON file into the app directory.
- Change to android mode
- Sync build gradle file
- Copy and paste the the lines of code shown on the firebase webpage
- Repeat the same step and copy the implementation line from firebase webpage into the app gradle file
- Then we sync everything
- Navigate to android studio
- We will do the authentication now
- Add a new dependencies in the build.gradle file

```
implementation 'com.google.firebase:firebase-auth:19.2.0'
```

- Sync now
- Navigate to the Register Activity Java file
- Follow the steps from the firebase tab on the right
- To give function to the register button , we need to create a method called performRegister() in the RegisterActivity.kt file:

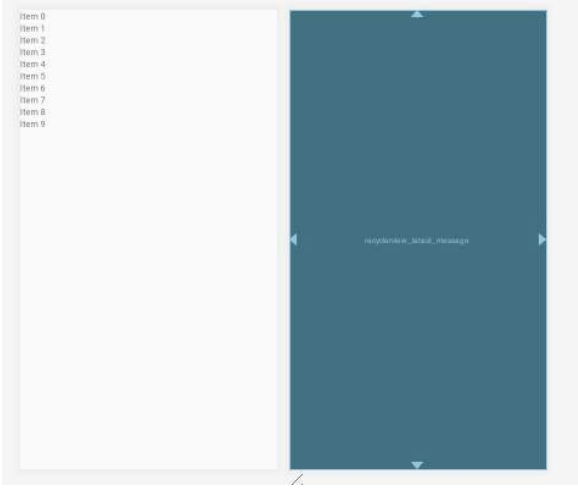
```
private fun performRegister() {

        val username  = username_register.text.toString()
        val email = email_register.text.toString()
        val password = password_register.text.toString()


        if(email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please enter text in email/pw", Toast.LENGTH_SHORT).show()
            return
        }

        Log.d("RegisterActivity", "Email is: " + email)
        Log.d("RegisterActivity", "Password is: $password")

        //Firebase authentication

        FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener {
                if (!it.isSuccessful) return@addOnCompleteListener

                //else if successful
                Log.d("RegisterActivity", "Successfully created user with uid: ${it.result?.user?.uid} ")

            }
            .addOnFailureListener{
```

```
                Log.d("RegisterActivity", "Fail to create user: ${it.message}")
                Toast.makeText(this, "Fail to create user: ${it.message}", Toast.LENGTH_SHORT).show()
            }
    }
```

- Run your app and check.
- Make sure these dependencies are added:

```
    implementation "com.xwray:groupie:2.7.0"
    implementation "com.xwray:groupie-kotlin-android-extensions:2.7.0"
```

- Create LatestMessages.kt file, so that when the user is successfully signed-in they will be taken to the activity_latest_messages.
- Right click layout > New > Activity > Empty Activity. This will create both the kotlin file and the layout resource file in one-click.
- Now make the design accordingly.



- Add the dependency for recycler view in the build.gradle (Module: app)

```
implementation 'com.android.support:recyclerview-v7:28.0.0-alpha3'
```

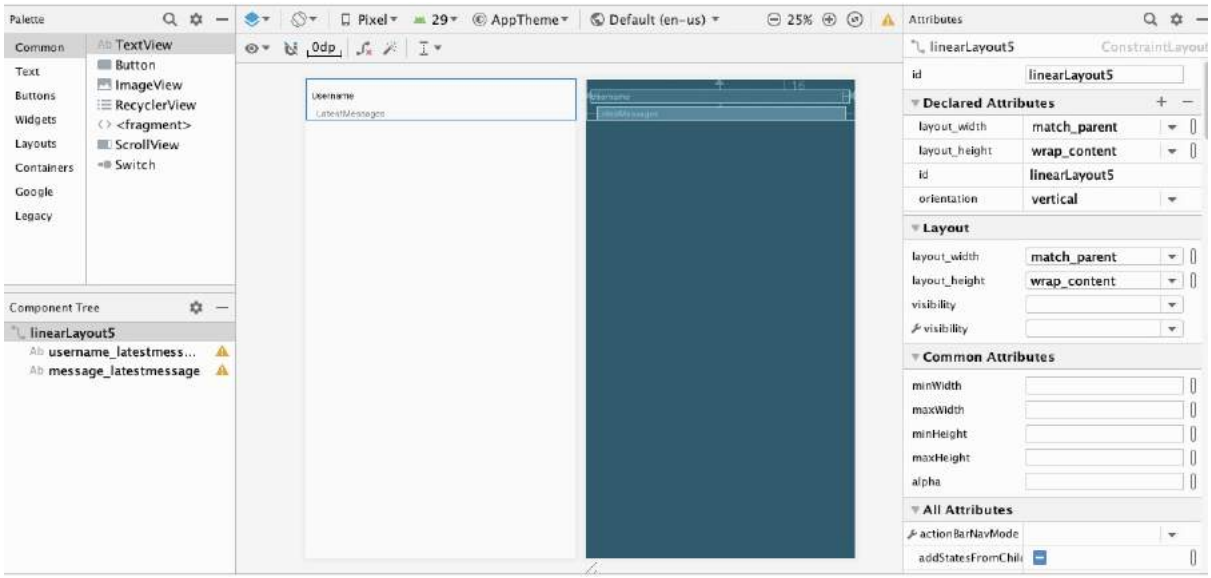
- In the activity_latest_messages.xml, drag & drop the Recycler View.

Recycler view may not show the attributes section. If this happens, try invalidating caches and restarting by going into File > Invalidate Caches / Restart

- Set the layout width and height to match_constraint in the attributes section
- Next, set the layoutmanager for the recyclerview as linear layout manager by entering the code below into the activity_latest_messages.xml.

```
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

- Rename recyclerview to recyclerview_latest_message
- Now, in order to design each rows for the recyclerview, we need to make another layout resource file by going into layout > New > Layout Resource File. Name the file as latest_message_row.
- Drag and drop the components as shown below and give it appropriate names



Make sure the layout_height for latest_message_row is wrap_content. For the rows inside recycler view, the height should be set to wrap_content otherwise each rows will be covering other rows and the rows won't be visible.

==========

- Now go back to the RegisterActivity.kt file. We have to save the user to firebase database.
- Make a User class by adding the below code to RegisterActivity.kt

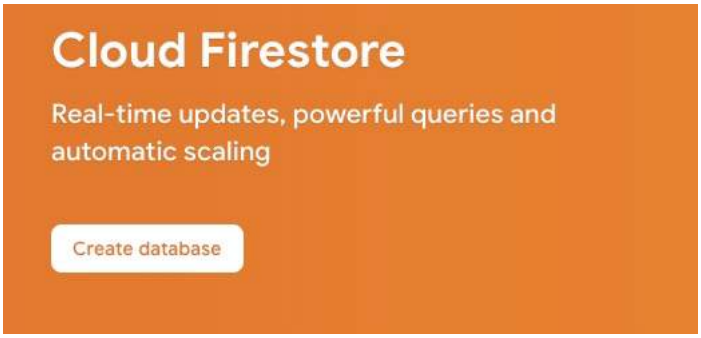
```
class User(val uid: String, val username: String) {
    constructor() : this("", "")
}
```

Step 3: Save user to Firebase database

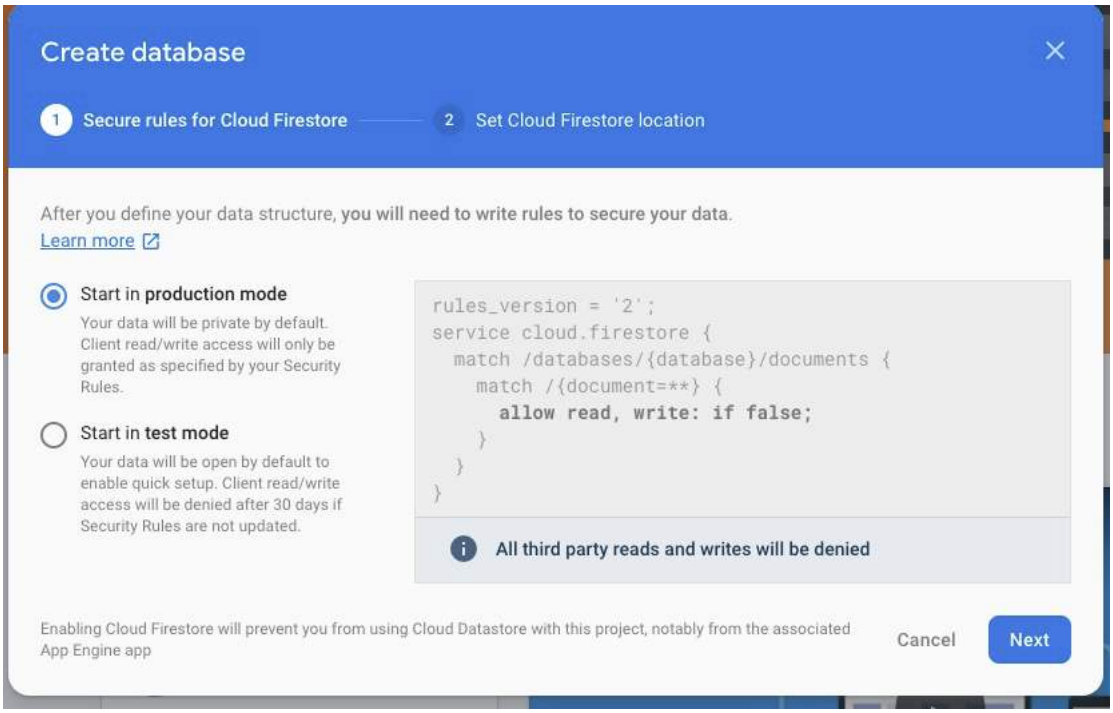
- Make the saveUserToFirebaseDatabase() function by adding the below code.

```
private fun saveUserToFirebaseDatabase() {
        val uid = FirebaseAuth.getInstance().uid ?: ""
        val ref =  FirebaseDatabase.getInstance().getReference("/users/$uid")

        val user = User(uid, username_register.text.toString())
        Log.d("RegisterActivity", "Saving user to database...")
        ref.setValue(user).addOnSuccessListener {
            Log.d("RegisterActivity", "Finally the user is saved to database")

            val intent = Intent(this, LatestMessages::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or((Intent.FLAG_ACTIVITY_NEW_TASK)) //clears the stack or when you c
            startActivity(intent)


        }
            .addOnFailureListener {
                Log.d("RegisterActivity", "Failed saving user to database")
            }
    }
```

- Now we have to check whether the user gets saved to the Firebase Database. Go to the Firebase console and create a database.



- Follow the prompt to setup the Firebase database & choose realtime database



If you run the app now and get the error "Failed saving user to database", change the rules of the firebase database with the code shown below & click publish.

```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    ".read": "auth!=null",
    ".write": "auth!=null"
  }
}
```

- Now go into the RegisterActivity.kt & LoginActivity.kt and make sure the code below is there. This code is for the app to go into the LatestMessages screen when the user successfully logs in.

```
val intent = Intent(this, LatestMessages::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or((Intent.FLAG_ACTIVITY_NEW_TASK)) //clears the stack or when y
                startActivity(intent)
```

- Make the LatestMessages activity your main launcher now for easier testing purposes by adding the code below to AndroidManifest.xml inside the LatestMessages activity.

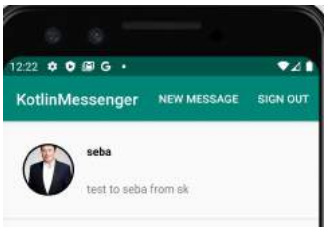
```
<intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Now in the LatestMessages.kt, add the verifyUserIsLoggedIn() method

```
private fun verifyUserIsLoggedIn() {
        val uid =  FirebaseAuth.getInstance().uid
        if (uid == null) {
            val intent = Intent(this, RegisterActivity::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
            startActivity(intent)
        }
    }
```

Step 4: Make the top menu bar

- Now we need to make buttons on the top panel as shown below:



- Now we have to create a new resource called menu by right clicking res > New > Directory > menu
- Right click menu > New > Menu resource file and name it nav_menu
- Drag and drop the component required as shown above in the screenshot.
- Make sure your nav_menu.xml is similar to the screenshot below:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/menu_newmessage"
        android:title="New Message"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/menu_signout"
        android:title="Sign Out"
        app:showAsAction="ifRoom" />
</menu>
```

- Now in the LatestMessagesActivity.kt override the fucntions

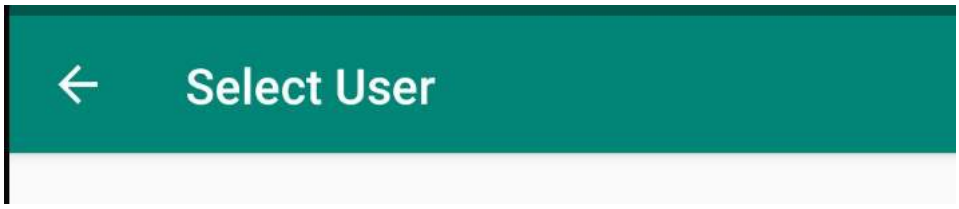
```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.nav_menu, menu)
        return super.onCreateOptionsMenu(menu)
    }
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item?.itemId) {
            R.id.menu_newmessage -> {
                val intent  =  Intent(this, NewMessageActivity::class.java)
                startActivity(intent)

            }
            R.id.menu_signout -> {
```

```
                FirebaseAuth.getInstance().signOut()
                val intent = Intent(this, RegisterActivity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
                startActivity(intent) //to go back to main page after sign out
            }
        }
        return super.onOptionsItemSelected(item)
    }
```

==========

- Now to have the back button for the NewMessageActivity as shown below in the screenshot:



- Go to manifest file. Add the code below to set the PARENT

```
                <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".LatestMessages" />
```

- To add the title to the nav bar add the code below:

```
supportActionBar?.title = "Select User"
```

- In the new message activity drag and drop the recycler view and set the constraints accordingly.
- Set the recycler view layout manager as LinearLayoutManager by adding the code below to the activity_new_message.xml

```
app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
```

Step 5: Implementing the RecyclerView

- We are going to use the groupie recycler view library for easier implementation of recyclerview

```
    implementation "com.xwray:groupie:2.7.0"
    implementation "com.xwray:groupie-kotlin-android-extensions:2.7.0"
```

- In the NewMessageActivity.kt we will add the function fetchUsers()

```
private fun fetchUsers(){
        val ref = FirebaseDatabase.getInstance().getReference("/users")
        ref.addListenerForSingleValueEvent(object: ValueEventListener {
            override fun onDataChange(p0: DataSnapshot) {

                val adapter = GroupAdapter<GroupieViewHolder>()

                p0.children.forEach {
                    Log.d("NewMessage", it.toString())
                    val user = it.getValue(User::class.java)
                    if (user!= null) {
                        adapter.add(UserItem(user))
                    }

                }

                adapter.setOnItemClickListener{item, view->

                    val intent = Intent(view.context, ChatLogActivity::class.java)
                    startActivity(intent)
                    finish()
```

```
                }

                recyclerview_newmessage.setAdapter(adapter)

            }
            override fun onCancelled(p0: DatabaseError) {

            }
        })

    }
```

- We need to UserItem class also:

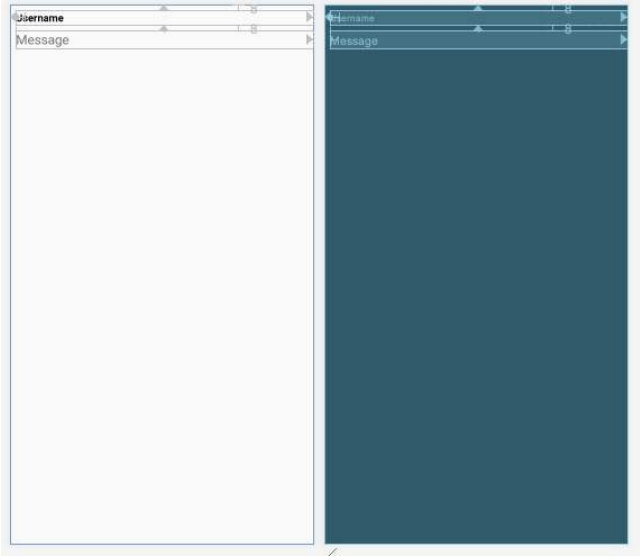
```
class UserItem(val user: User): Item<GroupieViewHolder>() {

        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
            //this will be called in our list 4 each user object
            viewHolder.itemView.username_newmessage.text = user.username

        }
        override fun getLayout(): Int {
            return R.layout.user_row_new_message
        }

    }
```

- In the actvity_chat_log.xml we need to have layout similar to the screenshot below:

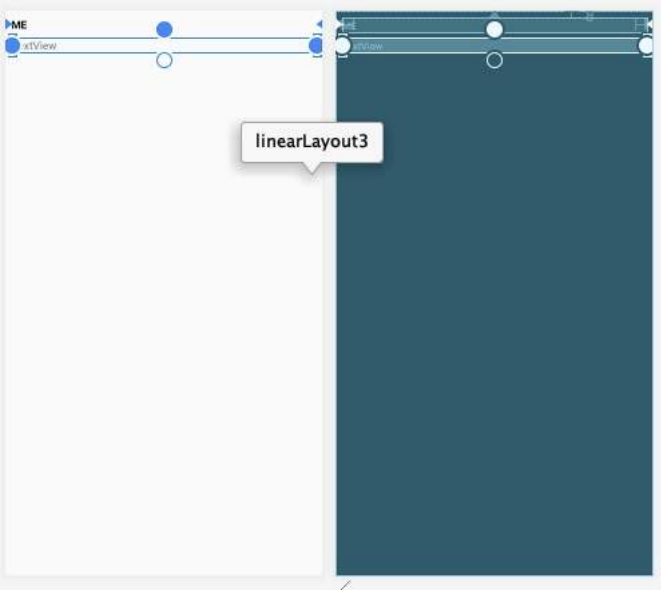
==========

Step 6: Chat message rows and parcelize extension

- Create the chat_from_row layout as shown below using the required components:



- Similarly create chat_to_row layout as shown below using the required components:

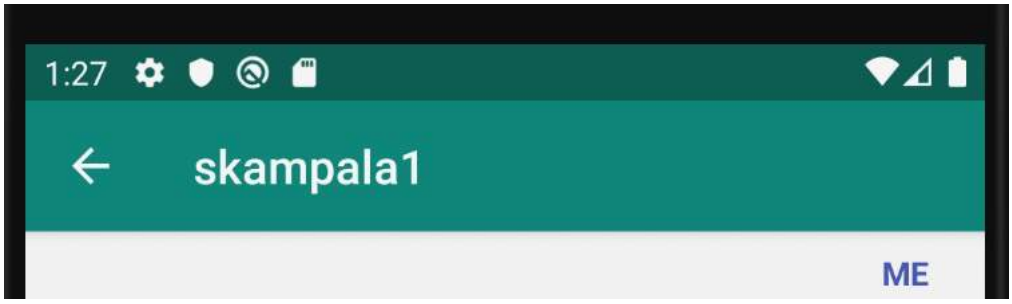


- We need to have separate classes for chat_for_row & chat_to_row as shown below:

```
class ChatFromItem(val text: String, val user: User) : Item<GroupieViewHolder>() {
        override fun bind(viewHolder: GroupieViewHolder, position: Int) {

            val u_name = user.username
            viewHolder.itemView.username_fromrow.text = u_name
            viewHolder.itemView.message_fromrow.text = text
        }

        override fun getLayout(): Int { ///row that renders chat rows
            return R.layout.chat_from_row
        }
    }

    class ChatToItem(val text: String, val user: User) : Item<GroupieViewHolder>() {
        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
            val u_name = user.username
            viewHolder.itemView.username_torow.text = u_name
            viewHolder.itemView.message_torow.text = text
        }

        override fun getLayout(): Int { ///row that renders chat rows
```

```
            return R.layout.chat_to_row
        }
    }
}
```

- Now, if we need to have the name of the user selected in the chat log action bar we need to add some extra code to the NewMessageActivity.kt




```
val userItem = item as UserItem
intent.putExtra(USER_KEY, userItem.user.username)
```

- And we can get the string using this code below at ChatLogActivity.kt

```
val username = intent.getStringExtra(NewMessageActivity.USER_KEY)
```

- We will parcelize to send the whole User object to the ChatLogActivity.
- Now, the user class will be:

```
@Parcelize
class User(val uid: String, val username: String): Parcelable {
    constructor() : this("", "")
}
```


Now, if there's an error asking us to implement writeToParcel(), we need to go to build.gradle (Module: app) and the line below:

```
androidExtensions {
    experimental = true
}
```

- To get the whole user object use this code:

```
toUser = intent.getParcelableExtra<User>(NewMessageActivity.USER_KEY)
```

Step 7 : Send and Listen for Chat Messages

Use groupie third party dependency to get multiple types of row in the recycler view

For Receiving Messages use the bind function

To add functionality to 'SEND' button and send message to firebase

```
private fun performSendMessage() {
        //send message to firebase
       val text = edittext_chat_log.text.toString()
       val fromId = FirebaseAuth.getInstance().uid
```

```
        val user = intent.getParcelableExtra<User>(NewMessageActivity.USER_KEY)
        val toId = user.uid

        if (fromId == null) return
       // val reference = FirebaseDatabase.getInstance().getReference("/messages").push()
        //.push() is for creating a new node

        val reference = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId").push()

        val toReference = FirebaseDatabase.getInstance().getReference("/user-messages/$toId/$fromId").push()


        val chatMessage = ChatMessage(reference.key!!, text,fromId, toId, System.currentTimeMillis() /1000 ) // iOS gives second

        reference.setValue(chatMessage).addOnSuccessListener {
            Log.d(TAG, "Saved our chat message: ${reference.key}")
            edittext_chat_log.text.clear()
            recyclerview_chat_log.scrollToPosition(adapter.itemCount -1)
        }

        toReference.setValue(chatMessage)

        val latestMessageRef = FirebaseDatabase.getInstance().getReference("/latest-messages/$fromId/$toId") //there is no push
        latestMessageRef.setValue(chatMessage)

        val latestMessageToRef = FirebaseDatabase.getInstance().getReference("/latest-messages/$toId/$fromId") //there is no pus
        latestMessageToRef.setValue(chatMessage)
    }
```

insert pic of firebase with all the new properties of data entered

Fetch all these data in recycler view by creating another method which contains 4 other method

```
private fun listenForMessages() {

        val fromId = FirebaseAuth.getInstance().uid
        val toId = toUser?.uid
        val ref = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId")

        //this guy below is going to notify us for every messages in the node
        ref.addChildEventListener(object: ChildEventListener {
            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onChildChanged(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildAdded(p0: DataSnapshot, p1: String?) {
                //we will work on this
              val chatMessage = p0.getValue(ChatMessage::class.java)

                if(chatMessage != null){
                    Log.d(TAG, chatMessage.text)

                    if (chatMessage.fromId == FirebaseAuth.getInstance().uid) {
                        val currentUser = LatestMessages.currentUser ?:return
                        adapter.add(ChatFromItem(chatMessage.text, currentUser))
                    } else {
                        adapter.add(ChatToItem(chatMessage.text, toUser!!))
                    }
                }

                recyclerview_chat_log.scrollToPosition(adapter.itemCount -1)

            }

            override fun onChildMoved(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildRemoved(p0: DataSnapshot) {

            }
        })

    }
```

Create ChatMessage class constructor

```
class ChatMessage(val id: String, val text: String, val fromId: String, val toId: String, val timestamp: Long) {
        constructor() : this("", "", "", "", -1)
    }
```

Create new adapter and set to recycler view

```
val adapter = GroupAdapter<GroupieViewHolder>()
recyclerview_chat_log.adapter = adapter
```

- To add data into the adapter → under the existing function listenForMessages() → look for the onChildAdded() function inside it. Spot the below code

```
                if(chatMessage != null){
                    Log.d(TAG, chatMessage.text)

                    if (chatMessage.fromId == FirebaseAuth.getInstance().uid) {
                        val currentUser = LatestMessages.currentUser ?:return
                        adapter.add(ChatFromItem(chatMessage.text, currentUser))
                    } else {
                        adapter.add(ChatToItem(chatMessage.text, toUser!!))
                    }
                }
```

Step 8 : User to User Messaging

- Introduce new node structure in the firebase to store messages of different user in different nodes
- To separate the to and from messages from the different users
- Under performSendMessage() function→ you will find these chunk of code

```
val reference = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId").push()

       val toReference = FirebaseDatabase.getInstance().getReference("/user-messages/$toId/$fromId").push()
```

Modify the reference for which we are listening for things

```
val fromId = FirebaseAuth.getInstance().uid
        val toId = toUser?.uid
        val ref = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId")
```

To save the messages after logging out

```
toReference.setValue(chatMessage)
```

To clear the edit text after you send the message

```
edittext_chat_log.text.clear()
```

To Scroll to the last message

```
recyclerview_chat_log.scrollToPosition(adapter.itemCount -1)
```

==========

- Into the LatestMessages.kt we need to add the required functions:

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



        recyclerview_latest_message.adapter = adapter
        recyclerview_latest_message.addItemDecoration(DividerItemDecoration(this, DividerItemDecoration.VERTICAL))


        //set item click listener on adapter

        adapter.setOnItemClickListener { item, view ->
            Log.d(TAG, "123")
            val intent = Intent(this, ChatLogActivity::class.java)



            val row = item as LatestMessageRow
            row.chatPartnerUser

            intent.putExtra(NewMessageActivity.USER_KEY,row.chatPartnerUser )
            startActivity(intent)
        }

        listenForLatestMessages()
        fetchCurrentUser()
        verifyUserIsLoggedIn()

    }
```

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.nav_menu, menu)
        return super.onCreateOptionsMenu(menu)
    }
```

```
private fun listenForLatestMessages(){

        val fromId = FirebaseAuth.getInstance().uid
        val ref = FirebaseDatabase.getInstance().getReference("/latest-messages/$fromId")
        ref.addChildEventListener(object: ChildEventListener {
            override fun onChildAdded(p0: DataSnapshot, p1: String?) {
                val chatMessage = p0.getValue(ChatMessage::class.java) ?:return
                latestMessagesMap[p0.key!!] = chatMessage
                refreshRecyclerViewMessages()


            }

            override fun onChildChanged(p0: DataSnapshot, p1: String?) {
                val chatMessage = p0.getValue(ChatMessage::class.java) ?: return
                latestMessagesMap[p0.key!!] = chatMessage
                refreshRecyclerViewMessages()
            }
            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onChildMoved(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildRemoved(p0: DataSnapshot) {

            }
        })


    }
```

```
class LatestMessageRow(val chatMessage: ChatMessage): Item<GroupieViewHolder>() {

        var chatPartnerUser: User? = null

        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
            viewHolder.itemView.message_latestmessage.text = chatMessage.text

            val chatPartnerId: String
```

```
            if (chatMessage.fromId == FirebaseAuth.getInstance().uid) {
                chatPartnerId = chatMessage.toId
            } else {
                chatPartnerId = chatMessage.fromId
            }

            val ref = FirebaseDatabase.getInstance().getReference("/users/$chatPartnerId")
            ref.addListenerForSingleValueEvent(object: ValueEventListener {
                override fun onCancelled(p0: DatabaseError) {

                }

                override fun onDataChange(p0: DataSnapshot) {
                    chatPartnerUser = p0.getValue(User::class.java)
                    viewHolder.itemView.username_latestmessages.text = chatPartnerUser!!.username


                }
            })

        }
        override fun getLayout(): Int {
            return R.layout.latest_message_row
        }
    }
```

```
val latestMessagesMap = HashMap<String, ChatMessage>()

    private fun refreshRecyclerViewMessages() {
        adapter.clear()
        latestMessagesMap.values.forEach {
            adapter.add(LatestMessageRow(it))
        }
    }
```

```
val adapter = GroupAdapter<GroupieViewHolder>()
    private fun fetchCurrentUser() {

        val uid = FirebaseAuth.getInstance().uid
        val ref = FirebaseDatabase.getInstance().getReference("/users/$uid")
        ref.addListenerForSingleValueEvent(object: ValueEventListener {

            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onDataChange(p0: DataSnapshot) {
                currentUser = p0.getValue(User::class.java)
                Log.d(TAG, "Current user: ${currentUser?.username}")
            }
        })

    }
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item?.itemId) {
            R.id.menu_newmessage -> {
                val intent  =  Intent(this, NewMessageActivity::class.java)
                startActivity(intent)

            }
            R.id.menu_signout -> {
                FirebaseAuth.getInstance().signOut()
                val intent = Intent(this, RegisterActivity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
                startActivity(intent) //to go back to main page after sign out
            }
        }
        return super.onOptionsItemSelected(item)
    }


    private fun verifyUserIsLoggedIn() {
        val uid =  FirebaseAuth.getInstance().uid
        if (uid == null) {
            val intent = Intent(this, RegisterActivity::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
            startActivity(intent)
```

```
        }
    }
}
```

Codebase for the Kotlin files:

- RegisterActivity.kt

```
package com.skapps12345

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Parcelable
import android.util.Log
import android.widget.Toast
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.FirebaseDatabase
import kotlinx.android.parcel.Parcelize
import kotlinx.android.synthetic.main.activity_main.*

class RegisterActivity : AppCompatActivity() {

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



        button_register.setOnClickListener {

            performRegister()

        }



        signin_register.setOnClickListener {
            val intent = Intent(this,LoginActivity::class.java )
            startActivity(intent)
        }


    }

    private fun performRegister() {

        val username  = username_register.text.toString()
        val email = email_register.text.toString()
        val password = password_register.text.toString()


        if(email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Please enter text in email/pw", Toast.LENGTH_SHORT).show()
            return
        }

        Log.d("RegisterActivity", "Email is: " + email)
        Log.d("RegisterActivity", "Password is: $password")

        //Firebase authentication
        FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener {
                if (!it.isSuccessful) return@addOnCompleteListener

                //else if successful
                Log.d("RegisterActivity", "Successfully created user with uid: ${it.result?.user?.uid} ")
                saveUserToFirebaseDatabase()

            }
            .addOnFailureListener{
                Log.d("RegisterActivity", "Fail to create user: ${it.message}")
                Toast.makeText(this, "Fail to create user: ${it.message}", Toast.LENGTH_SHORT).show()
            }


    }

    private fun saveUserToFirebaseDatabase() {
        val uid = FirebaseAuth.getInstance().uid ?: ""
        val ref =  FirebaseDatabase.getInstance().getReference("/users/$uid")
```

```
        val user = User(uid, username_register.text.toString())
        Log.d("RegisterActivity", "Saving user to database...")
        ref.setValue(user).addOnSuccessListener {
            Log.d("RegisterActivity", "Finally the user is saved to database")

            val intent = Intent(this, LatestMessages::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or((Intent.FLAG_ACTIVITY_NEW_TASK)) //clears the stack or when you c
            startActivity(intent)


        }
            .addOnFailureListener {
                Log.d("RegisterActivity", "Failed saving user to database")
            }
    }
}


@Parcelize
class User(val uid: String, val username: String): Parcelable {
    constructor() : this("", "")
}
```

- LoginActivity.kt

```
package com.skapps12345

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import kotlinx.android.synthetic.main.activity_login.*

class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)



        button_login.setOnClickListener {

            val email = email_login.text.toString()
            val password = password_login.text.toString()

            if(email.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Please enter text in email/pw", Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            Log.d("LoginActivity", "Attempt to login with email/pw $email/***")

            FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password).addOnCompleteListener {
                if (!it.isSuccessful)   return@addOnCompleteListener

                //else
                Log.d("LoginActivity", "Successfully logged in! ${it.result?.user?.uid}")
                val intent = Intent(this, LatestMessages::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or((Intent.FLAG_ACTIVITY_NEW_TASK)) //clears the stack or when y
                startActivity(intent)


            }.addOnFailureListener {
                Log.d("LoginActivity", "Failed to login :( ${it.message}")
            }


        }

        back_login.setOnClickListener {
            finish()
        }
    }


}
```

- NewMessageActivity.kt

```
package com.skapps12345

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.xwray.groupie.GroupAdapter
import com.xwray.groupie.GroupieViewHolder
import com.xwray.groupie.Item
import kotlinx.android.synthetic.main.activity_new_message.*
import kotlinx.android.synthetic.main.user_row_new_message.view.*

public fun add() {

}

class NewMessageActivity : AppCompatActivity() {


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


        supportActionBar?.title = "Select User"


/*        val adapter = GroupAdapter<GroupieViewHolder>()
        recylcerview_newmessage.setAdapter(adapter)
        //recylcerview_newmessage.layoutManager = LinearLayoutManager(this)

      */
        fetchUsers()




    }
    companion object {
        val USER_KEY = "USER_KEY"
    }
    private fun fetchUsers(){
        val ref = FirebaseDatabase.getInstance().getReference("/users")
        ref.addListenerForSingleValueEvent(object: ValueEventListener {
            override fun onDataChange(p0: DataSnapshot) {

                val adapter = GroupAdapter<GroupieViewHolder>()

                p0.children.forEach {
                    Log.d("NewMessage", it.toString())
                    val user = it.getValue(User::class.java)
                    if (user!= null) {
                        adapter.add(UserItem(user))
                    }
                }

                adapter.setOnItemClickListener{item, view->

                    val intent = Intent(view.context, ChatLogActivity::class.java)
                    val userItem = item as UserItem
                    intent.putExtra(USER_KEY, userItem.user)

                    startActivity(intent)
                    finish()

                }

               recyclerview_newmessage.setAdapter(adapter)

            }
            override fun onCancelled(p0: DatabaseError) {

            }
        })

    }

    class UserItem(val user: User): Item<GroupieViewHolder>() {

        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
            //this will be called in our list 4 each user object
```

```
            viewHolder.itemView.username_newmessage.text = user.username

        }
        override fun getLayout(): Int {
            return R.layout.user_row_new_message
        }

    }
}
```

- ChatLogActivity.kt

```
package com.skapps12345

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.ChildEventListener
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.xwray.groupie.GroupAdapter
import com.xwray.groupie.GroupieViewHolder
import com.xwray.groupie.Item
import kotlinx.android.synthetic.main.activity_chat_log.*
import kotlinx.android.synthetic.main.chat_from_row.view.*
import kotlinx.android.synthetic.main.chat_to_row.view.*

class ChatLogActivity : AppCompatActivity() {


    companion object {
        val TAG = "ChatLog"
    }


    val adapter = GroupAdapter<GroupieViewHolder>()
    var toUser: User? = null

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



        recyclerview_chatlog.setAdapter(adapter)


        toUser = intent.getParcelableExtra<User>(NewMessageActivity.USER_KEY)
        supportActionBar?.title = toUser?.username


        listenForMessages()

        send_chatlog.setOnClickListener {
            performSendMessage()
        }
    }

    private fun listenForMessages() {

        val fromId = FirebaseAuth.getInstance().uid
        val toId = toUser?.uid
        val ref = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId")

        //this guy below is going to notify us for every messages in the node
        ref.addChildEventListener(object: ChildEventListener {
            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onChildChanged(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildAdded(p0: DataSnapshot, p1: String?) {
                //we will work on this
                val chatMessage = p0.getValue(ChatMessage::class.java)

                if(chatMessage != null){
                    Log.d(TAG, chatMessage.text)

                    if (chatMessage.fromId == FirebaseAuth.getInstance().uid) {
```

```
                        val currentUser = LatestMessages.currentUser ?:return

                        adapter.add(ChatFromItem(chatMessage.text,currentUser ))
                    } else {
                       adapter.add(ChatToItem(chatMessage.text, toUser!!))
                    }




                }

                recyclerview_chatlog.scrollToPosition(adapter.itemCount -1)

            }

            override fun onChildMoved(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildRemoved(p0: DataSnapshot) {

            }
        })
    }

    private fun performSendMessage(){

        val text = edittext_chatlog.text.toString()
        val fromId = FirebaseAuth.getInstance().uid
        val user = intent.getParcelableExtra<User>(NewMessageActivity.USER_KEY)
        val toId = user.uid

        if (fromId == null) return
        // val reference = FirebaseDatabase.getInstance().getReference("/messages").push()
        //.push() is for creating a new node

        val reference = FirebaseDatabase.getInstance().getReference("/user-messages/$fromId/$toId").push()

        val toReference = FirebaseDatabase.getInstance().getReference("/user-messages/$toId/$fromId").push()


        val chatMessage = ChatMessage(reference.key!!, text,fromId, toId, System.currentTimeMillis() /1000 ) // iOS gives secon

        reference.setValue(chatMessage).addOnSuccessListener {
            Log.d(TAG, "Saved our chat message: ${reference.key}")
            edittext_chatlog.text.clear()
            recyclerview_chatlog.scrollToPosition(adapter.itemCount -1)
        }

        toReference.setValue(chatMessage)

        val latestMessageRef = FirebaseDatabase.getInstance().getReference("/latest-messages/$fromId/$toId") //there is no push
        latestMessageRef.setValue(chatMessage)

        val latestMessageToRef = FirebaseDatabase.getInstance().getReference("/latest-messages/$toId/$fromId") //there is no pu
        latestMessageToRef.setValue(chatMessage)

    }

    class ChatFromItem(val text: String, val user: User) : Item<GroupieViewHolder>() {
        override fun bind(viewHolder: GroupieViewHolder, position: Int) {



            viewHolder.itemView.message_fromrow.text = text
        }

        override fun getLayout(): Int { ///row that renders chat rows
            return R.layout.chat_from_row
        }
    }

    class ChatToItem(val text: String, val user: User) : Item<GroupieViewHolder>() {
        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
           val u_name = user.username
            viewHolder.itemView.username_torow.text = u_name
            viewHolder.itemView.message_torow.text = text
        }

        override fun getLayout(): Int { ///row that renders chat rows
            return R.layout.chat_to_row
        }
    }
}

class ChatMessage(val id: String, val text: String, val fromId: String, val toId: String, val timestamp: Long) {
```

```
    constructor() : this("", "", "", "", -1)
}
```

- LatestMessages.kt

```
package com.skapps12345

import android.content.Intent
import android.os.Bundle
import android.os.PersistableBundle
import android.util.Log
import android.view.Menu
import android.view.MenuItem
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.DividerItemDecoration
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.*
import com.xwray.groupie.GroupAdapter
import com.xwray.groupie.GroupieViewHolder
import com.xwray.groupie.Item
import kotlinx.android.synthetic.main.activity_latest_messages.*
import kotlinx.android.synthetic.main.latest_message_row.view.*

class LatestMessages : AppCompatActivity() {
    val adapter = GroupAdapter<GroupieViewHolder>()
    companion object{
        var currentUser: User? = null
        val TAG = "LatestMessages"
    }

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



        recyclerview_latest_message.adapter = adapter
        recyclerview_latest_message.addItemDecoration(DividerItemDecoration(this, DividerItemDecoration.VERTICAL))


        //set item click listener on adapter

        adapter.setOnItemClickListener { item, view ->
            Log.d(TAG, "123")
            val intent = Intent(this, ChatLogActivity::class.java)



            val row = item as LatestMessageRow
            row.chatPartnerUser

            intent.putExtra(NewMessageActivity.USER_KEY,row.chatPartnerUser )
            startActivity(intent)
        }

        listenForLatestMessages()
        fetchCurrentUser()
        verifyUserIsLoggedIn()

    }

    private fun verifyUserIsLoggedIn() {
        val uid =  FirebaseAuth.getInstance().uid
        if (uid == null) {
            val intent = Intent(this, RegisterActivity::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
            startActivity(intent)
        }
    }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean { //for nav bar menu
        menuInflater.inflate(R.menu.nav_menu, menu)
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item?.itemId) {
            R.id.menu_newmessage -> { //basically switch case
                val intent  =  Intent(this, NewMessageActivity::class.java)
                startActivity(intent)

            }
            R.id.menu_signout -> {
                FirebaseAuth.getInstance().signOut()
```

```
                val intent = Intent(this, RegisterActivity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
                startActivity(intent) //to go back to main page after sign out
            }
        }
        return super.onOptionsItemSelected(item)
    }

    private fun fetchCurrentUser() {

        val uid = FirebaseAuth.getInstance().uid
        val ref = FirebaseDatabase.getInstance().getReference("/users/$uid")
        ref.addListenerForSingleValueEvent(object: ValueEventListener {

            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onDataChange(p0: DataSnapshot) {
                currentUser = p0.getValue(User::class.java)
                Log.d(TAG, "Current user: ${currentUser?.username}")
            }
        })

    }

    private fun listenForLatestMessages(){

        val fromId = FirebaseAuth.getInstance().uid
        val ref = FirebaseDatabase.getInstance().getReference("/latest-messages/$fromId")
        ref.addChildEventListener(object: ChildEventListener {
            override fun onChildAdded(p0: DataSnapshot, p1: String?) {
                val chatMessage = p0.getValue(ChatMessage::class.java) ?:return
                latestMessagesMap[p0.key!!] = chatMessage
                refreshRecyclerViewMessages()


            }

            override fun onChildChanged(p0: DataSnapshot, p1: String?) {
                val chatMessage = p0.getValue(ChatMessage::class.java) ?: return
                latestMessagesMap[p0.key!!] = chatMessage
                refreshRecyclerViewMessages()
            }
            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onChildMoved(p0: DataSnapshot, p1: String?) {

            }

            override fun onChildRemoved(p0: DataSnapshot) {

            }
        })

    }
    class LatestMessageRow(val chatMessage: ChatMessage): Item<GroupieViewHolder>() {

        var chatPartnerUser: User? = null

        override fun bind(viewHolder: GroupieViewHolder, position: Int) {
            viewHolder.itemView.message_latestmessage.text = chatMessage.text

            val chatPartnerId: String

            if (chatMessage.fromId == FirebaseAuth.getInstance().uid) {
                chatPartnerId = chatMessage.toId
            } else {
                chatPartnerId = chatMessage.fromId
            }

            val ref = FirebaseDatabase.getInstance().getReference("/users/$chatPartnerId")
            ref.addListenerForSingleValueEvent(object: ValueEventListener {
                override fun onCancelled(p0: DatabaseError) {

                }

                override fun onDataChange(p0: DataSnapshot) {
                    chatPartnerUser = p0.getValue(User::class.java)
                    viewHolder.itemView.username_latestmessages.text = chatPartnerUser?.username


                }
            })
```

```
        }
        override fun getLayout(): Int {
            return R.layout.latest_message_row
        }
    }

    val latestMessagesMap = HashMap<String, ChatMessage>()

    private fun refreshRecyclerViewMessages() {
        adapter.clear()
        latestMessagesMap.values.forEach {
            adapter.add(LatestMessageRow(it))
        }
    }


}
```