CHAT CONNECT- A REAL-TIME CHAT AND COMMUNICATION APP

NAME : KARUPPASAMY.K.
ROLL NO: 712822104022

PROJECT OBJECTIVES

- ChatConnect is a sample project built using the Android Compose Ul 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 Ul 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.

FUNCTIONS

- User authentication : secure login.
- Real-time messaging : text chat with instant delivery.
- Media sharing : users can share images , videos and files.
- Push notifications : notifications for new messages.
- Typing indicators & Read receipts : to show when users are typing or when a message is read.
- Group chats : ability to chat in groups with multiple users.

MAIN ACTIVITY.kt :

```
package com.vivek.chatingapp.repository

import com.google.android.gms.tasks.OnCompleteListener
import com.google.firebase.firestore.*
import com.google.firebase.firestore.EventListener
import com.google.firebase.messaging.FirebaseMessaging
import com.google.gson.JsonObject
import com.vivek.chatingapp.model.MessageBody
import com.vivek.chatingapp.network.Api
import com.vivek.chatingapp.utils.Constant
import com.vivek.chatingapp.utils.Resource
import kotlinx.coroutines.tasks.await
import okhttp3.ResponseBody
import org.json.JSONObject
import retrofit2.Response
import java.util.*
import javax.inject.Inject
import kotlin.collections.HashMap
```

```
class MainRepository @Inject constructor(
    private val fireStore: FirebaseFirestore,
    private val fireMessage: FirebaseMessaging,
    private val fcmApi:Api,
    private val remoteHeader:HashMap<String,String>
) {

    suspend fun updateToken(token: String, userId: String): Boolean {
        return try {
            val documentReference =
fireStore.collection(Constant.KEY_COLLECTION_USERS).document(userId)
            documentReference.update(Constant.KEY_FCM_TOKEN, token).await()
            true
        } catch (e: Exception) {
            false
        }
    }
```

```
suspend fun getToken(): String {
 return fireMessage.token.await()
}

suspend fun userSignOut(userId: String, userData: HashMap<String, Any>): Boolean {
return try {
 val documentReference = fireStore.collection(Constant.KEY_COLLECTION_USERS)
.document(userId)
documentReference.update(userData).await()
true
} catch (e: Exception) {
false
 }
}

suspend fun getAllUsers(): Resource<QuerySnapshot> {
try {
val await = fireStore.collection(Constant.KEY_COLLECTION_USERS)
.get()
.await()
if (await.isEmpty) {
return Resource.Empty("No User Available")
```

```
}
return Resource.Success(await)
} catch (e: Exception) {
return Resource.Error(e.message ?: "An Unknown Error Occurred")
}
}

suspend fun sendMessage(message: HashMap<String, Any>): Boolean = try {
fireStore.collection(Constant.KEY_COLLECTION_CHAT).document().set(message).await()
true
} catch (e: Exception) {
false
}
fun observeChat(userId: String, receiverId: String, listener: EventListener<QuerySnapshot>) {
fireStore.collection(Constant.KEY_COLLECTION_CHAT)
```

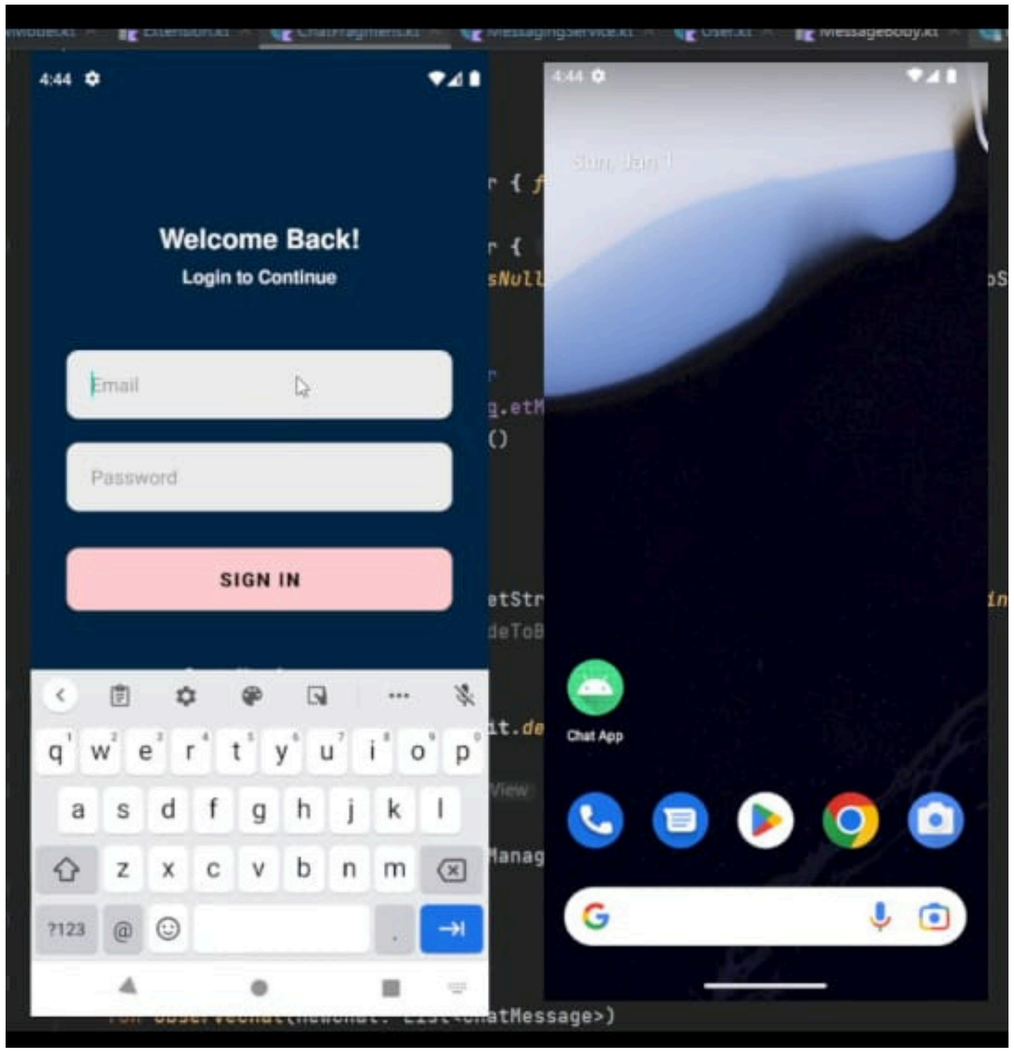
```
.whereEqualTo(Constant.KEY_SENDER_ID, userId)
.whereEqualTo(Constant.KEY_RECEIVER_ID, receiverId)
.addSnapshotListener(listener)
fireStore.collection(Constant.KEY_COLLECTION_CHAT)
.whereEqualTo(Constant.KEY_SENDER_ID, receiverId)
.whereEqualTo(Constant.KEY_RECEIVER_ID, userId)
.addSnapshotListener(listener)
}

suspend fun checkForConversionRemotely(
senderId: String,
receiverId: String,
listener: OnCompleteListener<QuerySnapshot>,
) {
```

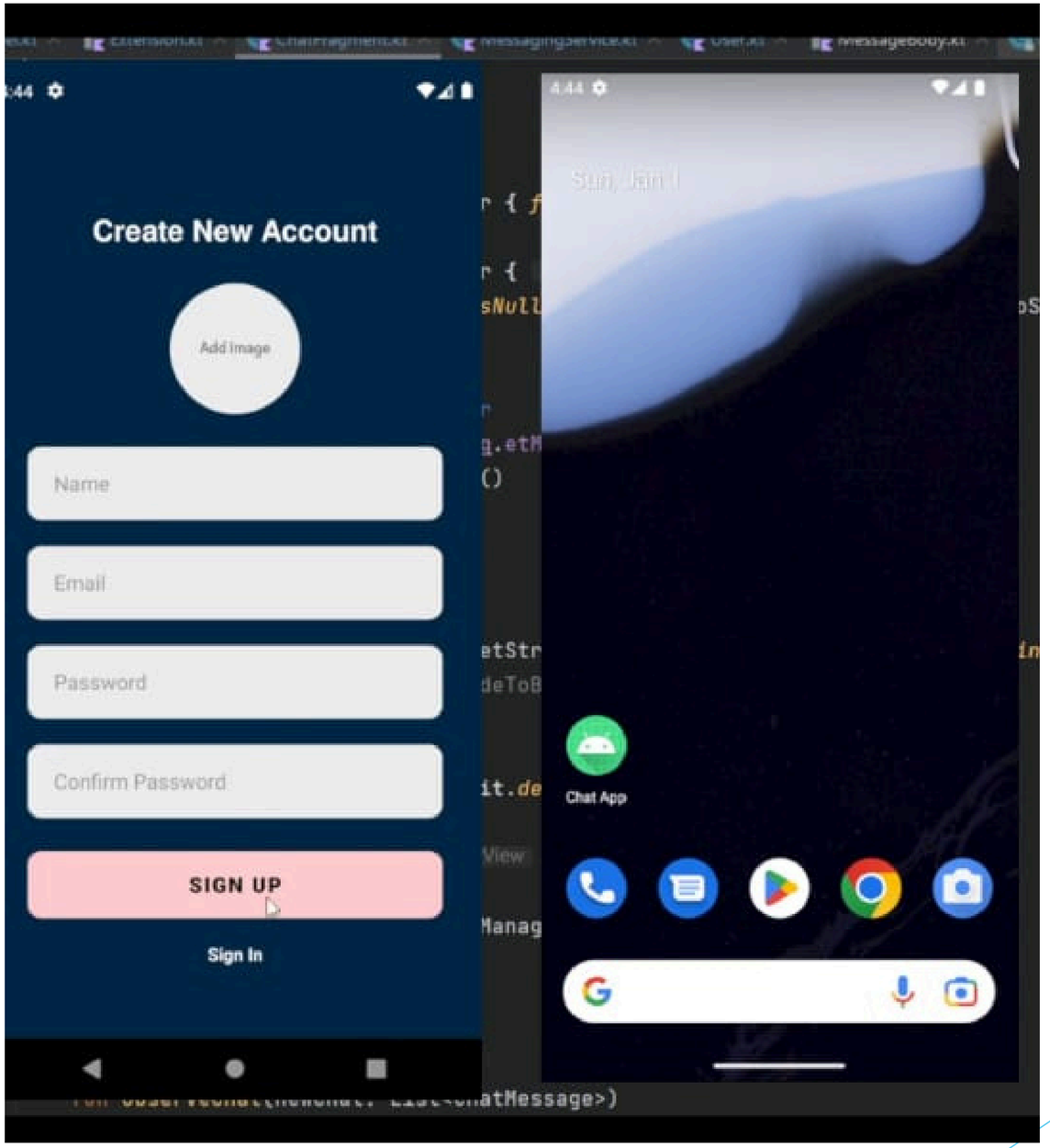
```
fireStore.collection(Constant.KEY_COLLECTION_CONVERSATIONS)
.whereEqualTo(Constant.KEY_SENDER_ID, senderId)
.whereEqualTo(Constant.KEY_RECEIVER_ID, receiverId)
.get()
.addOnCompleteListener(listener)
.await()
}

suspend fun updateConversation(message: String, conversionId: String) {
fireStore.collection(Constant.KEY_COLLECTION_CONVERSATIONS).document(conversionId)
.update(
Constant.KEY_LAST_MESSAGE,message,
Constant.KEY_TIMESTAMP,Date()
)
.await()
}
```

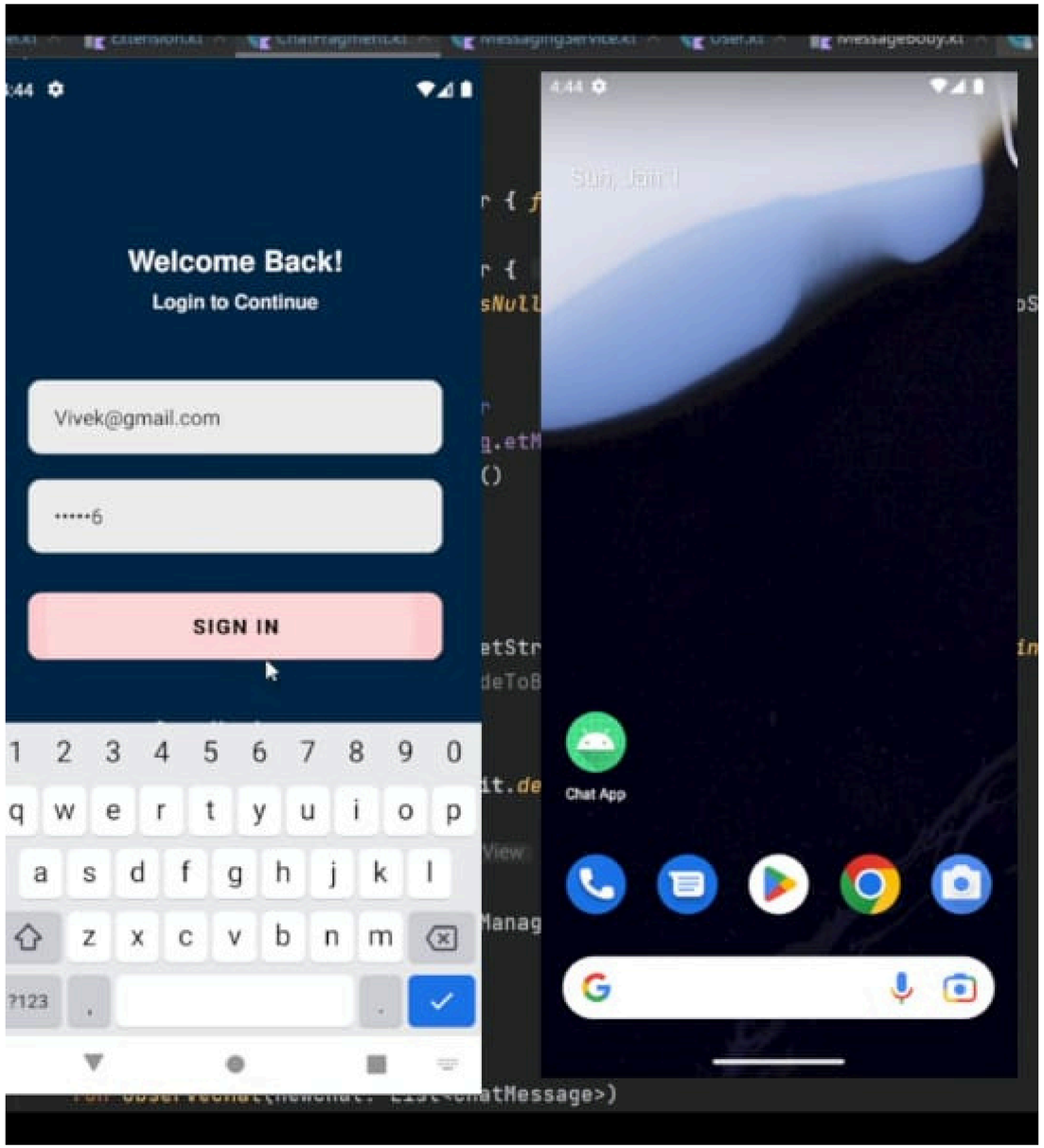
```
suspend fun updateRecentConversation(message: HashMap<String, Any>,
                                     onSuccessListener:(String)->Unit){
    fireStore.collection(Constant.KEY_COLLECTION_CONVERSATIONS)
        .add(message)
        .addOnSuccessListener {
            onSuccessListener(it.id)
        }
        .await()
}
fun observeRecentConversation(id:String, listener: EventListener<QuerySnapshot>) {
    fireStore.collection(Constant.KEY_COLLECTION_CONVERSATIONS)
        .whereEqualTo(Constant.KEY_SENDER_ID, id)
        .addSnapshotListener(listener)
    fireStore.collection(Constant.KEY_COLLECTION_CONVERSATIONS)
        .whereEqualTo(Constant.KEY_RECEIVER_ID, id)
        .addSnapshotListener(listener)
}
```

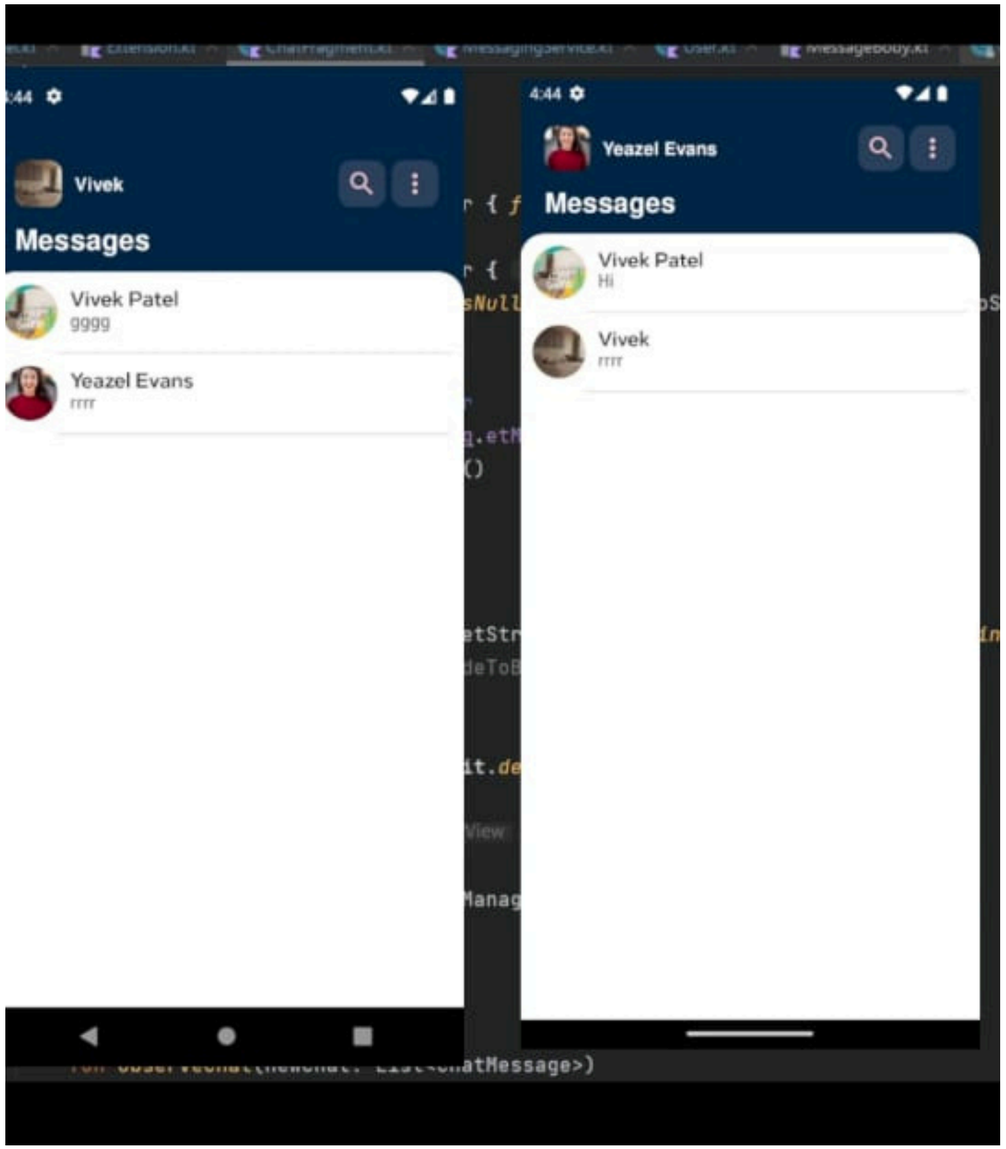
```
fun listenerAvailabilityOfReceiver(receiverId: String,listener:
EventListener<DocumentSnapshot>){
fireStore.collection(Constant.KEY_COLLECTION_USERS)
.document(receiverId)
.addSnapshotListener(listener)

}

suspend fun sendNotification(messageBody: MessageBody): Response<JsonObject> {
return fcmApi.sendMessage(messageBody =  messageBody, header =
remoteHeader)
}

}
```

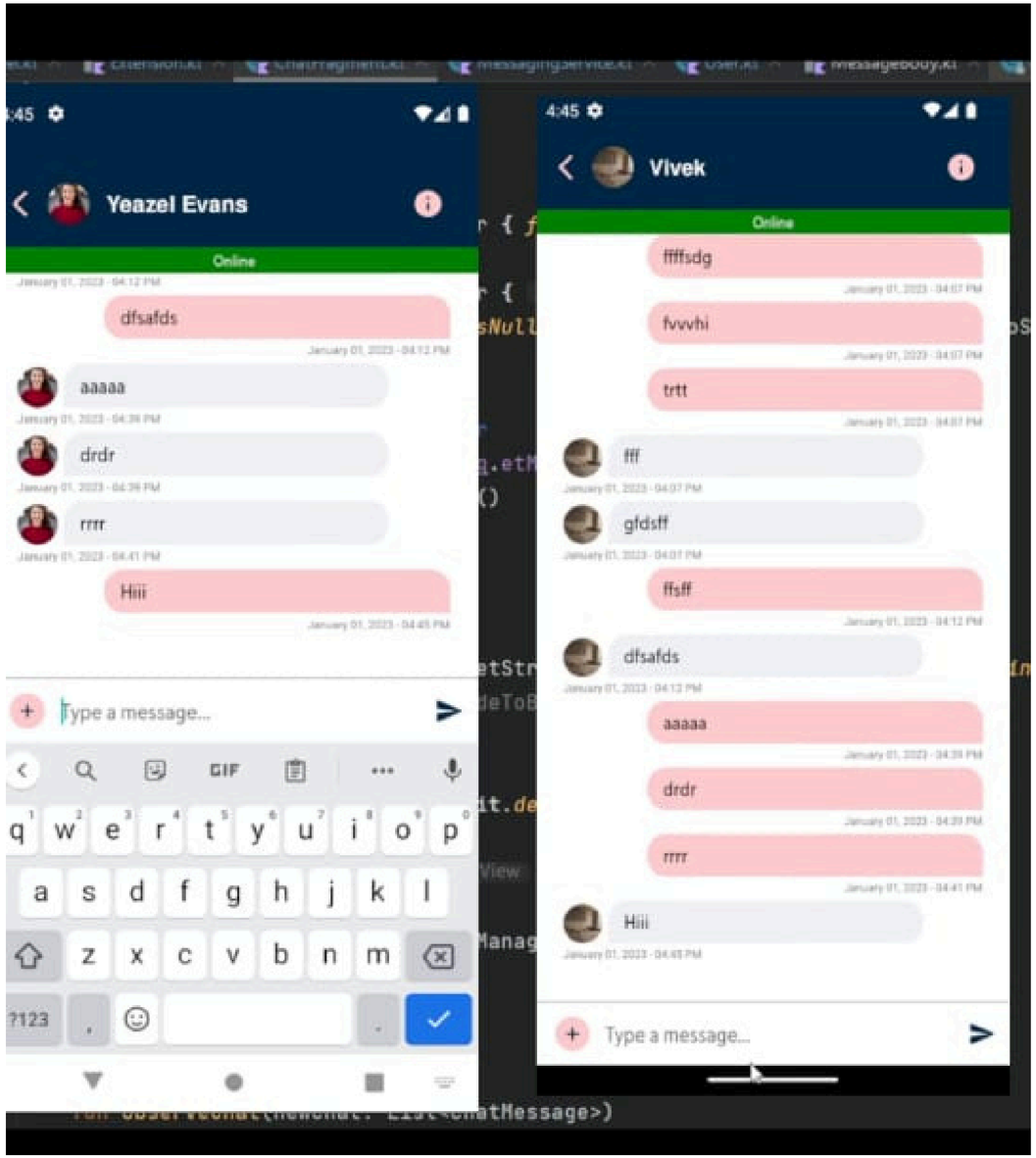
OUTPUT :

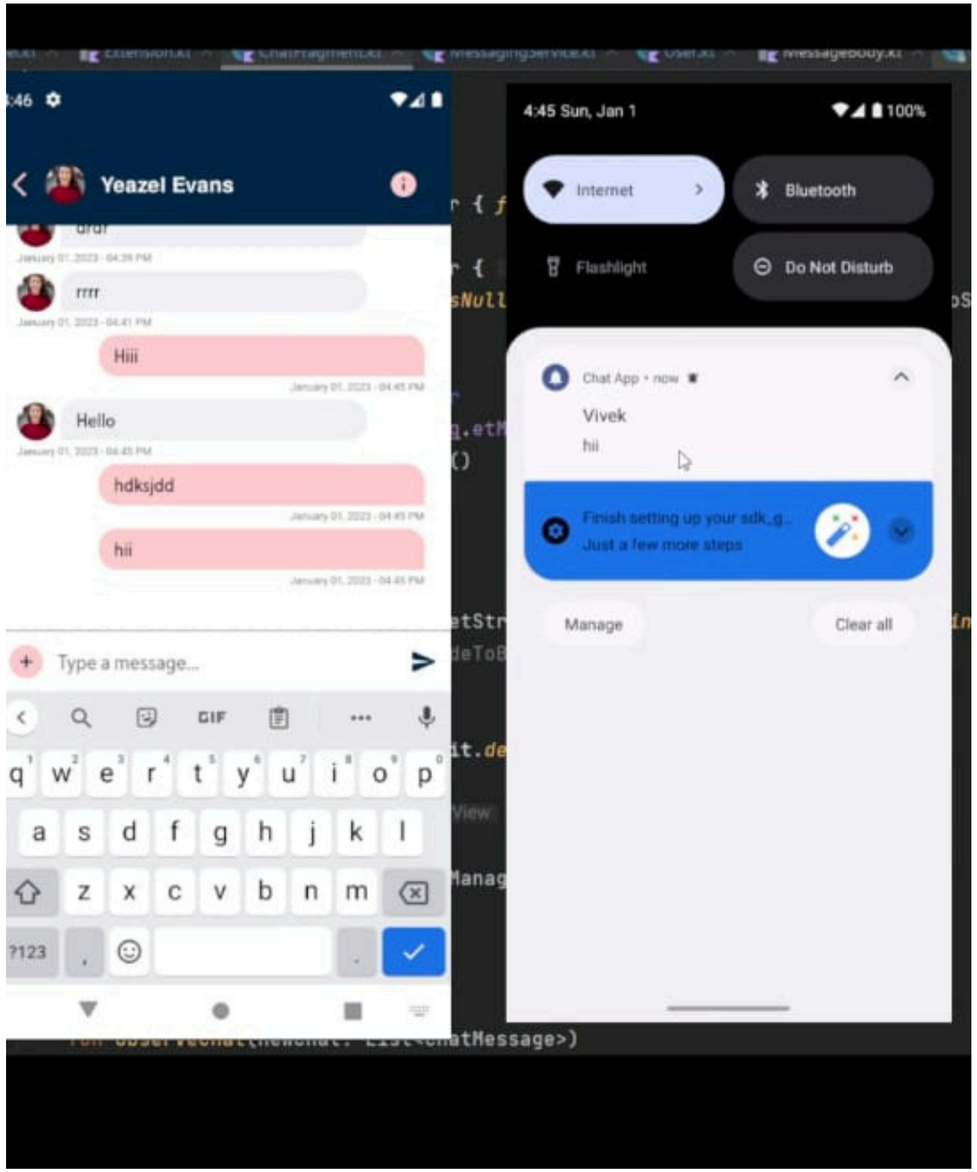

4:44
Welcome Back!
Login to Continue
Email
Password
SIGN IN
4:44
Chat App

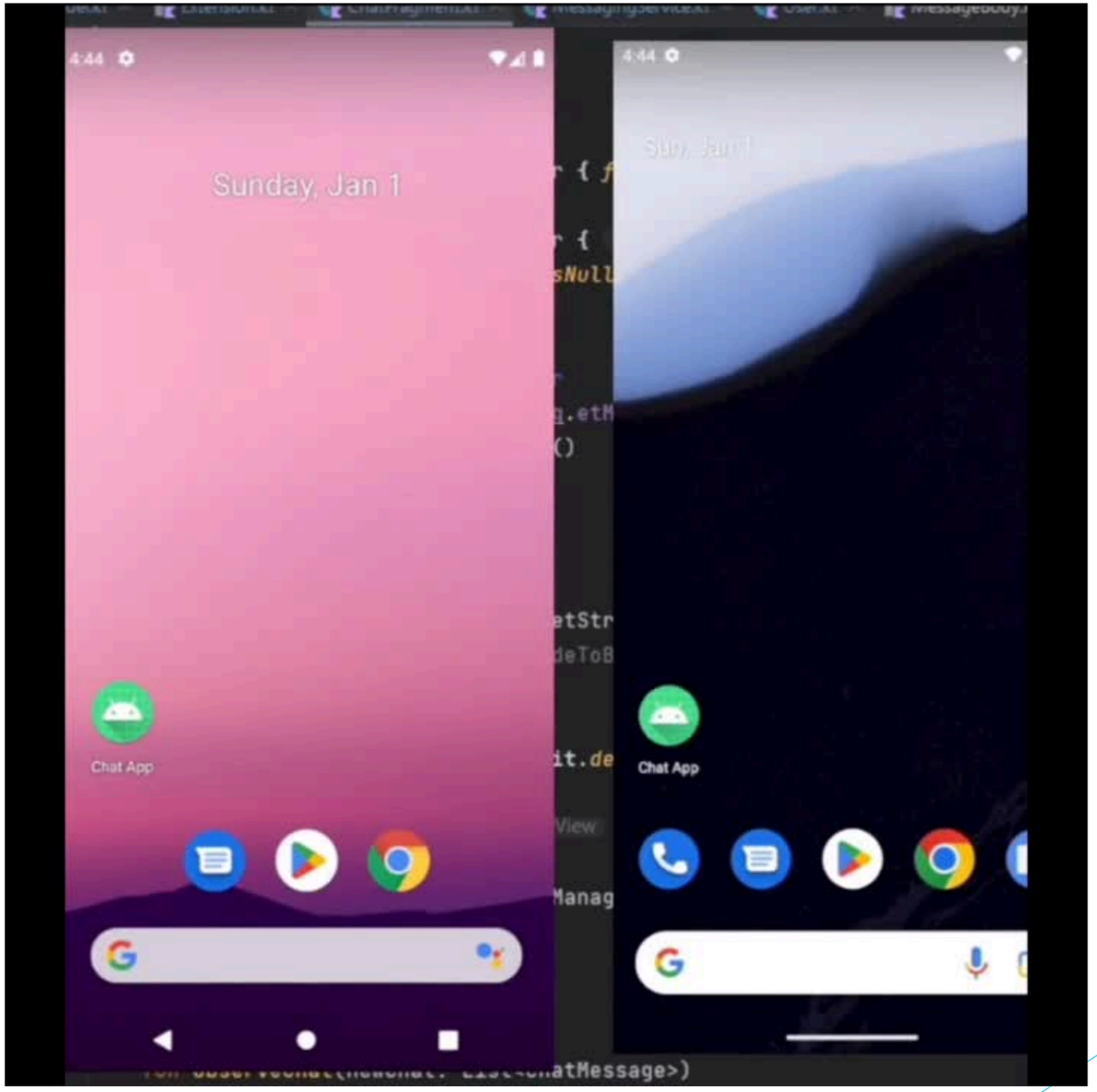

Create New Account
Add Image
Name
Email
Password
Confirm Password
SIGN UP
Sign In
Chat App


Welcome Back!
Login to Continue
Vivek@gmail.com
SIGN IN
Chat App


Vivek
Messages
Vivek Patel
gggg
Yeazel Evans
rrrr
Yeazel Evans
Messages
Vivek Patel
Hi
Vivek
rrrr


Yeazel Evans
Online
dfsafds
aaaaa
drdr
rrrr
Hiii
Type a message...
Vivek
Online
ffffsdg
fvvvhi
trtt
fff
gfdsff
ffsff
dfsafds
aaaaa
drdr
rrrr
Hiii
Type a message...


Yeazel Evans
rrrr
Hiii
Hello
hdksjdd
hii
Type a message...
4:45 Sun, Jan 1
100%
Internet
Bluetooth
Flashlight
Do Not Disturb
Chat App • now
Vivek
hii
Finish setting up your sdk,g...
Just a few more steps
Manage
Clear all


4:44
Sunday, Jan 1
Chat App
4:44
Chat App