CHAT CONNECT – A REAL TIME CHAT APP

INTRODUCTION

1.1 OVERVIEW:

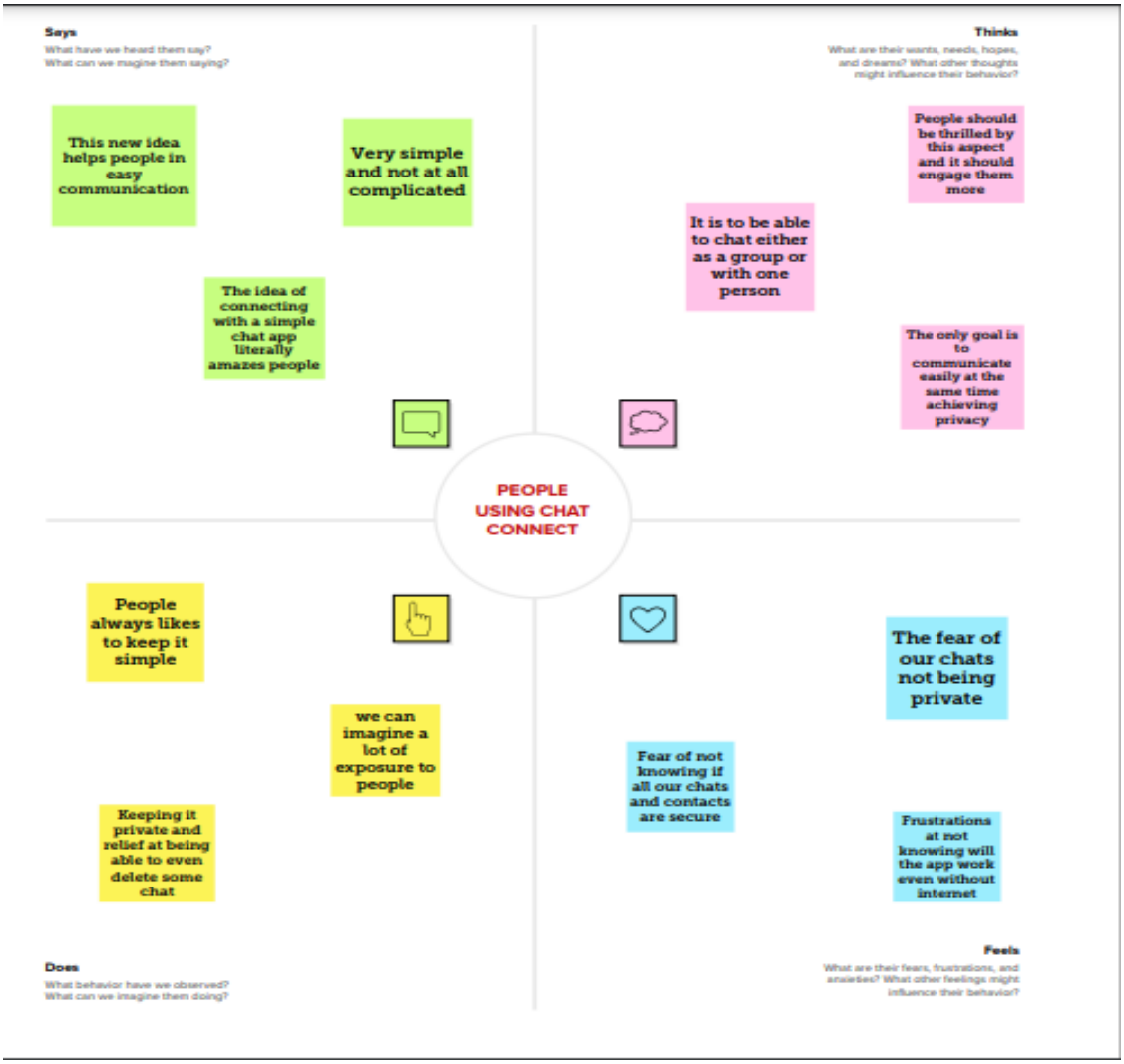
Communication is a mean for people to exchange messages. Messaging apps (a.k.a. social messaging or chat applications) are apps and platforms that enable instant messaging. According to the survey the group of users prefer WhatsApp and like to communicate using Emoji. 51% of the group uses the chat applications on an average of 1-2 hours a day. Messaging apps now have more global users than traditional social network which means they will play an increasingly important role in the distribution of digital journalism in the future.

1.2 PURPOSE:

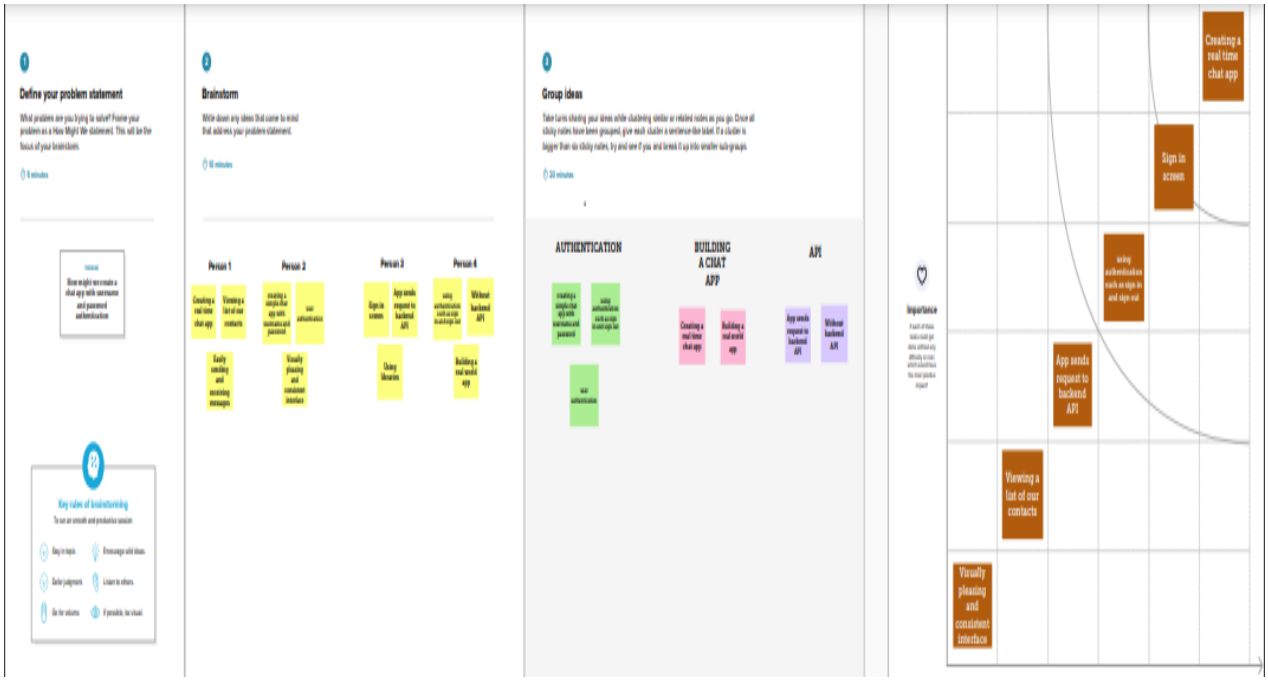
A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time. With a web or mobile chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person. This also keeps users conversing on your platform instead of looking elsewhere for a messaging solution. Whether it's private chat, group chat, or large-scale chat, adding personalized chat features to your app can help ensure that your users have a memorable experience.

PROBLEM DEFINITION & DESIGN THINKING

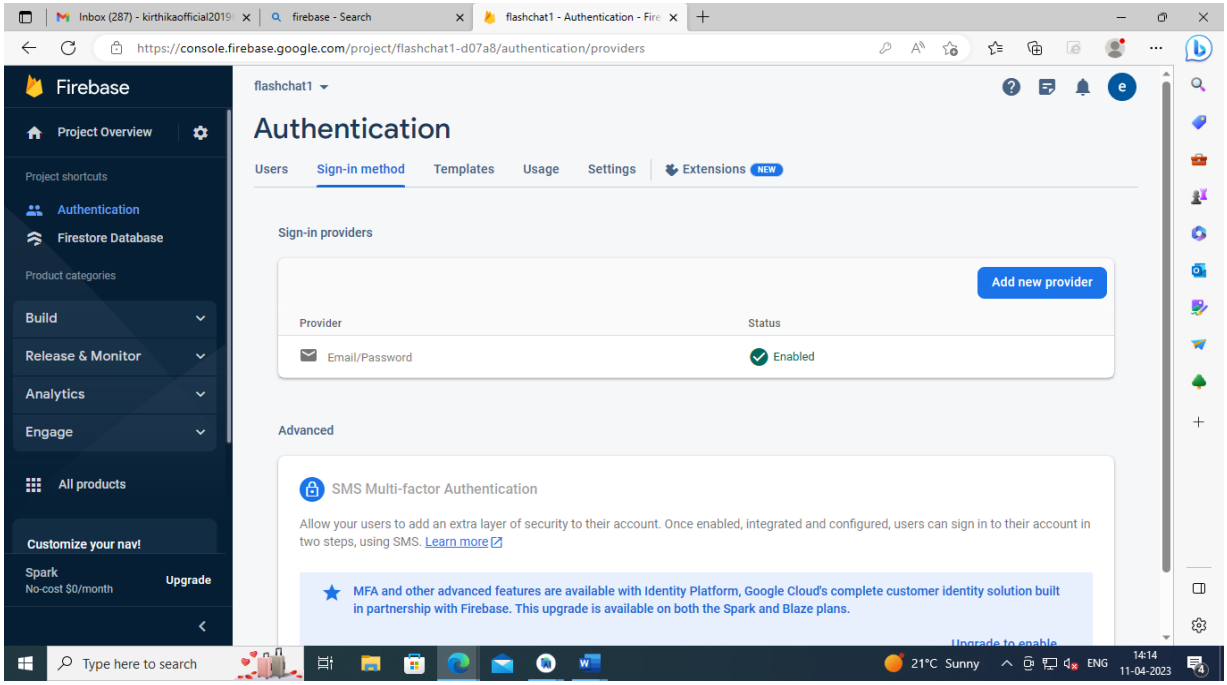
2.1 EMPATHY MAP:

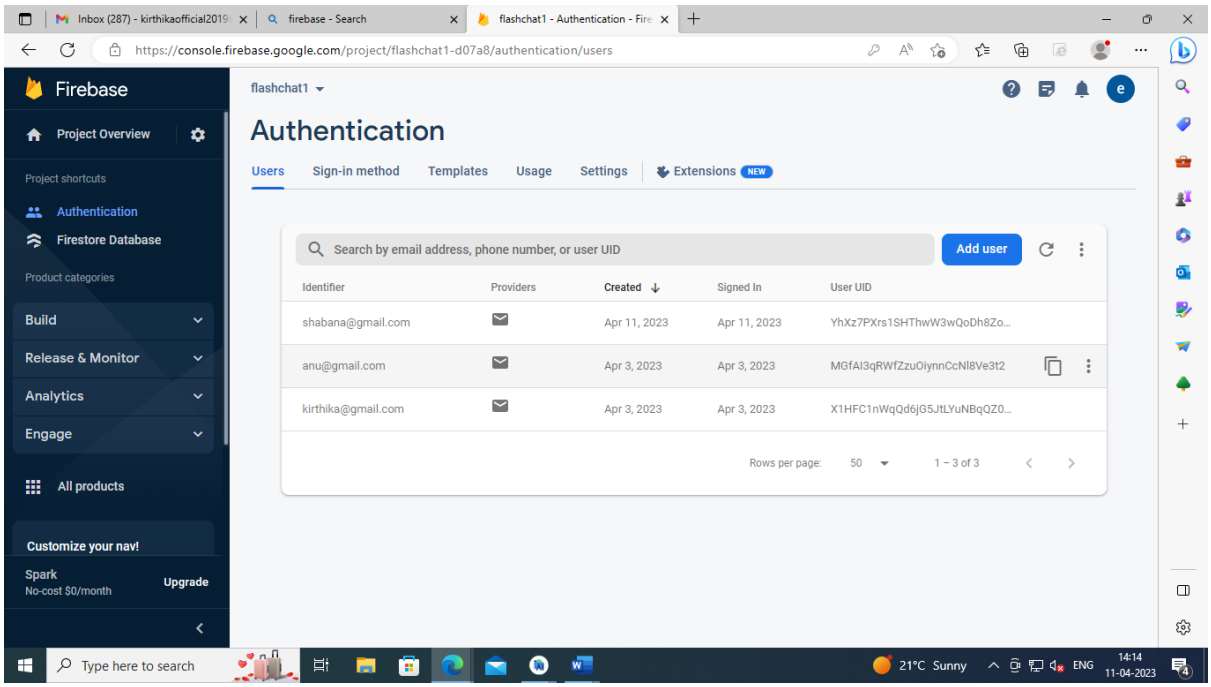


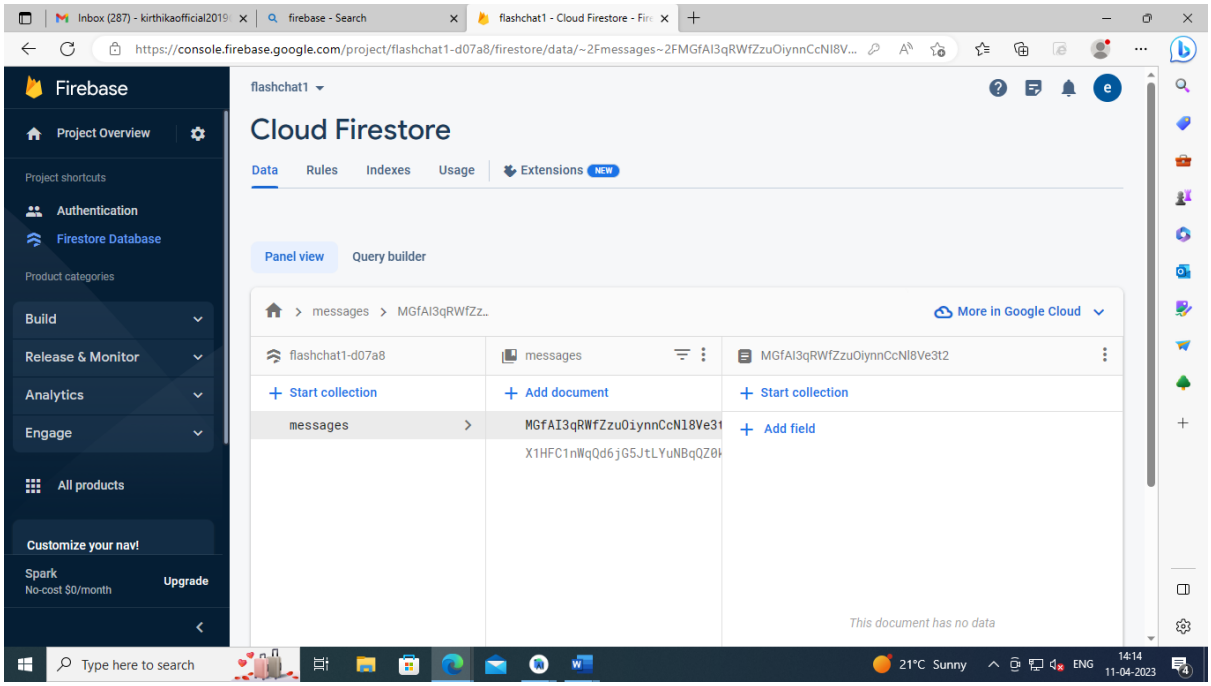
2.2 IDEATION & BRAINSTROMING MAP:

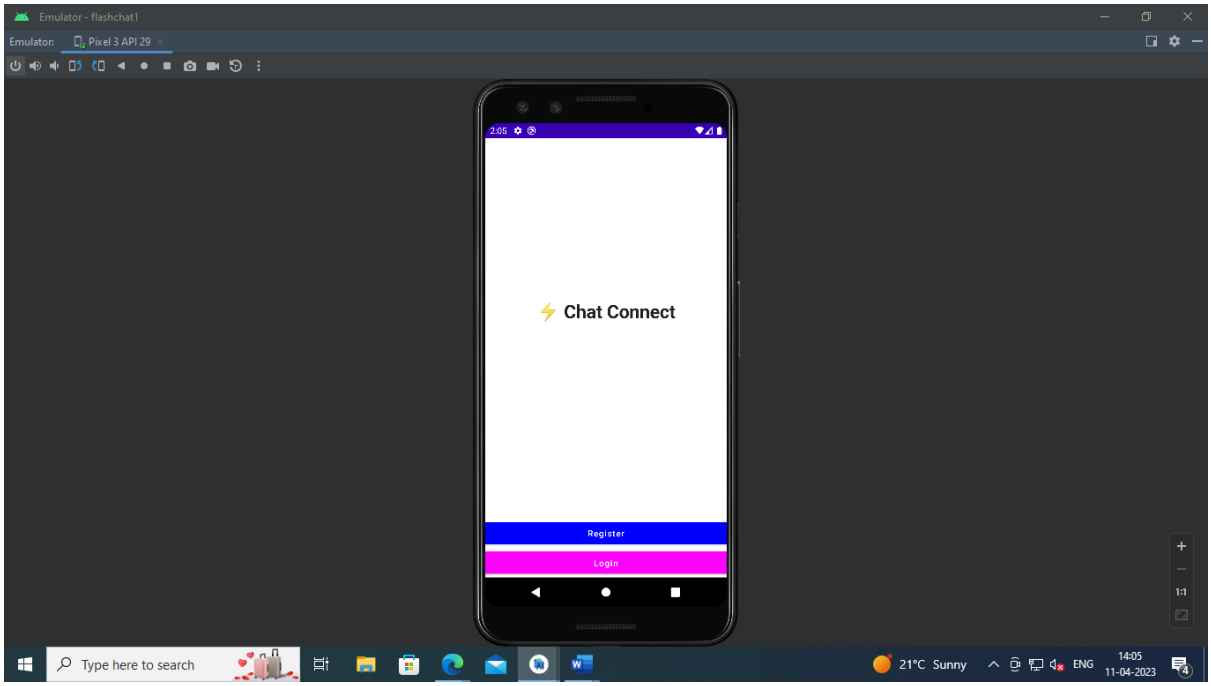


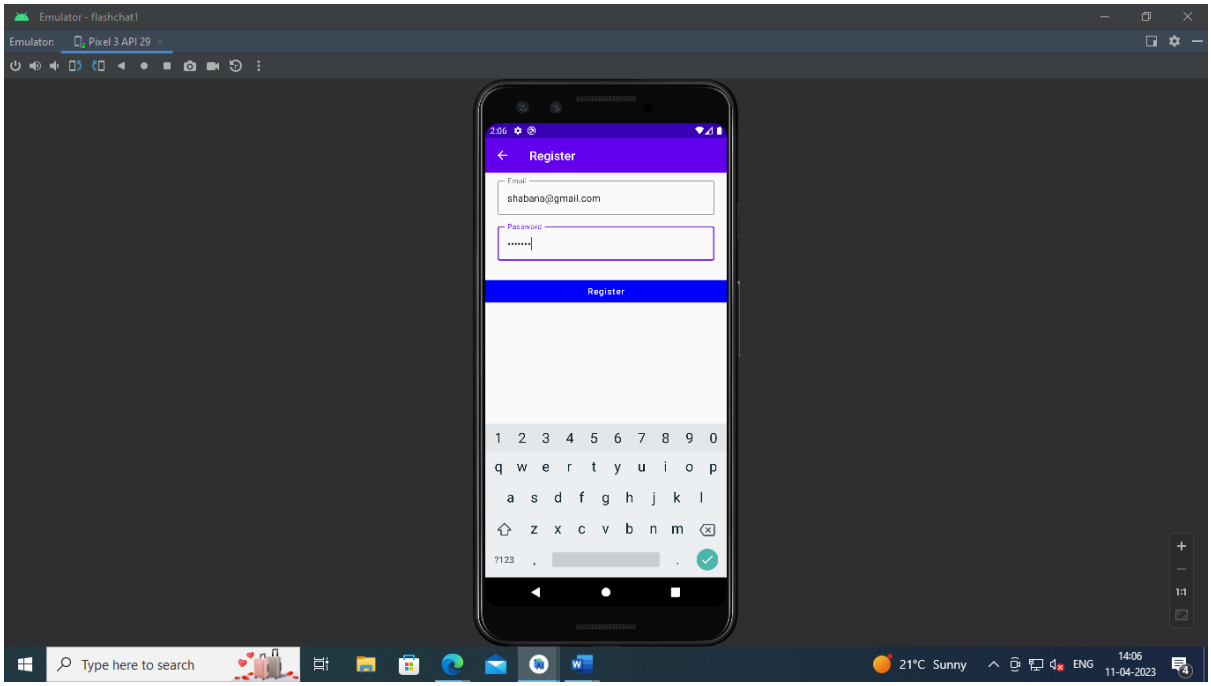
RESULT

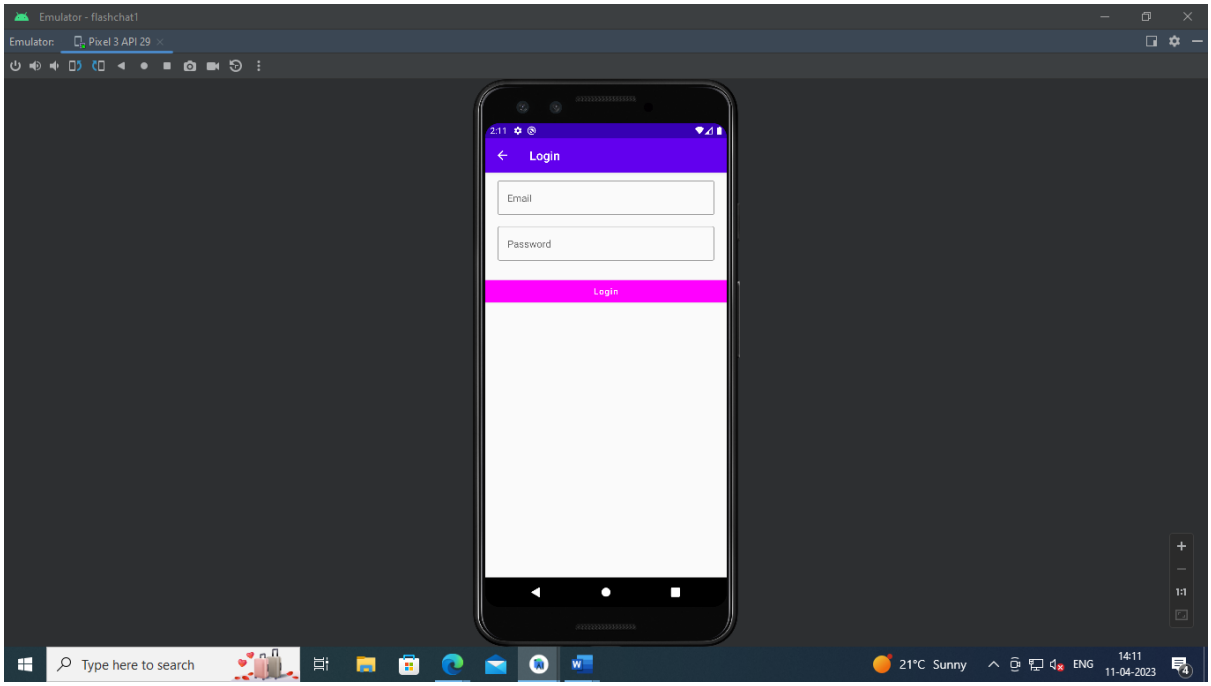

Inbox (287) - kirthikaofficial2019
firebase - Search
flashchat1 - Authentication - Fire
https://console.firebase.google.com/project/flashchat1-d07a8/authentication/providers
Firebase
flashchat1
Project Overview
Project shortcuts
Authentication
Firestore Database
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
Spark
No-cost $0/month
Upgrade
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions NEW
Sign-in providers
Add new provider
Provider
Status
Email/Password
Enabled
Advanced
SMS Multi-factor Authentication
Allow your users to add an extra layer of security to their account. Once enabled, integrated and configured, users can sign in to their account in two steps, using SMS. Learn more
MFA and other advanced features are available with Identity Platform, Google Cloud's complete customer identity solution built in partnership with Firebase. This upgrade is available on both the Spark and Blaze plans.
Type here to search
21°C Sunny
ENG
14:14
11-04-2023

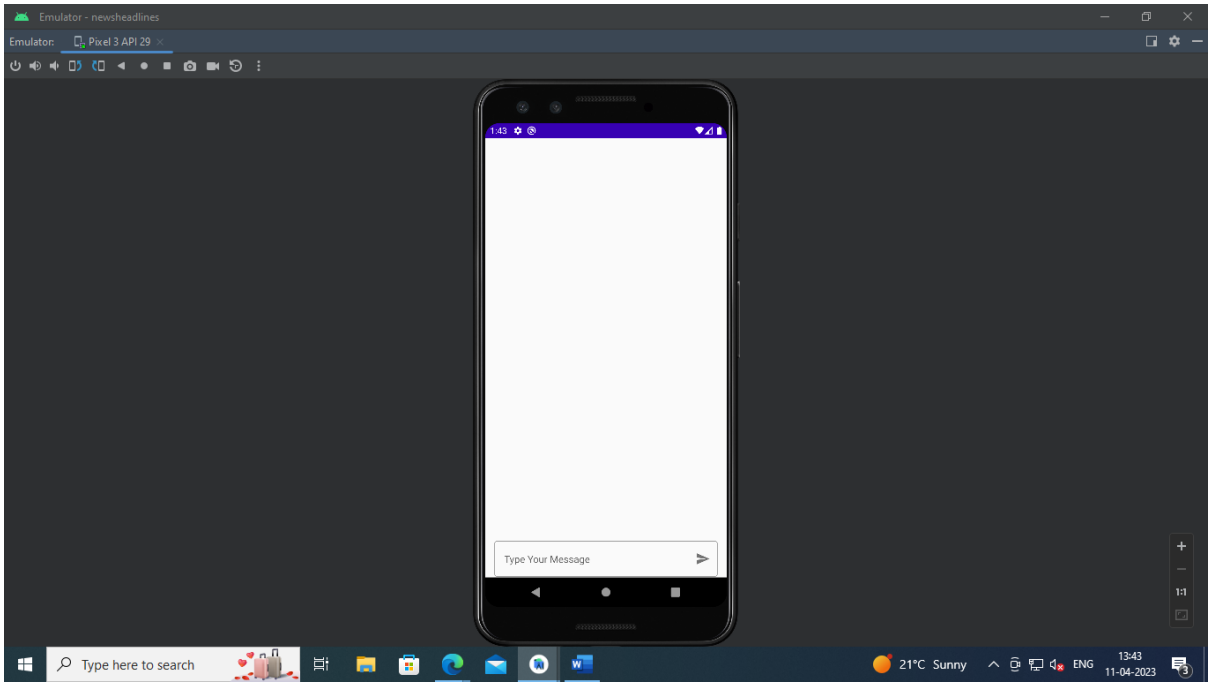

Inbox (287) - kirthikaofficial2019
firebase - Search
flashchat1 - Authentication - Fire
https://console.firebase.google.com/project/flashchat1-d07a8/authentication/users
Firebase
flashchat1
Project Overview
Project shortcuts
Authentication
Firestore Database
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
Spark
No-cost $0/month
Upgrade
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions NEW
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
shabana@gmail.com
Apr 11, 2023
Apr 11, 2023
YhXz7PXrs1SHThwW3wQoDh8Zo...
anu@gmail.com
Apr 3, 2023
Apr 3, 2023
MGfAI3qRWfZzuOiynnCcNl8Ve3t2
kirthika@gmail.com
Apr 3, 2023
Apr 3, 2023
X1HFC1nWqQd6jG5JtLYuNBqQZ0...
Rows per page: 50
1 – 3 of 3
Type here to search
21°C Sunny
ENG
14:14
11-04-2023


Firebase
Project Overview
Project shortcuts
Authentication
Firestore Database
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
Spark
No-cost $0/month
Upgrade
flashchat1
Cloud Firestore
Data
Rules
Indexes
Usage
Extensions NEW
Panel view
Query builder
messages > MGfAI3qRWfZz...
More in Google Cloud
flashchat1-d07a8
messages
MGfAI3qRWfZzuOiynnCcNI8Ve3t2
Start collection
Add document
Start collection
messages
MGfAI3qRWfZzuOiynnCcNI8Ve3t
X1HFC1nWqQd6jG5JtLYuNBqQZ0k
Add field
This document has no data
Type here to search
21°C Sunny
ENG
14:14
11-04-2023


Emulator - flashchat1
Emulator:
Pixel 3 API 29
2:05
Chat Connect
Register
Login
Type here to search
21°C Sunny
ENG
14:05
11-04-2023


Emulator - flashchat1
Emulator:
Pixel 3 API 29
2:06
Register
Email
shabana@gmail.com
Password
Register
Type here to search
21°C Sunny
ENG
14:06
11-04-2023


Emulator - flashchat1
Emulator:
Pixel 3 API 29
2:11
Login
Email
Password
Login
Type here to search
21°C Sunny
ENG
14:11
11-04-2023


Emulator - newsheadlines
Emulator:
Pixel 3 API 29
1:43
Type Your Message
1:1
Type here to search
21°C Sunny
ENG
13:43
11-04-2023

ADVANTAGES & DISDAVANTAGES

ADVANTAGES:

1. **FASTER SUPPORT:** Obviously chat is easy to reach for your customers, but what's more is that the average resolution time is significantly lower than with traditional service channels. It takes less than a minute to resolve a customer issue using live chat.

2. **REAL-TIME TEXT PREVIEW:** One of the handy advantages of live chat is the option to see a real-time preview of what the customer is typing before she hits enter. It gives your chat agent the chance to think about a solution, research and impress users with prompt, customized answers.

3. **INSTANT CUSTOMER FEEDBACK:** Feedback is easily collectable. Users can rate your chat service right after their interaction with you. This is a benefit for your service agents too because they get instant feedback on their performance. This makes it easy for them to connect the dots.

4. **NON-INTRUSIVE:** Customers can go about their day while being helped. Live chat allows users to continue browsing, posting and working – your agent solves the case along the way. Over 51% of customers like live chat because it allows them to multitask.

5. **INCREASED EFFICIENCY / REDUCED COST:** Entertaining a live chat team is cheaper than the traditional call centre. A 2010 Forrester study suggests that a live chat session is about 20%-50% cheaper than a phone call. The minimized resolution time is a main reason for this cost reduction.

DISADVANTAGES:

1. **TIME CONSUMING:** Various system includes time consuming process in their application.
2. **RISKS:** Invasion of privacy & bullying using digital technology.
3. **TIME ZONE ISSUES:** Live chat agents are not available 24/7. In this case, when a website visitor arrives at your site, they might face time zone issues. When there is no response on an online website, there is a chance of customers leaving the website.
4. **ONLINE TROLLS:** Because you can chat quite anonymously, internet trolls are a phenomenon. That's why User like offers features like blocking and ignoring. To protect the privacy of operators, it is possible to use operator aliases.
5. **NO INSTANT RESPONSE FOR MULTIPLE VISITORS**: An e-commerce site owner will have multiple visitors. At times, there is no response from live chat agents since they are handling multiple customers. Hence, if you have integrated live chat on your website, then it reduces instant responses for customers.

APPLICATIONS

Real-time chat uses Mobile-based apps, which permit communication that is usually addressed directly but is anonymous among users in a multi-user environment.

Real-time can be applied in:

- Personal
- Work settings
- Company settings

CONCLUSION

By working on this project, we came to know a lot about programming and knowledge about various languages which we have worked on.

Being students from programming background, we shad a huge challenge in front of us, but the technology helped us pave our way through it.

We learned how to apply logic at the correct point, at correct time. Where do we have to mainly focus and at what point we have to receive help of technology. Many concepts regarding programming and languages were cleared, and how does it apply while coding, could be known. With the help of our teachers, technology and our base of programming helped us to complete our project on time successfully.

In today's world, the person has to think deep, quick and be focused to the goal which is to be achieved. But it should also include the future result of its decision. Teamwork is important in each and every task which is to be completed, with the help of our team members. Thanks to all the members for their full cooperation during the project, till the end.

FUTURE SCOPE

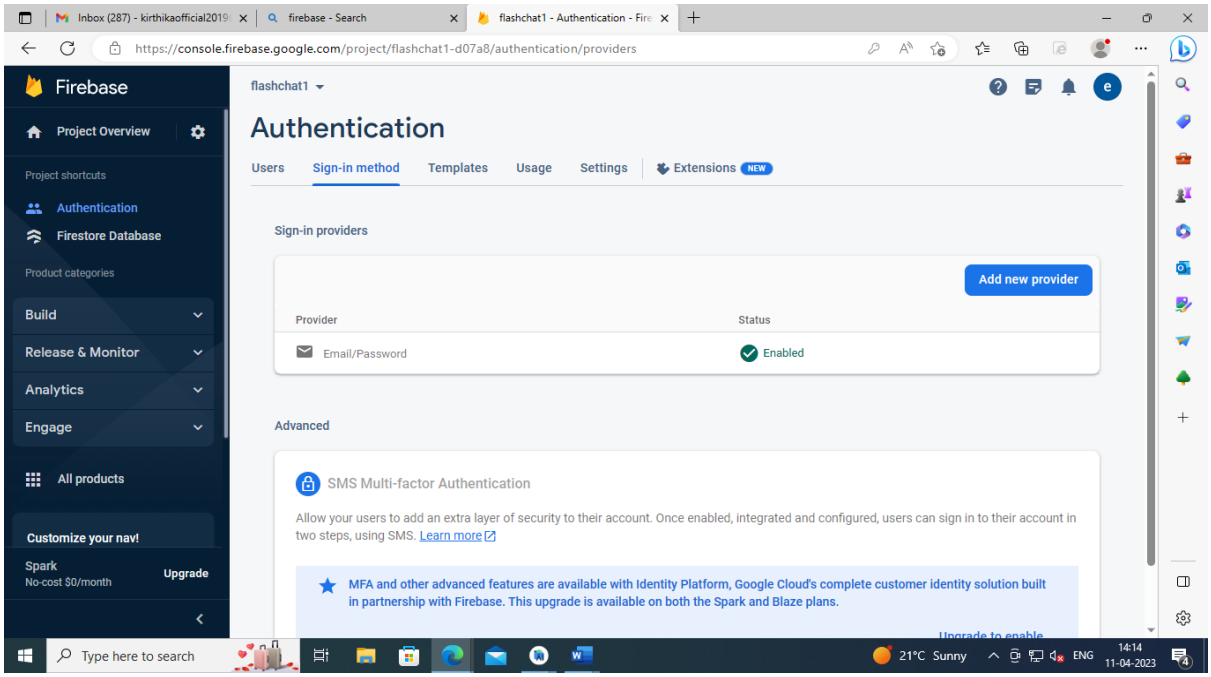
With the knowledge I have gained by developing this application, I am confident that in the future I can make the application more effectively by adding these services.

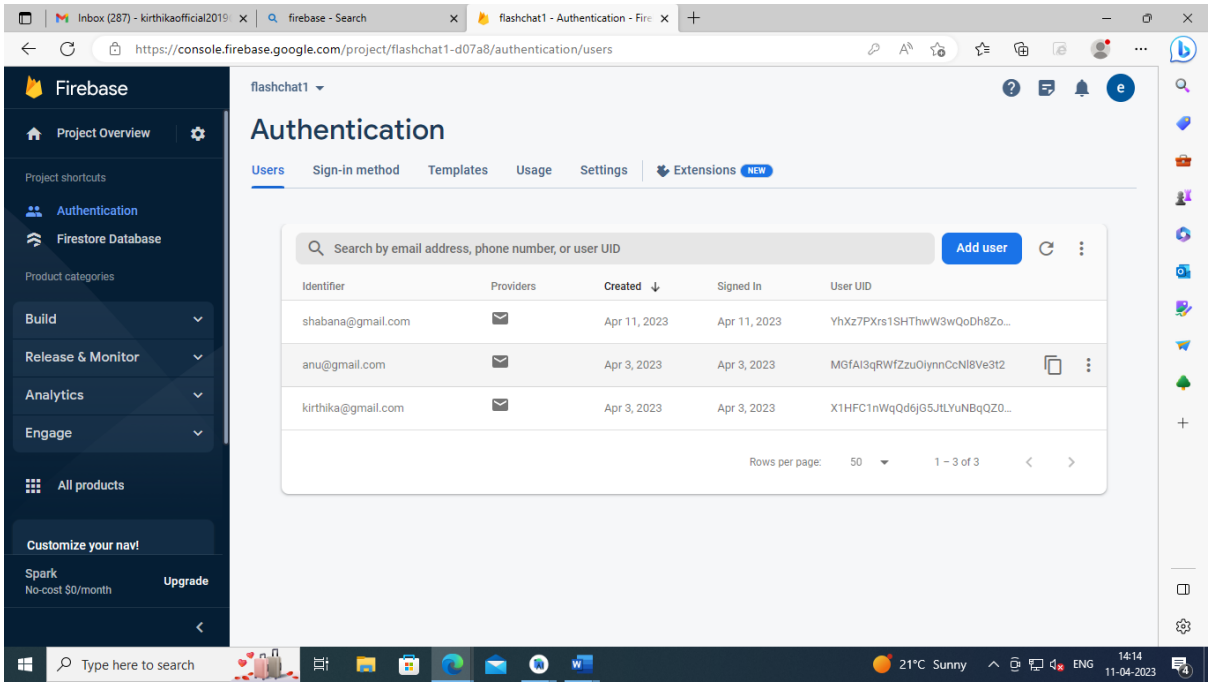
- Extending this application by providing Authorisation service.
- Creating Database and maintaining users.
- Increasing the effectiveness of the application by providing Voice Chat.
- Extending it to Web Support.

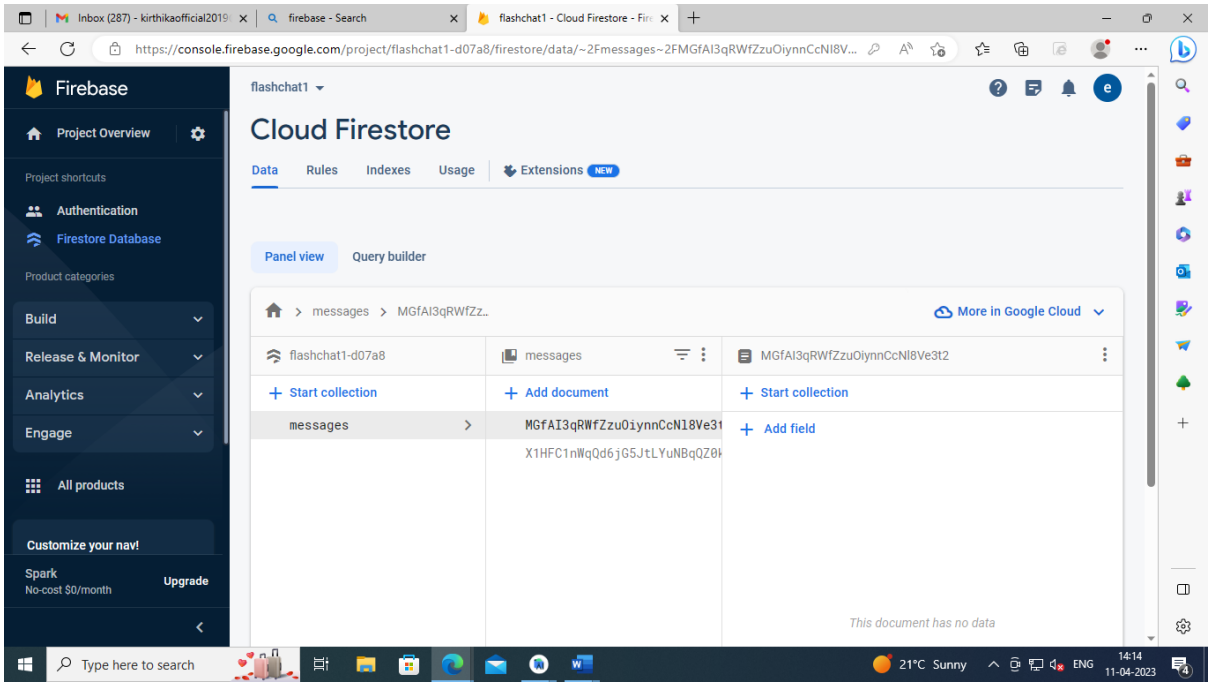
APPENDIX

SOURCE CODE

INTEGRATING FIREBASE:







GRADLE SCRIPTS:

```
plugins {

    id 'com.android.application'

    id 'org.jetbrains.kotlin.android'

    id 'com.google.gms.google-services'
```

```
}


android {
    namespace 'com.example.flashchat1'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.flashchat1"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
```

```
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.2.0'
    implementation "androidx.compose.ui:ui: $compose_ui_version"
    implementation "androidx.compose.material:material:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
    implementation "androidx.compose.runtime:runtime-
livedata:$compose_ui_version"
    implementation 'androidx.lifecycle:lifecycle-viewmodel-compose:1.0.0-
alpha07'
```

```
    implementation "androidx.navigation:navigation-compose:2.4.0-alpha06"



    implementation platform('com.google.firebase:firebase-bom:28.3.0')

    implementation 'com.google.firebase:firebase-analytics-ktx'

    implementation 'com.google.firebase:firebase-auth-ktx'

    implementation 'com.google.firebase:firebase-firestore-ktx'

    implementation 'com.google.firebase:firebase-auth:21.0.3'

    implementation 'com.google.firebase:firebase-firestore:24.1.1'



    testImplementation 'junit:junit:4.13.2'



    androidTestImplementation 'androidx.test.ext:junit:1.1.3'

    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"



    debugImplementation "androidx.compose.ui:ui-
tooling:$compose_ui_version"

}
```

MAINACTIVITY.Kt FILE:

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.OwlApplication"
        tools:targetApi="31">

        <activity
            android:name=".RegisterActivity"
            android:exported="false"
            android:label="@string/title_activity_register"
            android:theme="@style/Theme.OwlApplication" />
        <activity
```

```
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
            android:theme="@style/Theme.OwlApplication" />
        <activity
            android:name=".MainActivity5"
            android:exported="false"
            android:label="@string/title_activity_main5"
            android:theme="@style/Theme.OwlApplication" />
        <activity
            android:name=".MainActivity4"
            android:exported="false"
            android:label="@string/title_activity_main4"
            android:theme="@style/Theme.OwlApplication" />
        <activity
            android:name=".MainActivity3"
            android:exported="false"
            android:label="@string/title_activity_main3"
            android:theme="@style/Theme.OwlApplication" />
        <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:label="@string/title_activity_main2"
            android:theme="@style/Theme.OwlApplication" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.OwlApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

CREATING NAVCOMPOSEAPP.Kt FILE:

```
package com.example.flashchat1



import androidx.compose.runtime.Composable

import androidx.compose.runtime.remember

import androidx.navigation.compose.NavHost

import androidx.navigation.compose.composable

import androidx.navigation.compose.rememberNavController

import com.google.firebase.auth.FirebaseAuth
```

```
import com.example.flashchat1.nav.Action

import com.example.flashchat1.nav.Destination.AuthenticationOption

import com.example.flashchat1.nav.Destination.Home

import com.example.flashchat1.nav.Destination.Login

import com.example.flashchat1.nav.Destination.Register

import com.example.flashchat1.ui.theme.FlashChat1Theme

import com.example.flashchat1.view.AuthenticationView

import com.example.flashchat1.view.home.HomeView

import com.example.flashchat1.view.login.LoginView

import com.example.flashchat1.view.register.RegisterView


/**

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

 */


@Composable

fun NavComposeApp() {

    val navController = rememberNavController()

    val actions = remember(navController) { Action(navController) }

    FlashChat1Theme {

        NavHost(

            navController = navController,

            startDestination =

            if (FirebaseAuth.getInstance().currentUser != null)

                Home

            else

                AuthenticationOption

        ) {

            composable(AuthenticationOption) {

                AuthenticationView(
```

```
                    register = actions.register,
                    login = actions.login
                )
            }
            composable(Register) {
                RegisterView(
                    home = actions.home,
                    back = actions.navigateBack
                )
            }
            composable(Login) {
                LoginView(
                    home = actions.home,
                    back = actions.navigateBack
                )
            }
            composable(Home) {
                HomeView()
            }
        }
    }
}
```

CREATING CONSTANTS OBJECTS:

```
package com.example.flashchat1


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"

}
```

CREATING NAVIGATION.Kt IN NAV PACKAGE:

```
package com.example.flashchat1.nav


import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.material.MaterialTheme

import androidx.compose.material.Surface

import androidx.compose.material.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import androidx.compose.ui.tooling.preview.Preview

import com.example.flashchat1.nav.ui.theme.Flashchat1Theme


import androidx.navigation.NavHostController

import com.example.flashchat1.nav.Destination.Home

import com.example.flashchat1.nav.Destination.Login

import com.example.flashchat1.nav.Destination.Register


/**

 * 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) {
            popUpTo(Login) {
                inclusive = true
            }
            popUpTo(Register) {
                inclusive = true
            }
        }
    }
    val login: () -> Unit = { navController.navigate(Login) }
    val register: () -> Unit = { navController.navigate(Register) }
    val navigateBack: () -> Unit = { navController.popBackStack() }
}
```

CREATING HOME PACKAGE:

```
package com.example.flashchat1.view.home
```

```
import androidx.compose.foundation.background

import androidx.compose.foundation.layout.*

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.items

import androidx.compose.foundation.text.KeyboardOptions

import androidx.compose.material.*

import androidx.compose.material.icons.Icons

import androidx.compose.material.icons.filled.Send

import androidx.compose.runtime.Composable

import androidx.compose.runtime.getValue

import androidx.compose.runtime.livedata.observeAsState

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.text.input.KeyboardType

import androidx.compose.ui.unit.dp

import androidx.lifecycle.viewmodel.compose.viewModel

import com.example.flashchat1.Constants

import com.example.flashchat1.view.SingleMessage


/**

 * 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.observeAsState(initial =
"")

    val messages: List<Map<String, Any>> by
homeViewModel.messages.observeAsState(

        initial = emptyList<Map<String, Any>>().toMutableList()

    )



    Column(

        modifier = Modifier.fillMaxSize(),

        horizontalAlignment = Alignment.CenterHorizontally,

        verticalArrangement = Arrangement.Bottom

    ) {

        LazyColumn(

            modifier = Modifier

                .fillMaxWidth()

                .weight(weight = 0.85f, fill = true),

            contentPadding = PaddingValues(horizontal = 16.dp, vertical =
8.dp),

            verticalArrangement = Arrangement.spacedBy(4.dp),

            reverseLayout = true

        ) {

            items(messages) { message ->

                val isCurrentUser = message[Constants.IS_CURRENT_USER] as
Boolean



                SingleMessage(

                    message = message[Constants.MESSAGE].toString(),

                    isCurrentUser = isCurrentUser

                )

            }

        }
```

```
        OutlinedTextField(
            value = message,
            onValueChange = {
                homeViewModel.updateMessage(it)
            },
            label = {
                Text(
                    "Type Your Message"
                )
            },
            maxLines = 1,
            modifier = Modifier
                .padding(horizontal = 15.dp, vertical = 1.dp)
                .fillMaxWidth()
                .weight(weight = 0.09f, fill = true),
            keyboardOptions = KeyboardOptions(
                keyboardType = KeyboardType.Text
            ),
            singleLine = true,
            trailingIcon = {
                IconButton(
                    onClick = {
                        homeViewModel.addMessage()
                    }
                ) {
                    Icon(
                        imageVector = Icons.Default.Send,
                        contentDescription = "Send Button"
                    )
                }
```

```
            }
        )
    }
}
```

HOME VIEW MODEL:

```
package com.example.flashchat1.view.home


import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.ktx.auth
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase
import com.example.flashchat1.Constants
import java.lang.IllegalArgumentException


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


    private val _message = MutableLiveData("")
    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")
        if (message.isNotEmpty()) {

Firebase.firestore.collection(Constants.MESSAGES).document().set(
                hashMapOf(
                    Constants.MESSAGE to message,
                    Constants.SENT_BY to Firebase.auth.currentUser?.uid,
                    Constants.SENT_ON to System.currentTimeMillis()
                )
            ).addOnSuccessListener {
                _message.value = ""
            }
        }
    }


    /**
     * Get the messages
```

```
     */
    private fun getMessages() {
        Firebase.firestore.collection(Constants.MESSAGES)
            .orderBy(Constants.SENT_ON)
            .addSnapshotListener { value, e ->
                if (e != null) {
                    Log.w(Constants.TAG, "Listen failed.", e)
                    return@addSnapshotListener
                }

                val list = emptyList<Map<String, Any>>().toMutableList()

                if (value != null) {
                    for (doc in value) {
                        val data = doc.data
                        data[Constants.IS_CURRENT_USER] =
                            Firebase.auth.currentUser?.uid.toString() ==
data[Constants.SENT_BY].toString()

                        list.add(data)
                    }
                }

                updateMessages(list)
            }
    }

    /**
     * Update the list after getting the details from firestore
     */
    private fun updateMessages(list: MutableList<Map<String, Any>>) {
```

```
        _messages.value = list.asReversed()

    }

}
```

CREATING LOGIN PACKAGE:

```
package com.example.flashchat1.view.login


import androidx.compose.foundation.layout.*

import androidx.compose.material.CircularProgressIndicator

import androidx.compose.runtime.Composable

import androidx.compose.runtime.getValue

import androidx.compose.runtime.livedata.observeAsState

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.text.input.KeyboardType

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.text.input.VisualTransformation

import androidx.compose.ui.unit.dp

import androidx.lifecycle.viewmodel.compose.viewModel

import com.example.flashchat1.view.Appbar

import com.example.flashchat1.view.Buttons

import com.example.flashchat1.view.TextFormField


/**

 * 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("")

    val password: String by loginViewModel.password.observeAsState("")

    val loading: Boolean by loginViewModel.loading.observeAsState(initial =
false)



    Box(

        contentAlignment = Alignment.Center,

        modifier = Modifier.fillMaxSize()

    ) {

        if (loading) {

            CircularProgressIndicator()

        }

        Column(

            modifier = Modifier.fillMaxSize(),

            horizontalAlignment = Alignment.CenterHorizontally,

            verticalArrangement = Arrangement.Top

        ) {

            Appbar(

                title = "Login",

                action = back

            )

            TextFormField(

                value = email,

                onValueChange = { loginViewModel.updateEmail(it) },

                label = "Email",
```

```
                keyboardType = KeyboardType.Email,
                visualTransformation = VisualTransformation.None
            )
            TextFormField(
                value = password,
                onValueChange = { loginViewModel.updatePassword(it) },
                label = "Password",
                keyboardType = KeyboardType.Password,
                visualTransformation = PasswordVisualTransformation()
            )
            Spacer(modifier = Modifier.height(20.dp))
            Buttons(
                title = "Login",
                onClick = { loginViewModel.loginUser(home = home) },
                backgroundColor = Color.Magenta
            )
        }
    }
}
```

LOGIN VIEW MODEL:

```
package com.example.flashchat1.view.login


import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.lang.IllegalArgumentException
```

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

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

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

    private val _loading = MutableLiveData(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
    fun loginUser(home: () -> Unit) {
        if (_loading.value == false) {
            val email: String = _email.value ?: throw
IllegalArgumentException("email expected")
            val password: String =
```

```
                _password.value ?: throw IllegalArgumentException("password
expected")


            _loading.value = true


            auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener {
                    if (it.isSuccessful) {
                        home()
                    }
                    _loading.value = false
                }
        }
    }
}
```

CREATING REGISTER PACKAGE:

```
package com.example.flashchat1.view.register


import androidx.compose.foundation.layout.*
import androidx.compose.material.CircularProgressIndicator
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
```

```
import androidx.compose.ui.unit.dp

import androidx.lifecycle.viewmodel.compose.viewModel

import com.example.flashchat1.view.Appbar

import com.example.flashchat1.view.Buttons

import com.example.flashchat1.view.TextFormField


/**

 * 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("")

    val password: String by registerViewModel.password.observeAsState("")

    val loading: Boolean by
registerViewModel.loading.observeAsState(initial = false)


    Box(

        contentAlignment = Alignment.Center,

        modifier = Modifier.fillMaxSize()

    ) {

        if (loading) {

            CircularProgressIndicator()

        }

        Column(
```

```
            modifier = Modifier.fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Top
        ) {
            Appbar(
                title = "Register",
                action = back
            )
            TextFormField(
                value = email,
                onValueChange = { registerViewModel.updateEmail(it) },
                label = "Email",
                keyboardType = KeyboardType.Email,
                visualTransformation = VisualTransformation.None
            )
            TextFormField(
                value = password,
                onValueChange = { registerViewModel.updatePassword(it) },
                label = "Password",
                keyboardType = KeyboardType.Password,
                visualTransformation = PasswordVisualTransformation()
            )
            Spacer(modifier = Modifier.height(20.dp))
            Buttons(
                title = "Register",
                onClick = { registerViewModel.registerUser(home = home) },
                backgroundColor = Color.Blue
            )
        }
    }
}
```

REGISTER VIEW MODEL:

```
package com.example.flashchat1.view.register


import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.lang.IllegalArgumentException


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

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

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

    private val _loading = MutableLiveData(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
    fun registerUser(home: () -> Unit) {
        if (_loading.value == false) {
            val email: String = _email.value ?: throw
IllegalArgumentException("email expected")
            val password: String =
                _password.value ?: throw IllegalArgumentException("password
expected")

            _loading.value = true

            auth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener {
                    if (it.isSuccessful) {
                        home()
                    }
                    _loading.value = false
                }
        }
    }
}
```