A Project Report on

# Adaptive mail: A Real Time Chat Connect App

BACHELOER OF INFORMATION TECHNOLOGY(2020-2023)

Saiva Bhanu Kshatriya College,

(Affiliated to Madurai Kamaraj University)

Aruppukottai-626101

Submitted

By

| | |
|---|---|
| R.Mohana bairavi | Reg.No C0S24051 |
| S.Arunkumar | Reg.No C0S24052 |
| P.Eswaraganapathy | Reg.No C0S24053 |
| S.SanjayKumar | Reg.No C0S24054 |
| P.Veeruchamy | Reg.No C0S24055 |

# A REALTIME CHAT CONNECT-APP

## ABSTRACT:

- The online chatting application is a web-based management application. In this system, the user can review the chatting system. In this system, the owner can make their account online and have a good conversation in the chat. The emergence of computer networks and telecommunication technologies allows people to communicate in a new way.

- Chatting is a method of using technology to bring people and ideas together despite geographical barriers. The technology has been available for years but the acceptance was quite recent. The group chat application will allow multiple users to connect to the server and chat with all other online users. The app works in a broadcast fashion.

- This means that messages from a user are broadcasted to other users. Messaging apps are surging in popularity. The past few years have brought apps like WhatsApp, Telegram, etc.

# 1. INTRODUCTION

## 1.1 OverView:

Teleconferencing or Chatting, is a method of using technology to bring people and ideas "together" despite of the geographical barriers. The technology has been available for years but the acceptance it was quit recent. Our project is an example of a chat server. It is made up of 2 applications the client application, which runs on the user's Pc and server application, which runs on any Pc on the network. To start chatting client should get connected to server where they can practice two kinds of chatting, public one (message is broadcasted to all connected users) and private one (between any 2 users only) and during the last one security measures were taken.

## 1.2 Propose:

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.

## 2. Problem Definition & Design Thinking

### 2.1 Empathy Map

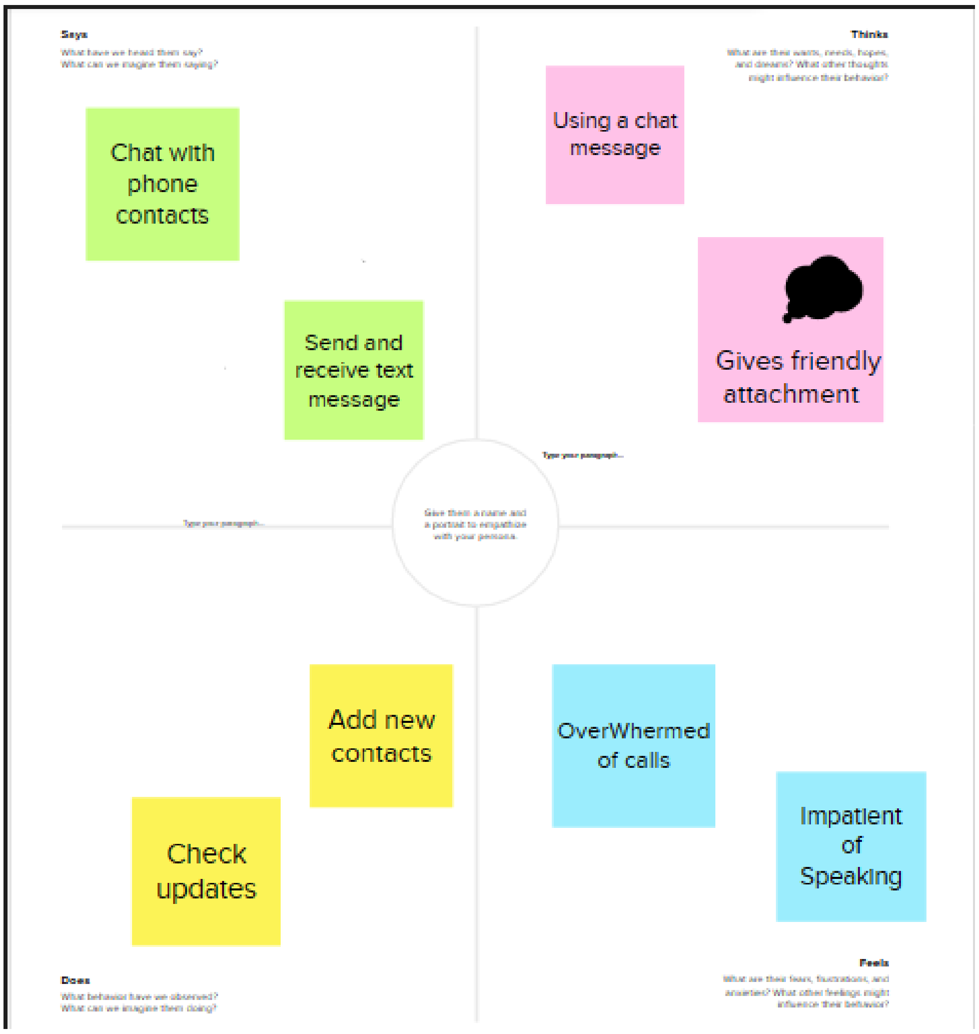
An empathy map is a template that organizes a user’s behaviours and feelings to create a sense of empathy between the user and your team. The empathy map represents a principal user and helps teams better understand their motivations, concerns, and user experience.

Empathy mapping is a simple yet effective workshop that can be conducted with a variety of different users in mind, anywhere from stakeholders, individual use cases, or entire teams of people. It can be conducted by many different teams such as design teams, sales, product development or customer service. Essentially, it is an exercise that seeks to get inside the head of the customer as they interact with your product/service.

While the main importance of an empathy map is creating empathy between you and the user, there are some other important facets of using one that offer different benefits to your team. Creating an empathy map takes many factors into consideration in relation to the customer’s overall experience. These could be the specific problems they handle, how they use the product/service within a larger team, and who really experiences the brunt of the problem.

` These details are important to creating a holistic view of their experience but also important because they illuminate the problem in the mind of your team. This is equally as important and helps build an overall understanding of how users interact with your product/service.

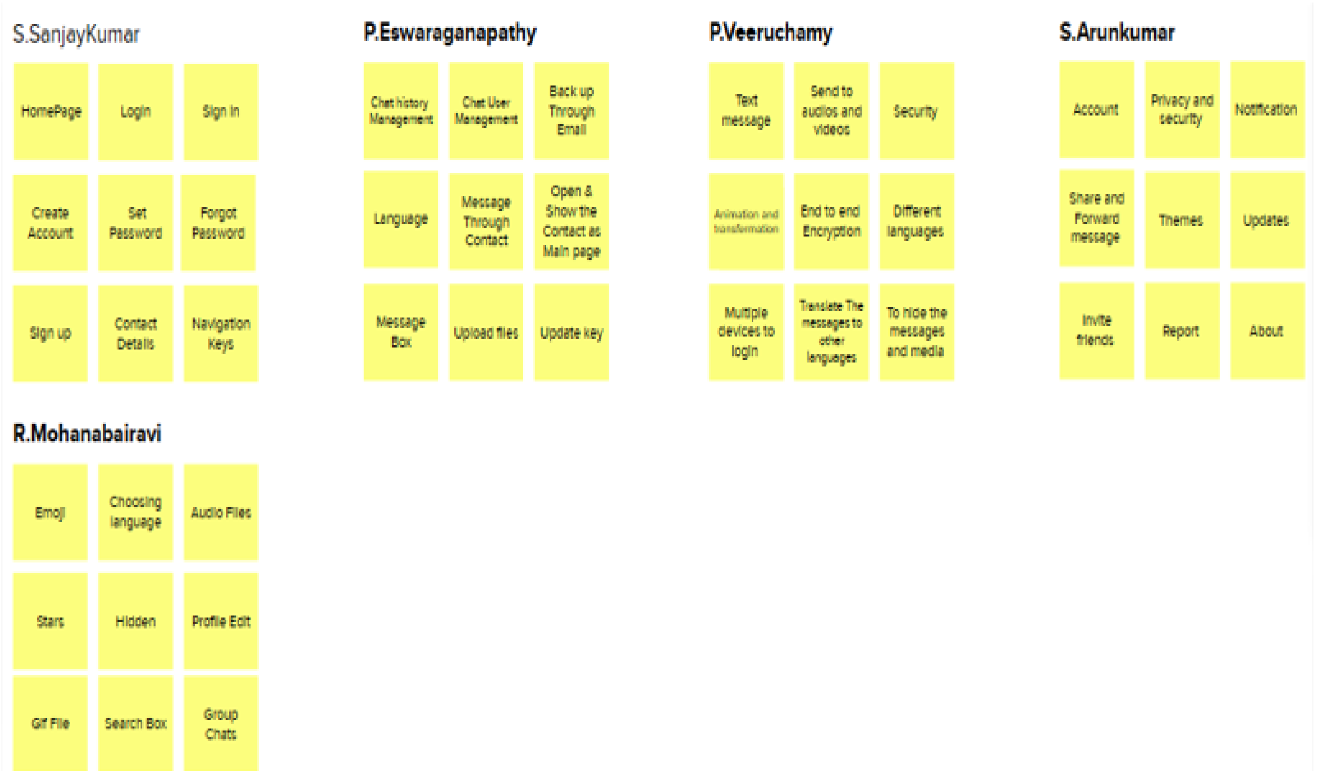
## Empathy Map


Says
What have we heard them say?
What can we imagine them saying?
Chat with phone contacts
Send and receive text message
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
Using a chat message
Gives friendly attachment
Type your paragraph...
Give them a name and a portrait to empathize with your persona.
Type your paragraph...
Add new contacts
Check updates
OverWhermed of calls
Impatient of Speaking
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

## 2.2 Brainstorming Map

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.

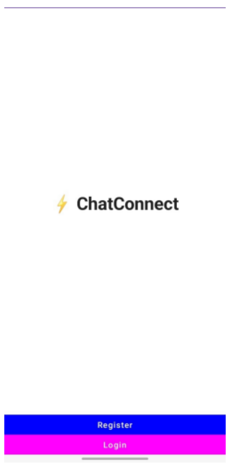
Brainstorming combines an informal approach to problem-solving with lateral thinking, which is a method for developing new concepts to solve problems by looking at them in innovative ways. Some of these ideas can be built into original, creative solutions to a problem, while others can generate additional ideas.

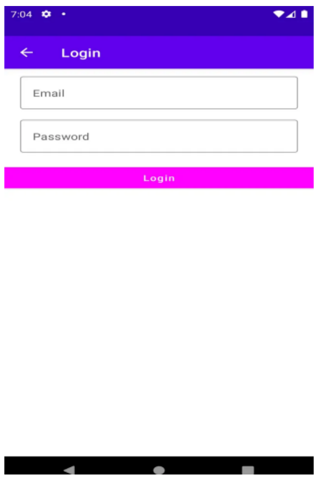


HomePage
Language
Profile Editin
Group Chat
Login
Share and Forward message 
Sign In

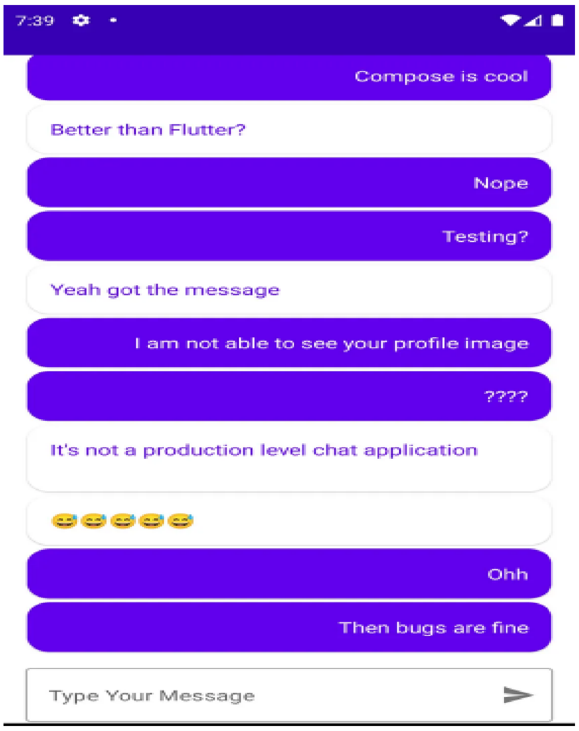
## 3. RESULT

Login Page :





## Home Screen:


7:39

Compose is cool

Better than Flutter?

Nope

Testing?

Yeah got the message

I am not able to see your profile image

????

It's not a production level chat application

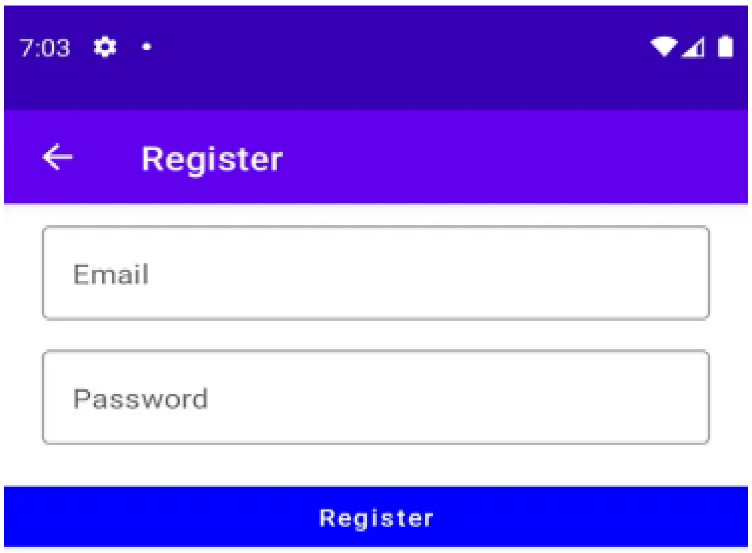
😅😅😅😅😅

Ohh

Then bugs are fine

Type Your Message

## RegisterPage :


7:03

Register

Email

Password

Register

## 4. ADVANTAGES &DISADVANTAGES

**ADVANTAGES:**

- Speed. A chat application allows you to message or contact a person in real-time. ...
- Familiarity
- Convenience
- Segmented Target Advertising
- Increased Productivity
- File Storage and Sharing
- Employee Engagement
- Privacy

**DISADVANTAGES:**

- You can't be sure other people are being honest or that they are who they say they are.
- If you are feeling vulnerable, people online might try to take advantage of you.
- Building relationships online can result in your spending less time with friends and family.

## 5. APPLICATIONS

**WhatsApp:**

WhatsApp Messenger is a free instant messaging app available on both Android and iphone. It allows you to send text messages to other users one-on-one or in groups. Importantly, WhatsApp chats go over the internet.

**Telegram:**

Telegram is a messaging app with a focus on speed and security, it’s super-fast, simple and free. You can use Telegram on all your devices at the same time — your messages sync seamlessly across any number of your phones, tablets or computers.

## 6. CONCLUSION

The chat app provides a better and more flexible chat system. Developed with the latest technology in the way of providing a reliable system. The main advantage of the system is instant messaging, real-world communication, added security, group chat, etc. This application may find the best demand in the market for most organizations that aim to have independent applications.

## 7. FUTURE SCOPE

We try to manage the private chat in this system as current system is based on broadcasting of messages.

We will try to design more interactive GUI and provide more facility for user e.g. to manage his/her account separately.

We will try to record sound of user.  In future we developed the full fletched database application associate with current system.

# 8. APPENDIX

MAIN ACTIVITY

```
package com.project.pradyotprakash.flashchat

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import com.google.firebase.FirebaseApp


/**

 * The initial point of the application from where it gets started.

 *

 * Here we do all the initialization and other things which will be required

 * thought out the application.

 */

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        FirebaseApp.initializeApp(this)

        setContent {

            NavComposeApp()

        }

    }

}
```

NAV COMPOSE APP

```
package com.project.pradyotprakash.flashchat

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.project.pradyotprakash.flashchat.nav.Action

import
com.project.pradyotprakash.flashchat.nav.Destination.AuthenticationOption

import com.project.pradyotprakash.flashchat.nav.Destination.Home

import com.project.pradyotprakash.flashchat.nav.Destination.Login

import com.project.pradyotprakash.flashchat.nav.Destination.Register

import com.project.pradyotprakash.flashchat.ui.theme.FlashChatTheme

import com.project.pradyotprakash.flashchat.view.AuthenticationView

import com.project.pradyotprakash.flashchat.view.home.HomeView

import com.project.pradyotprakash.flashchat.view.login.LoginView

import com.project.pradyotprakash.flashchat.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) }
    FlashChatTheme {
        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()
            }
        }
    }
}
```

ANDROID MANI FEST

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.project.pradyotprakash.flashchat">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.FlashChat">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.FlashChat.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

NAVIGATION

```
package com.project.pradyotprakash.flashchat.nav
import androidx.navigation.NavHostController
import com.project.pradyotprakash.flashchat.nav.Destination.Home
import com.project.pradyotprakash.flashchat.nav.Destination.Login
import com.project.pradyotprakash.flashchat.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() }
}
```

HOME

```
package com.project.pradyotprakash.flashchat.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.project.pradyotprakash.flashchat.Constants
import com.project.pradyotprakash.flashchat.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.project.pradyotprakash.flashchat.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.project.pradyotprakash.flashchat.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()
    } }
```

LOGIN

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

import com.project.pradyotprakash.flashchat.view.Buttons

import com.project.pradyotprakash.flashchat.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.project.pradyotprakash.flashchat.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

                }

        }

    }

}
```

REGISTER

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

import com.project.pradyotprakash.flashchat.view.Buttons

import com.project.pradyotprakash.flashchat.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.project.pradyotprakash.flashchat.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
                }
        }
    }
}
```

AUTHENTICATION OPTION

```
package com.project.pradyotprakash.flashchat.view
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import com.project.pradyotprakash.flashchat.ui.theme.FlashChatTheme

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

@Composable
```

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

WIDGETS

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

```
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.project.pradyotprakash.flashchat.Constants

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

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

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

@Composable
fun Appbar(title: String, action: () -> Unit) {
    TopAppBar(
        title = {
            Text(text = title)
        },
        navigationIcon = {
            IconButton(
                onClick = action
            ) {
                Icon(
                    imageVector = Icons.Filled.ArrowBack,
                    contentDescription = "Back button"
                )
            }
        }
    )
}

@Composable
```

```
fun TextFormField(value: String, onValueChange: (String) -> Unit, label:
String, keyboardType: KeyboardType, visualTransformation:
VisualTransformation) {
    OutlinedTextField(
        value = value,
        onValueChange = onValueChange,
        label = {
            Text(
                label
            )
        },
        maxLines = 1,
        modifier = Modifier
            .padding(horizontal = 20.dp, vertical = 5.dp)
            .fillMaxWidth(),
        keyboardOptions = KeyboardOptions(
            keyboardType = keyboardType
        ),
        singleLine = true,
        visualTransformation = visualTransformation
    )
}
@Composable
fun SingleMessage(message: String, isCurrentUser: Boolean) {
    Card(
        shape = RoundedCornerShape(16.dp),
        backgroundColor = if (isCurrentUser) MaterialTheme.colors.primary else
Color.White
    ) {
        Text(
            text = message,
            textAlign =
            if (isCurrentUser)
                TextAlign.End
            else
                TextAlign.Start,
            modifier = Modifier.fillMaxWidth().padding(16.dp),
            color = if (!isCurrentUser) MaterialTheme.colors.primary else
Color.White )
    }
}
```

CONSTANTS

```
package com.project.pradyotprakash.flashchat

object Constants
 {
    const val TAG = "flash-chat"
    const val MESSAGES = "messages"
    const val MESSAGE = "message"
    const val SENT_BY = "sent_by"
    const val SENT_ON = "sent_on"
    const val IS_CURRENT_USER = "is_current_user"
}
```