CHAT CONNECT - A REAL TIME CHAT AND COMMUNICATION APP

PROJECT PRESENTED BY:

CATEGORY: ANDROID APPLICATION DEVELOPMENT

TEAM ID: NM2023TMID10879

TEAM LEADER: PRAVEEN KUMAR S

TEAM MEMBERS:

NITHISWARAN S

PRASANNA C

RAJESHKANNA S

1) INTRODUCTION

1.1 Overview

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.

A messaging application is **a mobile-phone-based software programme that allows users to send and receive information using their phone's internet connection**. Messaging apps can transmit or receive a much wider range of data types than Short Message Service (SMS) or Multimedia Messaging Service (MMS).

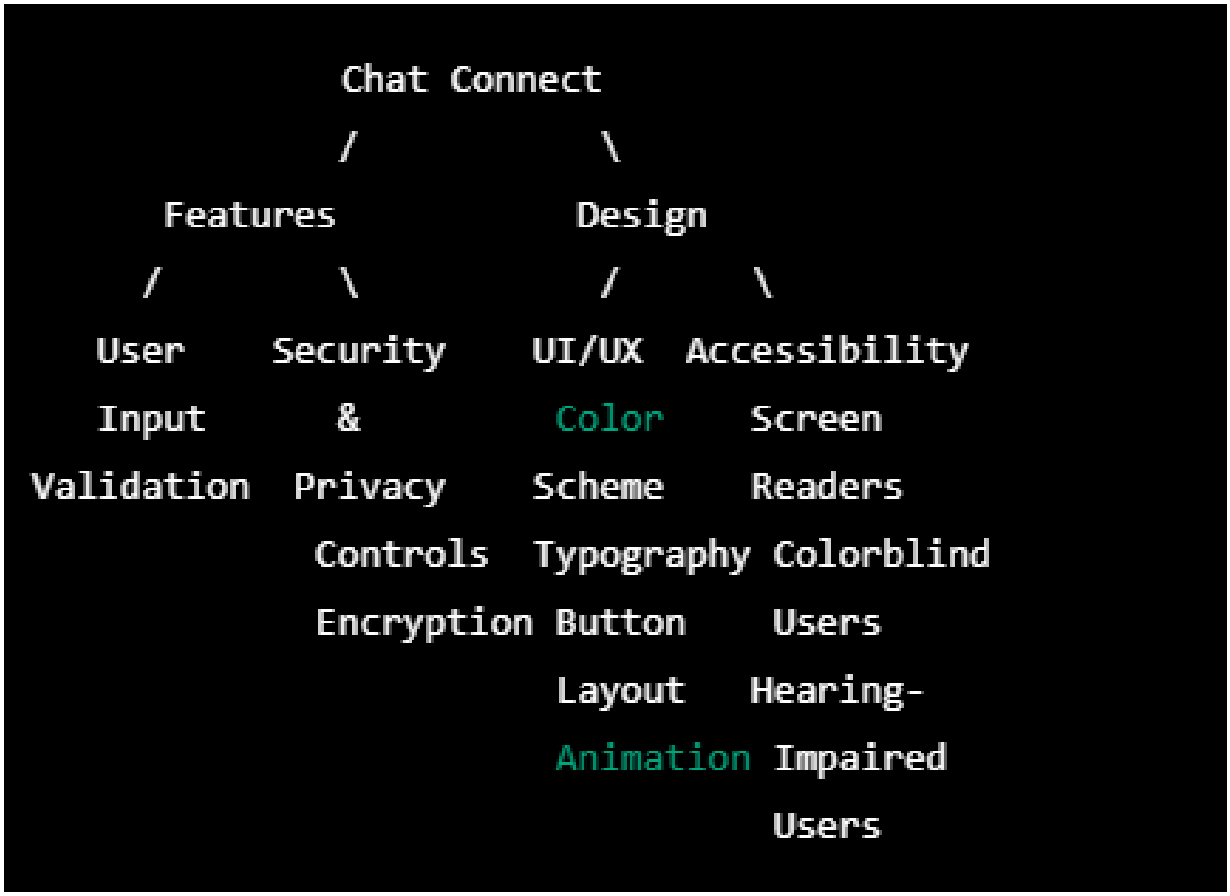
1.2 Purpose

Messaging has become a part of our everyday lives in part due to its convenience for real-time chat communication and simple-to-use functionality. For instance, an iOS or text message on an iPhone or Android device from a friend, an email from a co-worker on Microsoft or Gmail, a team chat in a Slack or Microsoft Teams workspace, or even instant messaging through social media. These messaging and real-time chat applications play an important role in how the world interacts today, due to their immediacy and vast capabilities.

2) Problem Definition & Design Thinking

2.1 Empathy Map

2.2 Ideation & Brainstorming Map



3) RESULT:

4) ADVANTAGES:

A chat connect app has several advantages, including:

Improved communication: A chat connect app allows users to communicate in real-time, making it easier and more efficient to exchange information and ideas.

Increased productivity: With the ability to quickly communicate and collaborate on tasks, a chat connect app can help teams work more efficiently and complete projects faster.

Remote work friendly: Chat connect apps are particularly useful for remote teams or those working from home, as they allow for easy communication across different locations and time zones.

Cost-effective: Chat connect apps are often free or low-cost, making them a cost-effective solution for businesses of all sizes.

Customizable: Many chat connect apps offer a range of customization options, allowing users to tailor the app to their specific needs and preferences.

Increased accessibility: With the ability to access the app from any device with an internet connection, chat connect apps make it easy for users to stay connected and up-to-date on important conversations and projects.

DISADVANTAGES:

some potential disadvantages to consider:

Overuse and distraction: Chat connect apps can be a source of distraction, as users may be tempted to check for new messages frequently or engage in non-work related conversations.

Lack of context: Without face-to-face interaction, it can be difficult to convey tone and context in a chat conversation.

5) APPLICATIONS:

Chat connect apps can be applied in many different areas, including:

Business: Chat connect apps can be used by businesses to improve internal communication, collaboration, and project management. They can also be used to communicate with customers and provide customer support.

Education: Chat connect apps can be used in the classroom to facilitate communication and collaboration between students and teachers. They can also be used for online learning and remote education.

Healthcare: Chat connect apps can be used by healthcare providers to communicate with patients, share information, and coordinate care.

Social networking: Chat connect apps can be used for social networking and online communities, allowing users to connect and communicate with others who share similar interests.

6) CONCLUSION

In conclusion, chat connect apps provide a convenient and efficient way for individuals and teams to communicate and collaborate in real-time. With features such as instant messaging, file sharing, and video conferencing, chat connect apps can help improve productivity and streamline workflows. However, it is important to be aware of potential disadvantages such as distraction, information overload, and security concerns, and to take steps to mitigate them. Chat connect apps have many potential applications across various industries and areas, including business, education, healthcare, social networking, government, and non-profit organizations. Overall, chat connect apps can be a valuable tool for enhancing communication and collaboration in today's digital age.

7) FUTURE SCOPE:

The future scope of chat connect apps is promising, as advances in technology and changes in the way we work and communicate are driving the need for more efficient and convenient communication tools. Some potential areas of future development for chat connect apps include:

Artificial intelligence: As chat connect apps become more sophisticated, they may incorporate AI-powered features such as natural language processing, sentiment analysis, and chatbots to improve communication and streamline workflows.

Virtual and augmented reality: Chat connect apps may incorporate virtual and augmented reality features to provide more immersive and interactive communication experiences.

Integration with other tools: Chat connect apps may become more integrated with other productivity tools such as project management software, calendar apps, and email, providing a more seamless workflow.

Privacy and security: With increasing concerns around privacy and security, chat connect apps may incorporate stronger security features such as end-to-end encryption and multi-factor authentication to protect sensitive information.

Personalization: Chat connect apps may become more personalized, allowing users to customize the app to their specific needs and preferences.

Overall, the future of chat connect apps looks bright, with potential for continued innovation and development to meet the evolving needs of users in various industries and areas.

8) APPENDIX

A.Source code

ChatConnect – A Real Time chat and Communication App

AndroidManifest.xml

```
<?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.kt

```
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() }
}
```

Color.kt

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)
```

Shape.kt

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(4.dp),
    large = RoundedCornerShape(0.dp)
)
```

Theme.kt

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200
)

@Composable
fun FlashChatTheme(darkTheme: Boolean = isSystemInDarkTheme(), content:
@Composable() () -> Unit) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

Type.kt

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

/**
 * Set of Material typography styles to start with
 */
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
)
```

Home.kt

```
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"
                    )
                }
            }
        )
    }
}
```

HomeViewModel.kt

```
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.kt

```
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
            )
        }
    }
}
```

LoginViewModel.kt

```
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.kt

```
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.kt

```
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
            )
        }
    }
}
```

RegisterViewModel.kt

```
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
                }
        }
    }
}
```

AuthndicationOption.kt

```
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.kt

```
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.kt

```
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"
}
```

MainActivity.kt

```
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()
        }
    }
}
```

NavComposeApp.kt

```
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()
            }
        }
```

```
        }
    }
```