Chatconnect-A real-time chat and communication App

Presented By

Team ID: NM2023TMID34977

Team Size: 4

Team Leader: MUGUNDAN K

Team member: lyappan K

Team member : Mohamed Jassim S

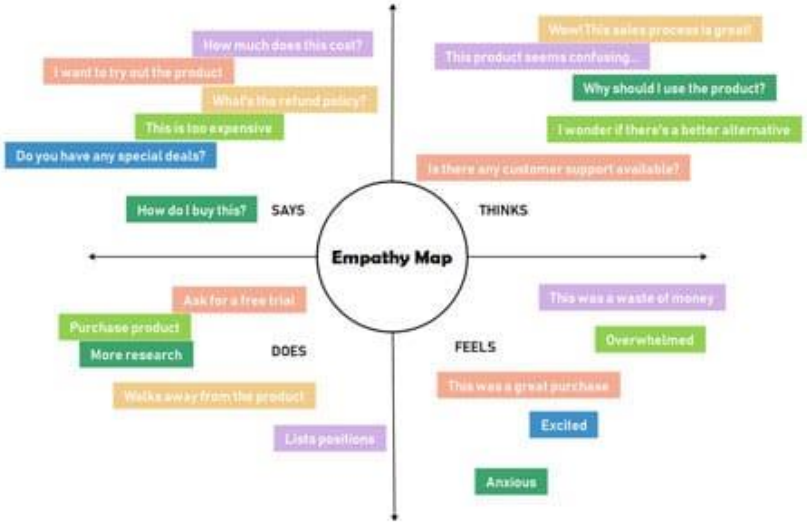
Team member : MOHAMED RIYAZ M H

INTRODUCTION:

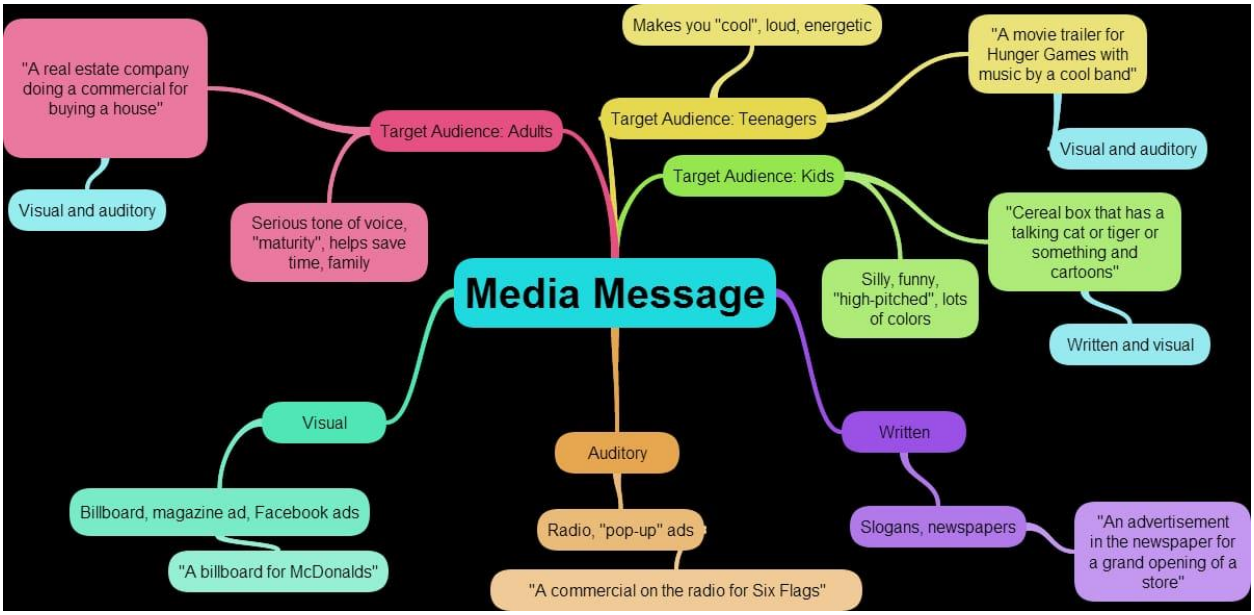
Chatconnect is a cutting-edge, user-friendly chat and communication app that allows individuals and groups to connect and communicate in real-time. With its intuitive interface and powerful features, Chatconnect is designed to provide seamless communication experiences across various platforms, including mobile devices and desktop computers.

With Chatconnect, users can send and receive text messages, voice messages, images, videos, and other multimedia files in real-time. The app also supports group chats, allowing users to create and join groups for specific topics, interests, or communities. Users can also engage in one-on-one private chats with friends, family, or colleagues. Chatconnect offers a range of features to enhance communication, such as read receipts, message reactions, message editing, and the ability to delete messages. Users can also customize their profile, set status updates, and manage their notification settings to suit their preferences.

EMPATHY MAP:



BRAINSTORM MAP:



Advantages:

Instant Communication: ChatConnect allows users to communicate in real-time, enabling quick and efficient conversations. Users can send and receive messages instantly, making it ideal for fast-paced communication scenarios where timely responses are crucial.

User-Friendly Interface: ChatConnect offers a user-friendly interface that is easy to navigate, making it accessible to users of all skill levels. It provides a smooth and seamless user experience, allowing users to chat and communicate without any technical barriers.

Multi-Platform Accessibility: ChatConnect is available across multiple platforms, including web browsers, mobile devices, and desktop applications. This allows users to access the app from various devices, making it convenient and accessible for communication on the go.

Real-Time Notifications: ChatConnect provides real-time notifications for new messages, ensuring that users are promptly alerted when they receive a new message. This feature helps users stay updated and respond promptly to incoming messages.

Rich Media Sharing: ChatConnect allows users to share various types of media, including text, images, videos, documents, and more. This makes it versatile for different types of communication, such as sharing files for collaboration, sharing multimedia content, and expressing ideas effectively.

Disadvantage:

Reliance on internet connection: Chatconnect requires a stable internet connection to function properly. If the internet connection is weak or unreliable, it may result in poor performance, lag, or even disconnection from the app.

Security and privacy concerns: Chatconnect may store user data, including personal information and chat logs. If the app does not have robust security

measures in place, it may be vulnerable to data breaches, hacking, or unauthorized access, potentially compromising user privacy.

Technical issues and bugs: Like any software application, Chatconnect may experience technical issues, bugs, or glitches that could impact its performance or usability. This may require frequent updates or patches to fix the issues, which can be inconvenient for users.

Compatibility and device limitations: Chatconnect may not be compatible with all devices or operating systems, or it may have limited functionality on certain platforms. This could limit its accessibility for some users or require them to use specific devices or software to access the app.

User experience and interface: The user interface of Chatconnect may not be intuitive or user-friendly for all users. It may take time for users to learn how to use the app effectively, which could result in a learning curve and potentially lead to frustration.

Appendix:

Supported Platforms: Chatconnect is available on iOS and Android mobile devices, as well as web browsers for desktop and laptop computers.

Future Scope:

Integration with emerging technologies: Chatconnect can explore integration with emerging technologies such as augmented reality (AR), virtual reality (VR), and Internet of Things (IoT) to enhance user experiences. For example, enabling AR/VR-powered communication or integrating with IoT devices for seamless communication and control.

XML CODES:

```
<?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()
        }
    }
  }
}
```