CHATCONNECT – A REAL – TIME CHAT AND COMMUNICATION APP

1. INTRODUCTION

1.1 Project Description:

ChatConnect is a sample project built using the Android Compose UI toolkit. It demonstrates how to create a simple chat app using the Compose libraries. The app allows users to send and receive text messages. The project showcases the use of Compose's declarative UI and state management capabilities. It also includes examples of how to handle input and navigation using composable functions and how to use data from a firebase to populate the UI.

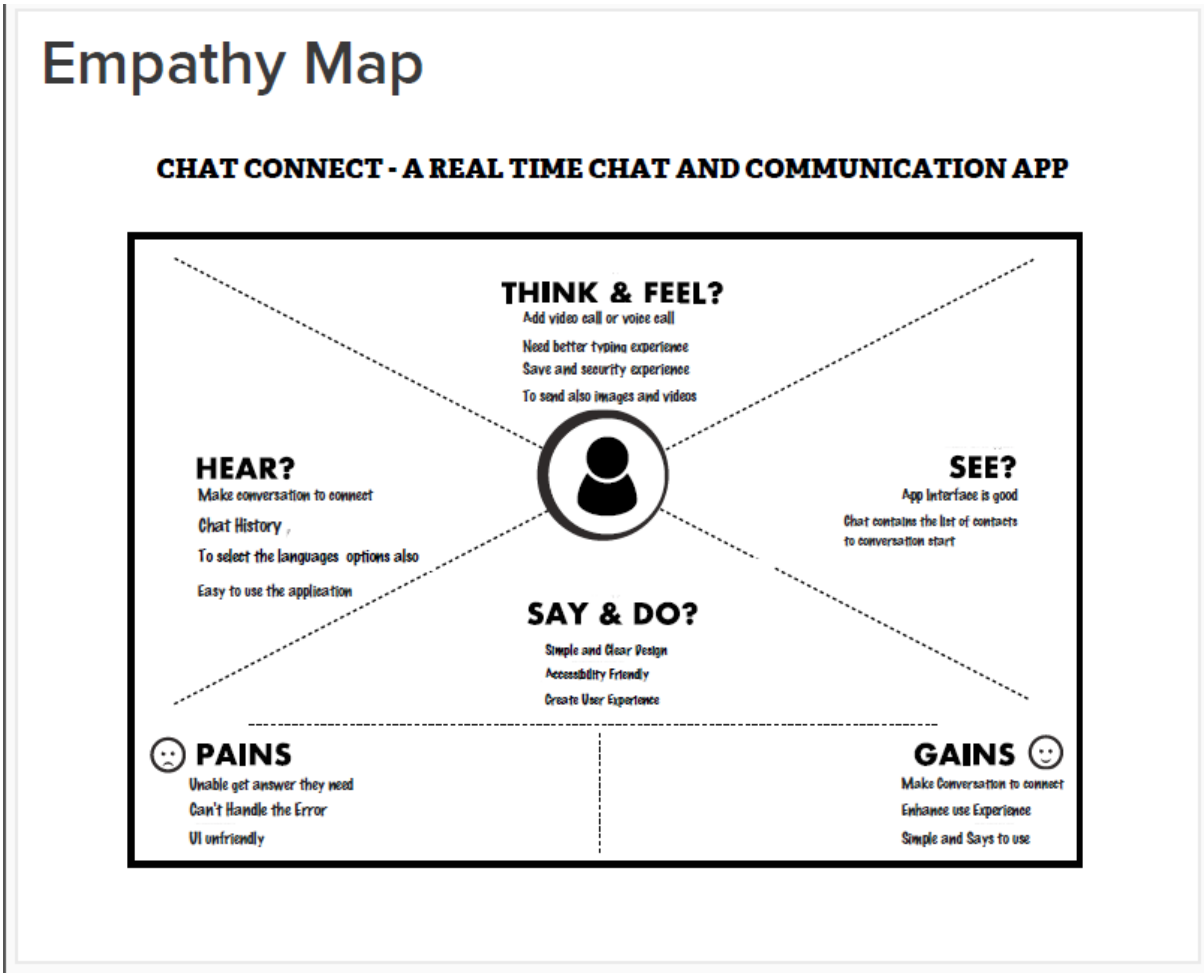
1.2 Purpose of this project:

The primary purpose of "Chat connect" is to allow people to stay connected and communicate effectively in real-time. The app can be used for a wide range of purposes, including personal communication, business communication, and online collaboration. It can be used by individuals, teams, organizations, and communities to facilitate communication and collaboration.

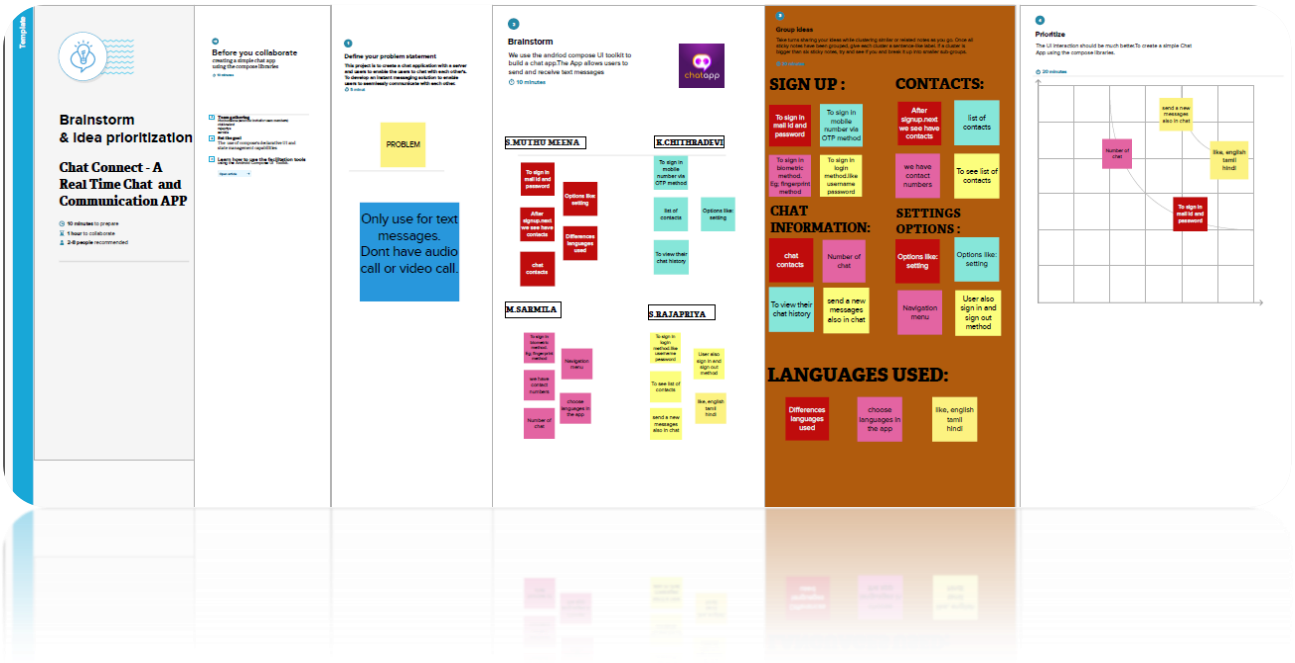
One of the key benefits of "Chat connect" is its ability to help people stay connected in real-time, regardless of their location. With the app, users can chat with friends, family, and colleagues from anywhere in the world. This can be especially useful for people who travel frequently or have family and friends living in different parts of the world.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map:

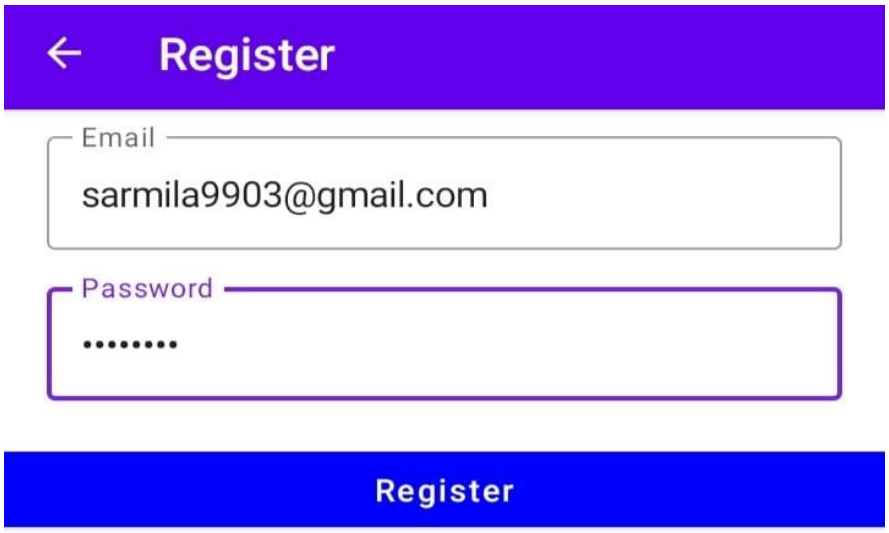


2.2 Brainstorming Map:

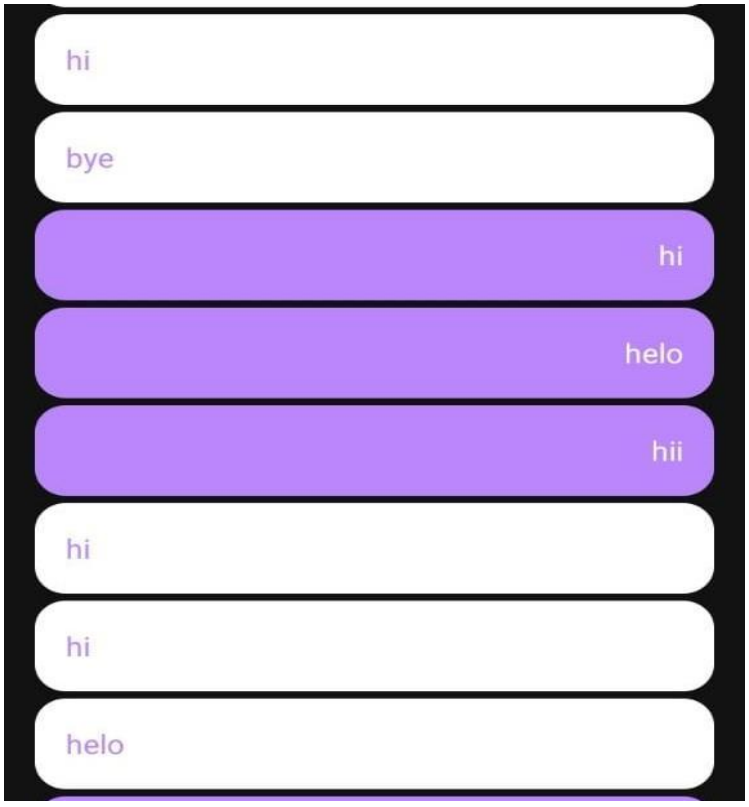


3. RESULT

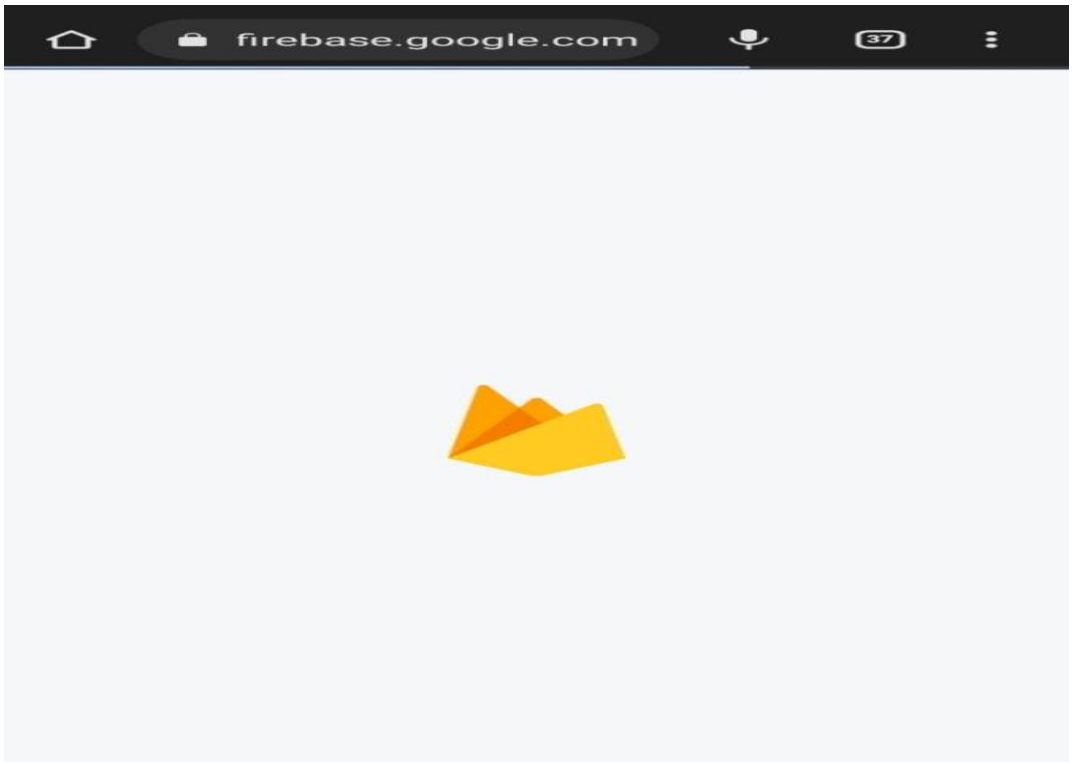

Chat Connect
Register
Login

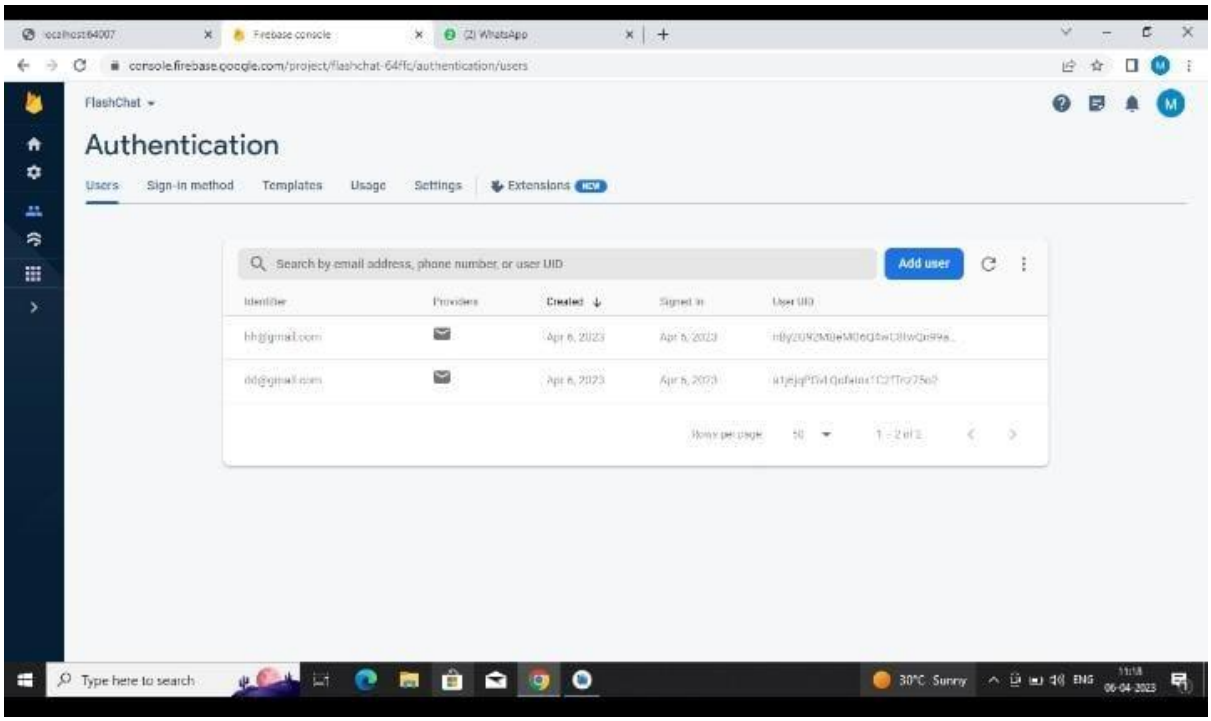

Register
Email
sarmila9903@gmail.com
Password
••••••••
Register

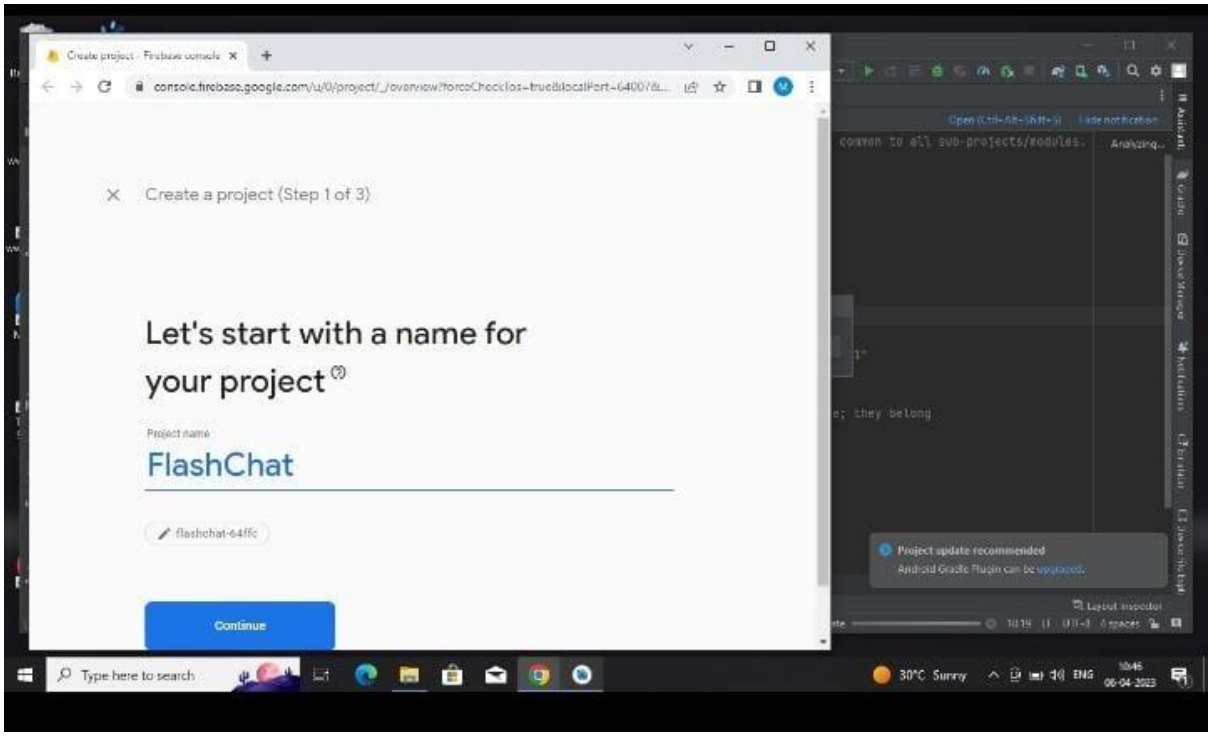

Login
Email
rajapriyasankar13@gmail.com
Password
Login

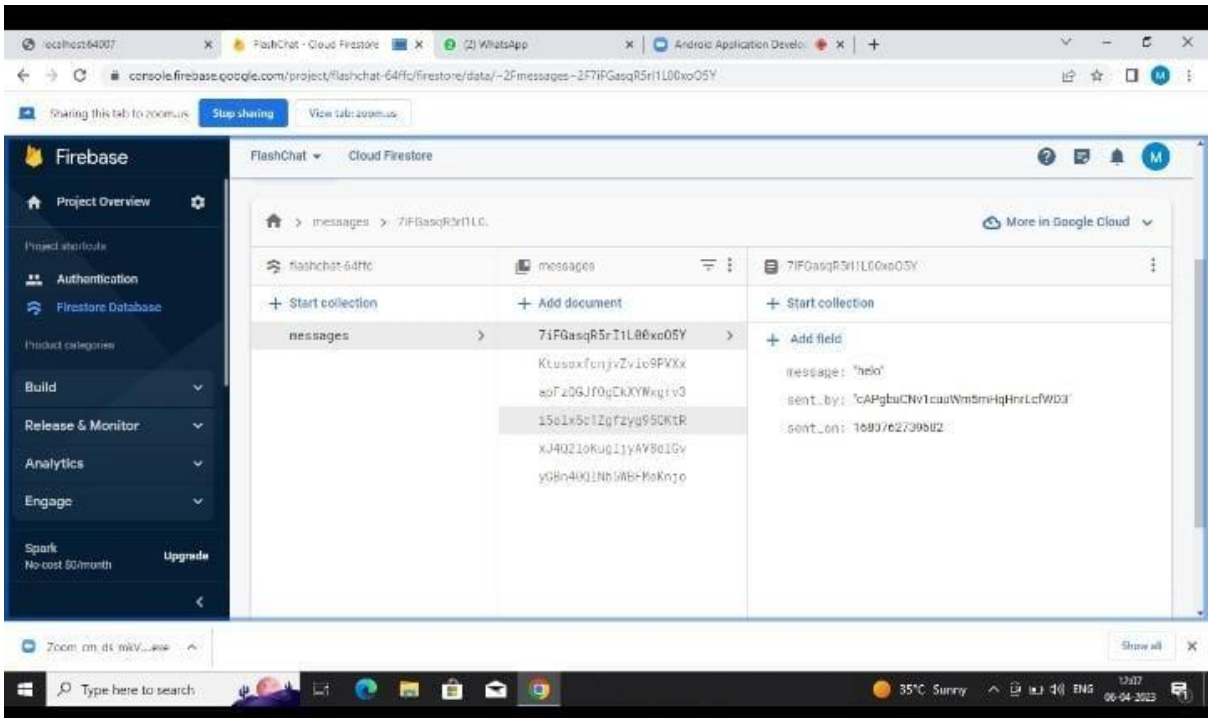

hi
bye
hi
helo
hii
hi
hi
helo

INTEGRATING FIREBASE






Create project - Firebase console
console.firebase.google.com/u/0/project/_/overview?forceChecklist=true&localPort=64007&...
Create a project (Step 1 of 3)
Let's start with a name for your project
Project name
FlashChat
flashchat-64ffc
Continue
Project update recommended
Android Gradle Plugin can be upgraded.
Type here to search
30°C Sunny
ENG


console.firebase.google.com/project/flashchat-64ffc/firestore/data/~2Fmessages~2F7iFGasqR5rI1L00xoO5Y
Stop sharing
Firebase
FlashChat
Cloud Firestore
Project Overview
Authentication
Firestore Database
Build
Release & Monitor
Analytics
Engage
Spark
No-cost $0/month
Upgrade
messages
flashchat-64ffc
Start collection
Add document
Add field
More in Google Cloud
Type here to search
35°C Sunny
ENG

ADVANTAGES AND DISADVANTAGES

Advantages:

- **Faster support**
- **Real-time test preview**
- **No waiting queues**
- **Familiarity**
- **Convenience**

Disadvantages:

- **No video calls**
- **No audio calls**
- **Not to send for the transfer**

5. APPLICATIONS

Personal communication:

"Chat connect" can be used for personal communication, allowing individuals to stay connected with their friends and family in real-time, regardless of their location.

Social networking:

"Chat connect" can be used for social networking, enabling like-minded individuals to connect and communicate with one another in real-time.

Healthcare:

"Chat connect" can be used in the healthcare sector, enabling doctors, nurses, and other healthcare professionals to communicate and collaborate more effectively, especially in emergency situations.

6. CONCLUSION

"Chat connect" is a real-time chat and communication app that provides users with a simple and user-friendly interface for staying connected and communicating with one another. The app can be used in various areas, including personal communication, business communication, online collaboration, education, healthcare, customer support, and social networking.

With its ability to facilitate real-time communication and collaboration, "Chat connect" can help individuals, teams, organizations, and communities stay connected and achieve their goals more efficiently. Its features, such as file sharing, chat rooms, and voice and video calling capabilities, make it a versatile tool for staying connected and collaborating in real-time.

Overall, "Chat connect" is a powerful solution for those seeking a user-friendly and efficient means of communication and collaboration in real-time. Its potential applications are numerous, and it has the potential to make a significant impact in many different fields.

7. FUTURE SCOPE

There is always some place for enhancements in any software application, however good and efficient the application may be.

Right now, we are dealing with only the instant messaging between the peers. In future the application may further developed to include some features such as

1. Voice messaging.
2. Group calling
3. Live streaming
4. Messages auto delete after a given time.
5. Personalized message tunes.

And a messaging application feature which allows the user to create chat room while in conversation with another user by just sending the chatroom name with the hash symbol at the beginning

8. APPENDIX

a) Source code

```
packagecom.project.pradyotprak
ash.flashchat

import android.os.Bundle
import
androidx.activity.Compone
ntActivity
import
androidx.activity.compose
.setContent
import
com.google.firebase.Fireb
aseApp
/**
* 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(savedInstanceSta
te: Bundle?) {
super.onCreate(savedInsta
nceState)
FirebaseApp.initializeApp
(this)
setContent {
NavComposeApp()
}
}
}
```

```
Package
com.project.prad
yotprakash.flash
chat

                    import
                    androidx.compose.runtime.Composabl
                    e
                    import
                    androidx.compose.runtime.remember
                    import
                    androidx.navigation.compose.NavHos
                    t
                    import
                    androidx.navigation.compose.compos
                    able
                    import
                    androidx.navigation.compose.rememb
                    erNavController
                    import
                    com.google.firebase.auth.FirebaseA
                    uth
                    import
                    com.project.pradyotprakash.flashch
                    at.nav.Action
                    import
                    com.project.pradyotprakash.flashch
                    at.nav.Destination.AuthenticationO
                    ption
                    import
                    com.project.pradyotprakash.flashch
                    at.nav.Destination.Home
                    import
                    com.project.pradyotprakash.flashch
                    at.nav.Destination.Login
                    import
                    com.project.pradyotprakash.flashch
                    at.nav.Destination.Register
                    import
                    com.project.pradyotprakash.flashch
```

```
at.ui.theme.FlashChatTheme
import
com.project.pradyotprakash.flashch
at.view.AuthenticationView
import
com.project.pradyotprakash.flashch
at.view.home.HomeView
import
com.project.pradyotprakash.flashch
at.view.login.LoginView
import
com.project.pradyotprakash.flashch
at.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().curren
tUser != 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()
}
}
}
}
```

```
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"
}
```

```
package com.project.pradyotprakash.flashchat.nav
```

```
package
com.project.pradyotprakash.flashchat.view
                                                    import androidx.compose.f
                                                    import androidx.compose.f
```

```
import androidx.compose.f
import androidx.compose.f
import androidx.compose.f
import androidx.compose.m
import androidx.compose.r
import androidx.compose.u
import androidx.compose.u
import androidx.compose.u
import
com.project.pradyotprakas
/**
* The authentication view
to choose between
* login and register.
*/
@Composable
fun AuthenticationView(re
Unit) {
FlashChatTheme {
// A surface container us
theme
Surface(color = MaterialT
Column(
modifier = Modifier
.fillMaxWidth()
.fillMaxHeight(),
horizontalAlignment = Ali
verticalArrangement = Arr
) {
Title(title = "⚡ Chat Co
Buttons(title    =    'Reg
backgroundColor = Color.B
Buttons(title = "Login",
Color.Magenta)
}
}
}
}
```

```
package
com.project.pradyotpraka
sh.flashchat.view
                                import
                                androidx.compose.foundation.lay
                                out.fillMaxHeight
                                import
                                androidx.compose.foundation.lay
                                out.fillMaxWidth
                                import
                                androidx.compose.foundation.lay
                                out.padding
                                import
                                androidx.compose.foundation.sha
                                pe.RoundedCornerShape
                                import
                                androidx.compose.foundation.tex
                                t.KeyboardOptions
                                import
                                androidx.compose.material.*
                                import
                                androidx.compose.material.icons
                                .Icons
                                import
                                androidx.compose.material.icons
                                .filled.ArrowBack
                                import
                                androidx.compose.runtime.Compos
                                able
                                import
                                androidx.compose.ui.Modifier
                                import
                                androidx.compose.ui.graphics.Co
                                lor
                                import
                                androidx.compose.ui.text.font.F
                                ontWeight
                                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.flas
hchat.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
```

```
                )
            }
        }
```

```
package
com.project.pradyotprakas
h.flashchat.view.home

import
androidx.compose.foundation.ba
ckground
import
androidx.compose.foundation.la
yout.*
import
androidx.compose.foundation.la
zy.LazyColumn
import
androidx.compose.foundation.la
zy.items
import
androidx.compose.foundation.te
xt.KeyboardOptions
import
androidx.compose.material.*
import
androidx.compose.material.icon
s.Icons
import
androidx.compose.material.icon
s.filled.Send
import
androidx.compose.runtime.Compo
sable
import
androidx.compose.runtime.getVa
lue
import
androidx.compose.runtime.lived
ata.observeAsState
import
```

```
androidx.compose.ui.Alignment
import
androidx.compose.ui.Modifier
import
androidx.compose.ui.graphics.C
olor
import
androidx.compose.ui.text.input
.KeyboardType
import
androidx.compose.ui.unit.dp
import
androidx.lifecycle.viewmodel.c
ompose.viewModel
import
com.project.pradyotprakash.fla
shchat.Constants
import
com.project.pradyotprakash.fla
shchat.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.observeA
sState(initial = "")
val messages: List<Map<String,
Any>> by
homeViewModel.messages.observe
AsState(
```

```
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_U
SER] as Boolean
SingleMessage(
message                         =
message[Constants.MESSAGE].toS
tring(),
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"
                )
                }
                }
                )
                }
                }
```

```
package
com.project.pradyotprak
ash.flashchat.view.home
                        import android.util.Log
```

```
import
androidx.lifecycle.LiveData
import
androidx.lifecycle.MutableLiveDat
a
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.flashc
hat.Constants
import
java.lang.IllegalArgumentExceptio
n
/**
* 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<Str
ing, 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(Con
stants.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(Con
stants.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.to
                String()                        ==
                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()
                }
                }
```

```
package
com.project.pradyotprakas
h.flashchat.view.login
                    import
                    androidx.compose.foundation.lay
                    out.*
                    import
                    androidx.compose.material.Circu
                    larProgressIndicator
                    import
                    androidx.compose.runtime.Compos
                    able
```

```
import
androidx.compose.runtime.getVal
ue
import
androidx.compose.runtime.liveda
ta.observeAsState
import
androidx.compose.ui.Alignment
import
androidx.compose.ui.Modifier
import
androidx.compose.ui.graphics.Co
lor
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.co
mpose.viewModel
import
com.project.pradyotprakash.flas
hchat.view.Appbar
import
com.project.pradyotprakash.flas
hchat.view.Buttons
import
com.project.pradyotprakash.flas
hchat.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.observeAsS
tate("")
val   password:   String   by
loginViewModel.password.observe
AsState("")
val   loading:   Boolean   by
loginViewModel.loading.observeA
sState(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(i
                t) },
                label = "Password",
                keyboardType                    =
                KeyboardType.Password,
                visualTransformation            =
                PasswordVisualTransformation()
                )
                Spacer(modifier                 =
                Modifier.height(20.dp))
                Buttons(
                title = "Login",
                onClick             =           {
                loginViewModel.loginUser(home  =
                home) },
                backgroundColor = Color.Magenta
                )
                }
                }
                }
```

```
package
com.project.pradyotprakash.flas
hchat.view.login
                        import
                        androidx.lifecycle.LiveDa
                        ta
                        import
                        androidx.lifecycle.Mutabl
                        eLiveData
                        import
```

```
androidx.lifecycle.ViewMo
del
import
com.google.firebase.auth.
FirebaseAuth
import
com.google.firebase.auth.
ktx.auth
import
com.google.firebase.ktx.F
irebase
import
java.lang.IllegalArgument
Exception
/**
* 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(newPasswor
d: 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.signInWithEmailAndPa
ssword(email, password)
.addOnCompleteListener {
if (it.isSuccessful) {
home()
}
_loading.value = false
}
}
}
}
```

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

                        import
                        androidx.compose.foundation.la
                        yout.*
                        import
```

```
androidx.compose.material.Circ
ularProgressIndicator
import
androidx.compose.runtime.Compo
sable
import
androidx.compose.runtime.getVa
lue
import
androidx.compose.runtime.lived
ata.observeAsState
import
androidx.compose.ui.Alignment
import
androidx.compose.ui.Modifier
import
androidx.compose.ui.graphics.C
olor
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.c
ompose.viewModel
import
com.project.pradyotprakash.fla
shchat.view.Appbar
import
com.project.pradyotprakash.fla
shchat.view.Buttons
import
com.project.pradyotprakash.fla
shchat.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.observ
eAsState("")
val password: String by
registerViewModel.password.obs
erveAsState("")
val loading: Boolean by
registerViewModel.loading.obse
rveAsState(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.updatePasswo
                    rd(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
                    )
                    }
                    }
                    }
```

```
package
com.project.pradyotprakash.fla
```

shchat.view.register

```
import
androidx.lifecycle.LiveDa
ta
import
androidx.lifecycle.Mutabl
eLiveData
import
androidx.lifecycle.ViewMo
del
import
com.google.firebase.auth.
FirebaseAuth
import
com.google.firebase.auth.
ktx.auth
import
com.google.firebase.ktx.F
irebase
import
java.lang.IllegalArgument
Exception
/**
* 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
}
}
}
}
```