CONTENTS **PAGE NO**

INTRODUCTION

Whether you're catching up with old friends or collaborating with colleagues, our app is perfect for any occasion. With our real-time messaging, you can stay connected with loved ones even when you're on the go.Our app is designed with your privacy in mind. We use the latest encryption technology to secure your messages, so you can chat with peace of mind. We also offer several customization options, such as personalizing your chat bubble color or setting custom notification tones.

1.1 Overview:

- Our real-time chat app allows users to connect with friends, family, and colleagues instantly from anywhere in the world.
- The app offers easy-to-use features, such as messaging, sharing photos and videos, and voice and video calls.
- It is perfect for catching up with old friends or collaborating with colleagues.
- Users have the option to personalize their chat experience and the app is designed with privacy and security in mind. The goal is to provide a seamless communication experience that is both easy to use and reliable.

1.2 Purpose:

- The purpose of our chat app is to provide an efficient and user-friendly communication platform for individuals and groups.
- It enables users to connect with each other in real-time, regardless of location or device.
- The app is intended to enhance communication and collaboration among friends, family, and colleagues, by providing features like messaging, voice and video calls, and media sharing.

- Our chat app aims to make communication more convenient, inclusive, and accessible, while ensuring the privacy and security of users.
- Offering a platform for publishers to distribute their content and reach a wider audience.

PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map:

Template

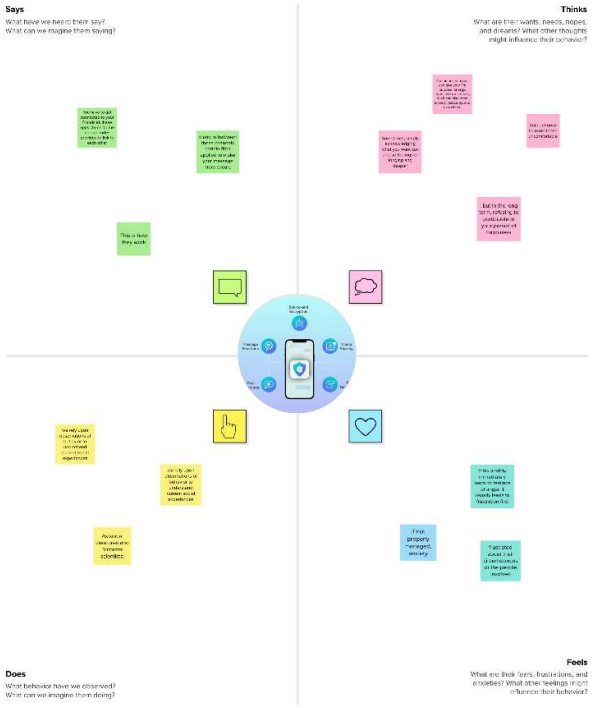
Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share template feedback

Build empathy

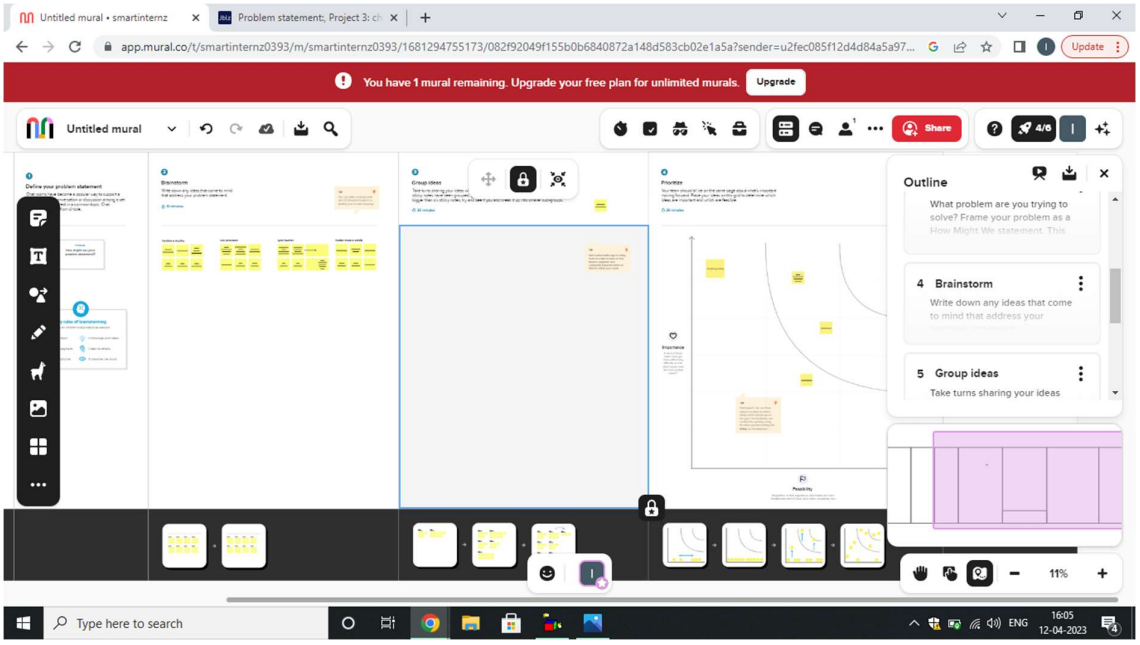
The information you add here should be representative of the observations and research you've done about your users.



Need some inspiration?

See a finished version of this template to kickstart your work.

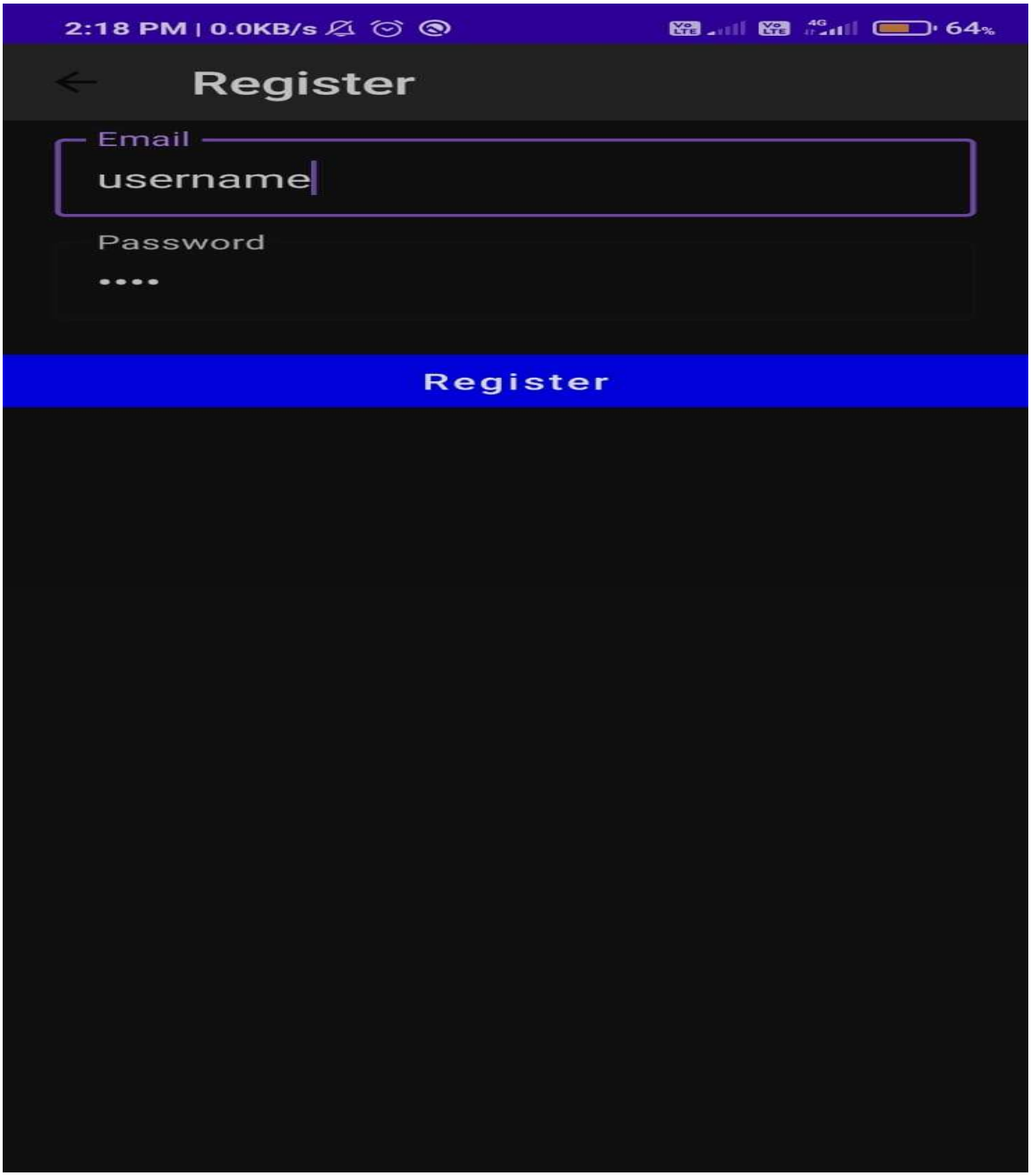
Open example →



2.2 Ideation & Brainstorming Map:

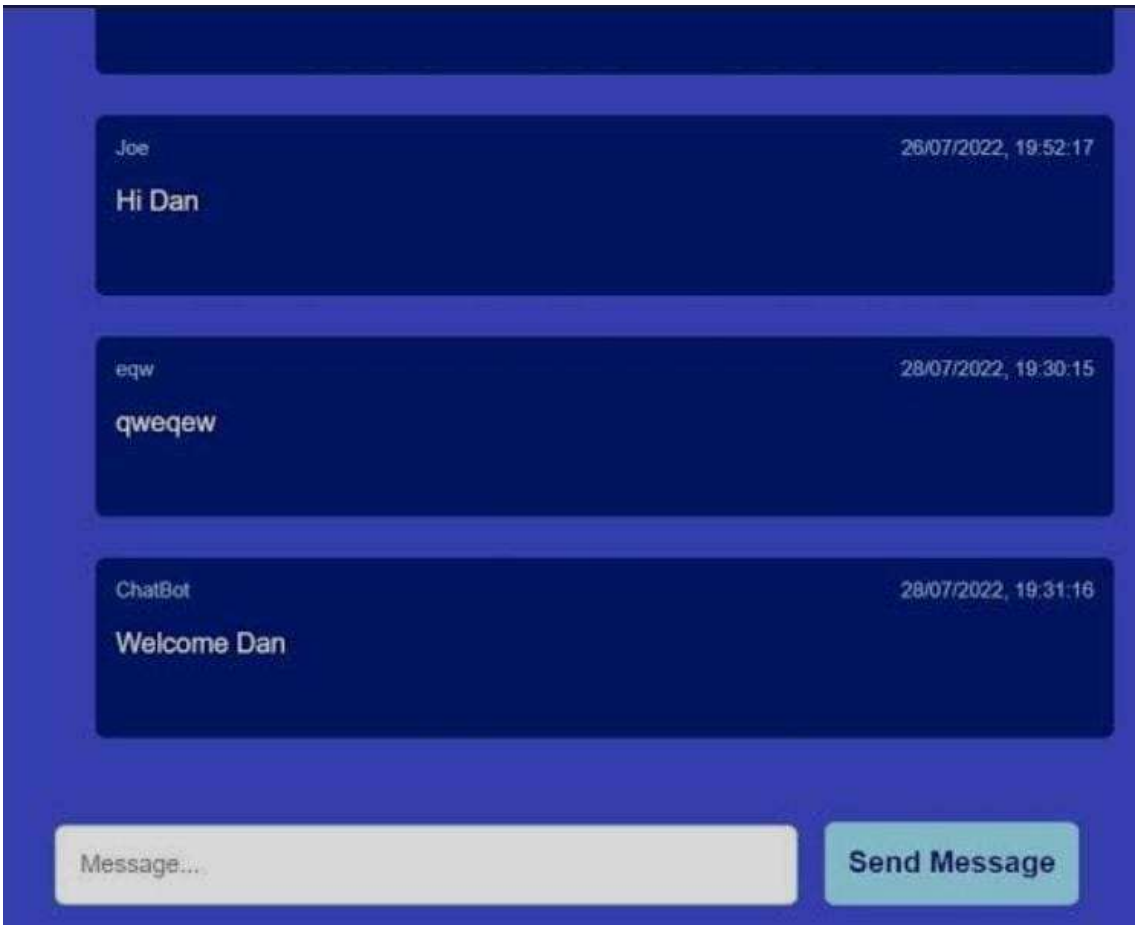
RESULT

Sign in Page:



Sign up page:

Home Page:


Joe
26/07/2022, 19:52:17
Hi Dan
eqw
28/07/2022, 19:30:15
qweqew
ChatBot
28/07/2022, 19:31:16
Welcome Dan
Message...
Send Message

ADVANTAGES & DISADVANTAGES

1.1 Advantages:

- Real-time communication: A chat app offers instant messaging and communication, allowing users to connect with each other immediately.
- Convenience: Chat apps can be accessed anytime, anywhere, making communication more convenient and accessible.
- Increased collaboration: Chat apps enable users to cooperate on projects, share ideas, and work together in real-time, which ultimately leads to increased productivity.
- Enhanced privacy and security: Chat apps typically use end-to-end encryption, making them more secure than traditional forms of communication methods.

1.2 Disadvantages:

- Distractions: With constant notifications and alerts, chat apps may cause distractions and hinder productivity.
- Miscommunication: The lack of face-to-face communication can lead to misunderstandings and misinterpretations of messages.
- Dependency on technology: As with any technology, chat apps may fail or experience technical difficulties, causing disruptions in communication.
- Cyberbullying and harassment: Chat apps can be used to bully or harass others online, which can be detrimental to individuals' mental health.

APPLICATIONS

- Business communication: Many businesses use chat apps for internal communication among employees and teams. This allows for quick and easy communication, regardless of location.
- Social networking: Chat apps can be used for social networking, allowing individuals to create groups and connect with friends and family.
- Customer service: Chat apps can be utilized for customer service, allowing customers to ask questions and receive real-time assistance from businesses.
- Business: A news app can be used in a business setting to provide employees with access to news articles and updates related to their industry or company.
- Personal communication: Chat apps can be used for personal communication between individuals, allowing for easy messaging and sharing of media**.**
- Education and learning: Chat apps can be used for remote learning and education, allowing teachers and students to communicate and collaborate on assignments and projects.

Conclusion

In conclusion, chat apps have a range of applications in various areas such as business communication, social networking, customer service, education and learning, personal communication, and mental health support. Chat apps are beneficial for easy and quick communication, allowing individuals to connect and interact with others regardless of their location. Chat apps have become an integral part of our daily lives, and their continued development and innovation are expected to bring more benefits to their users.

Future Scopes

The future of chat apps is expected to be even more exciting with the integration of emerging technologies such as AI, voice and video calling, virtual and augmented reality, and blockchain. These technologies are set to enhance the user experience and make communication even more seamless, secure, and interactive. Chat apps are also set to become more personalized to meet the unique needs of each user. Furthermore, businesses are expected to increasingly use chat apps for customer engagement, sales, and marketing. Hence, the future of chat apps is bright, and we can expect more innovative and exciting features in the years to come.

APPENDIX

Sources code :

https://github.com/abiabi00

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()
        }
    }
}
```

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() }
}
```

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"
                    )
                }
            }
        )
    }
}
```

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
            )
        }
    }
}
```

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
            )
        }
    }
}
```