**CONTENTS**

1. **INTRODUCTION** 
   1. Overview
   2. Purpose
2. **PROBLEM DEFINITION & DESIGN THINKING**
   1. Empathy Map
   2. Ideation & Brainstorming Map
3. **RESULT**
4. **ADVANTAGES & DISADVANTAGES**
5. **APPLICATIONS**
6. **CONCLUSION**
7. **FUTURE SCOPE**
8. **APPENDIX**
   1. Source Code

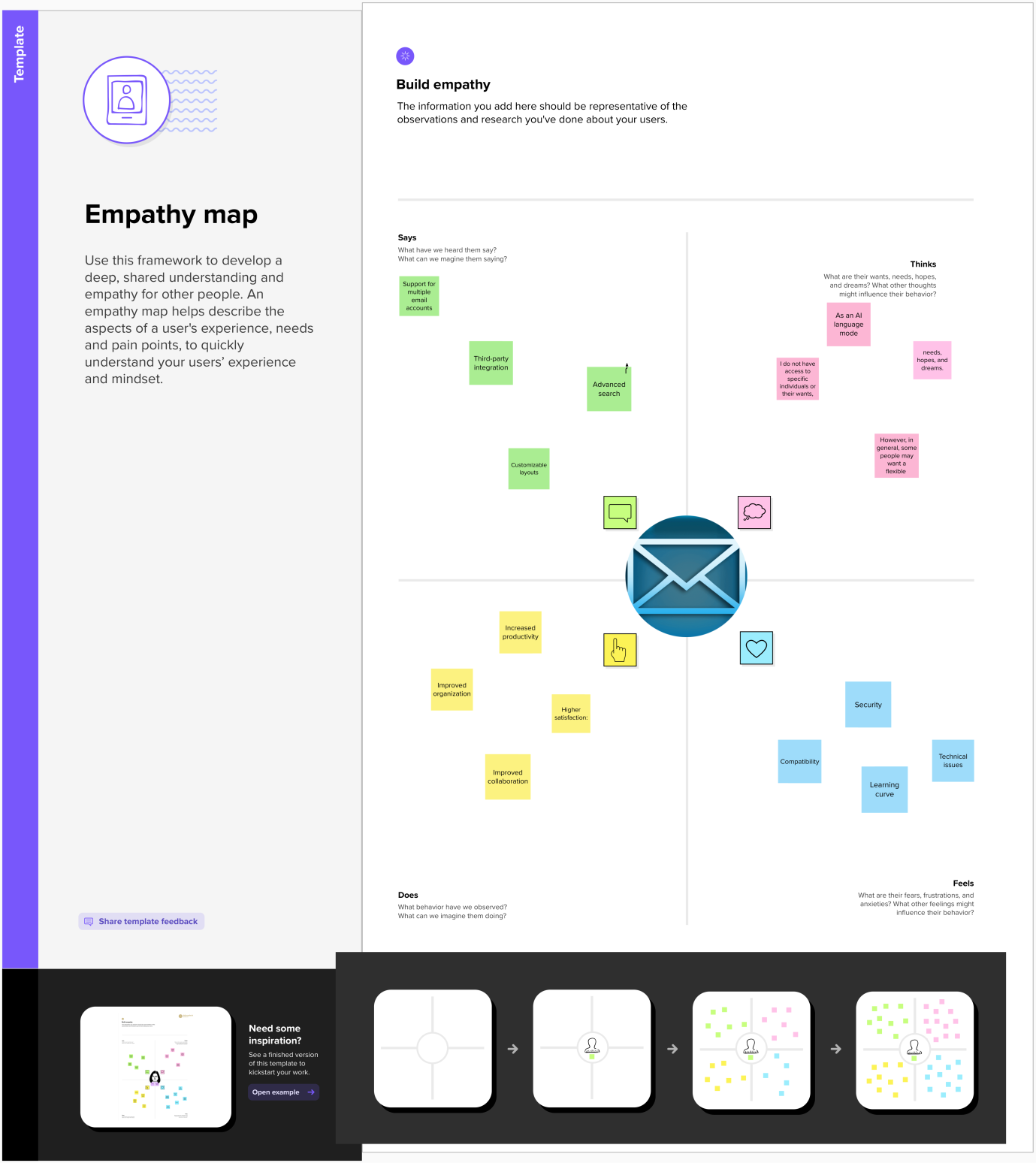
**INTRODUCTION**

The main objective of flexible email client apps, where you can customize and streamline your email experience like never before! Our email client app is designed with flexibility in mind, allowing you to tailor the app to meet your unique email needs.

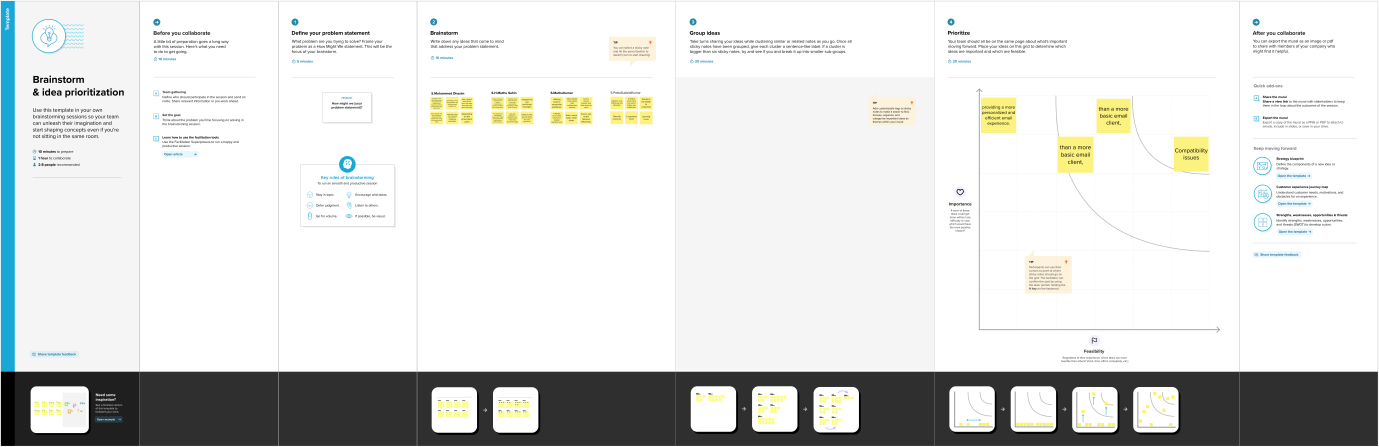
* 1. **Overview:**
* Multiple account support: The app should allow users to add and manage multiple email accounts from different providers, such as Gmail, Yahoo, or Outlook.
* Customizable interface: Users should be able to customize the interface of the app to suit their preferences, including the ability to change the colour scheme, layout, and font size..
* Advanced search options: The app should provide advanced search options that allow users to find specific emails quickly and easily.
* Email filtering and sorting: Users should be able to filter and sort their emails based on different criteria, such as sender, subject, date, and attachments.
  1. **Purpose:**
* Increased productivity: A flexible email client app can help users manage their emails more efficiently, allowing them to spend less time on email and more time on other important tasks.
* Customizability: A flexible email client app can be customized to suit the individual needs and preferences of users. This can include features such as customizable themes, font sizes, and email sorting options.
* Integration: A flexible email client app can integrate with other tools and apps, such as calendars, task managers, and project management software. This can help users streamline their workflows and stay organized.

**PROBLEM DEFINITION & DESIGN THINKING**

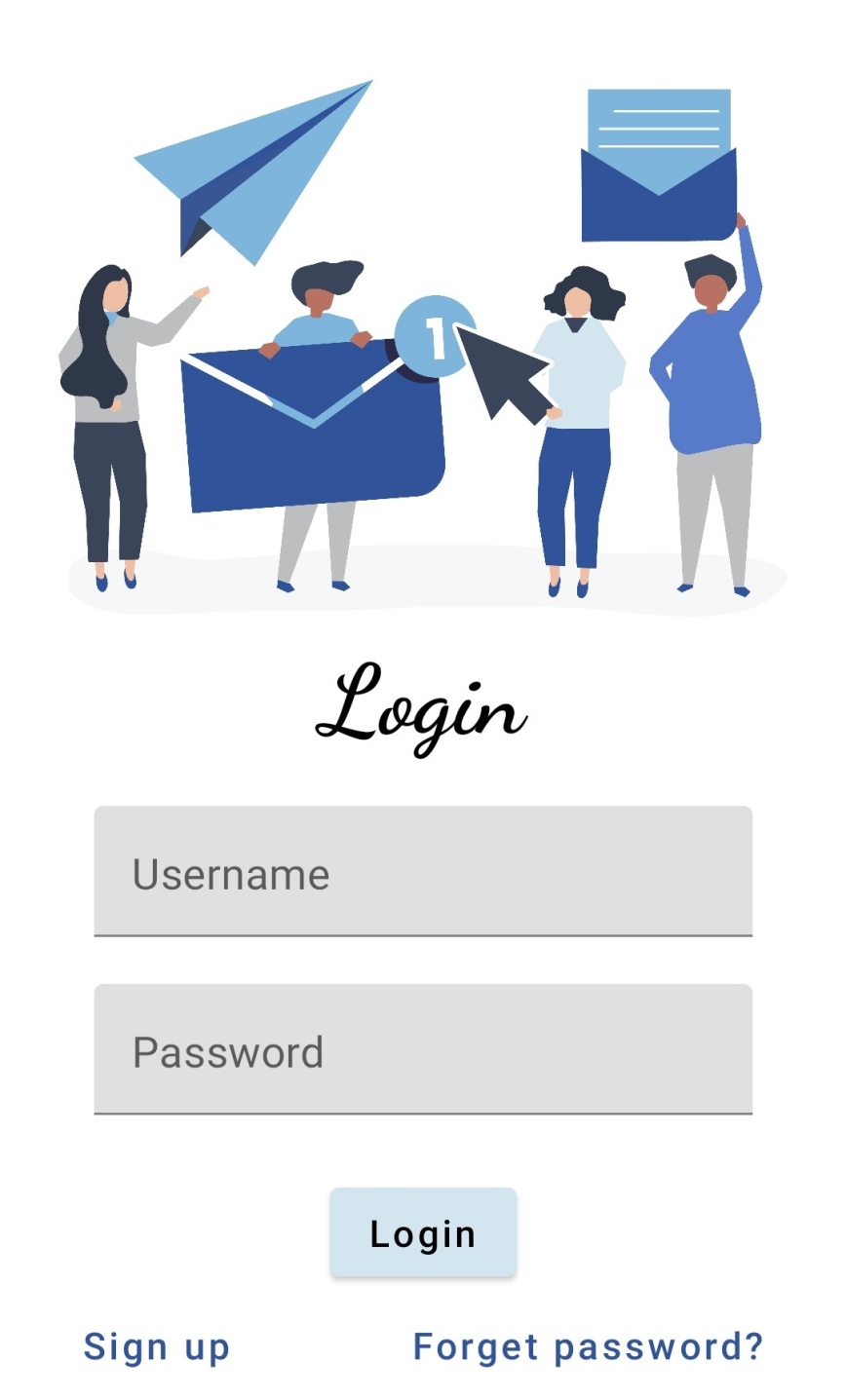
**2.1 Empathy Map:**

****

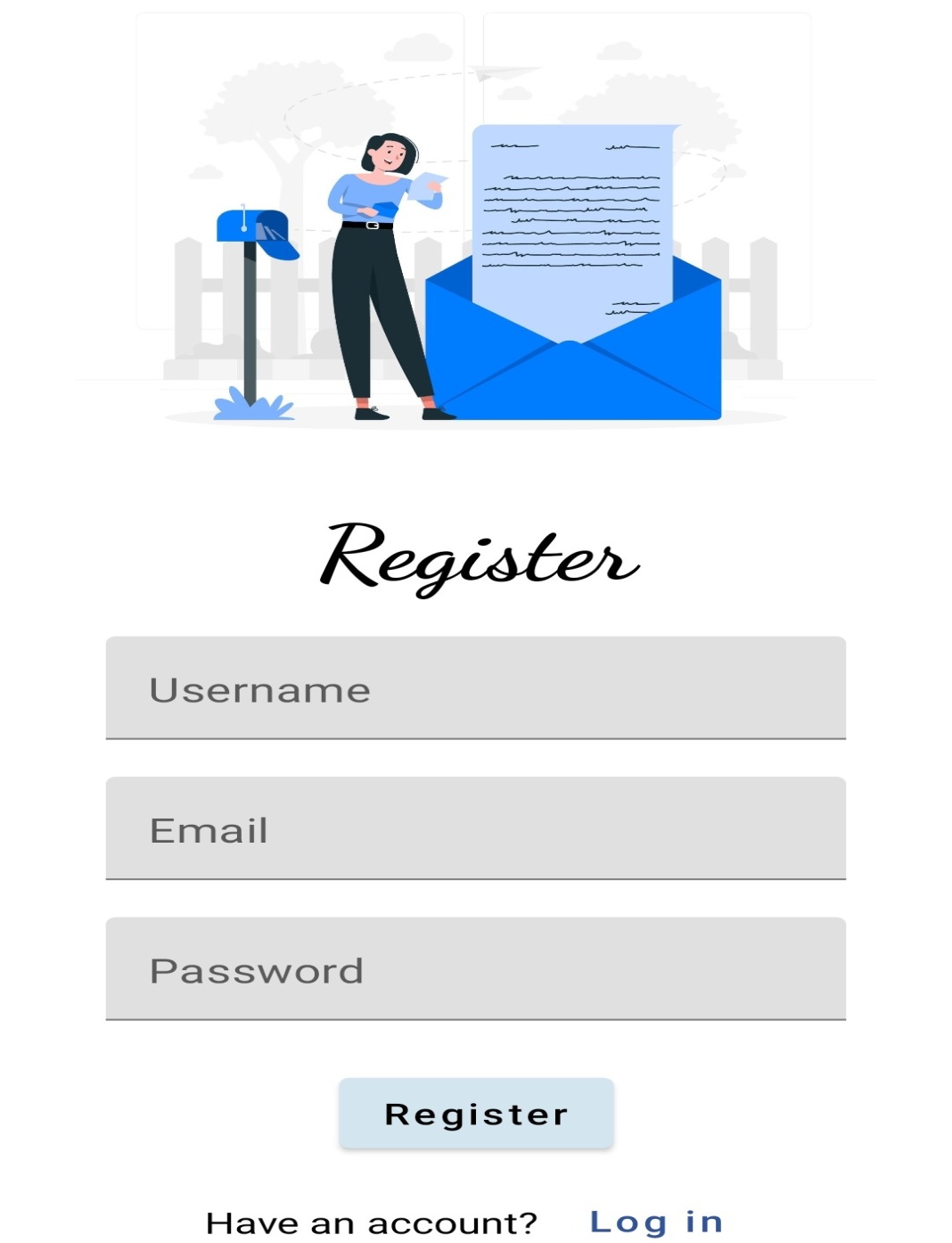
* 1. **Ideation & Brainstorming Map:**

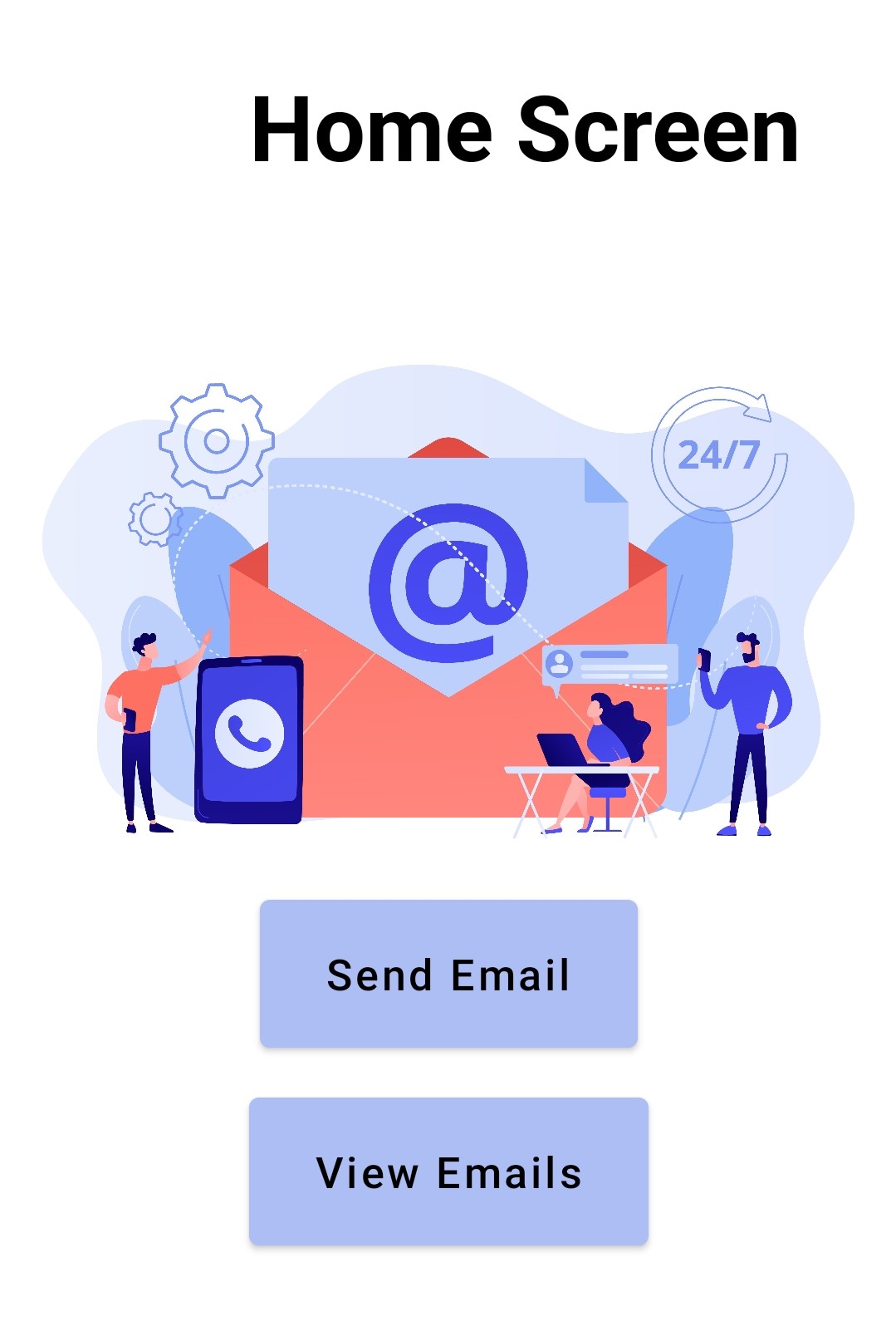


**log in Page:**

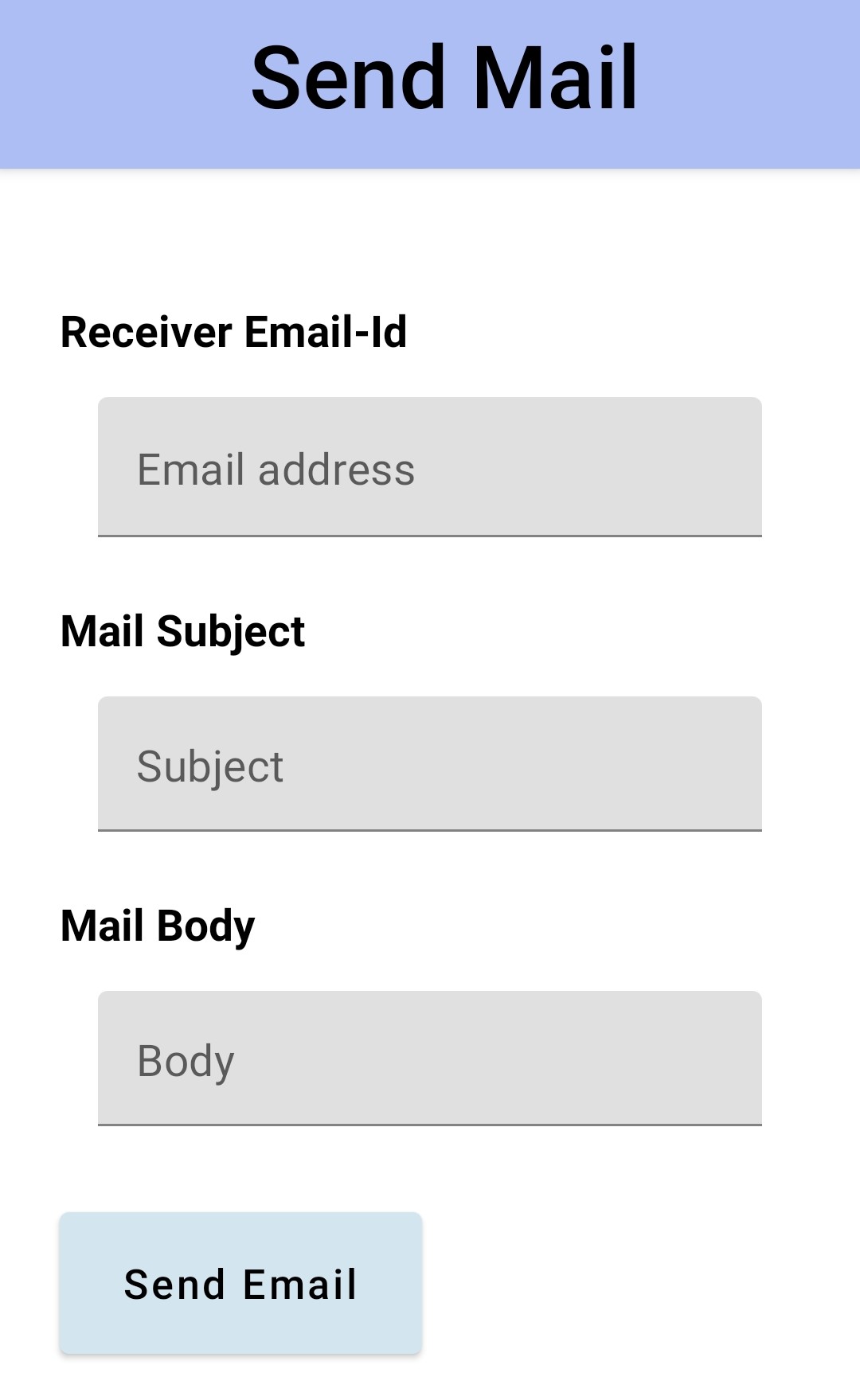
**RESULT**

**Register page**

****

**Home Page: **

**Send mail page**



**ADVANTAGES & DISADVANTAGES**

* 1. **Advantages:**
* Integration: A flexible email client app can integrate with other tools and apps, such as calendars, task managers, and project management software. This can help users streamline their workflows and stay organized..
* Productivity: A flexible email client app can help users manage their emails more efficiently, allowing them to spend less time on email and more time on other important tasks. This can include features such as automated replies, email scheduling, and snooze options.
* Accessibility: A flexible email client app can be designed to be accessible to all users, including those with disabilities. This can include features such as text-to-speech and keyboard shortcuts.
  1. **Disadvantages:**
* Compatibility issues: A flexible email client app may not be compatible with all email providers or email formats. This can limit its usefulness for some users and require them to switch to a different app.
* Cost: Some flexible email client apps may require users to pay for certain customization options or features, which can add up over time and make the app less accessible for some users.
* Overall, while a flexible email client app can offer many benefits, it is important for users to carefully consider their needs and the potential drawbacks before committing to using one.

**APPLICATIONS**

* Apple Mail: The default email client app on Apple devices, which offers advanced filtering options, the ability to customize swipe gestures, and integration with other Apple tools and services such as iCloud.
* Microsoft Outlook: A popular email client app that integrates with other Microsoft tools and services, such as Office 365, OneDrive, and Skype. It offers advanced filtering options, calendar integration, and the ability to create custom rules for managing emails.
* Gmail: Google's email client app, which offers advanced filtering options, labels, and integration with other Google tools and services. It also offers a powerful search function and the ability to snooze emails for later.
* Thunderbird: A free and open-source email client app that offers a high level of customization options, including themes, add-ons, and the ability to customize keyboard shortcuts. It also supports multiple email accounts and protocols.
* **Spike: A modern email client app that combines email, messaging, and collaboration features into a single platform. It offers advanced search options, the ability to create groups and share files, and an AI-powered smart inbox.**

**Conclusion**

We conclude that the flexible email client app is an essential tool for anyone who needs to manage their email accounts on the go. By providing users with a range of features and options that can be customized to meet their specific needs and preferences, a flexible email client app can help users stay organized, save time, and communicate more effectively with their contacts. Whether you're a busy professional, a student, or anyone else who relies on email for communication, a flexible email client app can be an invaluable asset.

**Future Scopes**

In future adding As the world becomes more conscious of environmental issues, email client apps will need to become more sustainable in their design and operation. This could include features such as energy-efficient modes, reduced data usage, and the ability to integrate with eco-friendly email hosting services. Overall, the future of flexible email client applications will be shaped by a combination of new technologies, changing user needs and preferences, and broader societal trends.

**APPENDIX**

**Source Code :** <https://github.com/MuthuSahin/Emailapp>

**Code : -**

1.LoginActivity.kt

**package com.example.emailapplication**

**import android.content.Context**

**import android.content.Intent**

**import android.os.Bundle**

**import androidx.activity.ComponentActivity**

**import androidx.activity.compose.setContent**

**import androidx.compose.foundation.Image**

**import androidx.compose.foundation.background**

**import androidx.compose.foundation.layout.\***

**import androidx.compose.material.\***

**import androidx.compose.runtime.\***

**import androidx.compose.ui.Alignment**

**import androidx.compose.ui.Modifier**

**import androidx.compose.ui.graphics.Color**

**import androidx.compose.ui.layout.ContentScale**

**import androidx.compose.ui.res.painterResource**

**import androidx.compose.ui.text.font.FontFamily**

**import androidx.compose.ui.text.font.FontWeight**

**import androidx.compose.ui.text.input.PasswordVisualTransformation**

**import androidx.compose.ui.tooling.preview.Preview**

**import androidx.compose.ui.unit.dp**

**import androidx.compose.ui.unit.sp**

**import androidx.core.content.ContextCompat**

**import com.example.emailapplication.ui.theme.EmailApplicationTheme**

**class LoginActivity : ComponentActivity() {**

**private lateinit var databaseHelper: UserDatabaseHelper**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**databaseHelper = UserDatabaseHelper(this)**

**setContent {**

**LoginScreen(this, databaseHelper)**

**}**

**}**

**}**

**@Composable**

**fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {**

**var username by remember { mutableStateOf("") }**

**var password by remember { mutableStateOf("") }**

**var error by remember { mutableStateOf("") }**

**Column(**

**modifier = Modifier.fillMaxSize().background(Color.White),**

**horizontalAlignment = Alignment.CenterHorizontally,**

**verticalArrangement = Arrangement.Center**

**) {**

**Image(**

**painterResource(id = R.drawable.email\_login), contentDescription = ""**

**)**

**Text(**

**fontSize = 36.sp,**

**fontWeight = FontWeight.ExtraBold,**

**fontFamily = FontFamily.Cursive,**

**text = "Login"**

**)**

**Spacer(modifier = Modifier.height(10.dp))**

**TextField(**

**value = username,**

**onValueChange = { username = it },**

**label = { Text("Username") },**

**modifier = Modifier.padding(10.dp)**

**.width(280.dp)**

**)**

**TextField(**

**value = password,**

**onValueChange = { password = it },**

**label = { Text("Password") },**

**visualTransformation = PasswordVisualTransformation(),**

**modifier = Modifier.padding(10.dp)**

**.width(280.dp)**

**)**

**if (error.isNotEmpty()) {**

**Text(**

**text = error,**

**color = MaterialTheme.colors.error,**

**modifier = Modifier.padding(vertical = 16.dp)**

**)**

**}**

**Button(**

**onClick = {**

**if (username.isNotEmpty() && password.isNotEmpty()) {**

**val user = databaseHelper.getUserByUsername(username)**

**if (user != null && user.password == password) {**

**error = "Successfully log in"**

**context.startActivity(**

**Intent(**

**context,**

**MainActivity::class.java**

**)**

**)**

**//onLoginSuccess()**

**}**

**} else {**

**error = "Please fill all fields"**

**}**

**},**

**colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)),**

**modifier = Modifier.padding(top = 16.dp)**

**) {**

**Text(text = "Login")**

**}**

**Row {**

**TextButton(onClick = {context.startActivity(**

**Intent(**

**context,**

**RegisterActivity::class.java**

**)**

**)}**

**)**

**{ Text(color = Color(0xFF31539a),text = "Sign up") }**

**TextButton(onClick = {**

**})**

**{**

**Spacer(modifier = Modifier.width(60.dp))**

**Text(color = Color(0xFF31539a),text = "Forget password?")**

**}**

**}**

**}**

**}**

**private fun startMainPage(context: Context) {**

**val intent = Intent(context, MainActivity::class.java)**

**ContextCompat.startActivity(context, intent, null)**

**}**

2.MainActivity.kt

**package com.example.emailapplication**

**import android.content.Context**

**import android.content.Intent**

**import android.os.Bundle**

**import androidx.activity.ComponentActivity**

**import androidx.activity.compose.setContent**

**import androidx.compose.foundation.Image**

**import androidx.compose.foundation.background**

**import androidx.compose.foundation.layout.\***

**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 androidx.compose.ui.layout.ContentScale**

**import androidx.compose.ui.res.painterResource**

**import androidx.compose.ui.text.font.FontWeight**

**import androidx.compose.ui.tooling.preview.Preview**

**import androidx.compose.ui.unit.dp**

**import androidx.compose.ui.unit.sp**

**import androidx.core.content.ContextCompat**

**import androidx.core.content.ContextCompat.startActivity**

**import com.example.emailapplication.ui.theme.EmailApplicationTheme**

**class MainActivity : ComponentActivity() {**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**setContent {**

**// A surface container using the 'background' color from the theme**

**Surface(**

**modifier = Modifier.fillMaxSize().background(Color.White),**

**) {**

**Email(this)**

**}**

**}**

**}**

**}**

**@Composable**

**fun Email(context: Context) {**

**Text(**

**text = "Home Screen",**

**modifier = Modifier.padding(top = 74.dp, start = 100.dp, bottom = 24.dp),**

**color = Color.Black,**

**fontWeight = FontWeight.Bold,**

**fontSize = 32.sp**

**)**

**Column(**

**horizontalAlignment = Alignment.CenterHorizontally,**

**verticalArrangement = Arrangement.Center**

**) {**

**Image(**

**painterResource(id = R.drawable.home\_screen), contentDescription = ""**

**)**

**Button(onClick = {**

**context.startActivity(**

**Intent(**

**context,**

**SendMailActivity::class.java**

**)**

**)**

**},**

**colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))**

**) {**

**Text(**

**text = "Send Email",**

**modifier = Modifier.padding(10.dp),**

**color = Color.Black,**

**fontSize = 15.sp**

**)**

**}**

**Spacer(modifier = Modifier.height(20.dp))**

**Button(onClick = {**

**context.startActivity(**

**Intent(**

**context,**

**ViewMailActivity::class.java**

**)**

**)**

**},**

**colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))**

**) {**

**Text(**

**text = "View Emails",**

**modifier = Modifier.padding(10.dp),**

**color = Color.Black,**

**fontSize = 15.sp**

**)**

**}**

**}**

**}**

3.RegisterActivity.kt

**package com.example.emailapplication**

**import android.content.Context**

**import android.content.Intent**

**import android.os.Bundle**

**import androidx.activity.ComponentActivity**

**import androidx.activity.compose.setContent**

**import androidx.compose.foundation.Image**

**import androidx.compose.foundation.background**

**import androidx.compose.foundation.layout.\***

**import androidx.compose.material.\***

**import androidx.compose.runtime.\***

**import androidx.compose.ui.Alignment**

**import androidx.compose.ui.Modifier**

**import androidx.compose.ui.graphics.Color**

**import androidx.compose.ui.layout.ContentScale**

**import androidx.compose.ui.res.painterResource**

**import androidx.compose.ui.text.font.FontFamily**

**import androidx.compose.ui.text.font.FontWeight**

**import androidx.compose.ui.text.input.PasswordVisualTransformation**

**import androidx.compose.ui.tooling.preview.Preview**

**import androidx.compose.ui.unit.dp**

**import androidx.compose.ui.unit.sp**

**import androidx.core.content.ContextCompat**

**import com.example.emailapplication.ui.theme.EmailApplicationTheme**

**class RegisterActivity : ComponentActivity() {**

**private lateinit var databaseHelper: UserDatabaseHelper**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**databaseHelper = UserDatabaseHelper(this)**

**setContent {**

**RegistrationScreen(this, databaseHelper)**

**}**

**}**

**}**

**@Composable**

**fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {**

**var username by remember { mutableStateOf("") }**

**var password by remember { mutableStateOf("") }**

**var email by remember { mutableStateOf("") }**

**var error by remember { mutableStateOf("") }**

**Column(**

**modifier = Modifier.fillMaxSize().background(Color.White),**

**horizontalAlignment = Alignment.CenterHorizontally,**

**verticalArrangement = Arrangement.Center**

**) {**

**Image(**

**painterResource(id = R.drawable.email\_signup), contentDescription = "",**

**modifier = Modifier.height(300.dp)**

**)**

**Text(**

**fontSize = 36.sp,**

**fontWeight = FontWeight.ExtraBold,**

**fontFamily = FontFamily.Cursive,**

**text = "Register"**

**)**

**Spacer(modifier = Modifier.height(10.dp))**

**TextField(**

**value = username,**

**onValueChange = { username = it },**

**label = { Text("Username") },**

**modifier = Modifier**

**.padding(10.dp)**

**.width(280.dp)**

**)**

**TextField(**

**value = email,**

**onValueChange = { email = it },**

**label = { Text("Email") },**

**modifier = Modifier**

**.padding(10.dp)**

**.width(280.dp)**

**)**

**TextField(**

**value = password,**

**onValueChange = { password = it },**

**label = { Text("Password") },**

**visualTransformation = PasswordVisualTransformation(),**

**modifier = Modifier**

**.padding(10.dp)**

**.width(280.dp)**

**)**

**if (error.isNotEmpty()) {**

**Text(**

**text = error,**

**color = MaterialTheme.colors.error,**

**modifier = Modifier.padding(vertical = 16.dp)**

**)**

**}**

**Button(**

**onClick = {**

**if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {**

**val user = User(**

**id = null,**

**firstName = username,**

**lastName = null,**

**email = email,**

**password = password**

**)**

**databaseHelper.insertUser(user)**

**error = "User registered successfully"**

**// Start LoginActivity using the current context**

**context.startActivity(**

**Intent(**

**context,**

**LoginActivity::class.java**

**)**

**)**

**} else {**

**error = "Please fill all fields"**

**}**

**},**

**colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)),**

**modifier = Modifier.padding(top = 16.dp)**

**) {**

**Text(text = "Register")**

**}**

**Spacer(modifier = Modifier.width(10.dp))**

**Spacer(modifier = Modifier.height(10.dp))**

**Row() {**

**Text(**

**modifier = Modifier.padding(top = 14.dp), text = "Have an account?"**

**)**

**TextButton(onClick = {**

**context.startActivity(**

**Intent(**

**context,**

**LoginActivity::class.java**

**)**

**)**

**})**

**{**

**Spacer(modifier = Modifier.width(10.dp))**

**Text(color = Color(0xFF31539a),text = "Log in")**

**}**

**}**

**}**

**}**

**private fun startLoginActivity(context: Context) {**

**val intent = Intent(context, LoginActivity::class.java)**

**ContextCompat.startActivity(context, intent, null)**

**}**

4.SendMainActivity.kt

**package com.example.emailapplication**

**import android.annotation.SuppressLint**

**import android.content.Context**

**import android.content.Intent**

**import android.os.Bundle**

**import androidx.activity.ComponentActivity**

**import androidx.activity.compose.setContent**

**import androidx.compose.foundation.layout.\***

**import androidx.compose.material.\***

**import androidx.compose.runtime.\***

**import androidx.compose.ui.Alignment**

**import androidx.compose.ui.Modifier**

**import androidx.compose.ui.graphics.Color**

**import androidx.compose.ui.platform.LocalContext**

**import androidx.compose.ui.text.TextStyle**

**import androidx.compose.ui.text.font.FontWeight**

**import androidx.compose.ui.text.style.TextAlign**

**import androidx.compose.ui.tooling.preview.Preview**

**import androidx.compose.ui.unit.dp**

**import androidx.compose.ui.unit.sp**

**import com.example.emailapplication.ui.theme.EmailApplicationTheme**

**class SendMailActivity : ComponentActivity() {**

**private lateinit var databaseHelper: EmailDatabaseHelper**

**@SuppressLint("UnusedMaterialScaffoldPaddingParameter")**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**databaseHelper = EmailDatabaseHelper(this)**

**setContent {**

**Scaffold(**

**// in scaffold we are specifying top bar.**

**topBar = {**

**// inside top bar we are specifying**

**// background color.**

**TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp),**

**// along with that we are specifying**

**// title for our top bar.**

**title = {**

**// in the top bar we are specifying**

**// title as a text**

**Text(**

**// on below line we are specifying**

**// text to display in top app bar.**

**text = "Send Mail",**

**fontSize = 32.sp,**

**color = Color.Black,**

**// on below line we are specifying**

**// modifier to fill max width.**

**modifier = Modifier.fillMaxWidth(),**

**// on below line we are**

**// specifying text alignment.**

**textAlign = TextAlign.Center,**

**)**

**}**

**)**

**}**

**) {**

**// on below line we are**

**// calling method to display UI.**

**openEmailer(this,databaseHelper)**

**}**

**}**

**}**

**}**

**@Composable**

**fun openEmailer(context: Context, databaseHelper: EmailDatabaseHelper) {**

**// in the below line, we are**

**// creating variables for URL**

**var recevierMail by remember {mutableStateOf("") }**

**var subject by remember {mutableStateOf("") }**

**var body by remember {mutableStateOf("") }**

**var error by remember { mutableStateOf("") }**

**// on below line we are creating**

**// a variable for a context**

**val ctx = LocalContext.current**

**// on below line we are creating a column**

**Column(**

**// on below line we are specifying modifier**

**// and setting max height and max width**

**// for our column**

**modifier = Modifier**

**.fillMaxSize()**

**.padding(top = 55.dp, bottom = 25.dp, start = 25.dp, end = 25.dp),**

**horizontalAlignment = Alignment.Start**

**) {**

**// on the below line, we are**

**// creating a text field.**

**Text(text = "Receiver Email-Id",**

**fontWeight = FontWeight.Bold,**

**fontSize = 16.sp)**

**TextField(**

**// on below line we are specifying**

**// value for our text field.**

**value = recevierMail,**

**// on below line we are adding on value**

**// change for text field.**

**onValueChange = { recevierMail = it },**

**// on below line we are adding place holder as text**

**label = { Text(text = "Email address") },**

**placeholder = { Text(text = "abc@gmail.com") },**

**// on below line we are adding modifier to it**

**// and adding padding to it and filling max width**

**modifier = Modifier**

**.padding(16.dp)**

**.fillMaxWidth(),**

**// on below line we are adding text style**

**// specifying color and font size to it.**

**textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),**

**// on below line we are**

**// adding single line to it.**

**singleLine = true,**

**)**

**// on below line adding a spacer.**

**Spacer(modifier = Modifier.height(10.dp))**

**Text(text = "Mail Subject",**

**fontWeight = FontWeight.Bold,**

**fontSize = 16.sp)**

**// on the below line, we are creating a text field.**

**TextField(**

**// on below line we are specifying**

**// value for our text field.**

**value = subject,**

**// on below line we are adding on value change**

**// for text field.**

**onValueChange = { subject = it },**

**// on below line we are adding place holder as text**

**placeholder = { Text(text = "Subject") },**

**// on below line we are adding modifier to it**

**// and adding padding to it and filling max width**

**modifier = Modifier**

**.padding(16.dp)**

**.fillMaxWidth(),**

**// on below line we are adding text style**

**// specifying color and font size to it.**

**textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),**

**// on below line we are**

**// adding single line to it.**

**singleLine = true,**

**)**

**// on below line adding a spacer.**

**Spacer(modifier = Modifier.height(10.dp))**

**Text(text = "Mail Body",**

**fontWeight = FontWeight.Bold,**

**fontSize = 16.sp)**

**// on the below line, we are creating a text field.**

**TextField(**

**// on below line we are specifying**

**// value for our text field.**

**value = body,**

**// on below line we are adding on value**

**// change for text field.**

**onValueChange = { body = it },**

**// on below line we are adding place holder as text**

**placeholder = { Text(text = "Body") },**

**// on below line we are adding modifier to it**

**// and adding padding to it and filling max width**

**modifier = Modifier**

**.padding(16.dp)**

**.fillMaxWidth(),**

**// on below line we are adding text style**

**// specifying color and font size to it.**

**textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),**

**// on below line we are**

**// adding single line to it.**

**singleLine = true,**

**)**

**// on below line adding a spacer.**

**Spacer(modifier = Modifier.height(20.dp))**

**// on below line adding a**

**// button to send an email**

**Button(onClick = {**

**if( recevierMail.isNotEmpty() && subject.isNotEmpty() && body.isNotEmpty()) {**

**val email = Email(**

**id = null,**

**recevierMail = recevierMail,**

**subject = subject,**

**body = body**

**)**

**databaseHelper.insertEmail(email)**

**error = "Mail Saved"**

**} else {**

**error = "Please fill all fields"**

**}**

**// on below line we are creating**

**// an intent to send an email**

**val i = Intent(Intent.ACTION\_SEND)**

**// on below line we are passing email address,**

**// email subject and email body**

**val emailAddress = arrayOf(recevierMail)**

**i.putExtra(Intent.EXTRA\_EMAIL,emailAddress)**

**i.putExtra(Intent.EXTRA\_SUBJECT,subject)**

**i.putExtra(Intent.EXTRA\_TEXT,body)**

**// on below line we are**

**// setting type of intent**

**i.setType("message/rfc822")**

**// on the below line we are starting our activity to open email application.**

**ctx.startActivity(Intent.createChooser(i,"Choose an Email client : "))**

**},**

**colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef))**

**) {**

**// on the below line creating a text for our button.**

**Text(**

**// on below line adding a text ,**

**// padding, color and font size.**

**text = "Send Email",**

**modifier = Modifier.padding(10.dp),**

**color = Color.Black,**

**fontSize = 15.sp**

**)**

**}**

**}**

5.ViewMailActivity.kt

**package com.example.emailapplication**

**import android.annotation.SuppressLint**

**import android.os.Bundle**

**import android.util.Log**

**import androidx.activity.ComponentActivity**

**import androidx.activity.compose.setContent**

**import androidx.compose.foundation.Image**

**import androidx.compose.foundation.layout.\***

**import androidx.compose.foundation.layout.R**

**import androidx.compose.foundation.lazy.LazyColumn**

**import androidx.compose.foundation.lazy.LazyRow**

**import androidx.compose.foundation.lazy.items**

**import androidx.compose.material.\***

**import androidx.compose.runtime.Composable**

**import androidx.compose.ui.Modifier**

**import androidx.compose.ui.graphics.Color**

**import androidx.compose.ui.layout.ContentScale**

**import androidx.compose.ui.res.painterResource**

**import androidx.compose.ui.text.font.FontWeight**

**import androidx.compose.ui.text.style.TextAlign**

**import androidx.compose.ui.tooling.preview.Preview**

**import androidx.compose.ui.unit.dp**

**import androidx.compose.ui.unit.sp**

**import com.example.emailapplication.ui.theme.EmailApplicationTheme**

**class ViewMailActivity : ComponentActivity() {**

**private lateinit var emailDatabaseHelper: EmailDatabaseHelper**

**@SuppressLint("UnusedMaterialScaffoldPaddingParameter")**

**override fun onCreate(savedInstanceState: Bundle?) {**

**super.onCreate(savedInstanceState)**

**emailDatabaseHelper = EmailDatabaseHelper(this)**

**setContent {**

**Scaffold(**

**// in scaffold we are specifying top bar.**

**topBar = {**

**// inside top bar we are specifying**

**// background color.**

**TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp),**

**// along with that we are specifying**

**// title for our top bar.**

**title = {**

**// in the top bar we are specifying**

**// title as a text**

**Text(**

**// on below line we are specifying**

**// text to display in top app bar.**

**text = "View Mails",**

**fontSize = 32.sp,**

**color = Color.Black,**

**// on below line we are specifying**

**// modifier to fill max width.**

**modifier = Modifier.fillMaxWidth(),**

**// on below line we are**

**// specifying text alignment.**

**textAlign = TextAlign.Center,**

**)**

**}**

**)**

**}**

**) {**

**val data = emailDatabaseHelper.getAllEmails();**

**Log.d("swathi", data.toString())**

**val email = emailDatabaseHelper.getAllEmails()**

**ListListScopeSample(email)**

**}**

**}**

**}**

**}**

**@Composable**

**fun ListListScopeSample(email: List<Email>) {**

**LazyRow(**

**modifier = Modifier**

**.fillMaxSize(),**

**horizontalArrangement = Arrangement.SpaceBetween**

**) {**

**item {**

**LazyColumn {**

**items(email) { email ->**

**Column(**

**modifier = Modifier.padding(**

**top = 16.dp,**

**start = 48.dp,**

**bottom = 20.dp**

**)**

**) {**

**Text("Receiver\_Mail: ${email.recevierMail}", fontWeight = FontWeight.Bold)**

**Text("Subject: ${email.subject}")**

**Text("Body: ${email.body}")**

**}**

**}**

**}**

**}**

**}**

**}**