A Project Report on

Adaptive mail: A Flexible Email Client App

BACHELOER OF COMPUTER APPLICATION

Saiva Bhanu Kshatriya College,

(Affiliated to Madurai Kamaraj University)

Aruppukottai-626101

Under the Guidance of

Mr

Saiva Bhanu Kshatriya Arts and Science College

(Affiliated to Madurai Kamaraj University)

Aruppukottai-626101

Submitted

BY

C. SELVAMANI (Reg.No:C0S23659)

M. VANMATHI (Reg.No:C0S23661)

M. RAMKUMAR (Reg.No:C0S23662)

N.VIGNESHWARAN (Reg.No:C0S23663)

Project Report Template

1. Introduction:

1.1 Overview

A brief description about your project

Adaptive Mail: A Flexible Email Client App

Project Description:

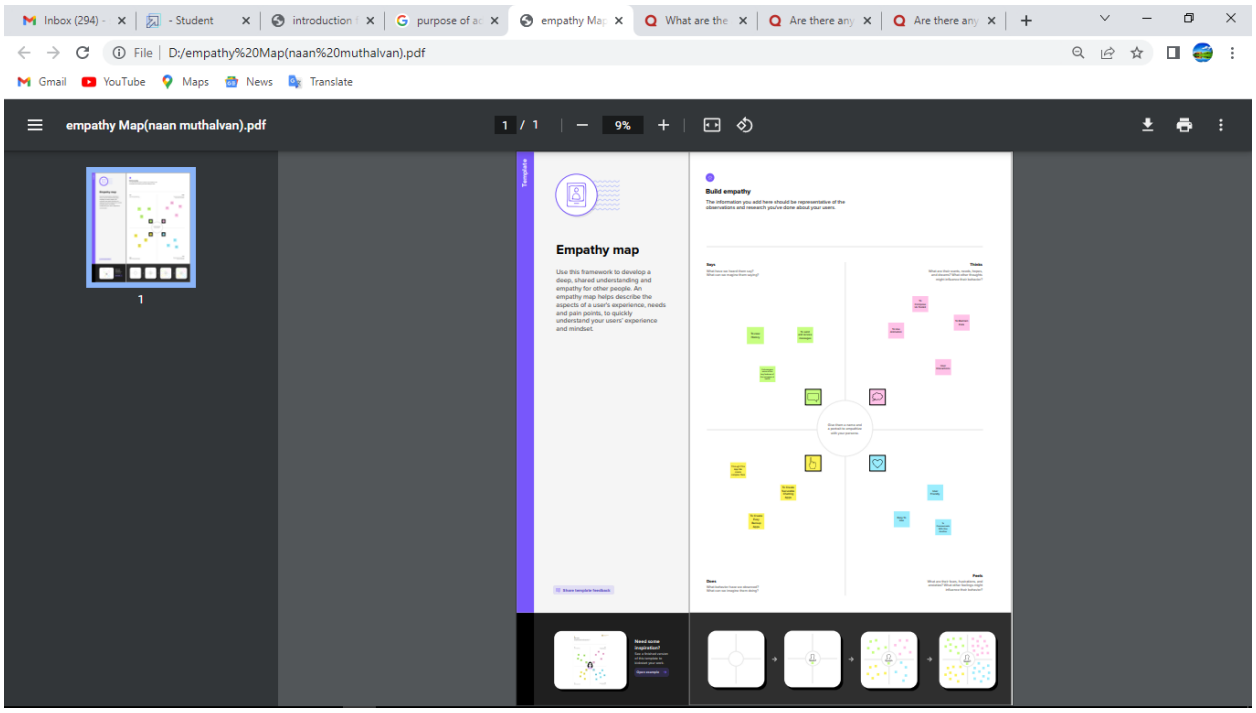
Adaptive Mail app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a conversational UI. The app simulates a messaging interface, allowing the user to send and receive messages, and view a history of previous messages. It showcases some of the key features of the Compose UI toolkit, data management, and user interactions.

1.2 Purpose

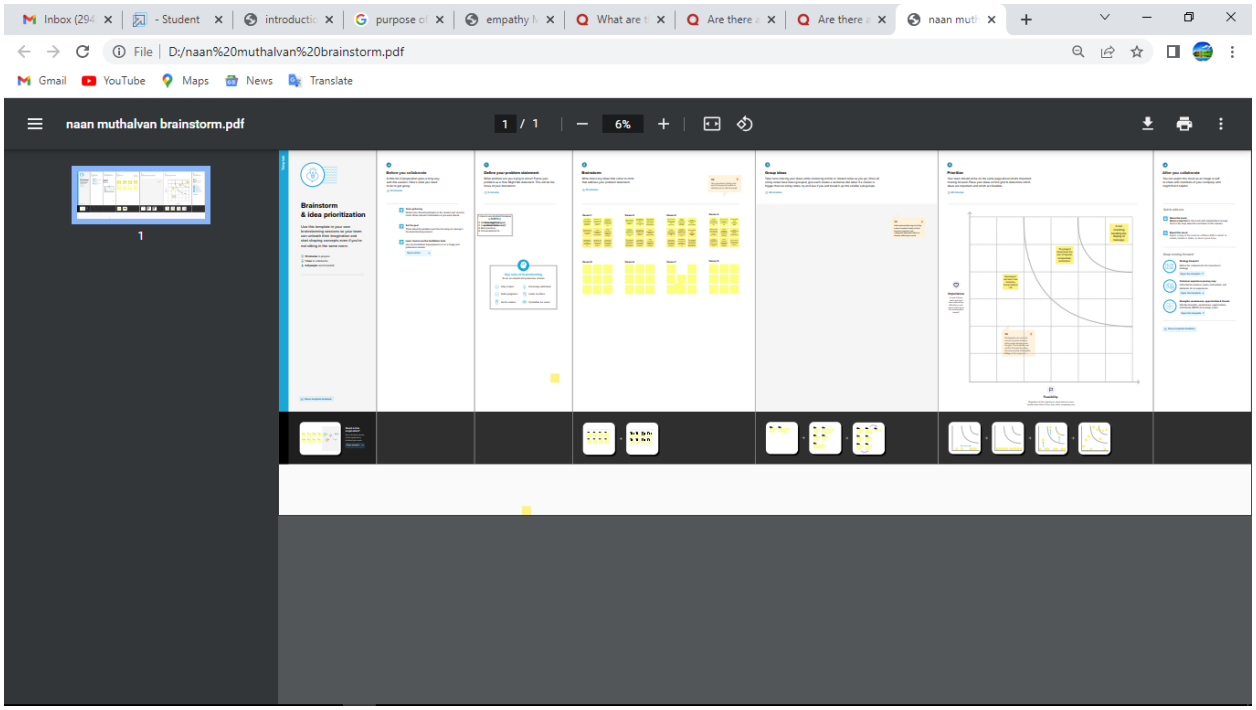
An email client is a program that lives on your computer and **lets you send or receive emails**. Typical examples include Outlook and Windows Live Mail. We don't recommend using email clients - they're less future-proof than accessing your emails through a browser or app via webmail.

2. Problem Definition&Design Thinking:

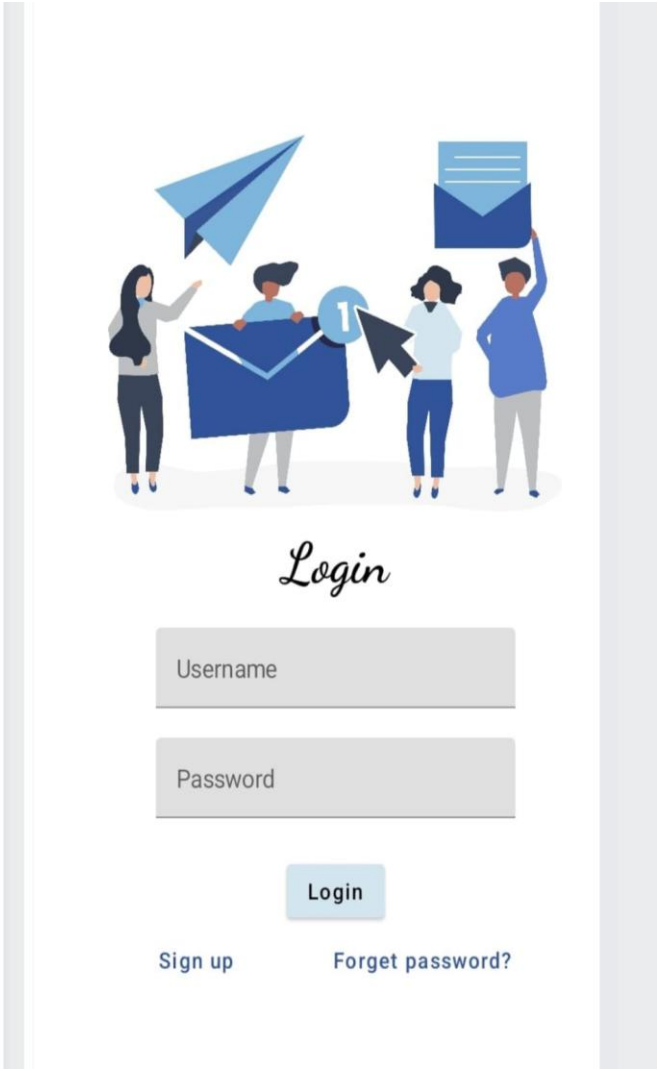
2.1 Empathy map



2.2 Ideation & brainstorming map screenshot



3. Result



Register

Username

Email

Password

Register

Home Screen

Send Email

View Emails

View Mails

Receiver_Mail: kavya78@gmail.com
Subject: Android
Body: This is an Adaptive Email app

Receiver_Mail: shirishbokka7@gmail.com
Subject: Order
Body: your courier has arrived

4. Advantages and Disadvantages

Advantages

- Access emails only via the internet. As the name suggests, “web” mail can only be accessed via your webmail website browser. This means that you’ll always need an internet connection in order to access the full features of your emails.
- Access emails within your web browser. Webmail allows you to access your emails when browsing your website provider, for example, Google and Gmail.
- Emails can be backed up in the email server. Webmails are automatically backed up by the email server or website provider, so you don’t have to manually backup your emails.
- Emails are automatically scanned for viruses by the service provider. When using webmail your emails and their attachments are automatically scanned by the website service provider.

Disadvantages:

Email lacks a personal touch. While some things are better off sent as written and typed messages, some things should be verbally relayed or written by hand in a note or letter. Email can be disruptive. Going through each email can be disruptive to work as it does require a bit of time.

5. Applications:

- Emails are accessible without internet access. Email clients allow you to access an “offline” feature, which allows you to view, reply or write professional emails without being connected to the internet on your desktop or mobile. Webmail does offer this but this is a built-in feature for desktop email clients. But you’ll only have access to your emails that were updated before you disconnect from the internet so keep that in mind.

- Easily customize your email client. Unlike webmail, most email clients allow you to customize the layout, color and other features so the user experience is more personalized.

- Emails can be backed up to the computer. Email clients allow you to back up your emails so you can securely find, archive or delete emails on your computer. This has to be done manually as not everyone wants their emails to be stored.
- More email management features. Email clients allow you to integrate or add extensions to your email provider for additional features so you can increase your email productivity.
- An up-to-date antivirus program is required. Email will need an antivirus program to help scan your emails and protect you from viruses; this isn't an automatic feature with email clients.

6.Conclution

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.
- We covered a lot in this article but hopefully you have a better understanding of the features, tech and benefits when it comes to learning 'what is an email client". To help here's a quick recap of what we covered:

- The pros and cons of using an email or webmail and how understanding the benefits of each can help you to decide what's best for you.
- How desktop email clients can proactively help you to manage your emails, access emails online, and offer more seamless integrations to better your inbox productivity.
- When looking for a good email client software, you should look out for: customization, email management/productivity features, affordability, paid subscription, and app integrations.
- The top-rated email clients are Mailbird, eM Client, Thunderbird, and Window Mail based on built-in features, integrations, and support.

With new and updated email clients emerging, you have a wide range of options to choose from. Selecting the right once can help make your life easier or add more hurdles to your workload. With the right information at your fingertips, you can select an email client that works best for you.

7 . Future scope

1. Gmail's app still doesn't support the <style> tag for non-Gmail accounts

This is the big one. The Gmail app is massively popular on iOS and Android, it can be set as users' default mail app on any smartphone, and it supports all non-Gmail accounts. But viewed in the Gmail app, emails for non-Gmail accounts don't support <style> tags, so media queries, which we can usually rely on to optimise our emails for small screens, aren't supported. This tutorial will show you how to make emails that are responsive, even in this scenario.

2. It's really hard to keep on top of email services and their level of CSS support

New email providers and apps appear all the time. Some of them have great CSS and media query support, but some of them focus more on email workflows and don't support <style> tags or media queries at all. Some of them differ across platforms – for example, the Yahoo mail app for iOS supports <style> in the head, whereas on Android it doesn't, unless you include your entire <head> twice, which isn't possible from every sending platform.

This tutorial will show you how to make an email that is *always* responsive, even in a worst-case scenario where CSS support is nonexistent, so you don't have to worry about how your emails render in all those unknown use cases.

8.Appendix

Source code:

Creating LoginActivity.KtWith Database

```
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.EmailApplicationThe
me
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)
}
```

Creating RegisterActivity.Kt With Database

```
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.EmailApplicationThe
me
```

```
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)
}
```

Creating MainActivity.Kt File

```
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.EmailApplicationThem
e
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
)
}
}
}
```

Creating SendMailActivity.Kt File

```
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.EmailApplicationThem
e
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
)
}
}
}
```

Creating ViewMailActivity.Kt File

```
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.EmailApplicationThe
me
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}")
}
}
}
}
}
}
```