Adaptive Mail: A Flexible Email Client App

1. INTRODUCTION

1.1 Overview

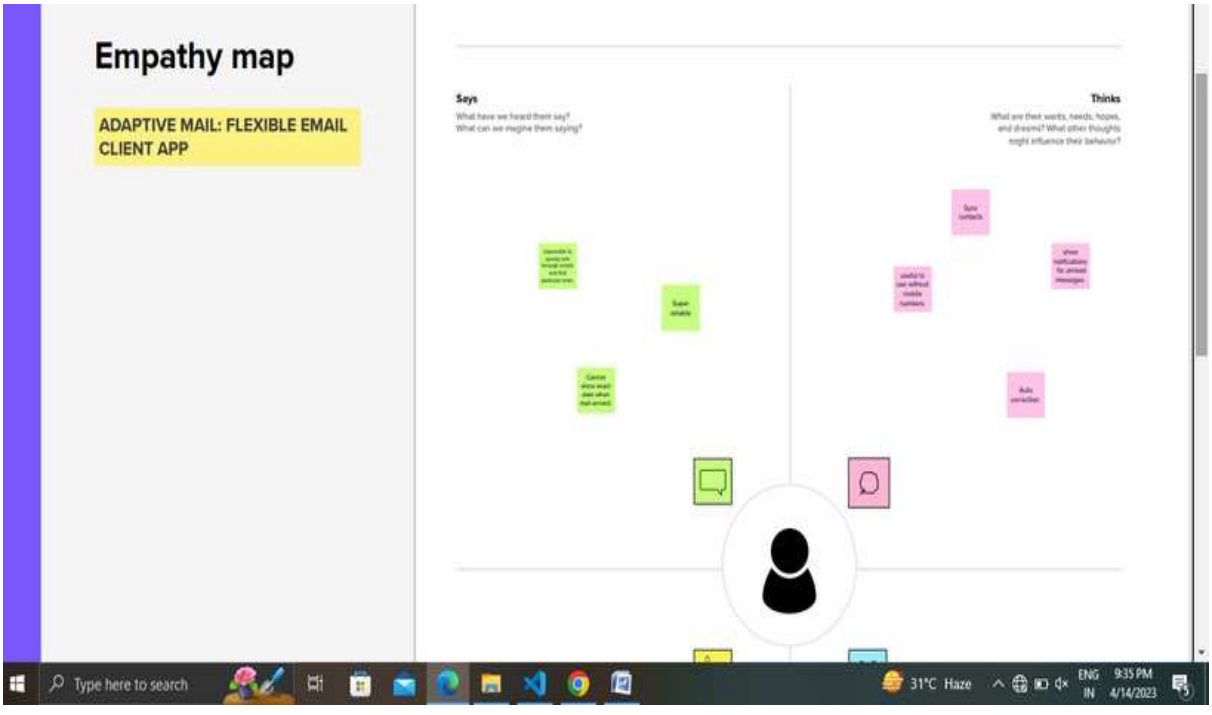
Adaptive Mail app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a conversational UI. The app simulates a messaging interface, allowing the user to send and receive messages, and view a history of previous messages. It showcases some of the key features of the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

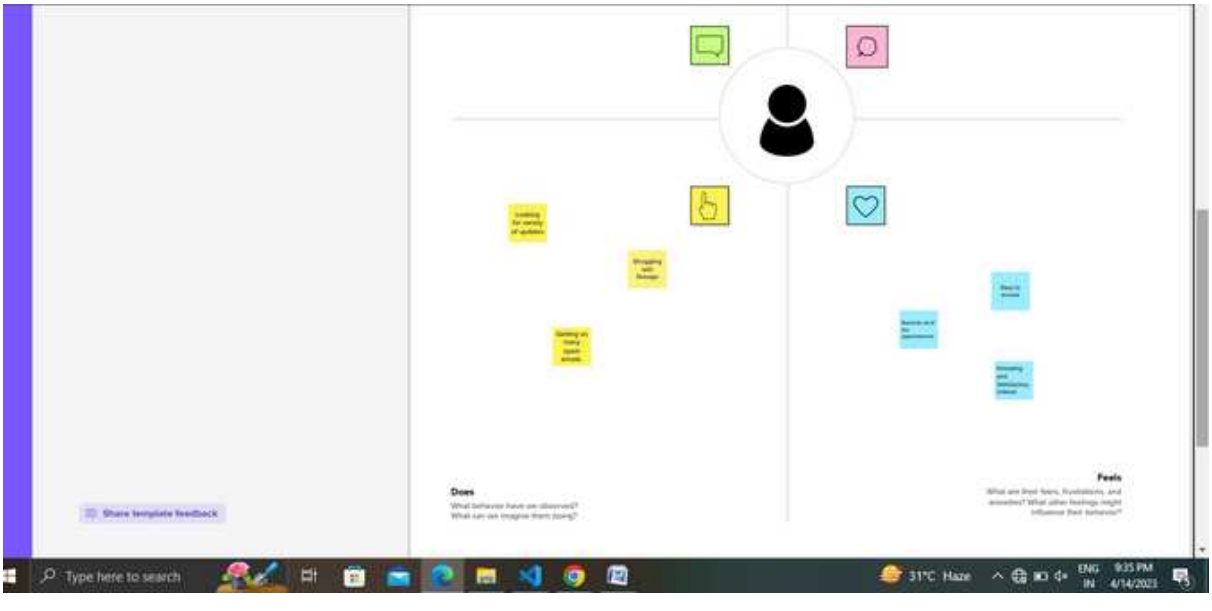
1.2 Purpose

The app stimulates a messaging interface, allowing the user to send and receive messages, and view a history of previous messages. It showcases some of the key featuresof the Compose UI toolkit, data management, and user information.

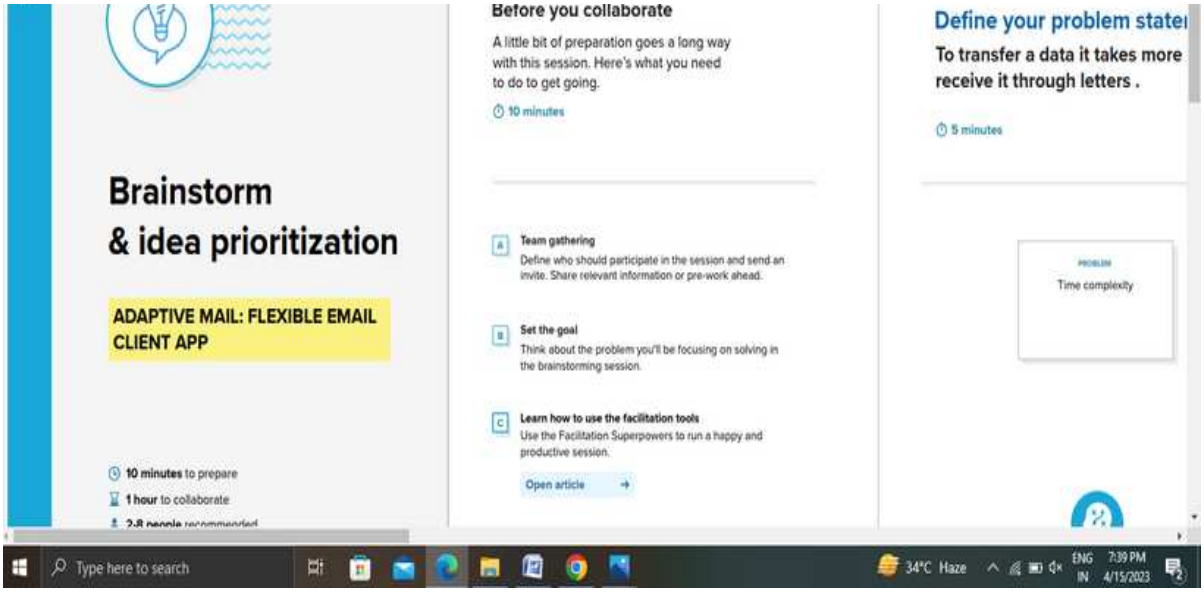
2. PROBLEM DEFINITION & DESIGN THINKING

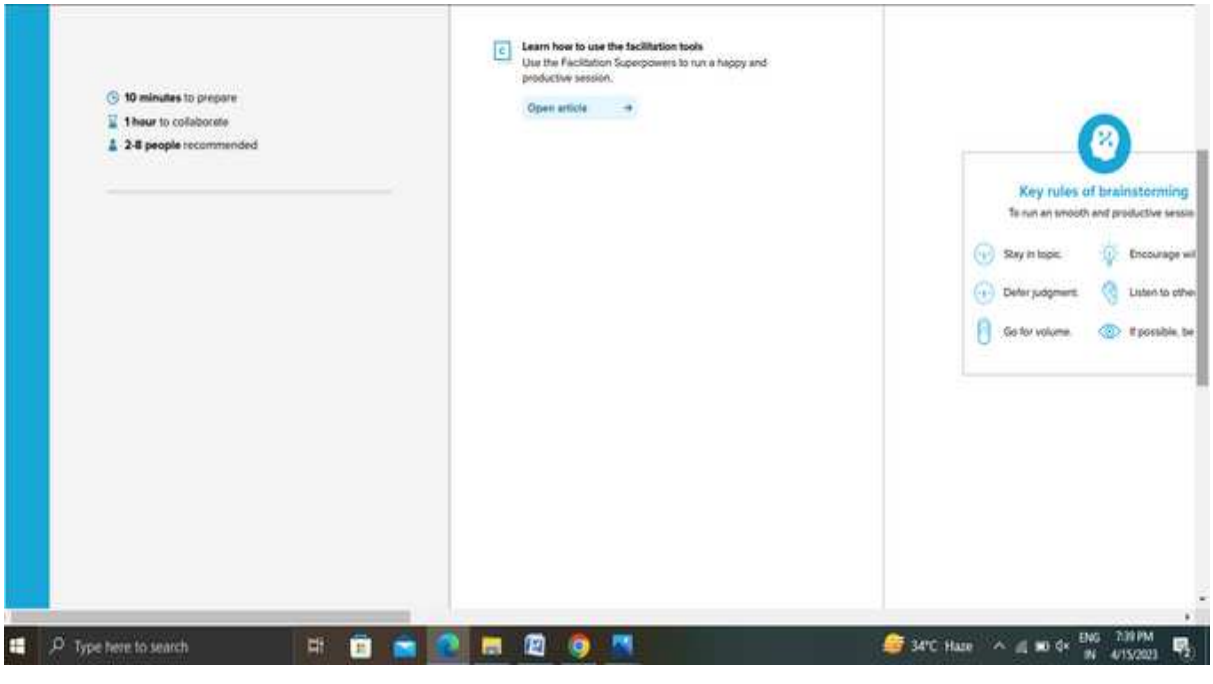
2.1 Empathy Map



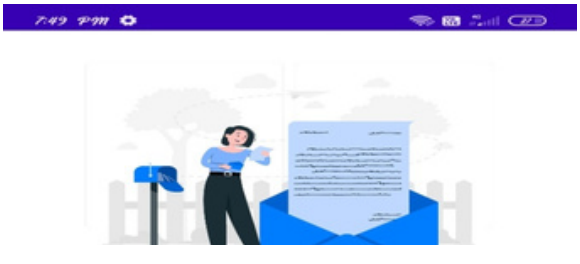


2.2 IDEATION & BRAINSTORMING MAP


Brainstorm
& idea prioritization
ADAPTIVE MAIL: FLEXIBLE EMAIL CLIENT APP
10 minutes to prepare
1 hour to collaborate
Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes
Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.
Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article
Define your problem state
To transfer a data it takes more
receive it through letters .
5 minutes
Time complexity
Type here to search
34°C Haze
ENG
IN
7:39 PM
4/15/2023


10 minutes to prepare
1 hour to collaborate
2-8 people recommended
Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article
Key rules of brainstorming
Stay in topic.
Defer judgment.
Go for volume.
Type here to search
34°C Haze
ENG
IN
7:39 PM
4/15/2023

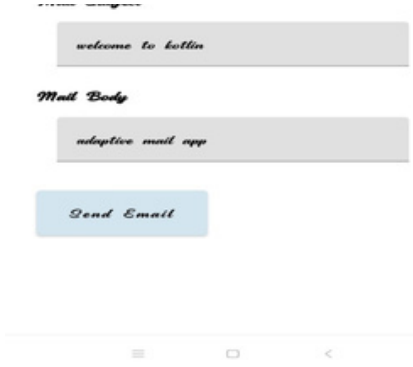
3. RESULT

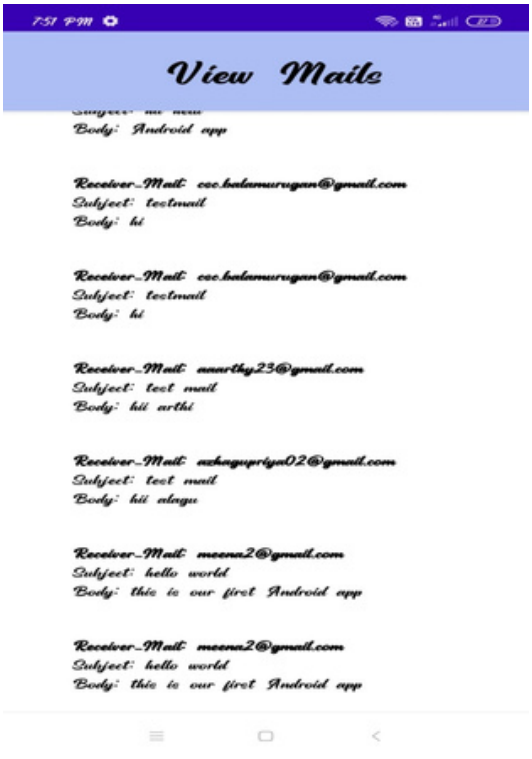



Register
Username
meena
Email
meena@gmail.com
Password
••••••••
Register
Have an account?


7:49 PM
Login
Username
Password
Login
Sign up
Forget password?


7:52 PM
Send Mail
Receiver Email-Id
Email address
meena2@gmail.com
Mail Subject





4. ADVANTAGES & DISADVANTAGES

4.1 Advantages:

1. Cost Effective :

The message sent through e - mail costs very low . It is very much cheaper than courier or fax or telegram . A distant courier may cost you hundred rupees or a fax can cost you the same price but the same message can be sent through e - mail at the cost of only one local call and proportionate service charges of Internet connection .

2. High Speed:

E-mail can be sent very fast and almost instantaneously. Similarly the message sent by the others to you can be received instantly

3. Easy to Use :

It is very easy to use E - mail tool on the Internet . Just you have to do is type the message and flash it at a click of the mouse. You don't have to go to post office for buying envelopes or postage stamps or other stationery. You do not have to write it on paper.

4. Time Saving:

It saves your lot of time. You do not have to use carbon papers to write multiple copies of the letter. You can send multiple copies of the letter to the same recipient or various other recipients at the click of a mouse.

5. Waste Reduction :

A lot of paper work is saved which also reduces file maintenance. For offices , it is very useful and beneficial to use the e - mail facility .

6. Record Maintenance :

Record maintenance of e - mails sent and received is also possible because all the messages are stored in the form of files in computer .

7. Message storing:

If for the reason , the recipients is away , the e - mail message waits for him until he received and reads the message .

4.2 Disadvantages

. It is source of viruses. It is capable to harm one's computer and read out user's e-mail address book and send themselves to number of people around the world.

2. It can be source of various spams. These spam mails can fill up inbox and to deletion of these mail consumes lot of time.

. It is informal method of communication. The documents those require signatures are not managed by e-mail.

4. To use facility of e-mail, user must have an access to internet and there are many parts of world where people does not have access to Internet.

5. Sometimes, e-mails becomes misunderstood as it is not capable of expressing emotions.

6. To be updated, user have to check inbox from time-to-time.

5. APPLICATIONS

- Email is a very popular way of communicating with others over the Internet.
- An application that allows users to send, receive, and read email is called an *email client*.
- Each of the email client applications is designed to suit specific types of users; so, you can choose one with the features that best suits your particular needs.
- Since all email clients perform the same basic tasks (send and receive email), you should choose one that is convenient and easy to use.

This chapter briefly discusses the following email clients:

- **Evolution**
- **Thunderbird**
- **Mutt**, a text-based email client

:

6. CONCLUSION

We are at a turning point in business today. Tools such as e-mail, instant messaging, and online chat have become an integral part of our everyday workplace. This influx of technology into almost every facet of business life has created a streamlined workplace that often collides with our increased need for human contact.

Throughout the past years we have interviewed more than 25,000 businesspeople in our seminars and presentations, asking them, "What's the single biggest challenge you face in today's business environment?" Time and again we hear variations on the same theme: "How do we stay in touch and connected to each other, in a high-tech world?"

7. FUTURE SCOPE

In the next two years alone, the number of worldwide email accounts is reportedly expected to continue growing at a slightly faster than the number of worldwide email users(3% per year- reaching 4.4 billion worldwide users by end of 2024).Email intelligence will reach.

8. APPENDIX

A. Source code

Email.kt

```
package com.example.emailapplication

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity
(tableName = "email_table")
data class Email(
@PrimaryKey(autoGenerate = true) val id: Int?,
@ColumnInfo(name = "receiver_mail") val recevierMail: String?,
@ColumnInfo(name = "subject") val subject: String?,
@ColumnInfo(name = "body") val body: String?,)
```

EmailDao.kt

```
package com.example.emailapplication

import androidx.room.*

@Dao
interface EmailDao {

@Query("SELECT * FROM email_table WHERE  subject= :subject")
suspend fun getOrderBySubject(subject: String): Email?

@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertEmail(email: Email)

@Update
suspend fun updateEmail(email: Email)

@Delete
suspend fun deleteEmail(email: Email)

}
```

EmailDatabase.kt

```
package com.example.emailapplication

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [Email::class], version = 1)
abstract class EmailDatabase : RoomDatabase() {

abstract fun emailDao(): EmailDao

companion object {

@Volatile
private var instance: EmailDatabase? = null
fun
getDatabase(context: Context): EmailDatabase {
return instance ?: synchronized(this) {
val newInstance = Room.databaseBuilder(                context.
applicationContext,
EmailDatabase::class.java,
"email_database"
).build()
instance = newInstance            newInstance

}
} }
}
```

EmailDatabseHelper.kt

```
package com.example.emailapplication

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class EmailDatabaseHelper(context: Context) :  SQLiteOpenHelper(context,
DATABASE_NAME, null,DATABASE_VERSION){
```

```
companion object {
private const val DATABASE_VERSION = 1
private const val DATABASE_NAME = "EmailDatabase.db"

private const val TABLE_NAME = "email_table"
private const val COLUMN_ID = "id"
private const val COLUMN_RECEIVER_MAIL = "receiver_mail"
private const val COLUMN_SUBJECT = "subject"
private const val COLUMN_BODY = "body"
}

override fun onCreate(db: SQLiteDatabase?) {
val createTable = "CREATE TABLE $TABLE_NAME (" +
"${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +
"${COLUMN_RECEIVER_MAIL} Text, " +
"${COLUMN_SUBJECT} TEXT ," +
"${COLUMN_BODY} TEXT " +
")"

db?.execSQL(createTable)    }


override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {       db?.execSQL(
"DROP TABLE IF EXISTS $TABLE_NAME")       onCreate(db)
}


fun insertEmail(email: Email) {
val db = writableDatabase
val values = ContentValues()
values.put(COLUMN_RECEIVER_MAIL, email.recevierMail)
values.put(COLUMN_SUBJECT, email.subject)
values.put(COLUMN_BODY, email.body)
db.insert(TABLE_NAME, null, values)
db.close()    }



@SuppressLint("Range")
fun getEmailBySubject(subject: String): Email? {
val db = readableDatabase
val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_SUBJECT = ?", arrayOf(subject))
var email: Email? = null       if
(cursor.moveToFirst()) {
```

```
                email = Email(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN_RECEIVER_MAIL)),
                    subject = cursor.getString(cursor.getColumnIndex(COLUMN_SUBJECT)),
                    body = cursor.getString(cursor.getColumnIndex(COLUMN_BODY)),
                )
            }

            cursor.close()
            db.close()
            return email
        }
    @SuppressLint("Range")
    fun getEmailById(id: Int): Email? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID = ?",
            arrayOf(id.toString()))
        var email: Email? = null
        if (cursor.moveToFirst()) {
            email = Email(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN_RECEIVER_MAIL)),
                subject = cursor.getString(cursor.getColumnIndex(COLUMN_SUBJECT)),
                body = cursor.getString(cursor.getColumnIndex(COLUMN_BODY)),
            )
        }

        cursor.close()
        db.close()
        return email
    }

    @SuppressLint("Range")
    fun getAllEmails(): List<Email> {
        val emails = mutableListOf<Email>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val email = Email(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    recevierMail = cursor.getString(cursor.getColumnIndex(COLUMN_RECEIVER_MAIL)),
                    subject = cursor.getString(cursor.getColumnIndex(COLUMN_SUBJECT)),
                    body = cursor.getString(cursor.getColumnIndex(COLUMN_BODY)),
                )

                emails.add(email)
            } while (cursor.moveToNext())
```

```
cursor.close()
db.close()
return emails
}
}
```

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)

}
```

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
                )
                  }

                   }

                  }
```

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)}

SendMailActivity.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
)
} }

}
```

User.kt

```
package com.example.emailapplication

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity
(tableName = "user_table")
data class User(
@PrimaryKey(autoGenerate = true) val id: Int?,
@ColumnInfo(name = "first_name") val firstName: String?,
@ColumnInfo(name = "last_name") val lastName: String?,
@ColumnInfo(name = "email") val email: String?,
@ColumnInfo(name = "password") val password: String?,
)
```

UserDao.kt

```
package com.example.emailapplication

import androidx.room.*

@Dao
interface UserDao {

@Query("SELECT * FROM user_table WHERE email = :email")
suspend fun getUserByEmail(email: String): User?

@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertUser(user: User)

@Update
suspend fun updateUser(user: User)

@Delete
suspend fun deleteUser(user: User)}
```

UserDatabase.kt

```
package com.example.emailapplication

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object {

@Volatile
private var instance: UserDatabase? = null
fun
getDatabase(context: Context): UserDatabase {
return instance ?: synchronized(this) {
val newInstance = Room.databaseBuilder( context.
applicationContext,
UserDatabase::class.java,
"user_database"
```

```
            ).build()
        instance = newInstance            newInstance

    }
    }   }
      }
```

UserDatabaseHelper.kt

```
package com.example.emailapplication

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :   SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)   }
```

```
override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {        db?.execSQL(
                "DROP TABLE IF EXISTS $TABLE_NAME")        onCreate(db)
                }


                fun insertUser(user: User) {
                val db = writableDatabase
                val values = ContentValues()
                values.put(COLUMN_FIRST_NAME, user.firstName)
                values.put(COLUMN_LAST_NAME, user.lastName)
                values.put(COLUMN_EMAIL, user.email)
                values.put(COLUMN_PASSWORD, user.password)
                db.insert(TABLE_NAME, null, values)
                db.close()    }


                @SuppressLint("Range")
                fun getUserByUsername(username: String): User? {
                val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
                $COLUMN_FIRST_NAME = ?", arrayOf(username))
                var user: User? = null        if
                (cursor.moveToFirst()) {
                user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
        firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
        lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
        email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
        password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),            )
                }

                cursor.close()
                db.close()
                return user
                }
                @SuppressLint("Range")
                fun getUserById(id: Int): User? {
                val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID = ?",
                arrayOf(id.toString()))
                var user: User? = null        if
                (cursor.moveToFirst()) {
                user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
        firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
```

```
lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),          )
}

cursor.close()
db.close()
return user
}

@SuppressLint("Range")
fun getAllUsers(): List<User> {
val users = mutableListOf<User>()
val db = readableDatabase
val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
if (cursor.moveToFirst()) {
do {
val user = User(
id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),          )

users.add(user)          }
while (cursor.moveToNext())       }

cursor.close()
db.close()
return users
}
}
```

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}")
                        }          }
                            }
                        }

                    }
```

}

1.
2.
3.
4.
5.
6.

-
-
-
-

-
-
-