**COMPOSE A DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE**

A major project report submitted to **Manonmaniam Sundaranar University** in partial fulfilment of the requirement for the award of Degree Bachelor of Science in Computer Science.

### SUBMITTED BY

**TEAM LEADER:**

**R.Varshan – ASMSU2022010824 (NANMUDHALVAN ID)**

**TEAM MEMBER:**

**K.Thanga kumar – ASMSU2022010823 (NANMUDHALVAN ID)**

**DEPARTMENT OF COMPUTER SCIENCE**

**ADITANAR COLLEGE OF ARTS AND SCIENCE**

**VIRAPANDIANPATNAM - 628216**

**PROJECT**

**REPORT TEMPLATE**

**1.** **INTRODUCTION**

* **Overview**

The flexible email Is A Personalized That Lets You Follow Trending Topics, transfer data Contents , Knowledge gaining News.

flexible mail is the exchange of computer-stored messages from one user to one or more recipients via the internet. Emails are a fast, inexpensive and accessible way to communicate for business or personal use.

**1.2) Purpose**

Email stands for electronic mail. Similar to a letter, it is sent via the internet to a recipient. An email address is required to receive email and that address is unique to the user.

Key benefits and feature of using email-

Emails are easy to use. You can organize your daily correspondence, send & receive electronic messages.

Emails are fast, no other form of written communication is as fast as an email. They are delivered at once around the world.

Simple & formal languages used in email.

We can use pictures & send birthday cards or newsletters through emails.

Emails are used for the purpose of communication, such as communicating with instructors and professors, keeping in touch with friends, requesting information from other people or businesses, applying for scholarships, jobs and internships. Even though email is a very valuable communication tool, its wide use in business and academic settings has led to the emergence of new challenges for the users.

**2. PROBLEM DEFINITION AND**

**DESIGN THINKING**

Email marketing is building strong relationships with the audience through effective emails. In the digital world, email marketing is a popular tool to promote your brand and engage with potential customers.

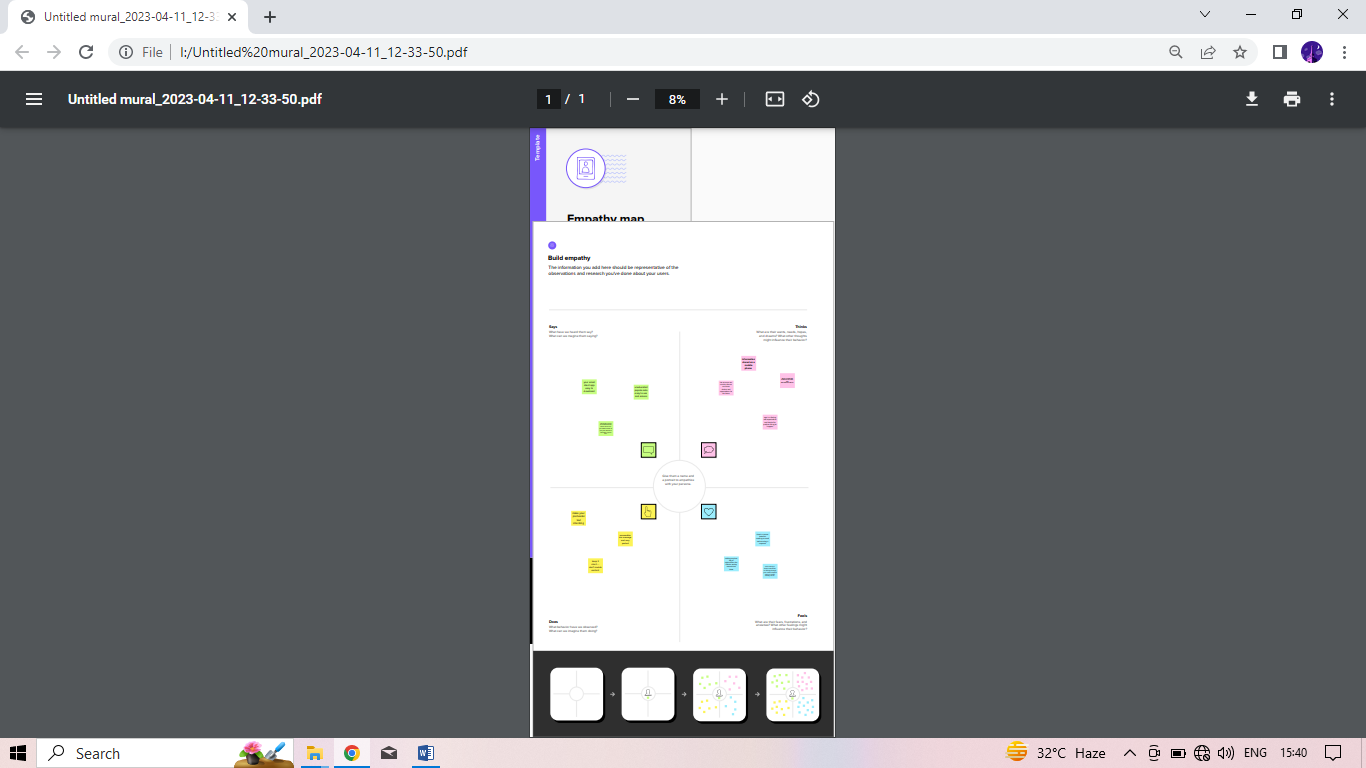
The whole reason for sending emails to the contacts is to convey your message with a personalized effect for a better customer experience.

But, there are 306+ billion emails sent every day (source). Some land in the spam, many are left unread.

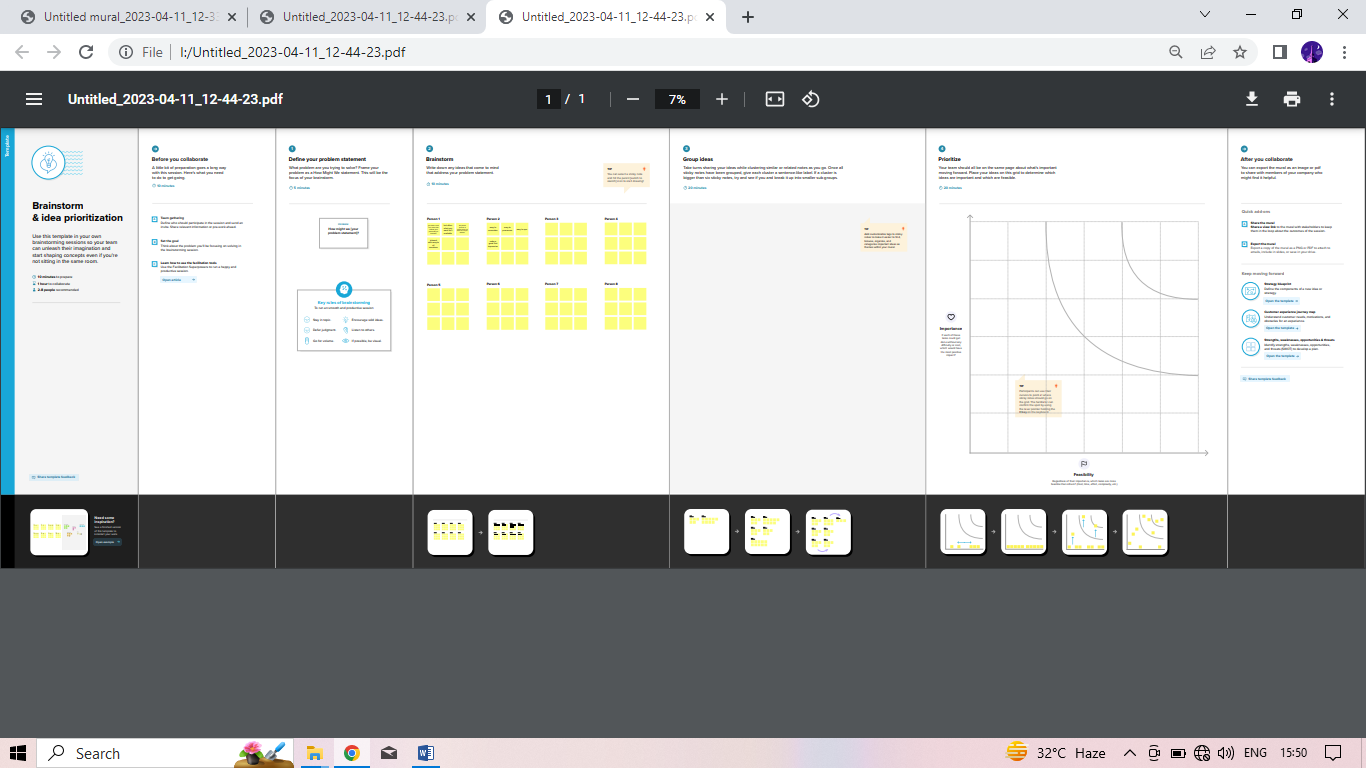
How can you hold the dwindling attention of your audience to make

your campaigns successful? Let's find out how you can do that with a well-designed email.

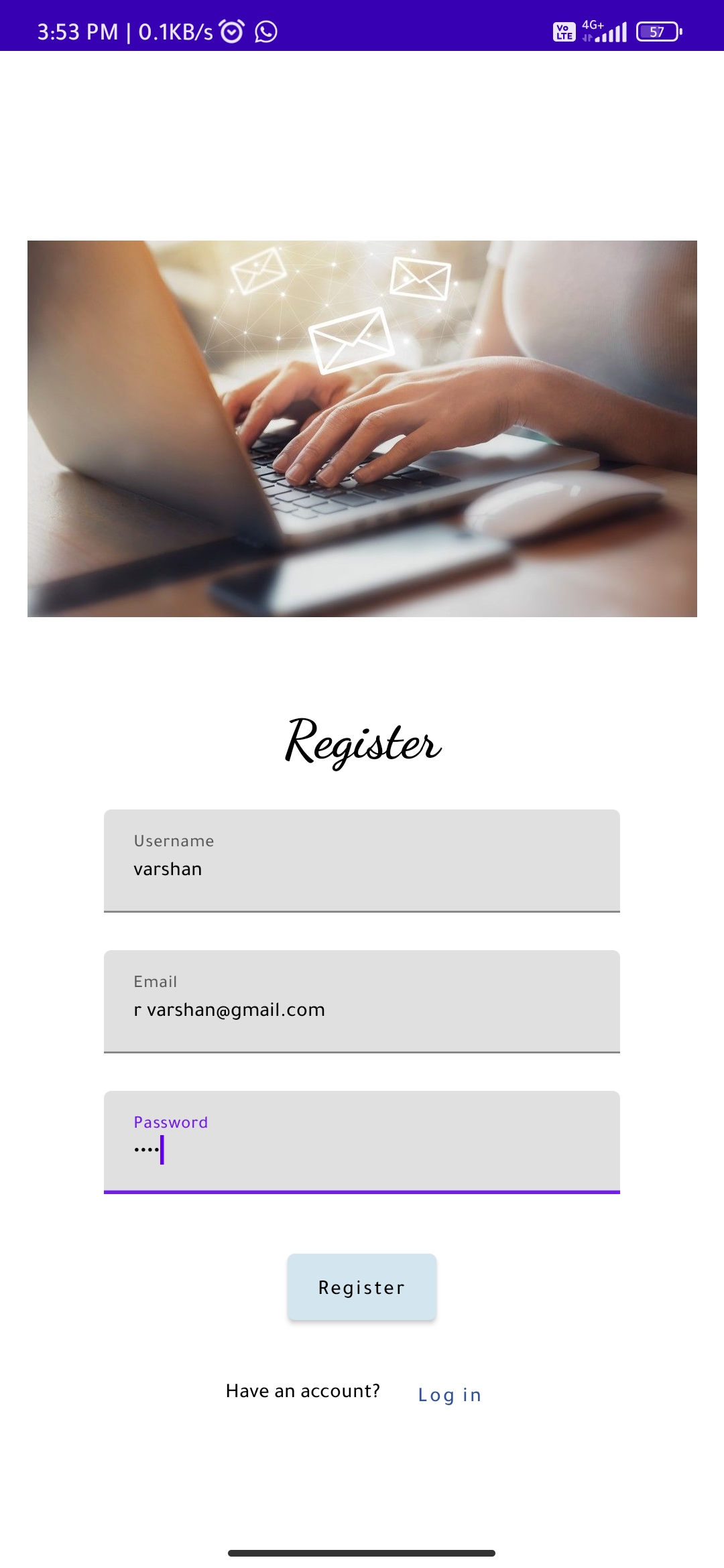
**2.1) Empathy Map**

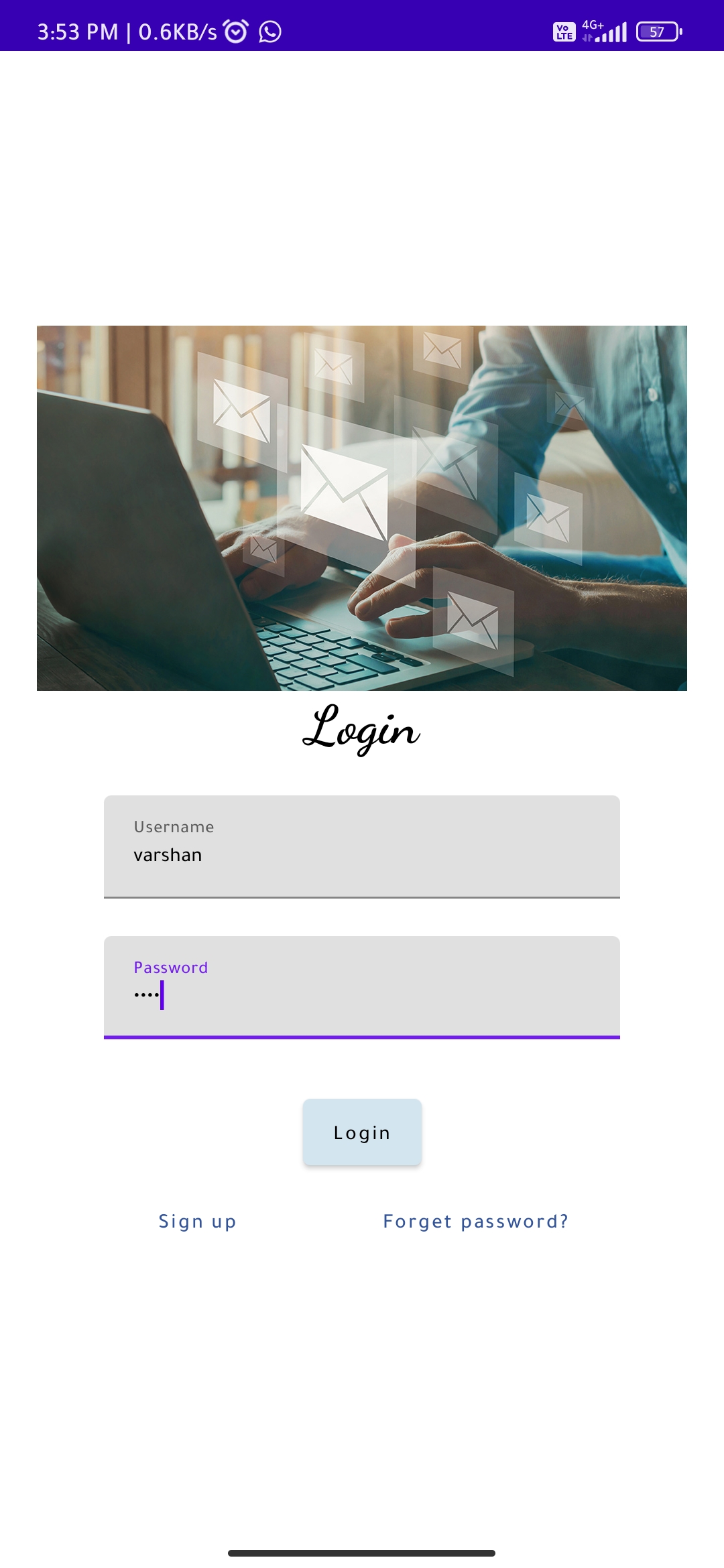
****

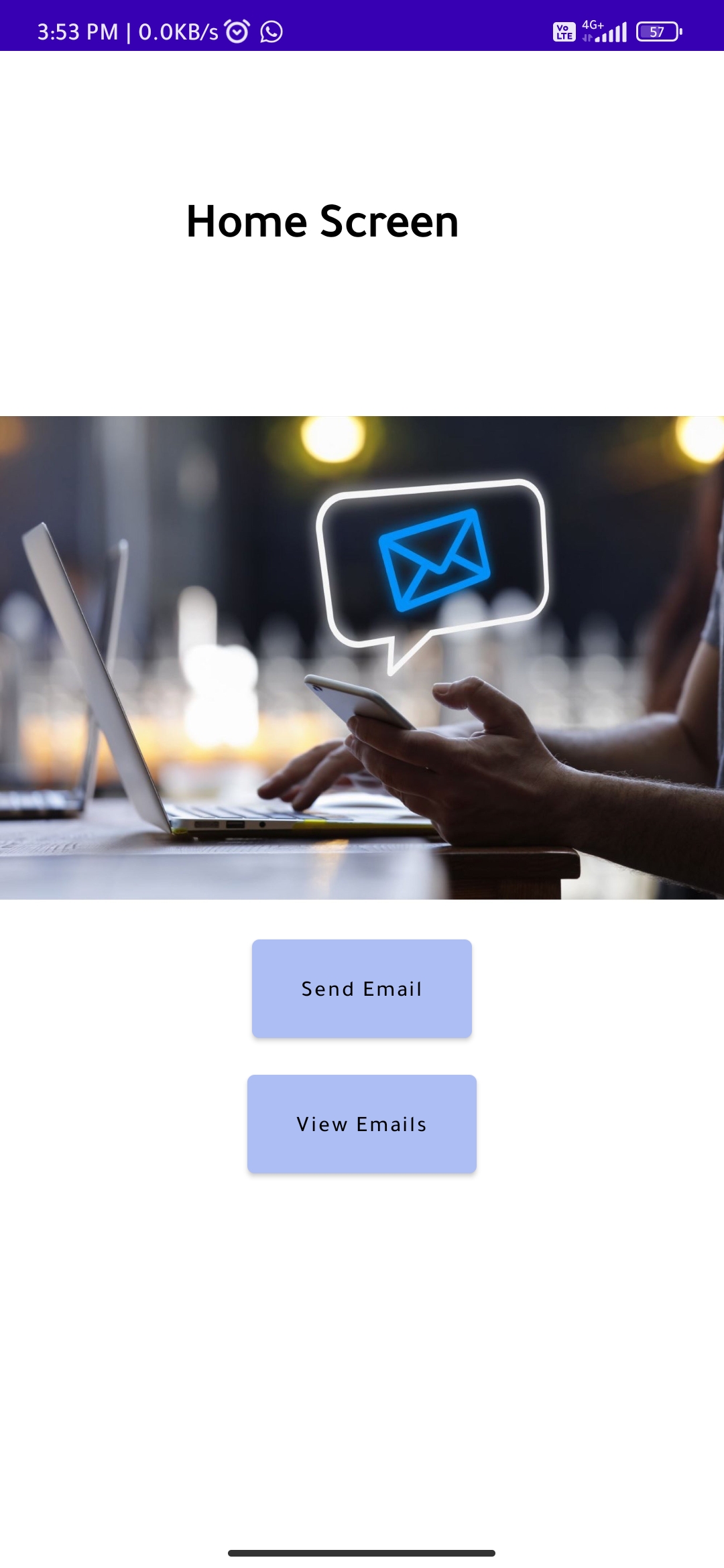
**2.2) Brainstorming Map**

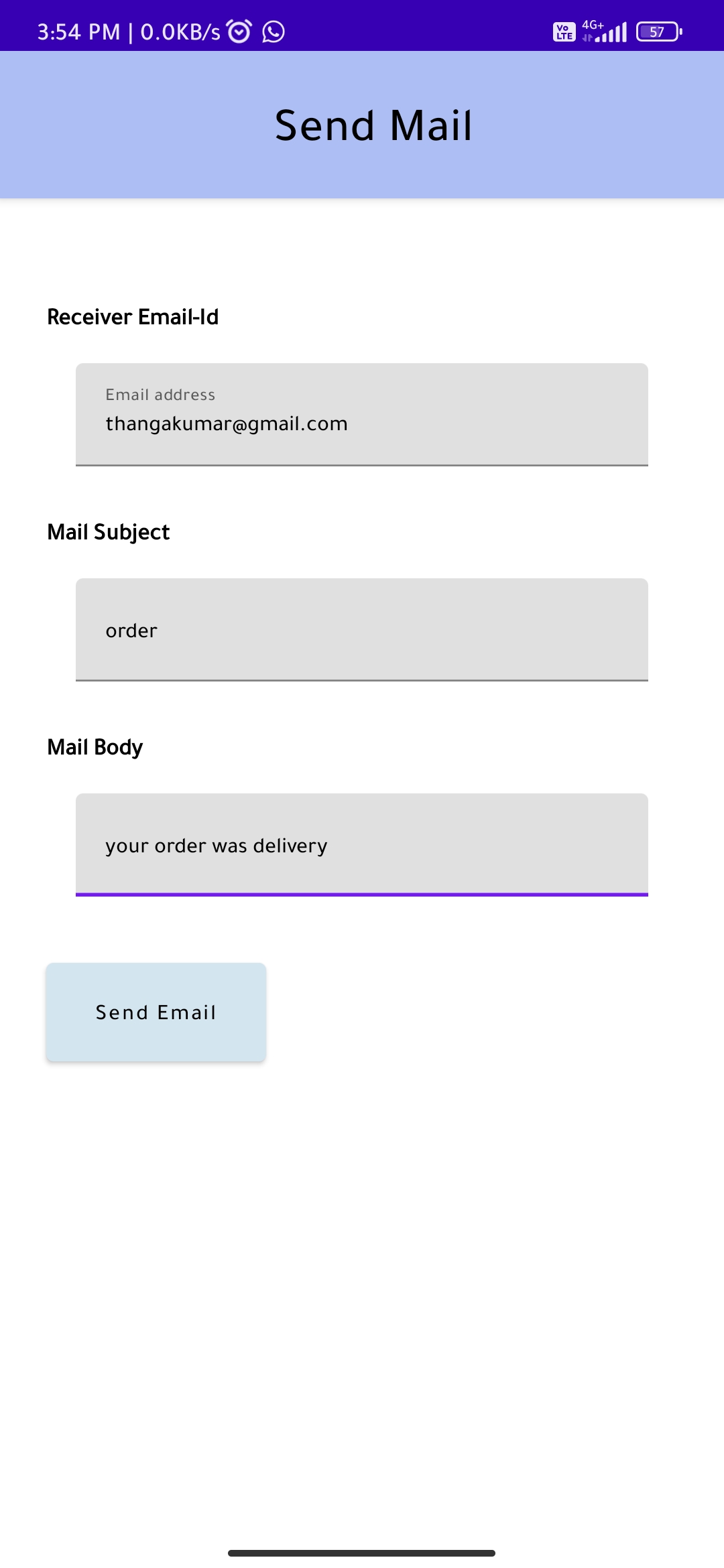


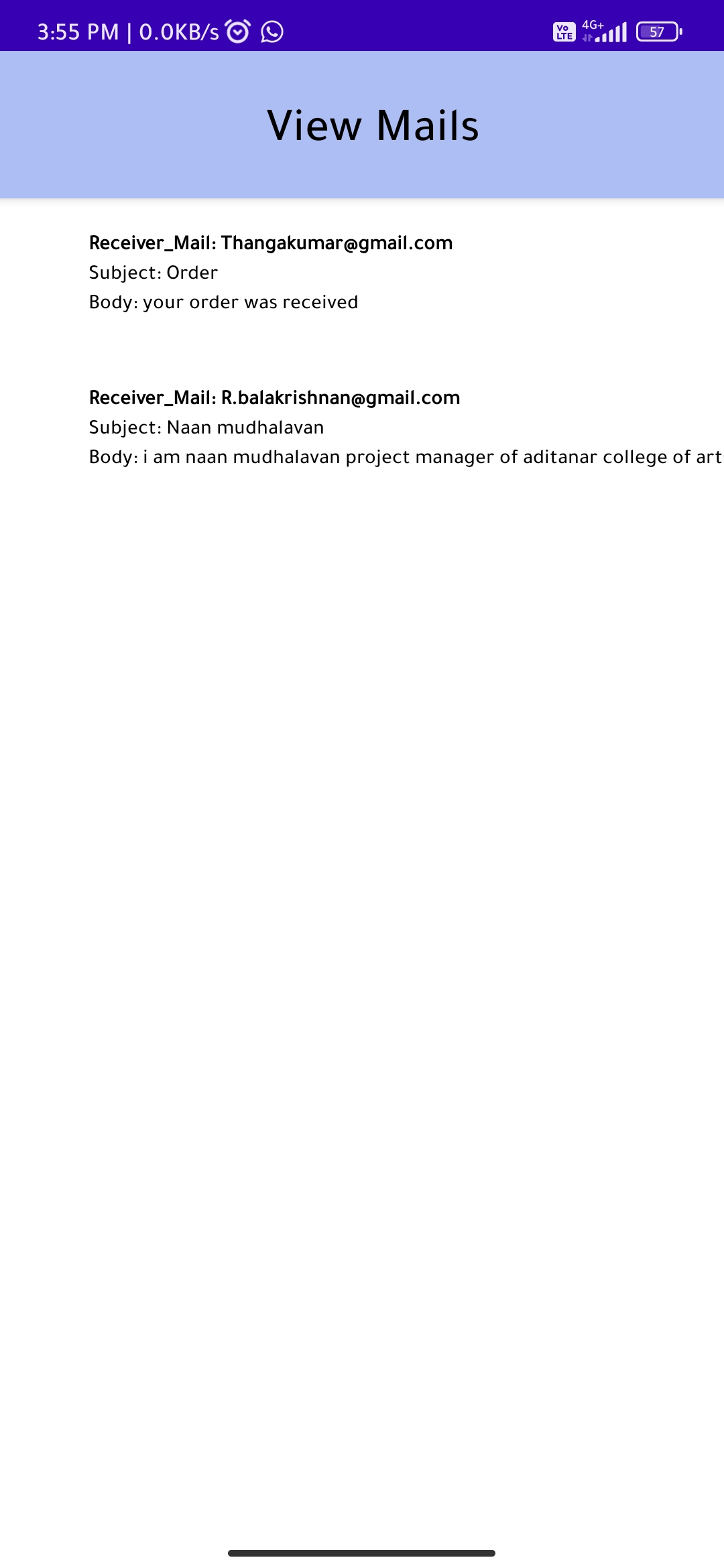
**Result:**

****

****

****

****

****

**Advantage :**

* **Features:**

**Email can increase efficiency, productivity and business readiness. Using email in business is:**

**cheap - sending email costs the same regardless of distance and the number of people you send it to**

**fast - an email should reach its recipient in minutes, or at the most within a few hours**

**convenient - your message will be stored until the recipient is ready to read it, and you can easily send the same message to a large number of people**

**permanent - you can keep a record of messages and replies, including details of when a message was received**

**One of the main advantages of email is that you can quickly and easily send electronic files such as text documents, photos and data sheets to several contacts simultaneously by attaching the file to an email. Check with your internet service provider if there is a limit to the size of email attachments you can send. Some businesses may also limit the type and size of attachments that they are willing to receive.**

**You can further increase your efficiency by setting up your email software to:**

**automatically create entries in your address book for every message you send or receive**

**respond to incoming emails automatically, eg to confirm receipt of an order, or to let people know that you are on leave or out of the office**

**Disadvantage :**

Email can pose certain risks that are worth being aware of, such as:

* spam
* viruses
* data storage issues
* data protection issues

Unsolicited email can easily overwhelm your email system unless you install a firewall and anti-spam software. Viruses can spread through email attachments or links, and other internet and email security issues may arise, especially if you're using the cloud or remote access. Electronic storing space can also become a problem, particularly where emails with large attachments are widely distributed.

The less formal nature of email can lead to careless or even libelous remarks being made which can damage your business. Equally risky is sending emails by mistake, where an email can go to the wrong person accidentally, potentially leaking confidential data and sensitive business information.

To minimize these risks, you should create and implement an email and internet acceptable use policy for your business and take steps to minimize the likelihood of business data breach and theft.

**Application :**

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.

**Conclusion :**

I look forward to speaking with you at [date and time]. I've sent over [materials you discussed]. Please review and let me know if you have any questions. Thank you again for meeting with me today.

**Future Scope :**

The future of email marketing is bright. It's still a viable form of promotion, and it's still arguably the best tool out there for delivering relevant content. The future of email marketing is going to be focused on delivering targeted content through segmentation…..

**Features of email :**

* automatic reply to messages.
* auto-forward and redirection of messages.
* facility to send copies of a message to many people.
* automatic filing and retrieval of messages.
* addresses can be stored in an address book and retrieved instantly.
* notification if a message cannot be delivered.

**8. Appendix:**

**Source code :**

**https://github.com/varshan03/A-Flexible-Email-Client-App/tree/main/A%20Flexible%20Email%20Client%20App**

**code :-**

**Email.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

}

}

}

}

**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.**

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

}

}

}

}

**EmaildatabaseHelper.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

}

}

**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.\*

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)

}

**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}")

}

}

}

}

}

}

THANK YOU