Adaptive Mail
A Flexible Email Client App

Project presented by

Team ID: NM2023TMID09333

Team leader: PRABAVATHY.S

Team member

PRAVEENA.B

PRIYANGA. M

RAMYA.A

1. INTRODUCTION:

1.1Overview

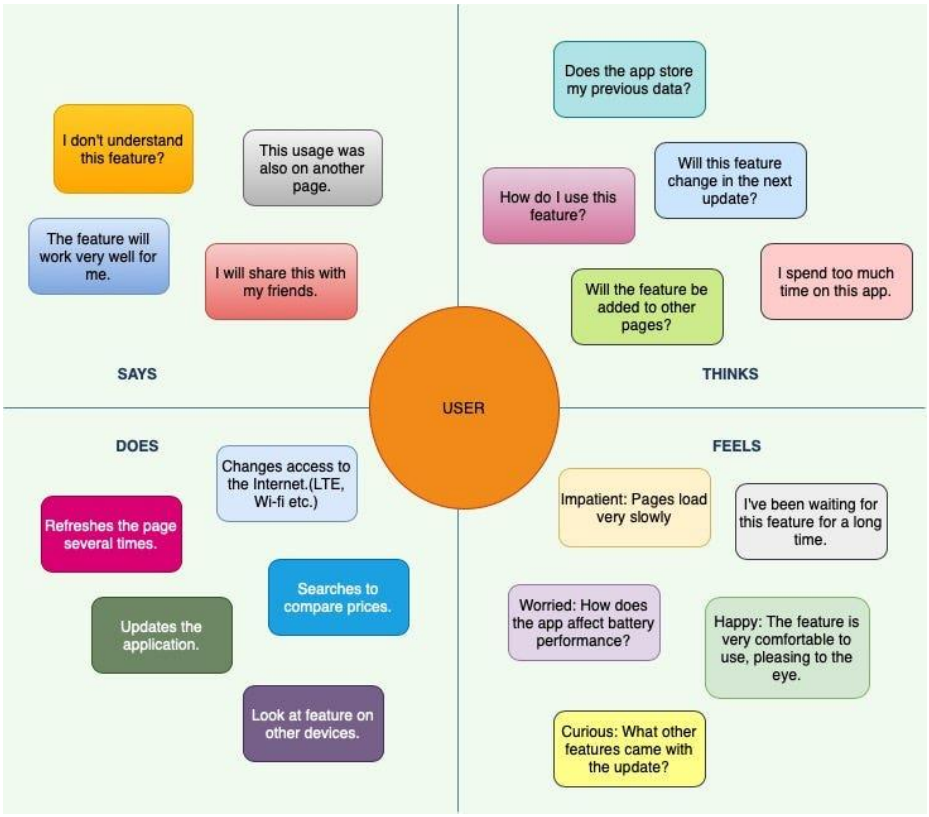
Adaptive Mail app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a conversational UI. The app simulates a messaging interface, allowing the user to send and receive messages, and view a history of previous messages. It showcases some of the key features of the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

1.2 Purpose

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.

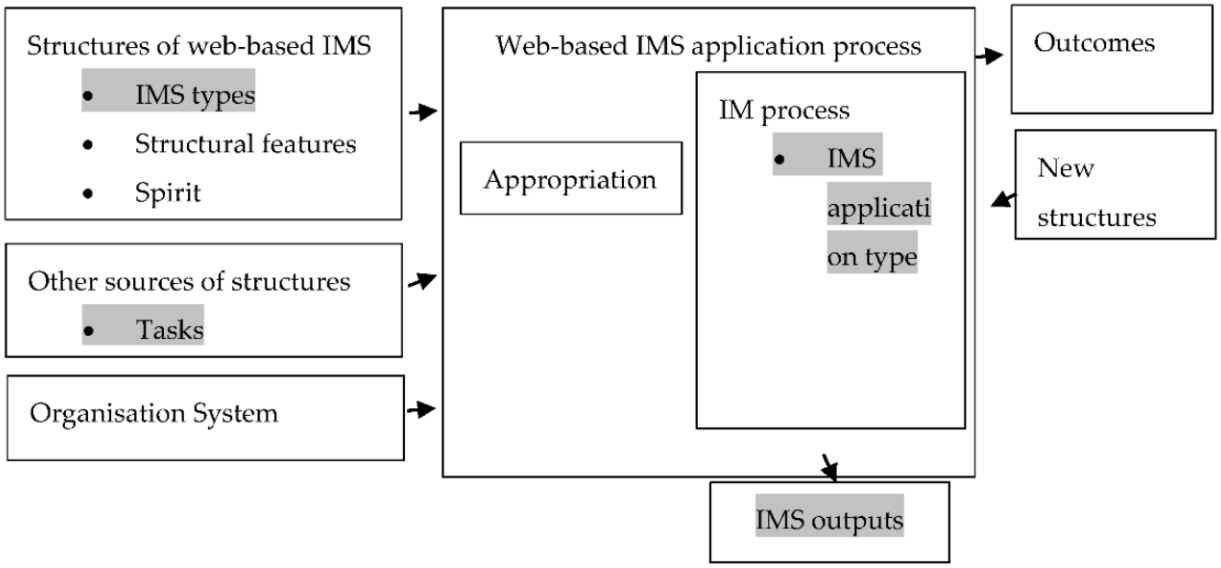
2. Problem Definition & Design Thinking

2.1 Empathy Map

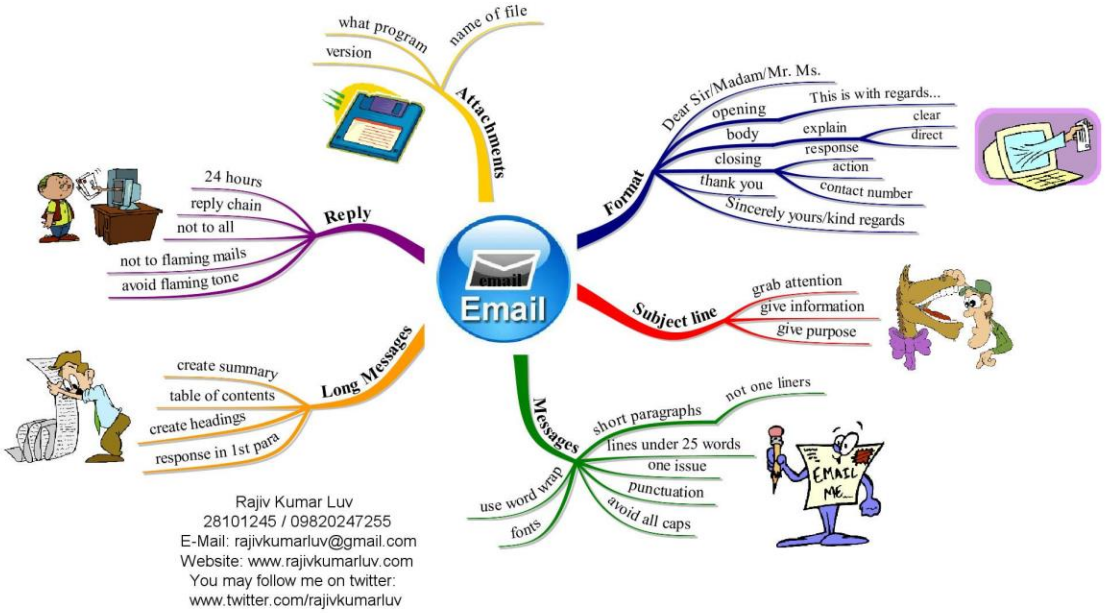


2.2 Ideation and Brainstorming Map

Ideation Map

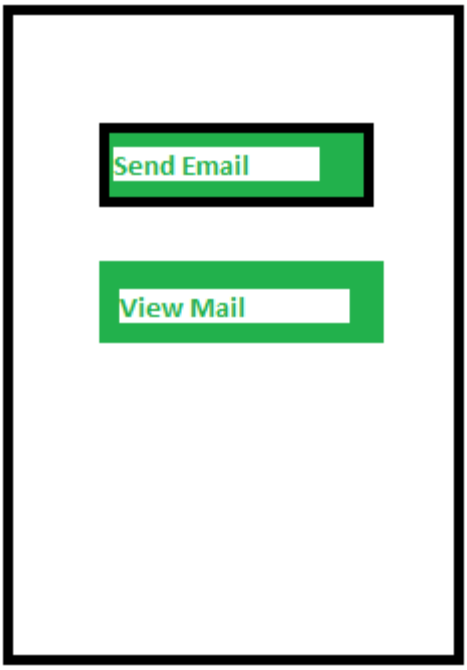


2.2Brain storming


Email
Attachments
what program
version
name of file
Format
Dear Sir/Madam/Mr. Ms.
opening
This is with regards...
body
explain
clear
direct
response
closing
action
thank you
contact number
Sincerely yours/kind regards
Reply
24 hours
reply chain
not to all
not to flaming mails
avoid flaming tone
Subject line
grab attention
give information
give purpose
Long Messages
create summary
table of contents
create headings
response in 1st para
Messages
short paragraphs
not one liners
lines under 25 words
one issue
punctuation
avoid all caps
use word wrap
fonts
EMAIL ME...
Rajiv Kumar Luv
28101245 / 09820247255
E-Mail: rajivkumarluv@gmail.com
Website: www.rajivkumarluv.com
You may follow me on twitter:
www.twitter.com/rajivkumarluv

3.Result:


Register
username
Email
Password
Have an account?
Login



4.Advantages and Disadvantages:

Advantages

- Fast Client Communication
- Availability and Portability
- Reduces Shipping and Mailing Costs

Disadvantages

- Vulnerability to Loss
- Disadvantage: Accessible to Others

5.Application

Most email services provide a webmail app as an interface for managing email. But you can only manage emails associated with this email service. For example, Gmail's webmail allows you to manage Gmail accounts only.

This email provider can be a third party, or a site provided by your ISP. You can access email features, and the website provider provides extensions.

6.Conclusion:

It's not hard to optimize emails for mobile email clients, and the result is worth it! Even while we still use table layouts and inline stylesheets for emails in 2011 it feels like a fresh breeze to drop in some media queries and advanced stylesheets for the clients that can support them.

7.Future scope:

The future of email looks bright, it's an essential tool for external and internal communication in 2021, about 319.6 billion emails were sent and received daily.

8.Appendix:

Source code

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:tools="http://schemas.android.com/tools" >

    <application
         android:allowBackup="true"
         android:dataExtractionRules="@xml/data_extraction_rules"
         android:fullBackupContent="@xml/backup_rules"
         android:icon="@mipmap/ic_launcher"
         android:label="@string/app_name"
         android:supportsRtl="true"
         android:theme="@style/Theme.EmailApplication"
         tools:targetApi="31" >
         <activity
             android:name=".RegisterActivity"
             android:exported="false"
             android:label="@string/title_activity_register"
             android:theme="@style/Theme.EmailApplication" />
         <activity
             android:name=".MainActivity"
             android:exported="false"
             android:label="MainActivity"
             android:theme="@style/Theme.EmailApplication" />
         <activity
             android:name=".ViewMailActivity"
             android:exported="false"
             android:label="@string/title_activity_view_mail"
             android:theme="@style/Theme.EmailApplication" />
         <activity
             android:name=".SendMailActivity"
             android:exported="false"
             android:label="@string/title_activity_send_mail"
             android:theme="@style/Theme.EmailApplication" />
         <activity
             android:name=".LoginActivity"
             android:exported="true"
             android:label="@string/app_name"
```

```
            android:theme="@style/Theme.EmailApplication" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

UI THEME

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?,

)
```

Color.kt

```
package com.example.emailapplication.ui.theme
```

```
import androidx.compose.ui.graphics.Color


val Purple200 = Color(0xFFBB86FC)

val Purple500 = Color(0xFF6200EE)

val Purple700 = Color(0xFF3700B3)

val Teal200 = Color(0xFF03DAC5)
```

Shape.kt

```
package com.example.emailapplication.ui.theme


import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.Shapes

import androidx.compose.ui.unit.dp


val Shapes = Shapes(

    small = RoundedCornerShape(4.dp),

    medium = RoundedCornerShape(4.dp),

    large = RoundedCornerShape(0.dp)

)
```

Theme.kt

```
package com.example.emailapplication.ui.theme


import androidx.compose.foundation.isSystemInDarkTheme
```

```
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200

    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
```

```
)


@Composable
fun EmailApplicationTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

Type.kt

```
package com.example.emailapplication.ui.theme


import androidx.compose.material.Typography
```

```
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp


// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
```

```
)
```

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
            }
        }
    }
}
```

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 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}")

                    }

                }

            }

        }


    }

}
```

ExampleInstrumentedTest.kt

```
package com.example.emailapplication


import androidx.test.platform.app.InstrumentationRegistry

import androidx.test.ext.junit.runners.AndroidJUnit4


import org.junit.Test

import org.junit.runner.RunWith


import org.junit.Assert.*
```

```
/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext =
InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.emailapplication", appContext.packageName)
    }
}
```

ExampleUnitTest.kt

```
package com.example.emailapplication


import org.junit.Test


import org.junit.Assert.*


/**
 * Example local unit test, which will execute on the development machine
(host).
```

```
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```