

EMAIL CLIENT APP

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.

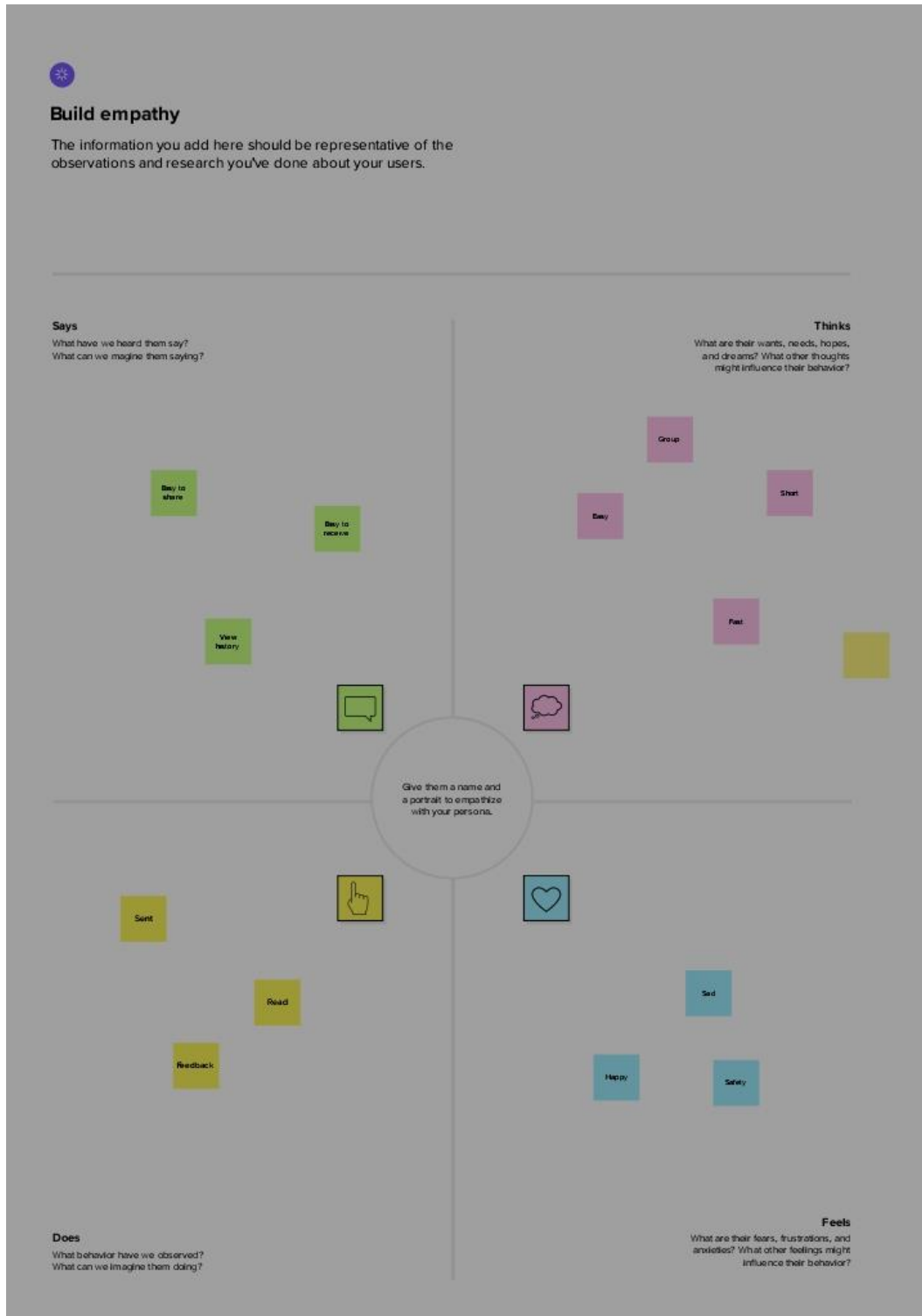
1.2 Purpose

Email is an open source, cross-platform email client. Mail is developed to provide a cross-platform alternative to email clients that already exists. Mail focuses on contemporary technologies as well as maintaining a user-friendly UI. There exist email clients which lack providing cross-platform software, smooth UI, modern design techniques, and trending technologies. Whereas a solution mail provides all such features. Mail is bundled with 2 classic dark and light themes along with a customized theme. Mail is lightweight, simple, OS friendly too. Mail provides a Rich Text Editing Experience to send mail. Only the IMAP protocol is supported in the beta version as many users don't seem to be using a POP protocol. Mail stores the messages as files under the user's home directory moved by the mail directory


Some features may be lacking as it is a beta version but the bundled features are assumed to work in all perspectives as long as a bug is detected. So feel free to propose changes by opening a new issue, or if you want to implement your own, feel free to fork this repo and create a pull request.

Problem Definition & Design Thinking

2.1 Empathy Map



RESULT



Login

[Sign up](#) [Forget password?](#)

Register Page :



Register

Register

Have an account? [Log in](#)

Main Page:

Home Screen



Send Email

View Emails

ViewMail Page:

View Mails

Receiver_Mail: kavya78@gmail.com

Subject: Android

Body: This is an Adaptive Email app

Receiver_Mail: shirishbokka7@gmail.com

Subject: Order

Body: your courier has arrived

ADVANTAGES

- Super fast .
- Light Weight.
- Cross platform email client that supports windows, MacOS, Linux.
- Specially Built as Desktop Application.
- Multiple Ways to Install the Application.
- New UI Design systems.
- Multiple User Account Switch.
- User customized Themes.
- Classical Themes: Dark and White.
- Multiple Layout of View Such as Grid View, and List View.
- A Better Card View and a Table View.
- Rich Text Editing Experience to Send Mail.
- General Settings to Configure APP Specs.
- Available in Five Different Languages.
- Download or Export Mail as CSV File.

DISADVANTAGS

- Emails are often used to send unwanted advertisements and other messages to the users.
- Emails can be intercepted by hackers.
- There is no personal touch.
- Internet access is required at all times.
- The emails inbox can become very crowded over a period.
- There is a chance of emails getting misinterpreted by the receivers.
- Viruses can be sent through emails.
- Email IDs can be hard to remember.
- Important emails might land up in spam or can be ignored.
- Not everyone has an email address.
- Emails scams are on the rise and one may easily be fooled by them.
- Some companies have strict email policies and don't allow attachments to be received.
- Need to check the inbox regularly to stay updated.

APPLICATION

Protect company mail and collaboration data

Centralised storage eases the backing up of important messaging records.

Stay in touch from anywhere

Check your emails at the office, from home, at the library, internet café or anywhere with an internet connected computer and recent web browser.

Have Email As You Want It

Choose from a variety of user themes, such as simple for optimized speed or lookout for outlook-like groupware collaboration.

Customize your content

Enter personalized signature, set up "out of office" messages. Compose messages in text or HTML, the later including images.

Automatically Organize Messages

Create unlimited personal folders and sorting filters so world client can automatically organize your messages.

Communicate in your language

World client uses Unicode fonts to support international languages.

Find lost messages

Search your messages by sender, recipient, subject and content.

Never miss a message

Let comagent inform you when new messages arrive in specified folders.

Chat securely online

Chat with other securely and privately using comagent's instant messaging.

Keep public and private contacts

Share some contact, keep other private.

Keep contacts current

Use comagent to keep synchronize your outlook or windows address books with your world client address book

CONCLUSION

At the end of the day, Emails is still one of the most effective forms of marketing. If you use it correctly, you can target prospective customers with relevant information, at precisely the right time gaining brand loyalists for life

APPENDIX

User.kt

```
import android.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?,
)
```

UserDae.kt

```
import android.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

```
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
            }
        }
    }
}

```

UserDatabase.kt

```

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

```

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"
    }
}

```

UserDatabaseDao.kt

```

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)
    }
    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()
    }

```

```

    }
    @SuppressWarnings("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
    }
    @SuppressWarnings("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
    }
}

```

Email.kt

```

import android.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 receiverMail:
String?,
    @ColumnInfo(name = "subject") val subject: String?,
    @ColumnInfo(name = "body") val body: String?,
)

```

EmailDao.kt

```

import android.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

```

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
            }
        }
    }
}

```

UserDatabaseHelper.kt

```

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()
    }

    @SuppressWarnings("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
    }
    @SuppressWarnings("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

```
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,
            onChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier.padding(10.dp)
                .width(280.dp)
        )
    }
}

```



```

        TextField(
            value = password,
            onChange = { 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

```

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,
            onChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier
                .padding(10.dp)

```

```

        .width(280.dp)
    )
    TextField(
        value = email,
        onChange = { email = it },
        label = { Text("Email") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )
    TextField(
        value = password,
        onChange = { 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"
        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)
}

```

MainActivity.kt

```

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 {
            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
        )
    }
}
}

```

SendMailActivity.kt

```

import android.annotation.SuppressLint
import android.content.Context

```

[illegible]

```

                textAlign = TextAlign.Center,
            )
        }
    )
}
) {
    openEmailer(this, databaseHelper)
}
}
}
}
}

```

```

}

```

```

@Composable

```

```

fun openEmailer(context: Context, databaseHelper:

```

```

EmailDatabaseHelper) {

```

```

    var recevierMail by remember {mutableStateOf("") }

```

```

    var subject by remember {mutableStateOf("") }

```

```

    var body by remember {mutableStateOf("") }

```

```

    var error by remember { mutableStateOf("") }

```

```

    val ctx = LocalContext.current

```

```

    Column(

```

```

        modifier = Modifier

```

```

            .fillMaxSize()

```

```

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

```

```

            horizontalAlignment = Alignment.Start

```

```

        ) {

```

```

            Text(text = "Receiver Email-Id",

```

```

                fontWeight = FontWeight.Bold,

```

```

                fontSize = 16.sp)

```

```

            TextField(

```

```

                value = recevierMail,

```

```

                onChange = { recevierMail = it },

```

```

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

```

```

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

```

```

                modifier = Modifier

```

```

                    .padding(16.dp)

```

```

        .fillMaxWidth(),
        textStyle = TextStyle(color = Color.Black, fontSize
= 15.sp),
        singleLine = true,
    )

    Spacer(modifier = Modifier.height(10.dp))
    Text(text = "Mail Subject",
        fontWeight = FontWeight.Bold,
        fontSize = 16.sp)
    TextField(
        value = subject,
        onValueChange = { subject = it },
        placeholder = { Text(text = "Subject") },
        modifier = Modifier
            .padding(16.dp)
            .fillMaxWidth(),
        textStyle = TextStyle(color = Color.Black, fontSize
= 15.sp),
        singleLine = true,
    )

    Spacer(modifier = Modifier.height(10.dp))
    Text(text = "Mail Body",
        fontWeight = FontWeight.Bold,
        fontSize = 16.sp)
    TextField(
        value = body,
        onValueChange = { body = it },

        placeholder = { Text(text = "Body") },
        modifier = Modifier
            .padding(16.dp)
            .fillMaxWidth(),
        textStyle = TextStyle(color = Color.Black, fontSize
= 15.sp),
        singleLine = true,
    )

```

```

        Spacer(modifier = Modifier.height(20.dp))
        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"
            }

            val i = Intent(Intent.ACTION_SEND)
            val emailAddress = arrayOf(recevierMail)
            i.putExtra(Intent.EXTRA_EMAIL, emailAddress)
            i.putExtra(Intent.EXTRA_SUBJECT, subject)
            i.putExtra(Intent.EXTRA_TEXT, body)
            i.setType("message/rfc822")
            ctx.startActivity(Intent.createChooser(i, "Choose an
Email client : "))
        },
        colors =
ButtonDefaults.buttonColors(backgroundColor =
Color(0xFFd3e5ef))
    ) {

        Text(
            text = "Send Email",
            modifier = Modifier.padding(10.dp),
            color = Color.Black,
            fontSize = 15.sp
        )
    }

```

```

        )
    }
}
}

```

ViewMailActivity.kt

```

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(
                topBar = {
                    TopAppBar(backgroundColor =
Color(0xFFadbef4), modifier = Modifier.height(80.dp),
                        title = {
                            Text(
                                text = "View Mails",
                                fontSize = 32.sp,
                                color = Color.Black,
                                modifier =
Modifier.fillMaxWidth(),
                                    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}")
    }
}
}
}
```

Andoridmanifest.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" />
    </application>
</manifest>
```



```

<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>

```