ADAPTIVE MAIL - FLEXIBLE EMAIL CLIENT APP

TEAM LEADER : NAVEENRAJ. B

TEAM ID : NM2023TMID14966

TEAM MEMBER: MANIKANDAN. R

TEAM MEMBER: NAVEEN KUMAR. V

TEAM MEMBER: NAVEEN. V

TEAM MEMBER: VASANTHKUMAR. S

1.INTRODUCTION:

A Software project comprising of a strongly built Email client that allows users to send Email address and at the same time allows to receive emails at the same time allows to receive Emails too. The project basically connects users existing account with the system. Thus user can send and receiving the message into the system's database. The protocol abolishes a reliable connection for transferring and receiving Emails.

OVERVIEW:

- Today, Emailing is the mopst basic need of communication for people and or organization. It’s secure system where user can access Emails in a more efficient way.
- The concept of putting mails into spam is according to the pre-defined keywords. If the mail consists of these keywords. Then the system puts theminto spam folder.

PURPOSE:

- ❖ The Email client, and the Email reader or more formally. Message user agent(MUA) or mail user agent is a computer program used to access and message user Email.

- ❖ It allows users to access their Emails on their computer without having to login via web.

2.PROBLEM DEFINITION &DESIGN THINKING:

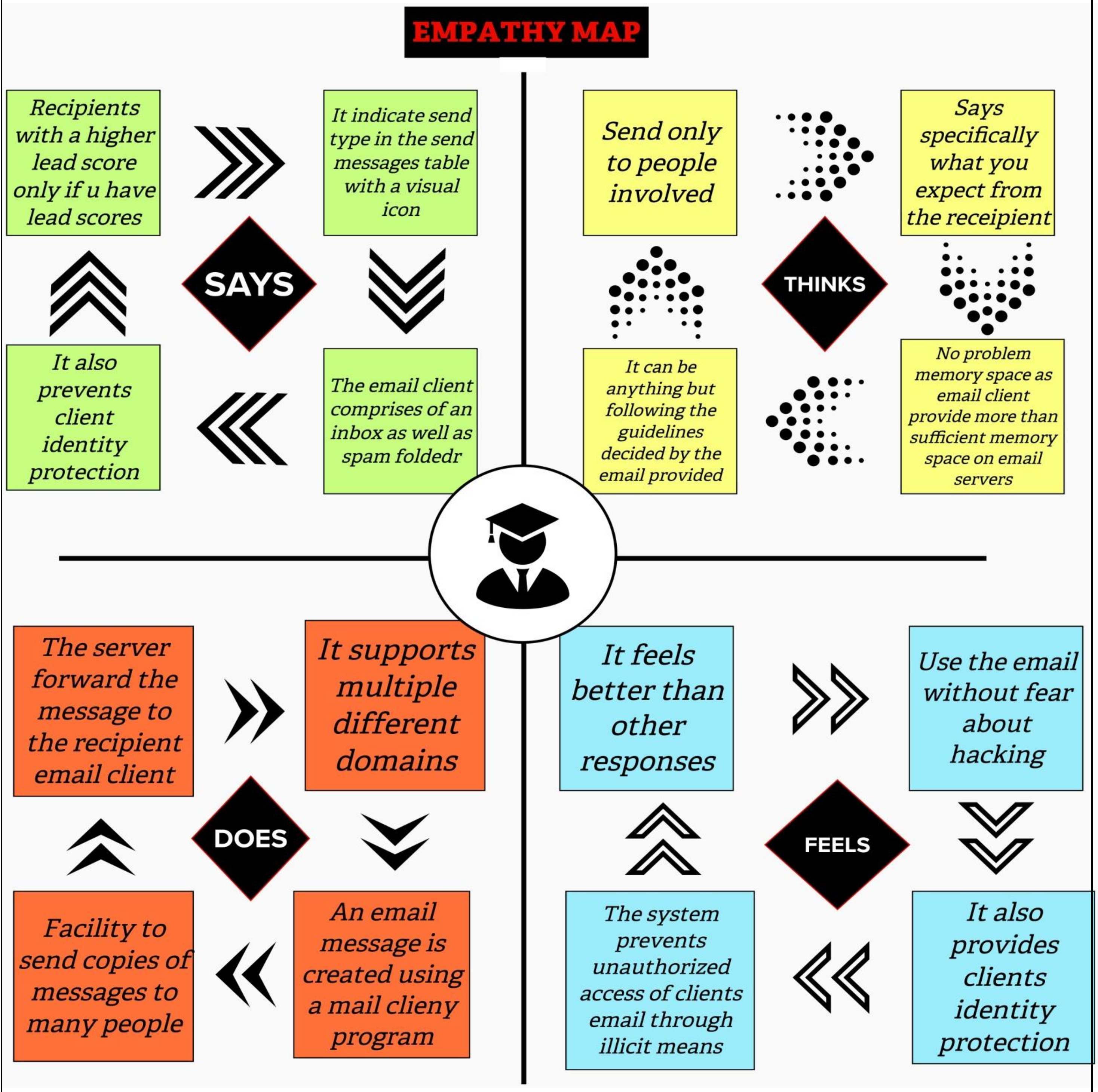

EMPATHY MAP
Recipients with a higher lead score only if u have lead scores
It indicate send type in the send messages table with a visual icon
SAYS
It also prevents client identity protection
The email client comprises of an inbox as well as spam foldedr
Send only to people involved
Says specifically what you expect from the receipient
THINKS
It can be anything but following the guidelines decided by the email provided
No problem memory space as email client provide more than sufficient memory space on email servers
The server forward the message to the recipient email client
It supports multiple different domains
DOES
Facility to send copies of messages to many people
An email message is created using a mail clieny program
It feels better than other responses
Use the email without fear about hacking
FEELS
The system prevents unauthorized access of clients email through illicit means
It also provides clients identity protection

FIGURE -1

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How to ask for help via Email ?

PROBLEM

Why getting too many Emails is such a problem.?

PROBLEM

How did i fix Email problems?

PROBLEM

What are the disadvantages were receiving or sending Email messages?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

NAVEEN.K

Verify your Email password

Some Emails cause upset or anger

It lost productivity because Email overload often leads to lost productivity

use a subject line to clearly express what your Emailis concerning especially if you dont know their receipient well

NAVEEN KUMAR.V

Verify your Email password

Too much Email can be also effect your focus and everytime a notification dings,it can distract

many peoples check Emails so it causes.

Determine the Email account type

NAVEEN RAJ.B

Too many people send to much information

Too many people send to much information

Verify your Email username

You should introduce yourself and show the value of your communication in the first sentance

MANIKANDAN.R

Fix misbehaving Email program or app

it makes misunderstanding and no respite

getting too many Emails can also causes excess stress and lead to brnout

The email seem more personal and set the tone for the content

VASANTH KUMAR.S

Determine the Email account type

word mistakes is an overflowing inbox can lead to more mistakes

it makes misunderstanding and no respite

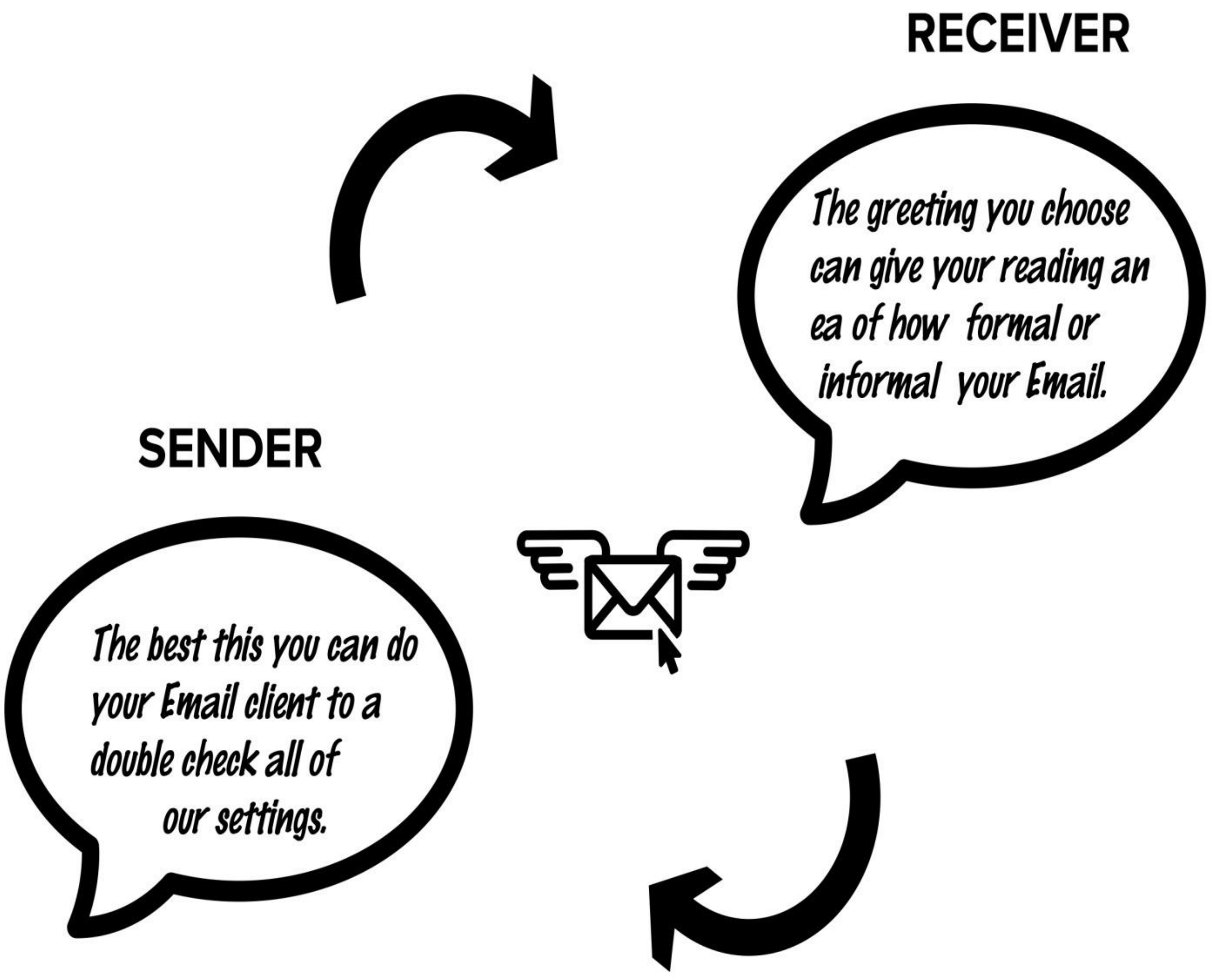
it makes misunderstanding and no respite

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

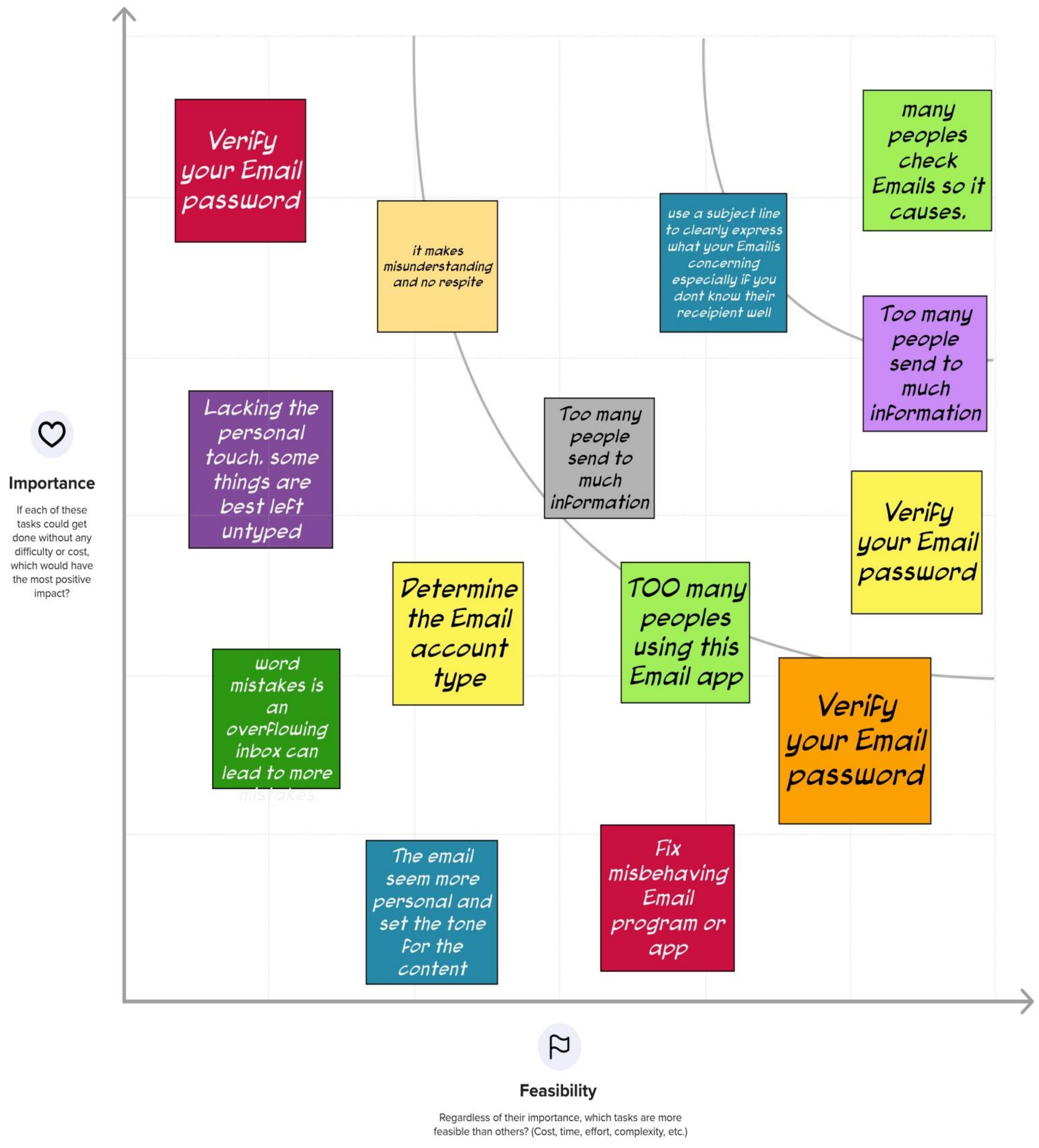

RECEIVER
The greeting you choose can give your reading an ea of how formal or informal your Email.
SENDER
The best this you can do your Email client to a double check all of our settings.

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

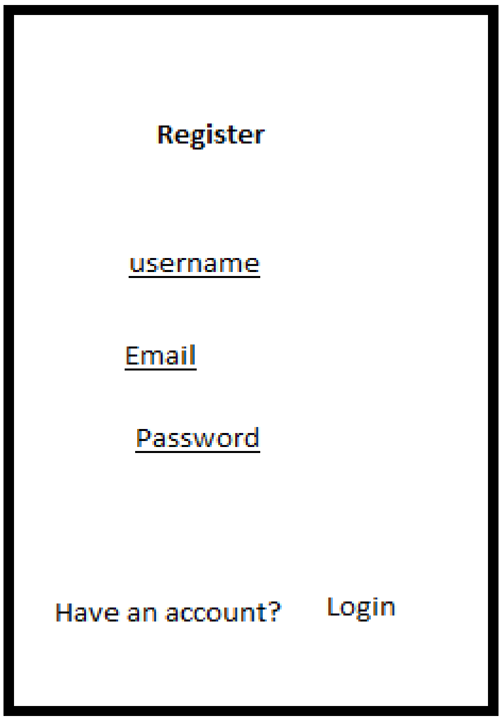
⏱ 20 minutes

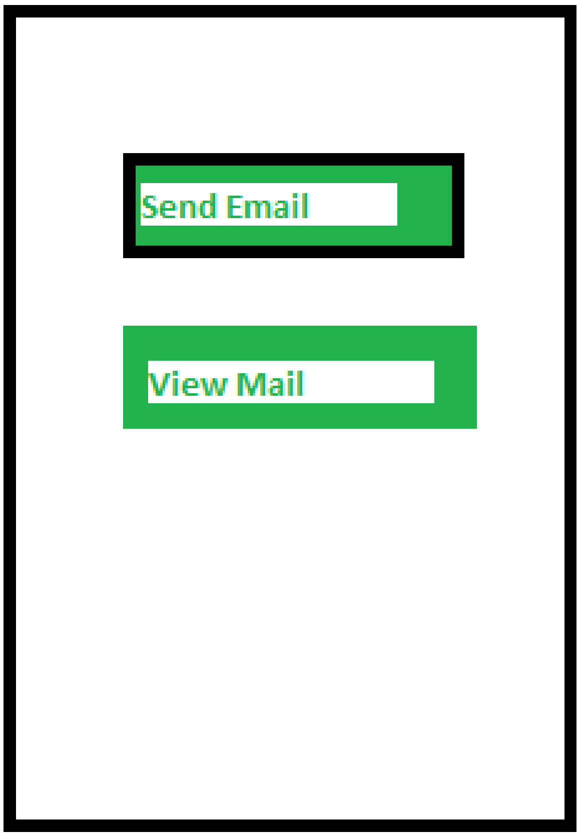


Verify your Email password
it makes misunderstanding and no respite
use a subject line to clearly express what your Emailis concerning especially if you dont know their receipient well
many peoples check Emails so it causes.
Too many people send to much information
Lacking the personal touch, some things are best left untyped
Too many people send to much information
Verify your Email password
Determine the Email account type
TOO many peoples using this Email app
word mistakes is an overflowing inbox can lead to more mistakes
Verify your Email password
The email seem more personal and set the tone for the content
Fix misbehaving Email program or app
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.RESULT:

<u>Sample Output:</u>





4.ADVANTAGES:

- **It allows provides clients identity protection.**
- **It prevents hacking.**
- **SMTP is a protocol used for sending Email and POP3 used for retrieving Emails.**

DISADVANTAGES:

- **Sometimes the system can consider the Email valid and put it in the inbox instead of putting in spam folder.**
- **It makes misunderstanding and sucks up yours time.**

5.APPLICATION:

- **The system can be used in any organization. Institutes for internal Emailing purpose.**
- **It can be used by any common man sending and receiving Emails and integrating the account with the system database.**
- **The system can also be implemented over the internet for public use**

6.Conclusion:

- A conclusion is an important part of the paper: It provides closure for the reader while reminding the reader of the contents and importance of the paper.

FUTURE SCOPE:

- Since most users tend to look At promotional Emails in the same light as spam. Future Email marketing campaigns should aim to be more personalized.

8.APPENDIX:

A.Source code:

<u>AndroidManifest.xml</u>

```
<?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
```

```kotlin
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

```kotlin
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

```kotlin
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

```kotlin
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

```kotlin
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

```kotlin
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

```kotlin
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("") }
```

```kotlin
    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)
                }

        }
    }
}
```

```kotlin
@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

```kotlin
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))
```

```kotlin
        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

```kotlin
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))
```

```kotlin
        ) {
            // 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

```kotlin
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

```kotlin
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

```kotlin
package com.example.emailapplication
```

```kotlin
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

```kotlin
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)
    }
}
```

---THE END---