Adaptive Mail : A Flexible Email Client App

Abstract

The flexible email client app for Android aims to redefine email management through a user-centric, customizable platform. Designed to support the basic send and receive operations of mails and offline access. With a focus on security, the app uses SSL encryption and offers passkey authentication. Its customizable interface allows users to tailor themes, layouts, and notifications to their needs, distinguishing it from existing market solutions. By combining robust features with intuitive navigation, this app is set to enhance productivity and offer a seamless, secure email experience for users across various segments.

Introduction

The Flexible Email Client App is designed to provide users with a simple, efficient, and user-friendly email management solution. The app offers features and functionality to cater to both personal and professional email needs. 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 register with their information and create a passcode, login to the app whenever necessary and send and receive messages. It showcases some of the key features of the Compose UI toolkit, data management, and user interactions. The flexible email client app aims to revolutionize how users interact with their. This Android app targets individual users, professionals, and businesses seeking a versatile and efficient email management solution.

Key Features:

1. **Multi-Account Support:**

Seamlessly manage multiple email accounts (Gmail, Outlook, Yahoo, etc.) from within a single interface.

Easily toggle between accounts and unify inboxes for a consolidated view.

2.Customizable Interface:

Choose from different themes (light/dark mode, custom color schemes) and layouts (classic, minimalist, or modern).

Personalize the home screen to display what matters most to you: inbox, calendar, contacts, or tasks.

3.Smart Inbox Management:

Use AI-powered filters to categorize and prioritize emails, automatically sorting them into groups like “Important”, “Social”, “Promotions”, or “Newsletters”.

Set rules for automatically tagging, archiving, or flagging certain types of emails.

Project description

Core Features:

1.Multi-Account Support:

Adaptive Mail allows users to manage multiple email accounts from various providers (e.g., Gmail, Outlook, Yahoo, etc.) in a single, unified inbox.

Users can quickly switch between accounts, view all emails in one place, or filter messages by account for a more focused experience.

2.Customizable Interface:

The app features a highly customizable interface with options to select between different themes (light or dark modes), color schemes, and layout styles.

Users can configure the home screen to prioritize the features that are most important to them, such as the inbox, calendar, tasks, or contacts.

3.Smart Email Organization:

Advanced AI algorithms automatically categorize and prioritize emails, sorting them into useful groups like “Important,” “Social,” “Promotions,” or “Newsletters.”

Email management rules allow for automatic tagging, archiving, or

flagging of specific messages, making inbox organization effortless.

4.Advanced Search & Sorting:

The search functionality within Adaptive Mail goes beyond simple keyword matching, allowing users to filter results based on date ranges, attachments, senders, and other criteria.

Emails can be sorted by date, sender, or importance, ensuring quick access to the most relevant messages

System requirements:

- Operating System: Android-7 or later
- Ram: 1Gb (Minimum)
- Rom:20Mb (Minimum)
- Internet connection

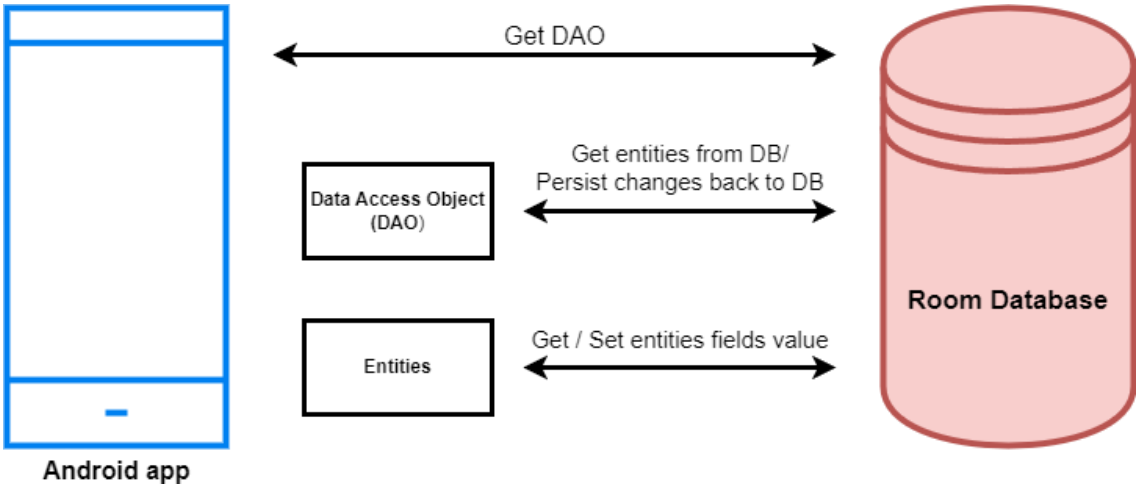
Tools Used:

- Android Studio
- Firebase
- Kotlin

Objectives

- To create an intuitive user interface that enhances user experience.
- To incorporate robust security features to protect user data.
- To provide seamless synchronization across multiple devices.
- To provide a user-friendly interface that offers intuitive navigation for all age groups.

Architecture



Features

- **Multiple Account Support**: Users can manage multiple email accounts within a single app.
- **Customizable Interface**: Users can personalize the app's appearance and layout according to their preferences.
- **Smart Filters**: Automated email categorization based on user-defined rules.
- **Push Notifications**: Real-time notifications for new emails with customizable alert settings.

Development Process

- **Planning**: Identifying user requirements and market research.
- **Design**: Creating wireframes and prototypes for the app interface.
- **Development**: Implementing the app using Android Studio and relevant programming languages.
- **Testing**: Conducting rigorous testing to ensure functionality, performance, and security.
- **Deployment**: Showcasing the app to the target audience.

User Interface (UI) and User Experience (UX)

- **Intuitive Navigation**: Simple and minimalistic design for easy navigation and accessibility.
- **Catchy Theme**:Attract the user attension.
- **Customizable Widgets**: Home screen widgets for quick access to emails functions.

Technologies Used

- **Programming Languages**: Kotlin, Java
- **Frameworks**: Android SDK, Retrofit, Room Database
- **Security**: SSL/TLS encryption, OAuth 2.0
- **Testing Tools**: JUnit, Espresso, Firebase Test Lab

Program

Login Activity

```
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
                    )
                )
            }

        } 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)
}
```

Main Activity

```
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 {
            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
        )
    }


  }
}
```

Register Activity

```
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"
            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)
}
```

Send Mail Activity

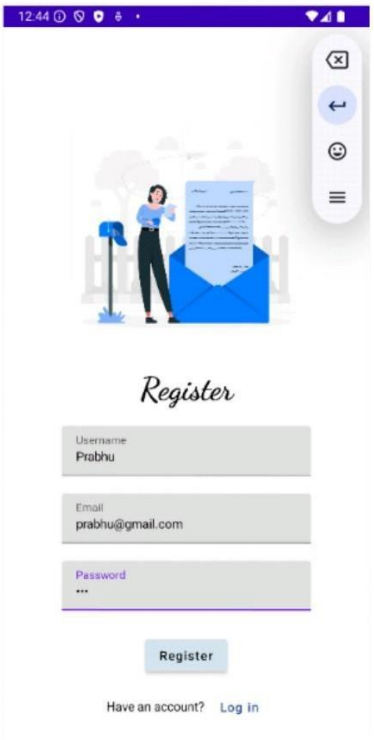
```
package com.example.emailapplication

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(
                topBar = {
                    TopAppBar(backgroundColor = Color(0xFFadbef4), modifier =
Modifier.height(80.dp),
                        title = {

                            Text(
```

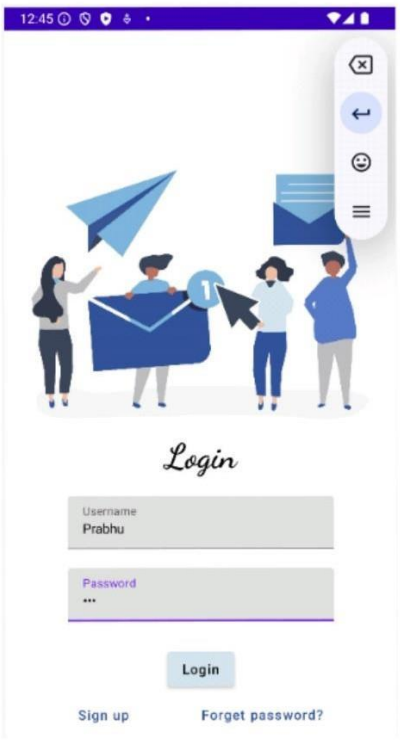
```
                    text = "Send Mail",
                    fontSize = 32.sp,
                    color = Color.Black,

                    modifier = Modifier.fillMaxWidth(),

                    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,
      onValueChange = { recevierMail = it },
      label = { Text(text = "Email address") },
      placeholder = { Text(text = "abc@gmail.com") },

      modifier = Modifier
        .padding(16.dp)
```

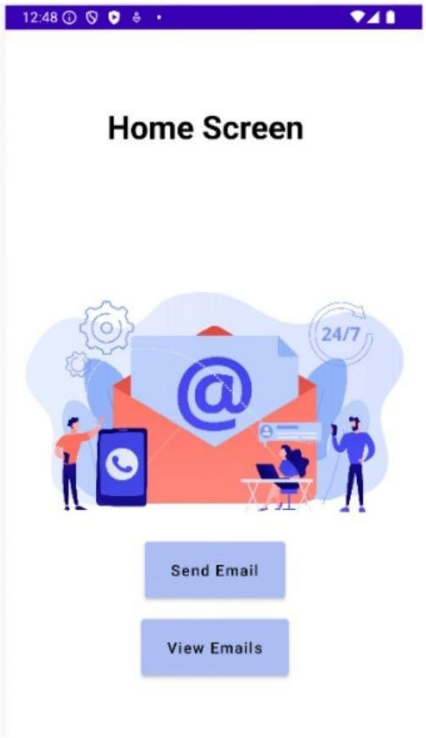
```
        .fillMaxWidth(),
    textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),
    singleLine = true,
)

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(
```

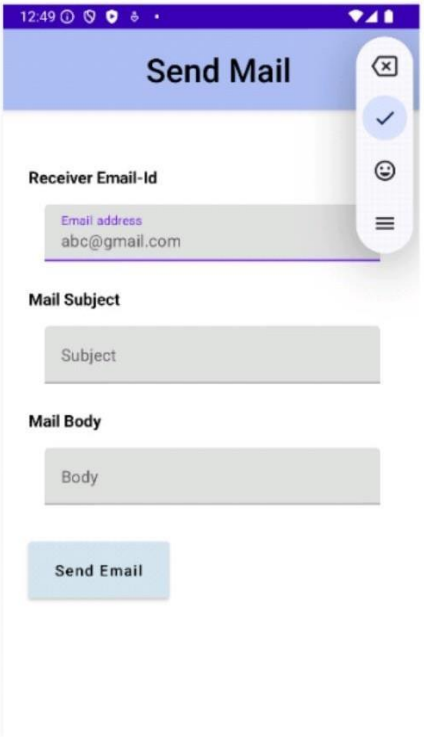
```
                id = null,
                recevierMail = recevierMail,
                subject = subject,
                body = body

            )
            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
        )
    }
  }
}
```

Output


12:44
Register
Username
Prabhu
Email
prabhu@gmail.com
Password
Register
Have an account? Log in


12:45
Login
Username
Prabhu
Password
Login
Sign up
Forget password?


12:48
Home Screen
24/7
Send Email
View Emails


12:49
Send Mail
Receiver Email-Id
Email address
abc@gmail.com
Mail Subject
Subject
Mail Body
Body
Send Email

Challenges and Solutions

- **User Interface Design**: Ensuring the interface is both aesthetically pleasing and easy to navigate.

  Solution: Conducting user testing and iterative design improvements.

- **Security**: Protecting user data from breaches.

  Solution: Implementing industry-standard encryption and authentication methods.

- **Performance**: Maintaining app performance with growing features.

  Solution: Optimizing code and using efficient algorithms.

Future Enhancements

- **AI-Powered Features**: Incorporating AI for predictive email sorting and responses.
- **Cross-Platform Support**: Expanding the app to iOS and web platforms.
- **Enhanced Collaboration**: Integrating with more productivity and collaboration tools.

Conclusion

The Flexible Email Client App aims to revolutionize the way users manage their emails by providing a comprehensive, secure, and customizable platform. With its advanced features and user-centric design, the app is poised to become an essential tool for efficient email management. This flexible email client app offers a modern, user-centric approach to email management. With basic options, features, and security, it can used by a wide range of users, ultimately transforming the way they manage their emails.