1-INTRODUCTION:

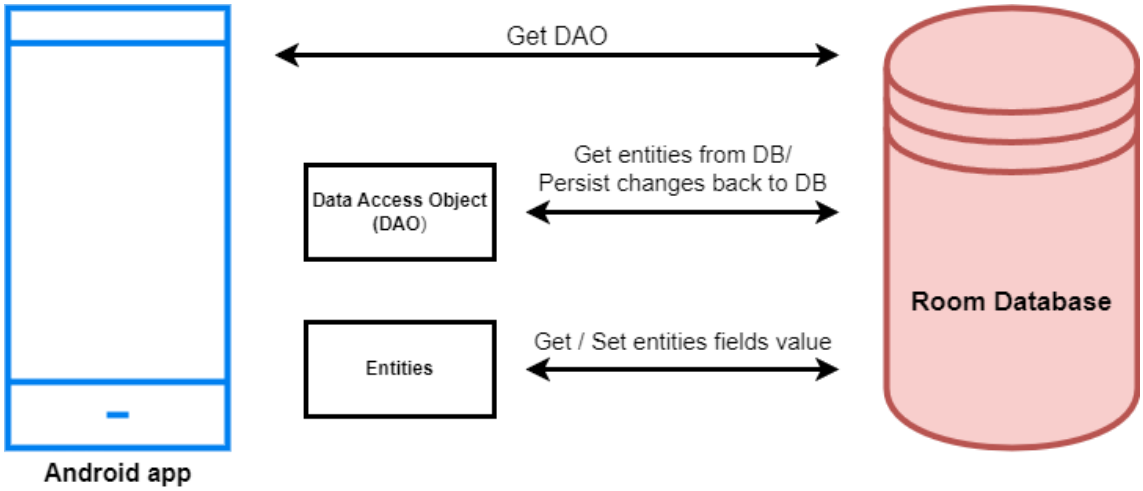
1.1 OVERVIEW

Adaptive Mail: A Flexible Email Client App

Project Description:

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.

Architecture



Learning Outcomes :

By end of this project:

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.

Project Workflow:

- Users register into the application.
- After registration , user logins into the application.

- User enters into the main page
- User can View previously sent emails.
- User can give subject and email body to send email.

Note:

To complete the project you need to finish up the tasks listed below:

Tasks:

1.Required initial steps
2.Creating a new project.
3.Adding required dependencies.
4.Creating the database classes.
5.Building application UI and connecting to database.
6.Using AndroidManifest.xml
7.Running the application.

1.2 PURPOSE

With a flexible email client app, users can manage their emails more efficiently. This may include features like automated email categorization, intelligent search functions, and integration with other productivity tools like calendars and task managers.

2-PROBLEM DEFINATION & DESIGN THINKING

2.1 EMPATHY MAP

Template

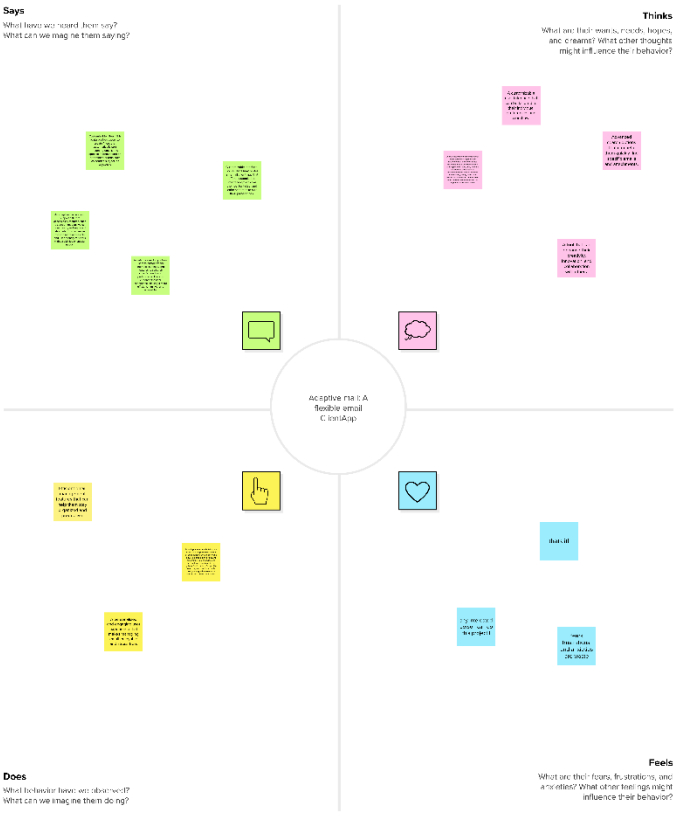
Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share template feedback

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

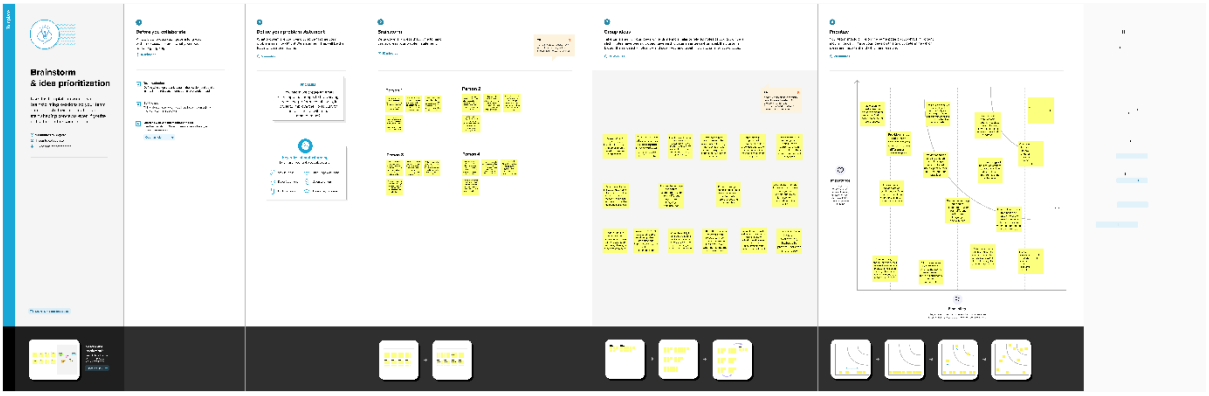


Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

2.2-Ideation and Brainstorming Map



3-RESULT

Login

Username

Password

Login

Sign up

Forget password?

Register

Username

Email

Password

Register

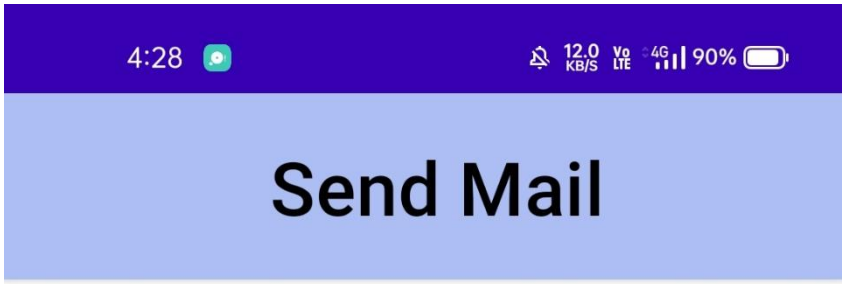
Have an account? Log in

Home Screen

Send Email

View Emails

Receiver_Mail: fayasmohammed336@gmail.com
Subject: fayas
Body: hsgs



Receiver Email-Id

Email address

Mail Subject

Subject

Mail Body

Body

Send Email

4-Advantage and disadvantage:

Advantages of a flexible email client app:

Customizability: Users can customize the app to their specific needs, preferences, and workflow.

Improved productivity: Features like automated email categorization, intelligent search functions, and integration with other productivity tools can help users manage their emails more efficiently.

Accessibility: The app can be accessed from multiple devices and platforms, making it easier for users to stay connected and productive on-the-go.

Security: Advanced security features like two-factor authentication, email encryption, and spam filtering can help protect users from email-related threats.

Disadvantages of a flexible email client app:

Complexity: With a flexible email client app, there may be a learning curve to understand and utilize all the features and customization options.

Maintenance: Customizing and updating the app may require technical knowledge or support, which can be time-consuming and expensive.

Compatibility issues: As the app is designed to work across multiple devices and platforms, there may be compatibility issues with certain devices or software systems.

Cost: A flexible email client app may require a subscription or purchase fee, which may not be feasible for some users or organizations.

5-Application:

Business: A flexible email client app can be used by businesses of all sizes to manage their email communications more efficiently and securely. Customizable features like automated email categorization and integration with other productivity

tools can help streamline workflows and improve productivity.

Education: A flexible email client app can be used by schools and universities to communicate with students and faculty, manage class schedules and assignments, and collaborate on group projects.

Healthcare: A flexible email client app can be used by healthcare providers to communicate with patients and other healthcare professionals securely, manage patient records, and schedule appointments.

Non-profits: A flexible email client app can be used by non-profit organizations to manage donor communications, fundraising campaigns, and volunteer coordination.

Government: A flexible email client app can be used by government agencies to communicate with citizens, manage public records, and coordinate public services.

6-Conclution:

In conclusion, a flexible email client app offers a wide range of benefits for individuals, businesses, and organizations of all sizes. With customizable features, improved productivity, accessibility, and advanced security options, a flexible email client app can help users manage their email communications more efficiently, securely, and effectively.

7-Future Scope:

Artificial intelligence integration: The integration of artificial intelligence (AI) in a flexible email client app could improve the app's capabilities for automated email categorization, prioritization, and response suggestions based on the user's habits and preferences.

Enhanced collaboration: The app's collaboration features could be improved to enable real-time collaboration on emails, documents, and other files.

Augmented reality integration: Augmented reality (AR) integration could allow users to visualize and interact

with their emails and attachments in a more immersive and intuitive way.

Blockchain integration: The integration of blockchain technology could improve the app's security features, ensuring the authenticity and privacy of email communications.

Virtual assistant integration: The integration of virtual assistant technology, such as voice assistants, could enable hands-free email management and response.

8-appendix:

A. source code:

MainActivity.kt

```
package com.example.emailapplication

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
```

```
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import androidx.core.content.ContextCompat.startActivity
import com.example.emailapplication.ui.theme.EmailApplicationTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier =
Modifier.fillMaxSize().background(Color.White),
                ) {
                    Email(this)
                }

        }
    }
}

@Composable
fun Email(context: Context) {
    Text(
        text = "Home Screen",
        modifier = Modifier.padding(top = 74.dp, start = 100.dp, bottom =
24.dp),
        color = Color.Black,
        fontWeight = FontWeight.Bold,
        fontSize = 32.sp
    )

    Column(
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Image(
            painterResource(id = R.drawable.home_screen),
contentDescription = ""
        )


        Button(onClick = {
            context.startActivity(
                Intent(
                    context,
                    SendMailActivity::class.java
                )
            )
        },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFFadbef4))
        ) {
            Text(
                text = "Send Email",
                modifier = Modifier.padding(10.dp),
                color = Color.Black,
                fontSize = 15.sp
            )
        }
```

```
        Spacer(modifier = Modifier.height(20.dp))

        Button(onClick = {
            context.startActivity(
                Intent(
                    context,
                    ViewMailActivity::class.java
                )
            )
        },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFFadbef4))
        ) {
            Text(
                text = "View Emails",
                modifier = Modifier.padding(10.dp),
                color = Color.Black,
                fontSize = 15.sp
            )
        }


    }
}
```

LoginActivity.kt

```
package com.example.emailapplication

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.emailapplication.ui.theme.EmailApplicationTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        databaseHelper = UserDatabaseHelper(this)
        setContent {

            LoginScreen(this, databaseHelper)
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {


    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Image(
            painterResource(id = R.drawable.email_login),
contentDescription = ""
        )


        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            text = "Login"
        )
        Spacer(modifier = Modifier.height(10.dp))

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

        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.padding(10.dp)
                .width(280.dp)
        )

        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
```

```
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user = databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainActivity::class.java
                            )
                        )
                        //onLoginSuccess()
                    }

                } else {
                    error = "Please fill all fields"
                }
            },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFFd3e5ef)),
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    RegisterActivity::class.java
                )
            )}
            )
            { Text(color = Color(0xFF31539a),text = "Sign up") }
            TextButton(onClick = {
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color(0xFF31539a),text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```