

**ANDROID STUDIO – EXPERIENCE BASED PROJECTLEARNING**

**Adaptive Mail: A flexible email client app**

Submitted by

RAGHAVI.R - 711022104039

SHARON RUFINA.B - 711022104047

SUJI.N - 711022104054

SIKHAMOL.S - 711022104048

BACHELOR OF COMPUTER SCIENCE AND ENGINEERING

IN

FIFTH SEMESTER

COMPUTER SCIENCE AND ENGINEERING

INFO INSTITUTE OF ENGINEERING, COIMBATORE – 641107

NOVEMBER/DECEMBER - 2024

**BONAFIDE CERTIFICATE**

Certified that this project “Adaptive Mail: A flexible email client app”is the Bonafide work of RAGHAVI R(711022104039),SUJI N(711022104054),SHARON RUFINA B (711022104047), SIKHAMOL(711022104048), who carried out the project work under any supervision.

**SIGNATURE**

**STAFF COORDINATOR**

**Mrs. A. SARANYA M.E.,**

**ASSISTANT PROFESSOR**

DEPT. COMPUTER SCIENCE AND ENGINEERING

INFO INSTITUTE OF ENGINEERING,

KOVILPALAYAM COIMBATORE - 641107

**SIGNATURE**

**HEAD OF THE DEPARTMENT**

**Dr. G. SELVAVINAYAGAM Ph.D.,**

**HEAD OF THE DEPARTMENT**

DEPT. COMPUTER SCIENCE AND ENGINEERING

INFO INSTITUTE OF ENGINEERING,

KOVILPALAYAM COIMBATORE - 641107

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We sincerely thank to Tamil Nadu Skill Development Corporation (TNSDC), Naan Mudhalvan" Platform and ANDROID STUDIO – EXPERIENCE BASED PROJECT LEARNING(EBPL) for encouragement towards our project work for providing necessary skill training.

We sincerely thank our Principal Dr.N.KOTTISWARAN,M.E., Ph.D., and Head of the Department Dr. G. SELVAVINAYAGAM,M.E., Ph.D., and also Staff Coordinator Mrs. A. SARANYA M.Efor herencouragement towards our project works.

We also thank our project guide and our parents for the complete and whole hearted support, motivation guidance and help in making our project activities.

**ABSTRACT:**

This paper introduces an adaptive e-mail client application designed to enhance user productivity and streamline communication. The app leverages machine learning algorithms and user behavior analysis to provide personalized features such as intelligent email prioritization, context-based categorization, and adaptive user interface customization.

By integrating sentiment analysis and natural language processing, the client offers smart responses and proactive suggestions to improve email interactions.

Additional features include cross-platform compatibility, robust security measures, and seamless integration with productivity tools, catering to the diverse needs of both personal and professional users.

This adaptive e-mail client aims to redefine email management, delivering a more efficient and user-centric experience.

**INTRODUCTION:**

An adaptive email client app is a smart, dynamic application designed to revolutionize the way you manage your emails.

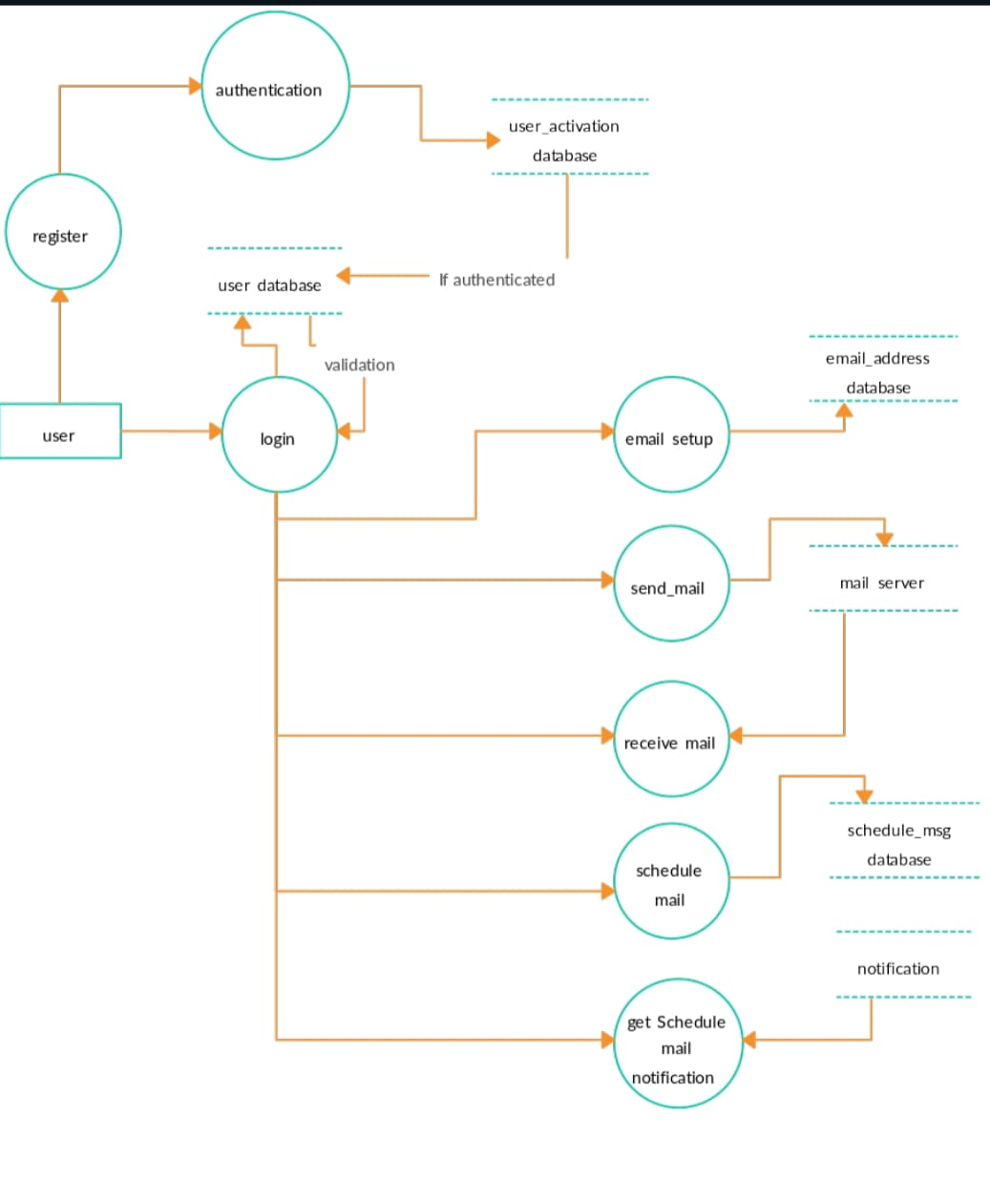
It leverages machine learning and advanced algorithms to provide a personalized experience tailored to your preferences and work habits.

Key features include Smart Inbox Management, Context-Aware Replies, Cross-Platform Synchronization, Customizable Interface, Security Features.

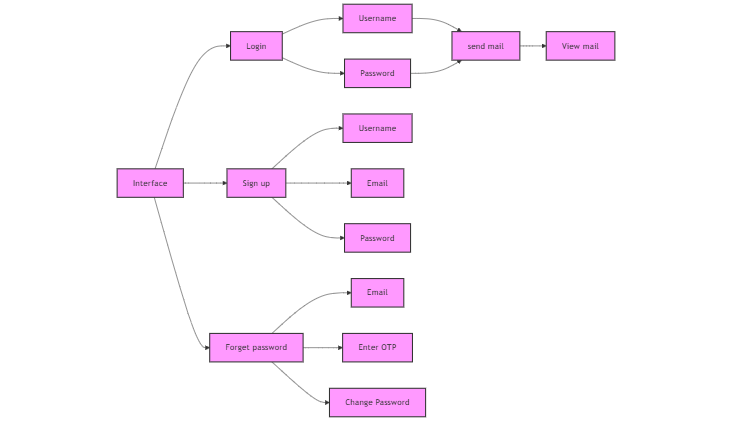
\*Smart Inbox Management\*: Automatically categorizes and prioritizes emails, ensuring that critical messages stand out while reducing clutter.

\*Context-Aware Replies\*: Suggests contextually relevant responses and allows for quick replies, saving time.

**DATA FLOW DIAGRAM:**



**USE CASE DIAGRAM:**



**SOFTWARE REQUIREMENT:**

**Front-End Technologies**:

UI Frameworks

Customizable UI Components

Animations

**Machine Learning and AI:**

Natural Language Processing

Behavioral Analytics

Recommendation Systems

**Backend & Database Technologies:**

Database Management

Storage

APIs for Email Retrieval

Push Notifications

**Security & Privacy:**

Authentication

Data Privacy

**PROGRAM CODE:**

**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?,

)

**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

)

}

}

}

**AndroidManifest.xml:**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools" >

<application

android:allowBackup="true"

android:dataExtractionRules="@xml/data\_extraction\_rules"

android:fullBackupContent="@xml/backup\_rules"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/Theme.EmailApplication"

tools:targetApi="31" >

<activity

android:name=".RegisterActivity"

android:exported="false"

android:label="@string/title\_activity\_register"

android:theme="@style/Theme.EmailApplication" />

<activity

android:name=".MainActivity"

android:exported="false"

android:label="MainActivity"

android:theme="@style/Theme.EmailApplication" />

<activity

android:name=".ViewMailActivity"

android:exported="false"

android:label="@string/title\_activity\_view\_mail"

android:theme="@style/Theme.EmailApplication" />

<activity

android:name=".SendMailActivity"

android:exported="false"

android:label="@string/title\_activity\_send\_mail"

android:theme="@style/Theme.EmailApplication" />

<activity

android:name=".LoginActivity"

android:exported="true"

android:label="@string/app\_name"

android:theme="@style/Theme.EmailApplication" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

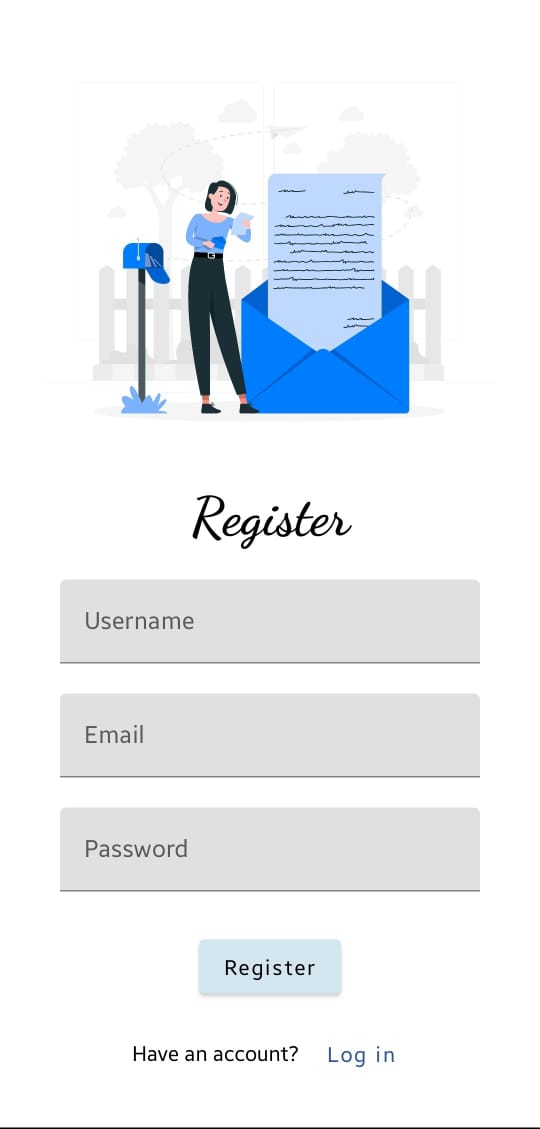
</intent-filter>

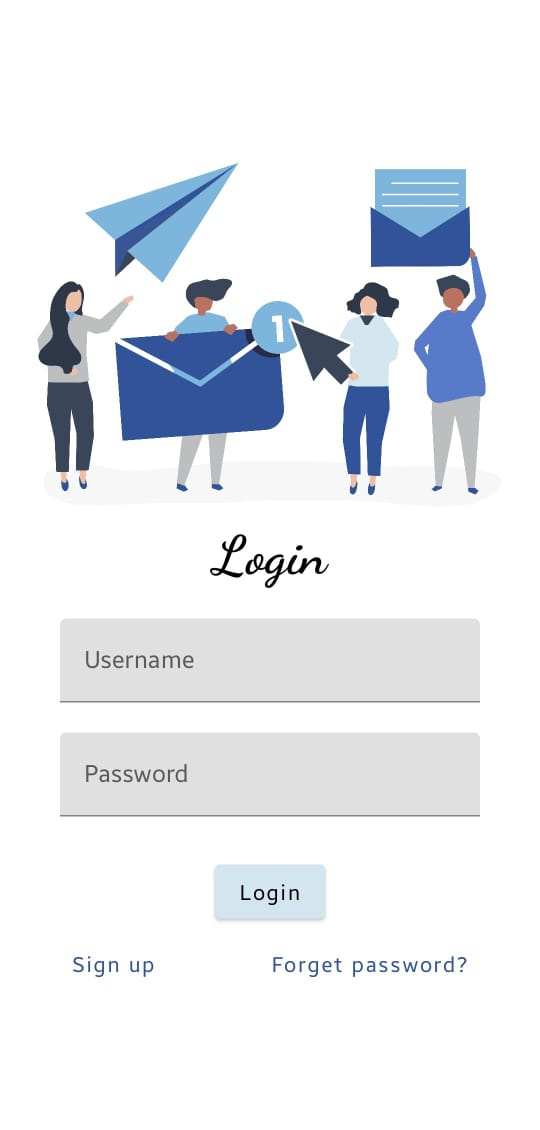
</activity>

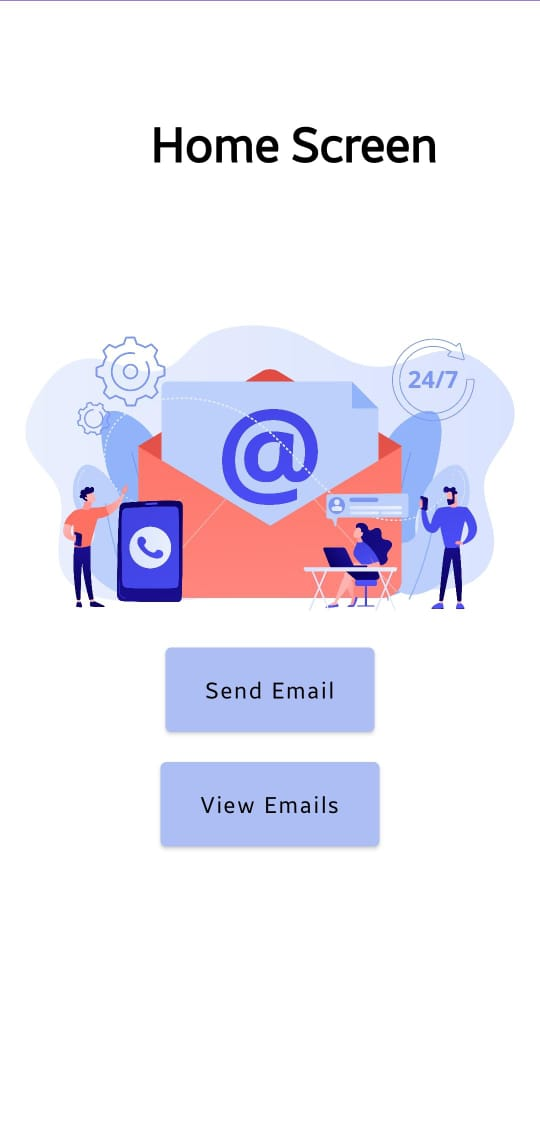
</application>

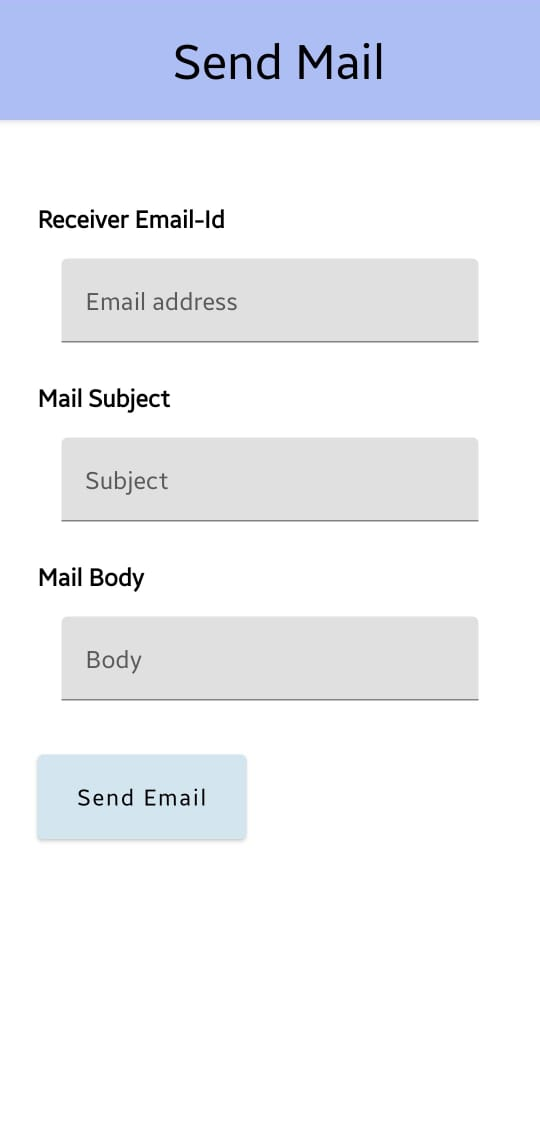
</manifest>

**OUTPUT:**

****

****

****

****

**CONCLUSION:**

An adaptive email client app represents the future of efficient and personalized email management.

By leveraging advanced technologies like machine learning, natural language processing, and behavioral analytics, the app can intelligently prioritize emails, categorize messages, and adapt to individual user needs over time.

As users demand s marter tools to manage growing email volumes, an adaptive email client offers a transformative solution, setting a new standard in digital correspondence**.**

**FUTURE ENHANCEMENT:**

### **Offline Functionality:**

* Allow users to read, reply, and compose emails offline, with automatic syncing when back online.

**Advanced Security Features:**

* **End-to-End Encryption:** Offer encrypted email options for sensitive communications.
* **Phishing Detection:** Use machine learning to identify and alert users about phishing or spam emails.

### **Customizable User Experience:**

* **Dynamic Themes:** Allow users to personalize the app with themes or modes (dark mode, minimal mode).
* **Widget Support:** Add email widgets for quick access on mobile or desktop.

### **Collaborative Features:**

* **Shared Inbox:** Enable shared access to specific email threads for teams.
* **Email Notes:** Allow users to add private notes or comments to emails for internal tracking.

### **Unified Communication Platform:**

* **Integrated Messaging:** Combine emails with other communication channels like chat (Slack, WhatsApp) or video calls (Zoom, Teams) in one interface.
* **Cross-Account Unification:** Manage multiple email accounts from different providers in a single, streamlined inbox.