ADAPTIVE MAIL: A FLEXIBLE EMAIL CLIENT APP

ANDROID APPLICATION DEVELOPMENT

Prepared by:

Gnanesh G

Harish V

Kamala Kannan S

Yogesh R

Piyush Kumar Mahto

TABLE OF CONTENTS

S.No	Description	Page no
1	Introduction	2
2	Problem Statement	2
3	Objectives	2
4	System Requirements	2
5	System Architecture	3
6	Features	3
7	Implementation Details	4
8	Testing	4
9	Challenges Faced	5
10	Learning Outcomes	5
11	Future Enhancements	5
12	Conclusion	6
13	References	6
14	Appendices	6

1. INTRODUCTION

The **Adaptive Mail** application is an Android-based email client designed using modern development practices. It showcases a conversational and user-friendly interface built with **Jetpack Compose** and **Kotlin**. The app enables users to register, log in, compose emails, and view email history. It is a flexible solution that integrates a database for efficient data storage and management.

2. PROBLEM STATEMENT

Email clients often face challenges such as poor UI design, lack of flexibility, and inefficient data management. The Adaptive Mail app addresses these issues by providing an intuitive interface and reliable functionality to improve user experience.

3. OBJECTIVES

- Create a modern Android application using Jetpack Compose for a clean and reactive UI.
- Enable secure user registration and login functionality.
- Provide a platform for users to compose emails and manage email history.
- Integrate a database for persistent storage of user and email data.

4. SYSTEM REQUIREMENTS

Hardware Requirements:

- Processor: Intel i3 or higher
- RAM: 4 GB or above
- Storage: Minimum 10 GB free space
- Android Device/Emulator: Android API level 21 or above

Software Requirements:

- Android Studio (latest version)
- Kotlin programming language

- Jetpack Compose toolkit
- SQLite or Room Database
- ADB tools for testing

5. SYSTEM ARCHITECTURE

1. **User Interface Layer:**
    - Jetpack Compose components like TextField, Button, and LazyColumn for login, email history, and email composition screens.
2. **Database Layer:**
    - Room Database with entities for User and Email.
    - DAO for database operations.
3. **Logic Layer:**
    - View Model for managing UI-related data and ensuring lifecycle awareness.
    - Kotlin Coroutines for handling asynchronous tasks like database queries.

6. FEATURES

- **User Registration and Login:** Secure access with credentials stored in the database.
- **Email History:** Displays a list of previously composed emails.
- **Email Composition:** Allows users to input the subject and body of an email.
- **Persistent Data Storage:** All data is saved in the Room Database.
- **Responsive UI:** Built with Jetpack Compose for a seamless user experience.

7. IMPLEMENTATION DETAILS

Technologies Used:

- **Programming Language:** Kotlin
- **UI Toolkit:** Jetpack Compose
- **Database:** Room Database

Development Steps:

1. **Initial Setup:**
    - Configure Android Studio and create a new project.
2. **Database Integration:**
    - Define entities for User and Email.
    - Create a DAO for CRUD operations.
    - Set up the Room database instance.
3. **UI Design:**
    - Implement Compose UI for screens: Registration, Login, Email History, and Compose Email.
    - Use navigation components to link screens.
4. **Logic Implementation:**
    - Add user authentication logic in ViewModel.
    - Implement email history retrieval and email composition logic.
5. **Testing and Debugging:**
    - Test each feature on an emulator and physical devices.

8. TESTING

Test Cases:

1. **Registration Screen:**
    - Test valid and invalid input for user registration.
    - Ensure data is correctly stored in the database.

2. **Login Screen:**
    - Verify user authentication with correct and incorrect credentials.
3. **Email History Screen:**
    - Test the display of previously composed emails.
4. **Compose Email Screen:**
    - Verify input validation for email subject and body.
    - Ensure new emails are stored in the database.

Tools Used:

- Android Emulator
- ADB for debugging

9. CHALLENGES FACED

- Integrating Room Database with Jetpack Compose UI.
- Managing lifecycle-aware components with View Model.
- Debugging database queries during email retrieval.

10. LEARNING OUTCOMES

- Acquired skills in Android development using Jetpack Compose.
- Learned how to integrate and manage a Room Database.
- Gained experience in Kotlin programming and debugging.

11. FUTURE ENHANCEMENTS

- Add real email-sending functionality using SMTP.
- Implement push notifications for new emails.
- Introduce multi-user support with role-based access.

12. Conclusion

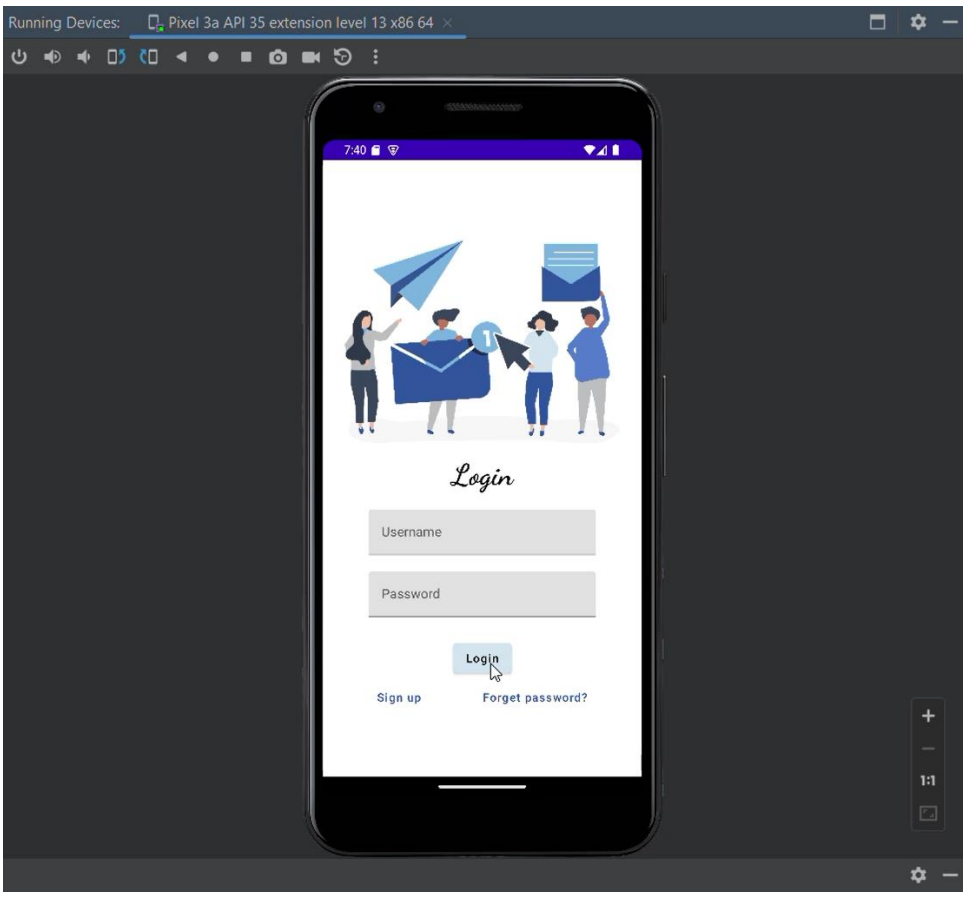
The Adaptive Mail application successfully demonstrates the development of a functional Android email client using Jetpack Compose and Kotlin. The project achieves its goals of providing secure login, email composition, and history management with persistent storage.

13. References

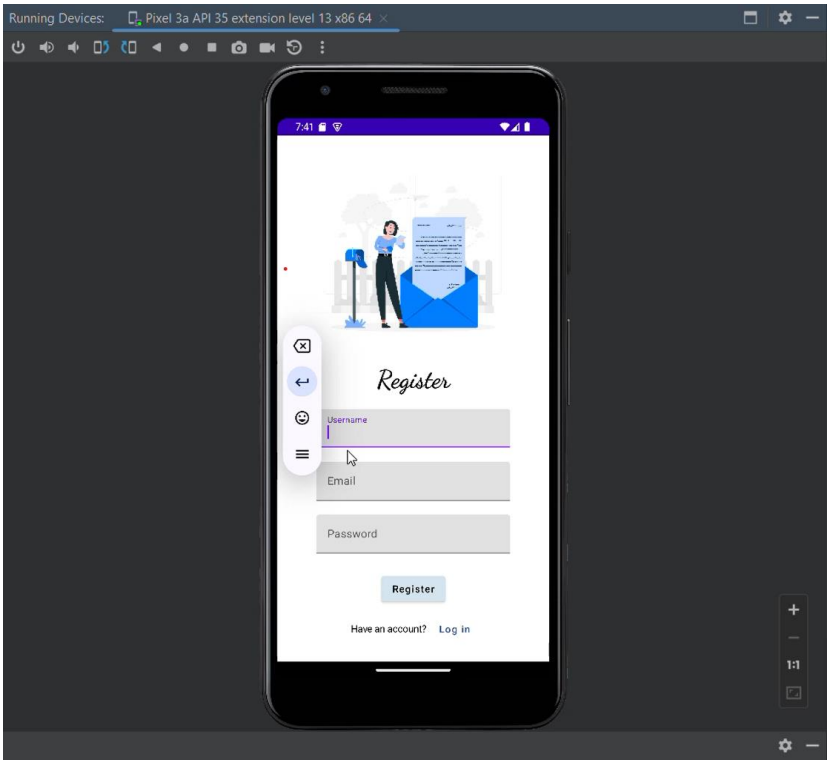
- Official Android Developer Documentation
- Jetpack Compose Documentation
- Room Database Documentation

14. Appendices

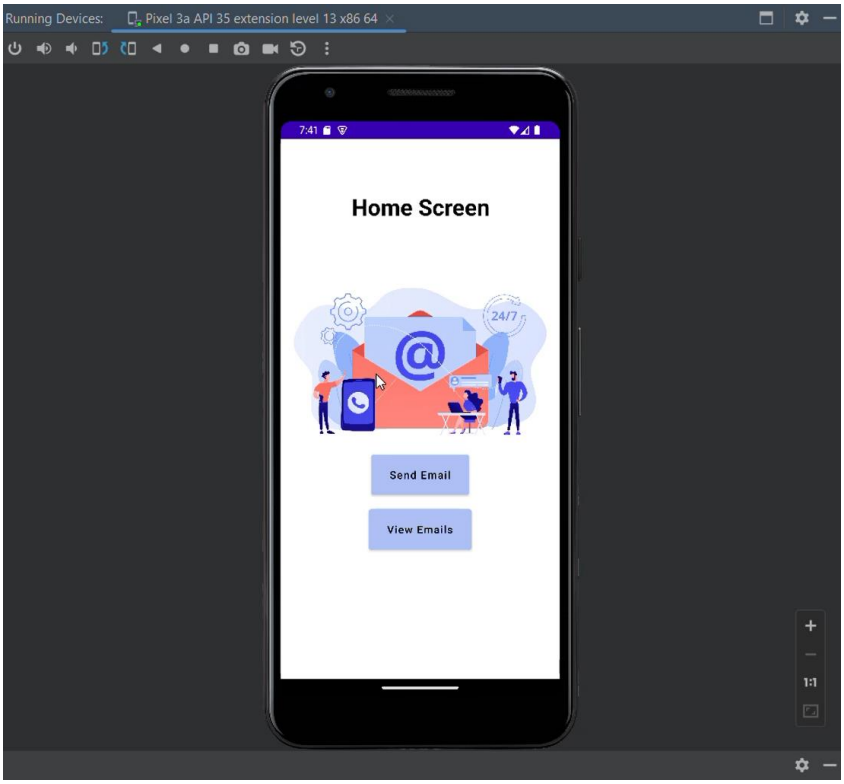
Login Screen



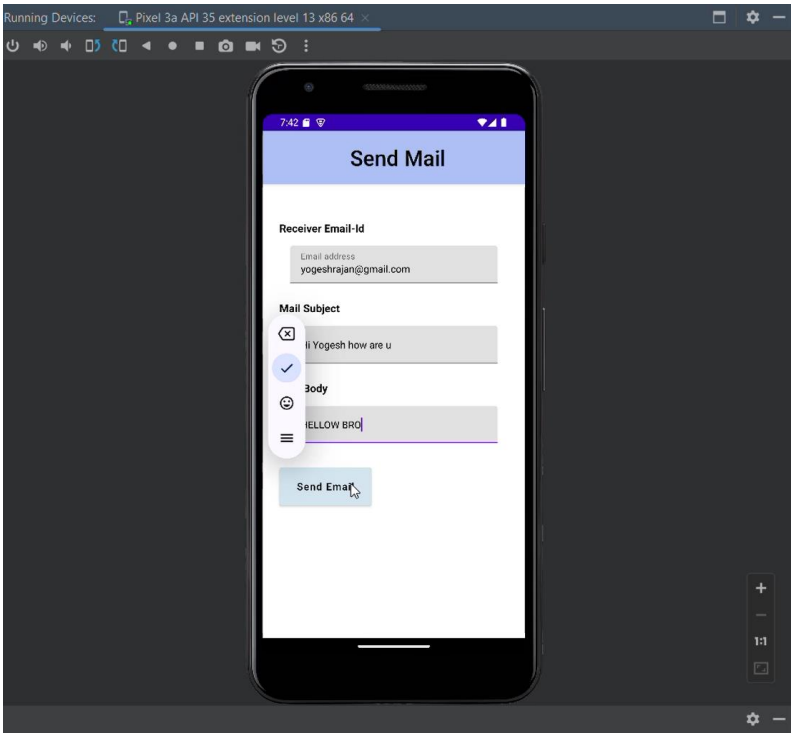
Register screen



Home Screen



Send Email page



View Emails