CMPE 277
SMARTPHONE APPLICATION DEVELOPMENT

FINAL PROJECT REPORT

Roommate Finder Application

TEAM UNSOCIAL

SATYA ASHISH VEDA
SHIREESH VENNAMANENI
UPASANA KUMAR

Github: https://github.com/upasanakr/FindRoomie

Table of Contents

1. INTRODUCTION

1.1. ABSTRACT

In today's fast-paced world, finding the perfect roommate or accommodation can be a daunting task, especially for international students and young professionals relocating to new cities. Students often spam messages across multiple messaging platforms to find roommates. There is no unified exclusive platform for students to find roommates. The Roommate Finder App aims to simplify this process by leveraging advanced technologies such as Android Application Development, Machine Learning (ML), Cloud Computing, and Generative Pre-trained Transformers (GPTs) to offer a personalized, efficient, and user-friendly solution. By understanding user preferences, both explicit and unstated, our app not only connects users with their ideal living situation but also fosters a community of like-minded individuals. Our primary goal is to streamline the Roommate and Room Finding Process: Make it easier, faster, and more reliable for users to find compatible roommates and desirable accommodations. We are utilizing ML to offer personalized room and roommate suggestions based on a comprehensive understanding of user preferences. Efficient User Onboarding is implemented using an interactive chatbot with GPT. We created a trusted platform where users can confidently find and connect with potential roommates, backed by AI-driven matching scores. Our solution integrates three key technological components to address the challenges in finding compatible roommates and accommodations:

● Machine Learning: By analyzing both explicitly stated preferences and deducing unstated preferences from user interactions, our ML models, hosted on AWS and powered by services like SageMaker, provide a sophisticated matching algorithm. This algorithm not only matches users with potential rooms or roommates but also ranks these matches based on compatibility, learning and adapting over time to improve suggestions.

● Cloud Infrastructure: Our robust cloud-based architecture ensures scalability, reliability, and security. User profiles, preferences, and interactions are securely stored in a database hosted on AWS. This setup supports the real-time execution of our ML models and hosts APIs that facilitate seamless app functionality across devices.

● GPT-Powered Chatbots: From the moment users engage with our app, they are greeted by an intuitive chatbot that assists with onboarding, gathers user preferences, and provides instant answers to queries. This not only enhances the user experience but also ensures that profiles are detailed and preferences are accurately captured.

.

1.2. OBJECTIVES

- To provide a user-friendly platform for finding compatible roommates and accommodations.
- Utilize ML for personalized roommate and accommodation suggestions.
- Enhance user experience with a GPT-powered chatbot for onboarding and support.
- Establish a trusted platform with AI-driven matching.

1.3. SCOPE

- Development of an Android application.
- Integration with cloud services for data management and ML model deployment.
- Implementation of a chatbot for user interaction and support.

1.4. FUNCTIONAL REQUIREMENTS

- User registration and profile creation.
- Personalized roommate and accommodation matching.
- Chatbot for assistance and profile setup.

1.5. NON-FUNCTIONAL REQUIREMENTS

- Scalability to handle growing user numbers.
- Security measures for user data protection.
- High availability and minimal downtime.
- Responsive design for various device size.

2. PROJECT DETAILS

2.1. TECHNOLOGIES USED

- Android Studio(JAVA)
- AWS EC2
- AWS Lambda
- AWS RDS
- AWS Sagemaker
- Python
- GPT

2.2. FEATURES

User Registration

The application offers a dual registration mode allowing users to sign up either as individuals seeking accommodation or as listers offering their places. This flexibility caters to a broad user base with varying needs related to housing.

Detailed User Preferences

Upon registration, the app prompts users to fill out a comprehensive preferences form. This includes budget, preferred area, specific requirements (e.g., landmarks, proximity to grocery stores), and lifestyle choices (e.g., smoking and drinking habits). These details are crucial for tailoring user experiences and improving the relevance of recommendations.

Interactive User Interface

Utilizing Android Studio, the app features a variety of activities and layouts to ensure a user-friendly and interactive interface. This design enhances user engagement by providing a seamless navigation experience throughout different sections of the app.

Intelligent Chatbot

The application integrates a GPT-based chatbot that enhances user interaction by dynamically rephrasing questions. This not only aids in maintaining engagement but also improves user support by offering varied conversational structures.

Landmark Classification

The app leverages GPT's capabilities to classify user inputs regarding landmarks into broader categories. For example, preferences like "near to Walmart" or "close to Target" are generalized to "near a grocery store," which simplifies the data processing and enhances matching accuracy.

2.3. METHODOLOGY

Data Generation

The initial dataset was crafted using ChatGPT, comprising four main CSV files: `Users.csv` for user details, `Listings.csv` for accommodation listings, `Seeker Preferences.csv` for individual accommodation-seeking preferences, and `Matches.csv` for recording matches and user actions (accept, reject, no action).

Data Preprocessing

Data preprocessing involved handling null values to maintain data integrity, categorical encoding to transform non-numeric data into a machine-readable format, and feature selection to eliminate redundant or irrelevant features. These steps were critical for preparing the dataset for effective machine learning modeling.

Collaborative Filtering Using SVD

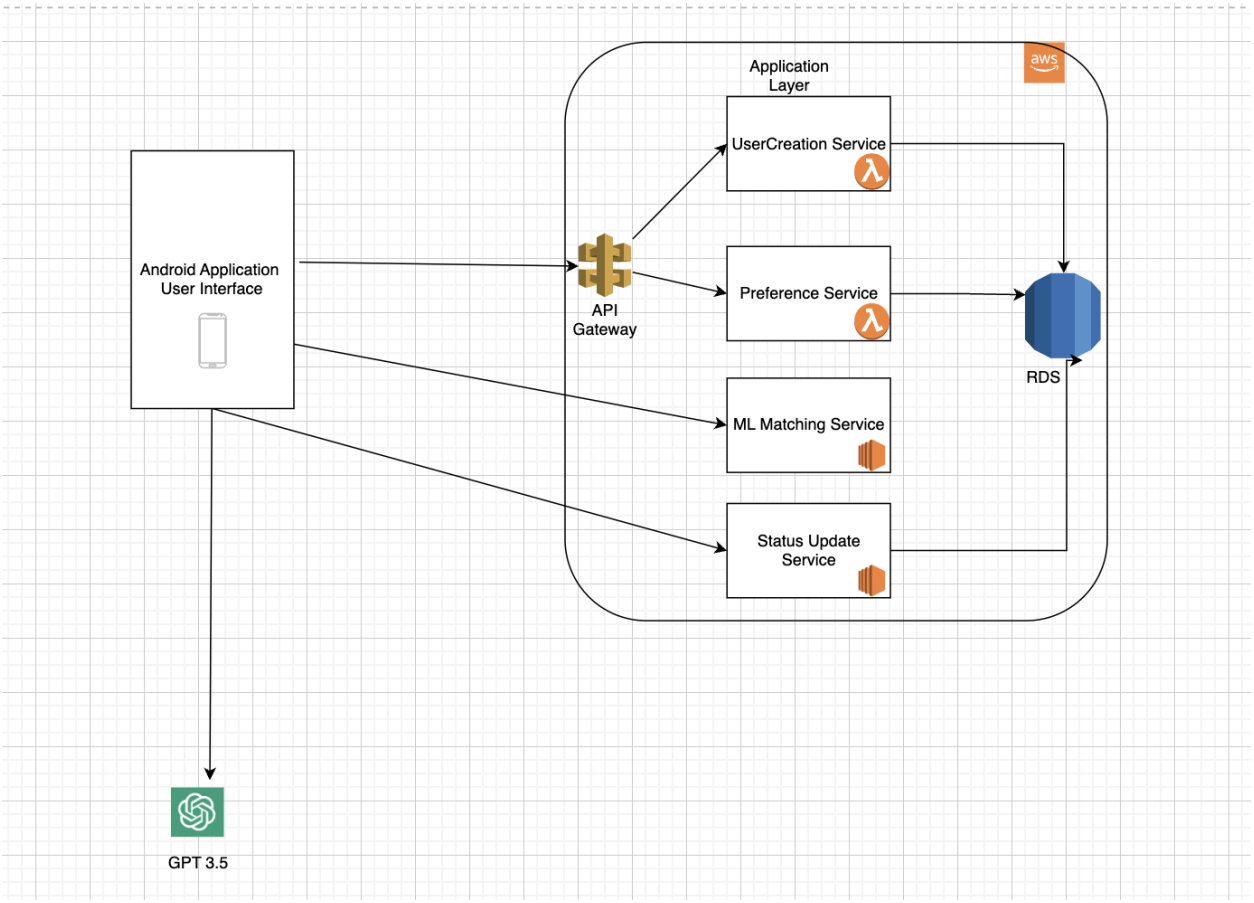
The core of the recommendation system is based on Singular Value Decomposition (SVD) collaborative filtering. This technique analyzes both user and listing data to identify latent factors that influence preferences and behavior. By decomposing the data, the model can predict how well a user might like a listing not yet considered, thus personalizing the recommendations.

Match Generation and Interaction

Once the recommendation model processes the preferences, potential matches are displayed on the home screen for seekers. Users can then accept or decline these listings. Accepted matches allow seekers to view the contact details of the listers, while listers can see which users have accepted their listings. This feature facilitates direct communication and accelerates the roommate finding process.

3. ARCHITECTURE AND DESIGN

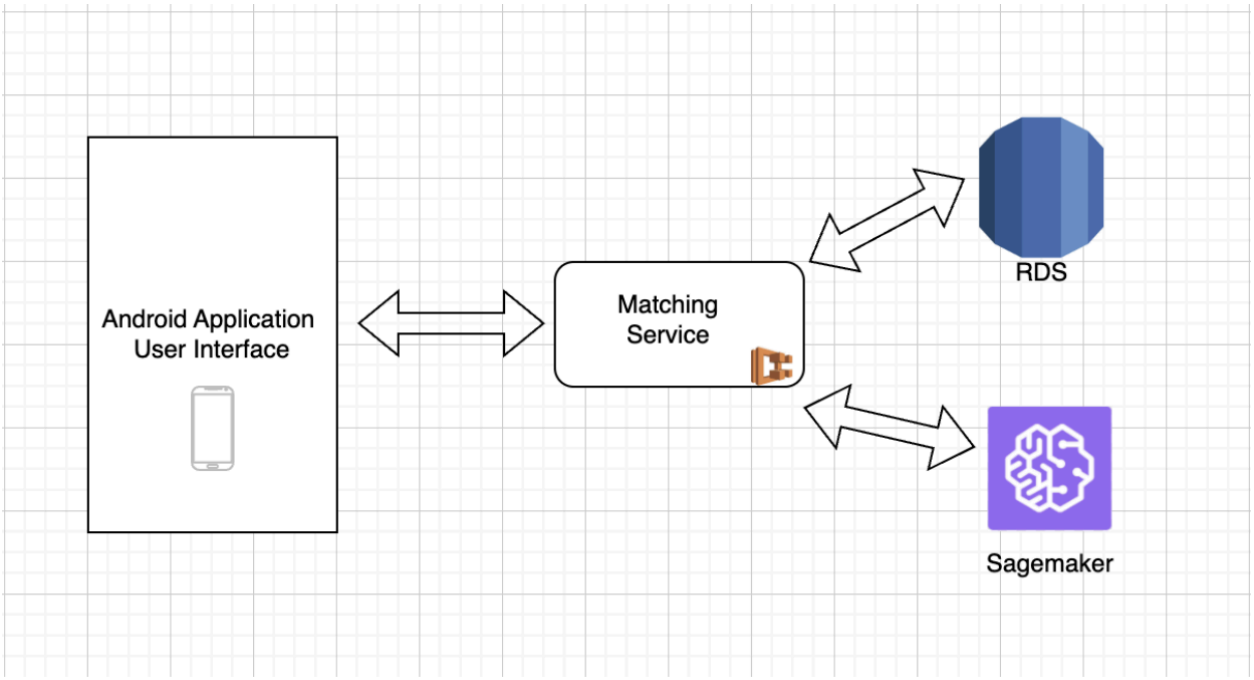
3.1. ARCHITECTURAL DESIGN



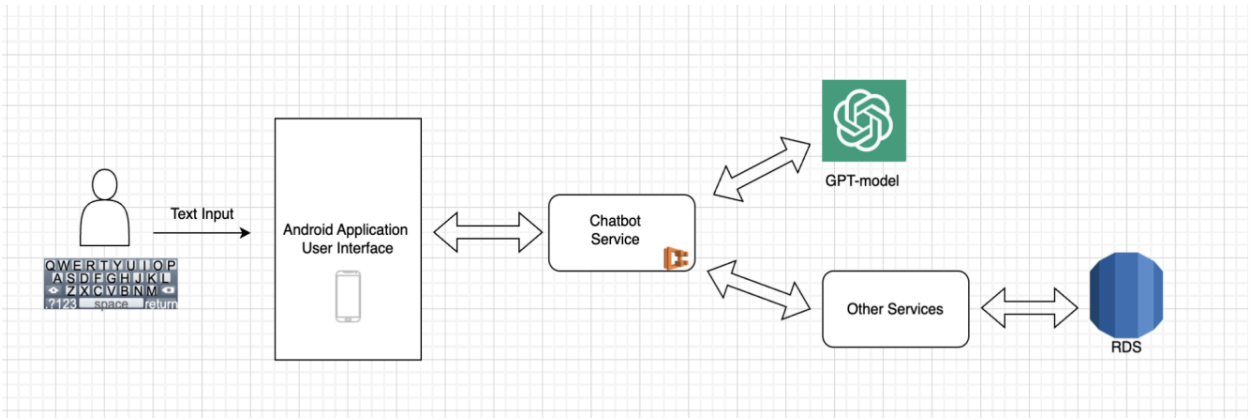
- *User Interface Layer*: This depicts the top most layer where users interact with the mobile application. We have different screens for User registration, Profile Setup, Match Suggestions etc.

- *Application Layer*: This is where the backend services for User management, Matching Service, Status Update Service, and Preference Service reside. The User Interface layer interacts with the Application layer to fetch appropriate information.

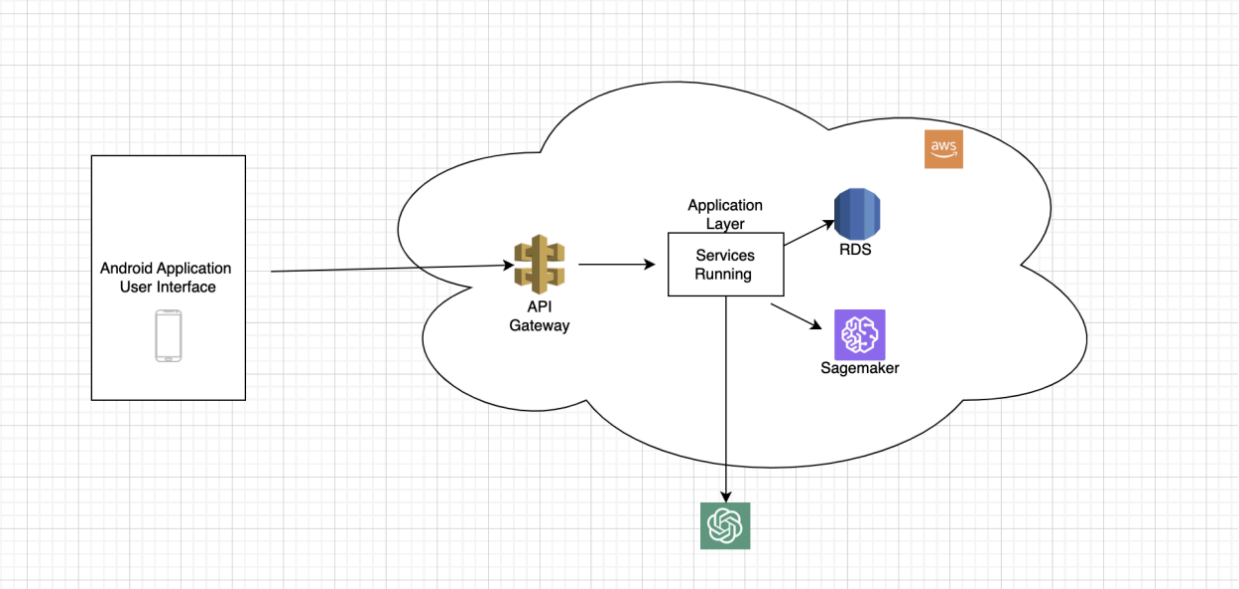
- *Machine Learning Layer*: The Matching Service interacts with the UI layer, DB and the ML model running on AWS Sagemaker to retrieve the correct matches and send them back to the UI layer.



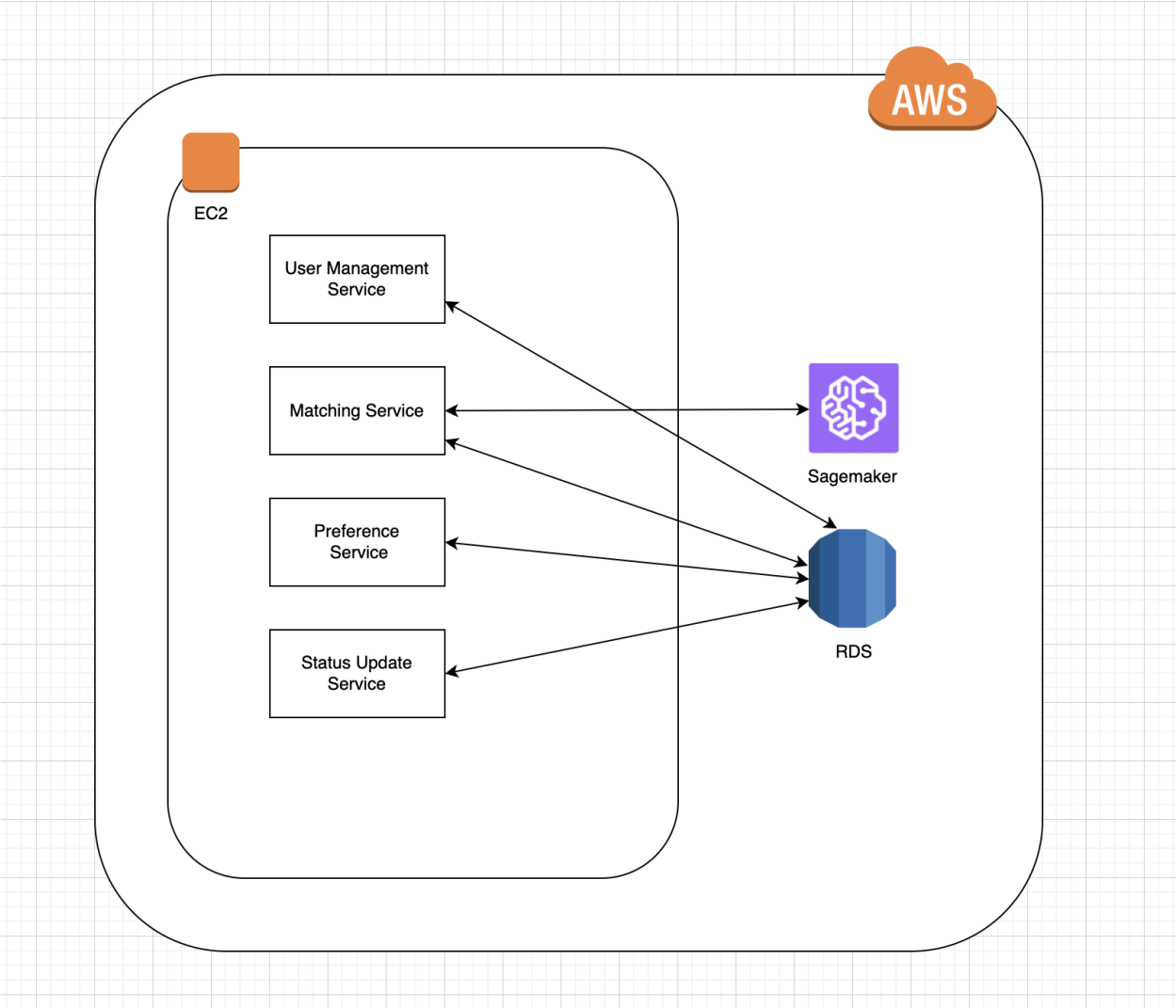
- *Chatbot Interface*: Users interact with the chatbot while signing up. The preferences of the users are captured using the chatbot. GPT-based models will be used.



- *Cloud Infrastructure and Database:* All the backend services, databases, ML models will be running on AWS Cloud. These support each other and also the UI layer in fetching and displaying correct data as required.



3.2. SERVER SIDE DESIGN



Server side design of our mobile application consists of the following components:

Amazon EC2 Instance: Illustrates the servers within the AWS cloud, which hosts the various services.

Application Services:

● User Management Service: Manages user account creation, authentication, profile management, and preference storage.

● Matching Service: Runs the algorithms to match roommates based on their preferences, utilizing ML model insights.

● Notification Service: Handles the triggering and delivery of notifications to users about matches, messages, and other relevant updates.

● Chatbot Service: Facilitates user interactions with the system, gathers user preferences, and provides support through a GPT-based conversational interface.

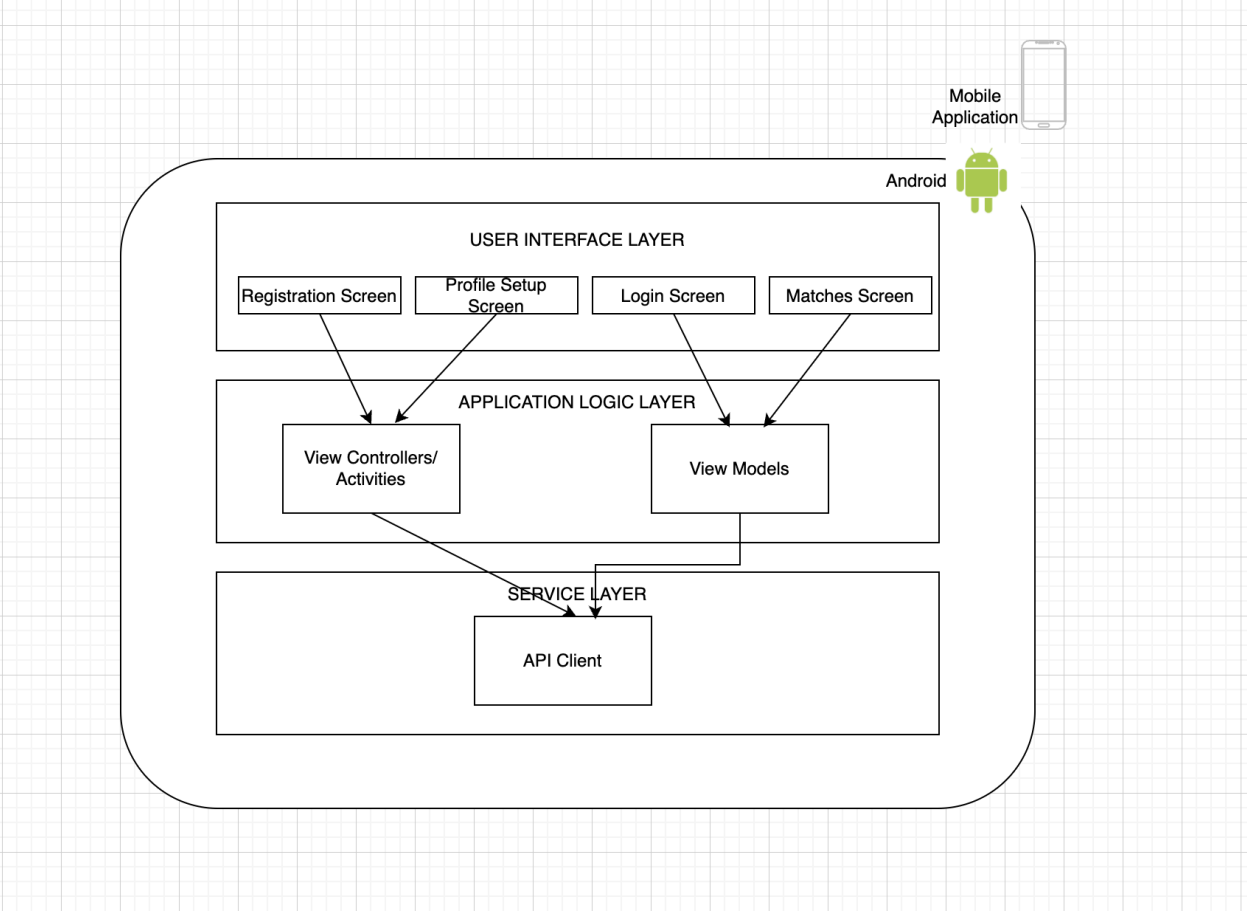
● Machine Learning Services

  ○ AWS SageMaker: Hosts and runs the ML models for roommate matching. Directly interacts with the Matching Service for generating match suggestions.

● Data Storage

  ○ Amazon RDS/DynamoDB: Stores user profiles, preferences, match histories, and chat logs securely. Interacts with all the application services for data retrieval and storage.

3.3. CLIENT SIDE DESIGN

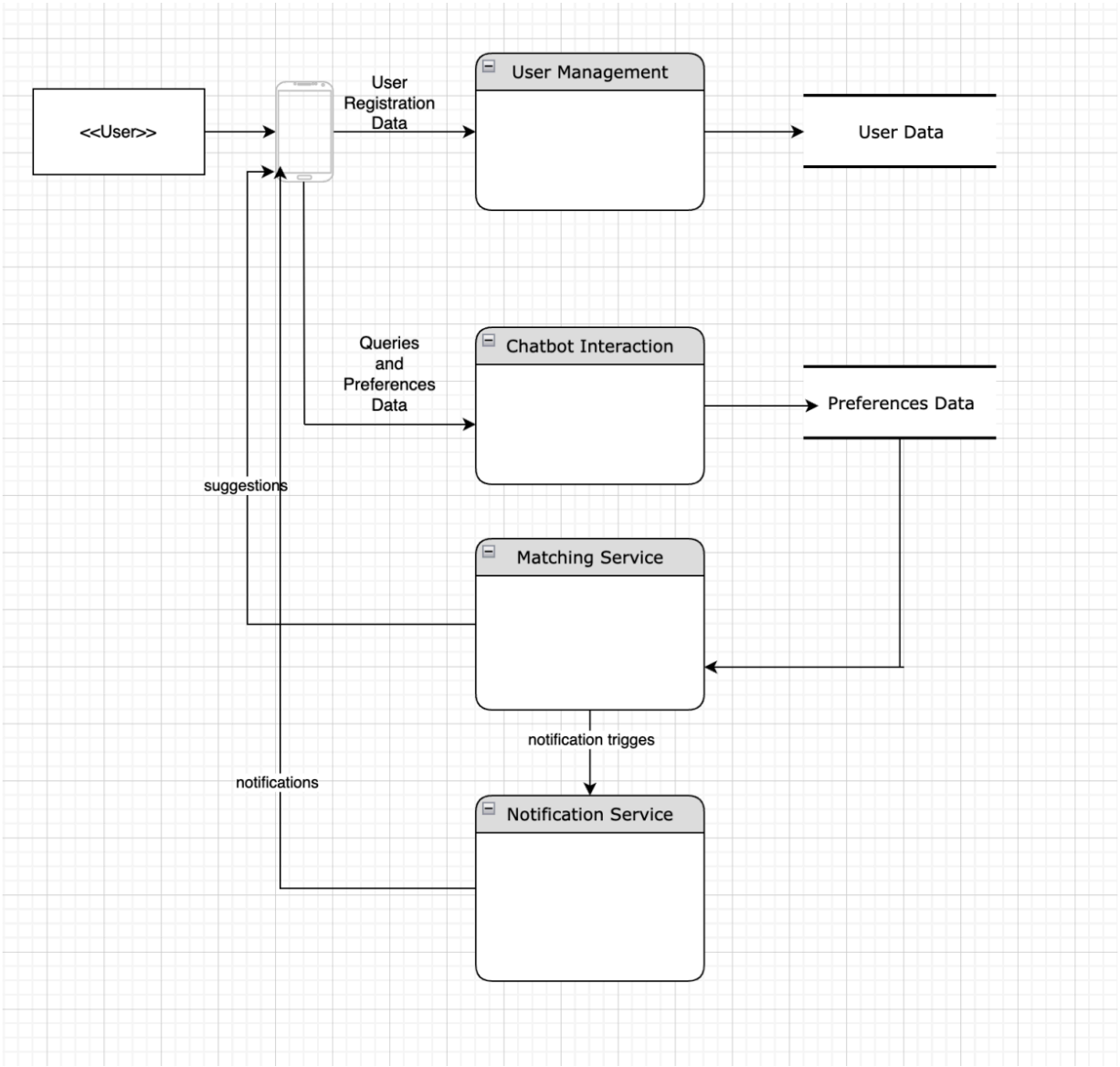


Client Side Design consists of the following components:

Mobile Application (Client-Side)

- Operating System: The base layer where the app is installed. Typically not detailed in client-side diagrams but understood to be Android/iOS.
    - User Interface Components
        - Registration Screen: Allows new users to create an account. Interacts with the Chatbot Interface for dynamic preference capture
        - Login Screen: For returning users to access their accounts.
        - Profile Setup Screen: Enables users to add or edit their personal and accommodation preferences.
        - Match Suggestions Screen: Displays potential roommate matches to the user.
    - Application Logic Components
        - View Controllers/Activities/Fragments: Manage the interactions between the UI components and the Application Service Layer.
        - ViewModels: Hold and manage UI-related data, serving as a communication center between the UI and the business logic.
    - Service Layer:
        - API Client: Handles all network communication with backend services.

3.4. DATAFLOW DIAGRAM



3.5. COMPONENT LEVEL DESIGN

User Interface Layer

Components:

Registration Screen: Captures user information and preferences. Interacts with the Chatbot Interface for dynamic preference capture.
Profile Setup Screen: Allows users to provide detailed personal and accommodation Preferences.
Match Suggestions Screen: Displays roommate or accommodation matches. Interacts with the Matching Service to fetch suggestions.

Interactions:

● The Registration and Profile Setup screens send user data to the User Management Service in the Application Layer.
● The Match Suggestions Screen requests data from the Matching Service, receiving filtered matches based on user preferences.

Application Layer

Components:

User Management Service: Handles user registration, login, and profile updates.
Matching Service: Utilizes Machine Learning models to provide personalized match suggestions.
Notification Service: Sends alerts and messages to users about new matches or messages.
Chatbot Service: Facilitates user interaction during sign-up and preference setting using a GPT-based model.

Interactions:

● The User Management Service communicates with the Cloud Database to store or retrieve user information.
● The Matching Service interacts with the Machine Learning Layer to process and refine match suggestions, sending the results back to the UI Layer.
● The Notification Service listens for triggers from other services (like new matches from

the Matching Service) to send notifications to users.
● The Chatbot Service engages users through the UI Layer, capturing preferences that are stored with the help of the User Management Service.

Machine Learning Layer

Components:

ML Model for Matching: Analyzes user data and preferences to suggest compatible roommates or accommodations.

Interactions:

● Receives user preference data from the Matching Service, processes this data to find suitable matches, and returns these suggestions back to the Matching Service.
● Chatbot Interface, considered part of both the User Interface and Application Layers, specifically interacts with users to gather preferences using natural language processing and feeds this data to the User Management Service for profile setup and refinement.

Cloud Infrastructure and Database

Components:

AWS Services: EC2, RDS/DynamoDB, SageMaker, Lambda, etc.

Database: Stores user profiles, preferences, match histories, and chat logs.

Interactions:

● Hosts the Application Layer services, providing the computational resources needed for the Matching Service's ML models and the Chatbot Service.

● Stores and manages data across the system, ensuring high availability, scalability, and security for the app's operations.

Detailed Component Interactions

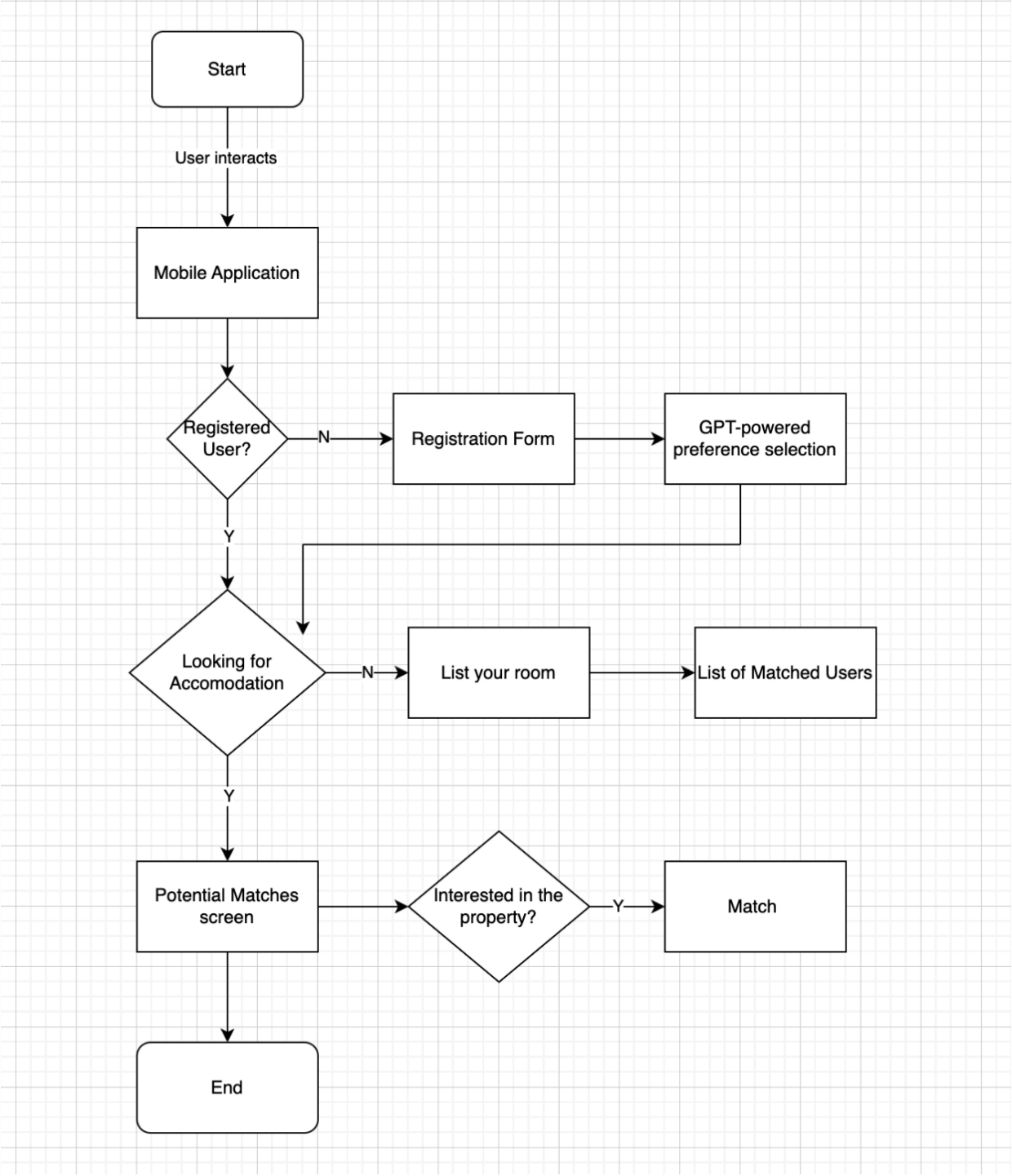
● User Registration and Preference Capture:
    ○ Flow: User interacts with the Registration Screen → Chatbot Interface captures detailed preferences → User Management Service stores this data in the database.

- Match Suggestion Process:
    - Flow: User requests matches → Matching Service fetches user preferences from Database → ML Model processes these preferences to suggest matches → Suggestions are sent back to the UI Layer for display.

- Notification and Updates:
    - Flow: Matching Service finds a new match → Notification Service alerts the user through the UI Layer.

- Chatbot Service:
    - Flow: User interacts with the Chatbot during registration/profile updates → Chatbot Service processes the conversation, utilizing GPT models for natural language understanding → Preferences are updated in the user's profile via the User Management Service.

3.6. WORKFLOW DIAGRAM



● User interacts with the mobile application.

● If the user is not a registered user, the application provides a form to register followed by an interactive way of providing the preferences based on the functionality(looking for

accommodation or providing accommodation)

● If the user is already registered and is looking for accommodation, it takes him directly to the potential matches screen where the user can see his potential matches and match the ones he is interested in.

● If the user is registered and is providing accommodation, then he is notified and listed all his matches.

Mobile Interfaces:

Overview:
Mobile interfaces are designed to ensure a user-friendly, intuitive, and accessible user experience on mobile devices. For the Roommate Finder App, the interface includes screens for registration, profile setup, and match suggestions, all integrated with a chatbot for dynamic interaction.

Key Screens:
Login Screen: The Login Screen is the entry point for returning users to access their accounts and interact with the app's core features.
● Elements: Username/Email Field, Password Field, Sign In Button
● Actions: Authenticate User, Navigate to Profile, Redirect to Registration.

Registration Screen: Captures user information and preferences interactively through the chatbot.
● Elements: Form fields for user details, chat interface for collecting preferences.
● Actions: Submit data, interact with chatbot.

Profile Setup Screen: Allows users to provide detailed personal and accommodation preferences post-registration.
● Elements: Detailed form fields, option selectors for preferences.
● Actions: Update profile, save preferences.

Match Suggestions Screen: Displays potential roommate or accommodation matches based on user preferences.
● Elements: List of matches, filter options, match details view.
● Actions: View match profile, send message, save favorite match.

4. TEST-CASES

1. User Registration and Login

- **Test Case 1A:** Register as a new user seeking accommodation.
    - **Expected Output:** User is registered in the database and a success message is displayed.
    - **Original Output:** User is registered in the database and a success message is displayed.
    - **Status:** PASS
- **Test Case 1B:** Register as a new lister.
    - **Expected Output:** Lister is registered in the database and a success message is displayed.
    - **Original Output:** Lister is registered in the database and a success message is displayed.
    - **Status:** PASS
- **Test Case 1C:** Register with incomplete details.
    - **Expected Output:** Error message displayed and registration fails.
    - **Original Output:** Error message displayed and registration fails.
    - **Status:** PASS
- **Test Case 1D:** Log in with valid credentials.
    - **Expected Output:** User successfully logs in and is directed to the home page.
    - **Original Output:** User successfully logs in and is directed to the home page.
    - **Status:** PASS
- **Test Case 1E:** Log in with invalid credentials.
    - **Expected Output:** Error message about invalid credentials and login fails.
    - **Original Output:** Error message about invalid credentials and login fails.
    - **Status:** PASS

2. User Preferences Submission

- **Test Case 2A:** Submit complete set of preferences as a seeker.
    - **Expected Output:** Preferences are saved, and the user is directed to the recommendations page.
    - **Original Output:** Preferences are saved, and the user is directed to the recommendations page.
    - **Status:** PASS
- **Test Case 2B:** Submit preferences as a lister.
    - **Expected Output:** Preferences are saved, and the lister's listing is updated accordingly.
    - **Original Output:** Preferences are saved, and the lister's listing is updated accordingly.

    - **Status:** PASS
- **Test Case 2C:** Submit preferences with mandatory fields missing.
    - **Expected Output:** Error message displayed, and preferences are not saved.
    - **Original Output:** Error message displayed, and preferences are not saved.
    - **Status:** PASS

3. Chatbot Interaction

- **Test Case 3A:** Standard query interaction.
    - **Expected Output:** Chatbot responds appropriately with rephrased question.
    - **Original Output:** Chatbot responds appropriately with rephrased question.
    - **Status:** PASS
- **Test Case 3B:** Uncommon query interaction.
    - **Expected Output:** Chatbot attempts to handle query or suggests contacting support.
    - **Original Output:** Chatbot provides a generic response, slightly off-topic.
    - **Status:** FAIL

4. Recommendation System

- **Test Case 4A:** Recommendations for a seeker based on submitted preferences.
    - **Expected Output:** Recommendations closely match the preferences.
    - **Original Output:** Recommendations closely match the preferences.
    - **Status:** PASS
- **Test Case 4B:** Lister views matches based on their listing preferences.
    - **Expected Output:** Appropriate matches are displayed.
    - **Original Output:** Appropriate matches are displayed.
    - **Status:** PASS

5. Match Acceptance and Contact Information Exchange

- **Test Case 5A:** Accept a listing as a seeker.
    - **Expected Output:** Contact details of the lister are made available to the seeker.
    - **Original Output:** Contact details of the lister are made available to the seeker.
    - **Status:** PASS
- **Test Case 5B:** Lister checks accepted matches.
    - **Expected Output:** Lister can view all users who have accepted their listing.
    - **Original Output:** Lister can view all users who have accepted their listing.
    - **Status:** PASS

5. INDIVIDUAL CONTRIBUTION

The success of the Roommate Finder app is a result of the collaborative efforts of our team members, each bringing specialized skills to various aspects of the project. Below is a detailed breakdown of individual contributions:

Satya Ashish Veda

- **Chatbot Development:** Designed and implemented the chatbot component using the GPT model. This involved integrating the chatbot into the app's UI and ensuring its ability to dynamically rephrase and handle user queries effectively.
- **User Interface for Chatbot:** Developed the user interface specifically for the chatbot interaction, ensuring a seamless and engaging user experience that complements the overall design of the app.
- **Backend Development for Preferences:** Created the backend functionality required to insert and manage user preferences within the database. This included setting up the necessary endpoints and ensuring data integrity and security during data transmission.

Shireesh Vennamaneni

- **Dataset Generation:** Took the lead in generating the initial datasets required for the application using ChatGPT, which formed the basis for our collaborative filtering model.
- **Data Preprocessing:** Handled the preprocessing of data which included cleaning, encoding categorical data, and preparing the dataset for the recommendation system.
- **Model Development and Backend Recommendations:** Developed the SVD collaborative filtering model used for generating user recommendations. Additionally, managed the backend processes involved in returning recommendations and updating match statuses based on user interactions.

Upasana Kumar

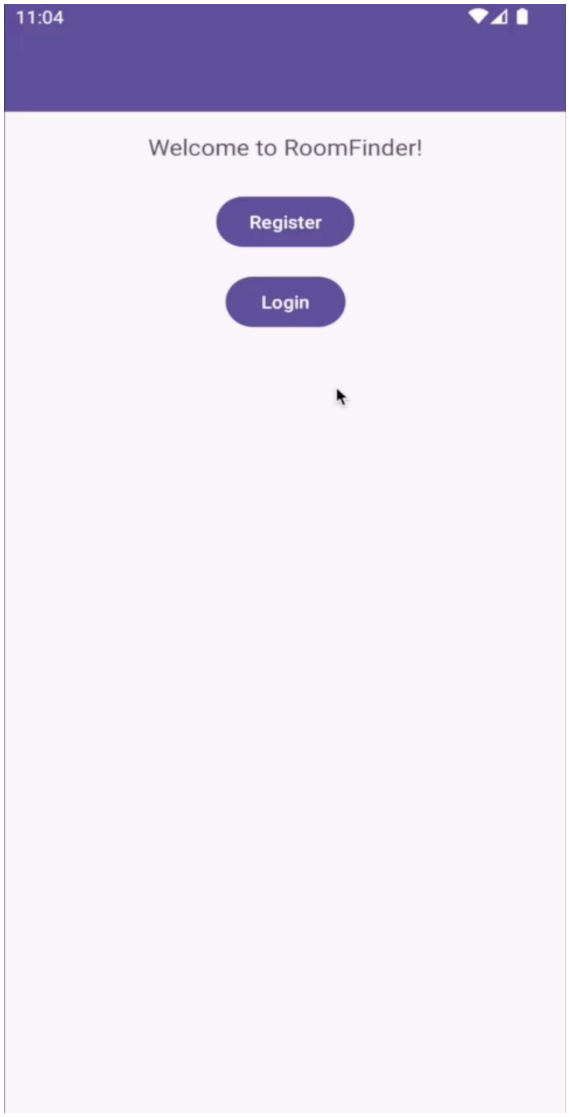
- **User Interface Development:** Responsible for designing and implementing the user interfaces for key screens other than the chatbot, including the login page, home page for seekers, and home page for listers.
- **Database Creation and Management:** Established and managed the cloud database for the application, setting up structures that efficiently store and retrieve data.
- **Backend for User Creation:** Developed the backend functionality for user account creation, ensuring the app effectively handles new user registrations and data storage.

Combined Efforts

- **Integration:** Together, we worked on integrating all the individual components into a cohesive application. This included aligning the frontend and backend functionalities, ensuring smooth data flow between the UI, the chatbot, the recommendation system, and the database. Our collaborative efforts were crucial in troubleshooting, refining the user experience, and achieving seamless functionality across different parts of the app.

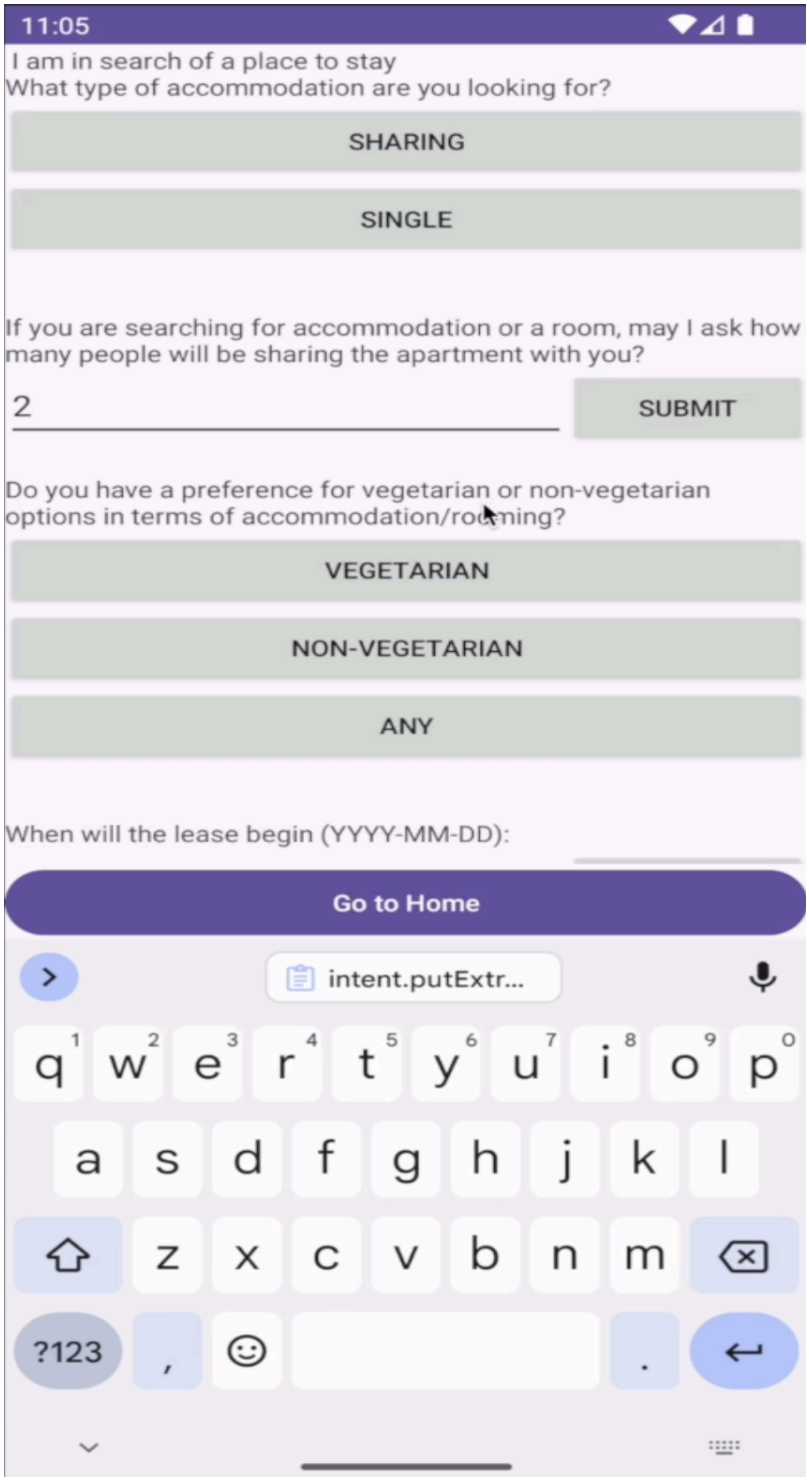
6. SCREENSHOTS

- Homepage when App is launched

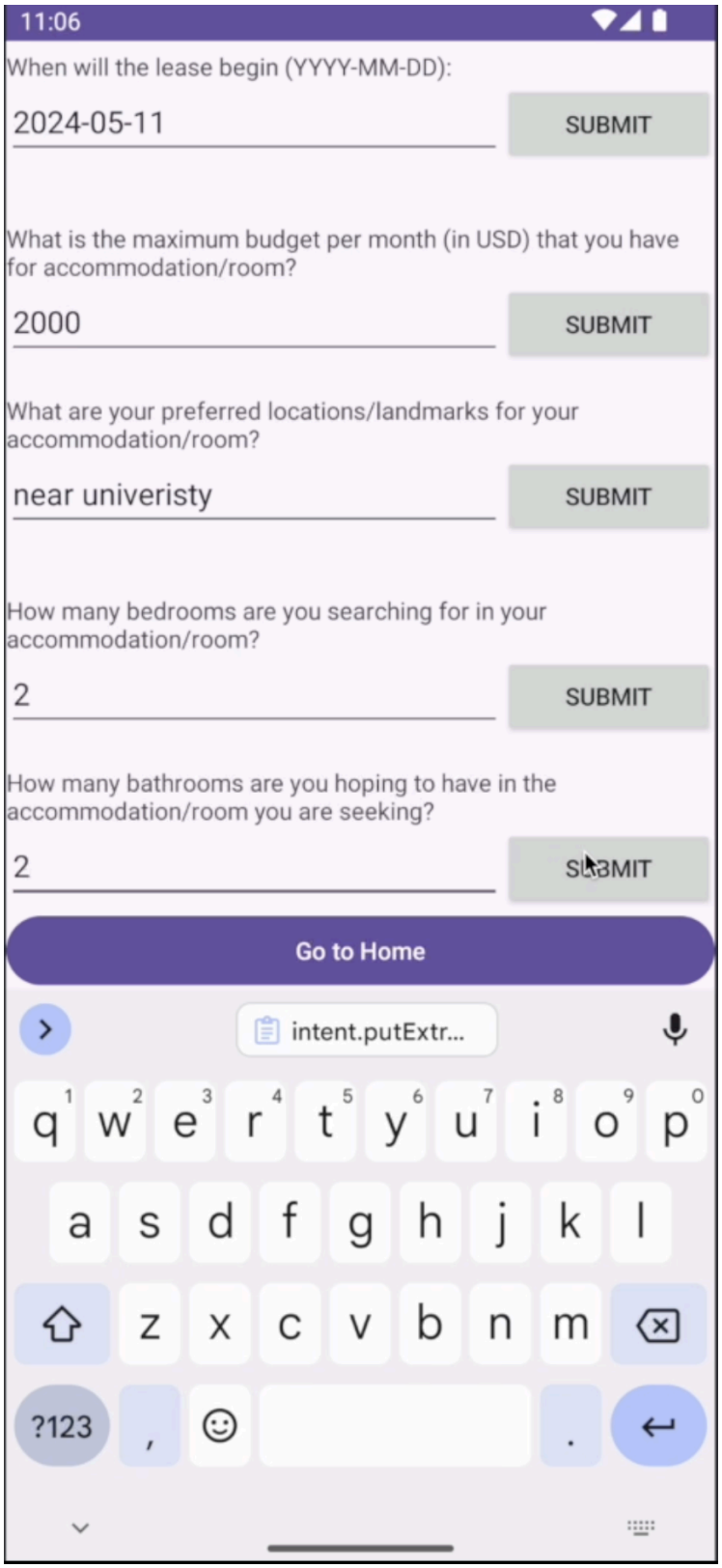


- Registration screen

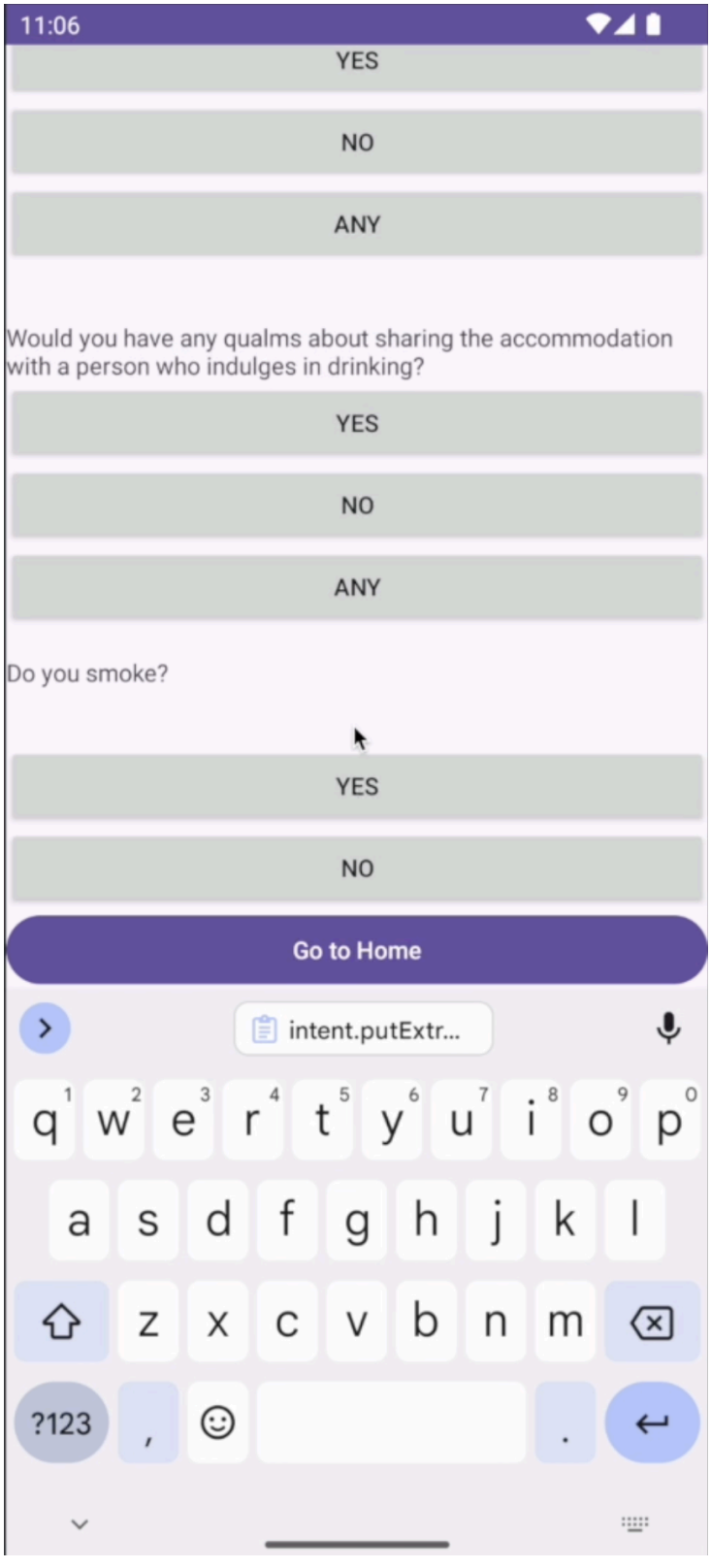
- Preference selection using interactive chatbot



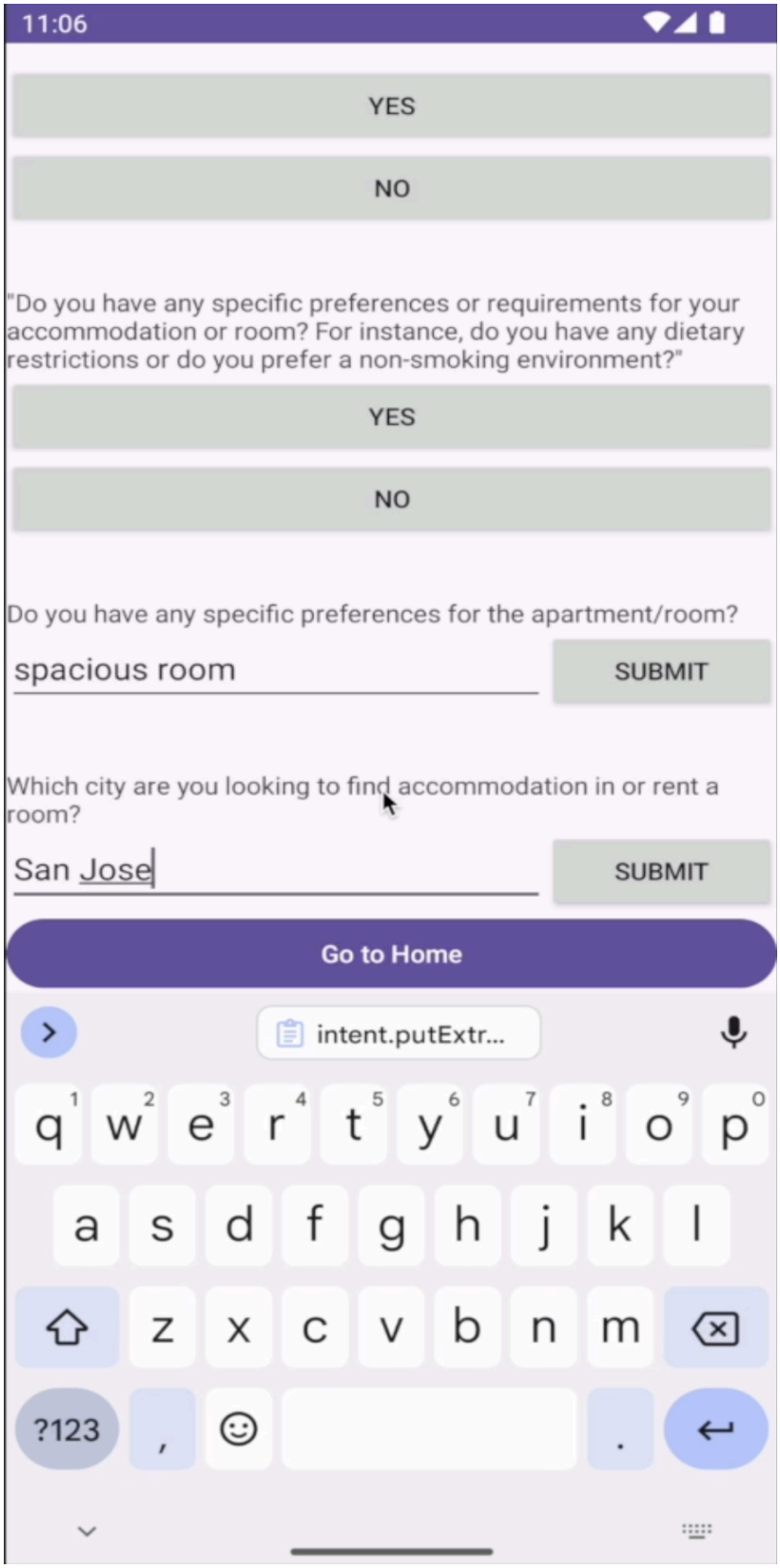
- More preferences like Lease start date, Budget, Landmarks, No. of bedrooms and bathrooms



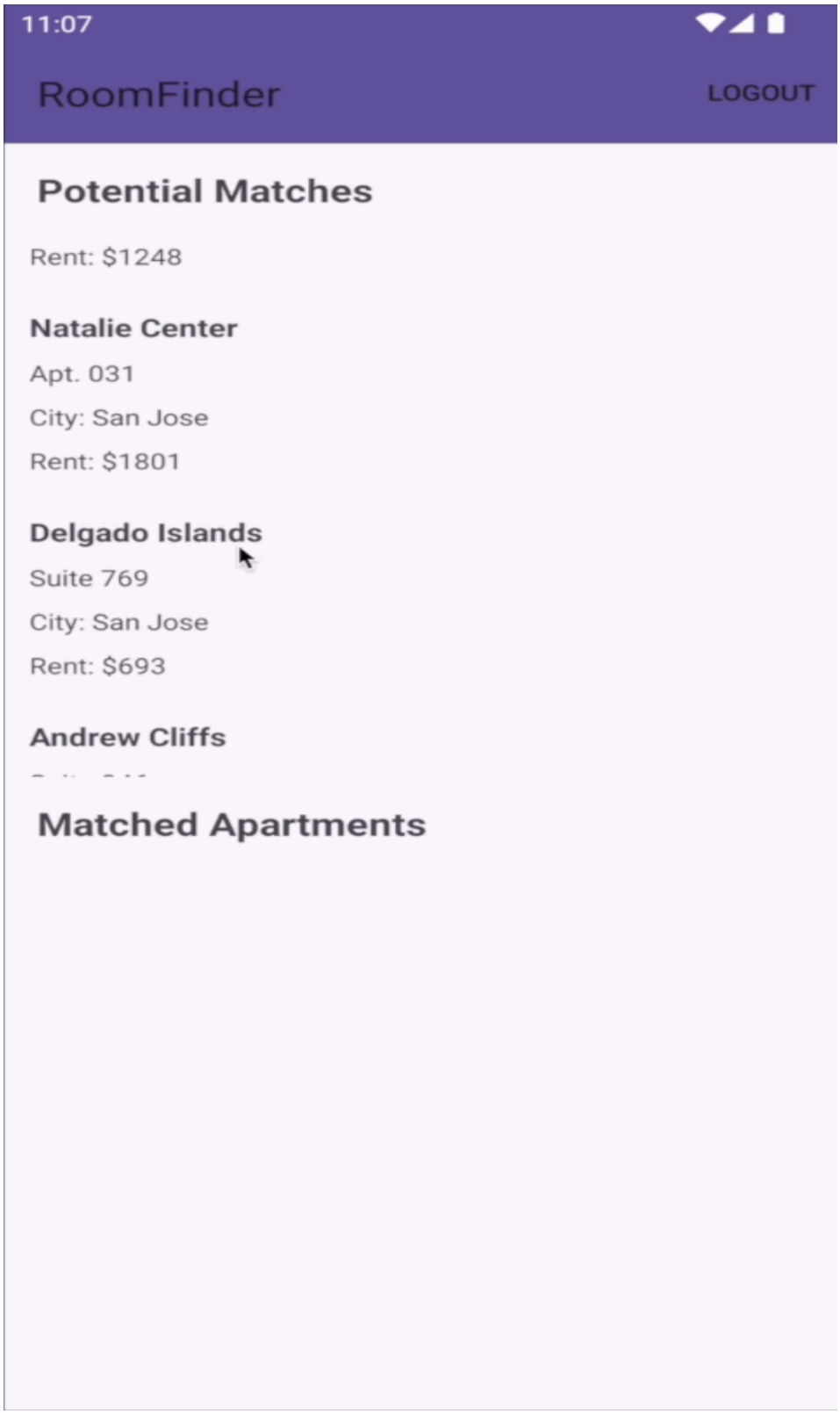
- More preferences like smoking and drinking



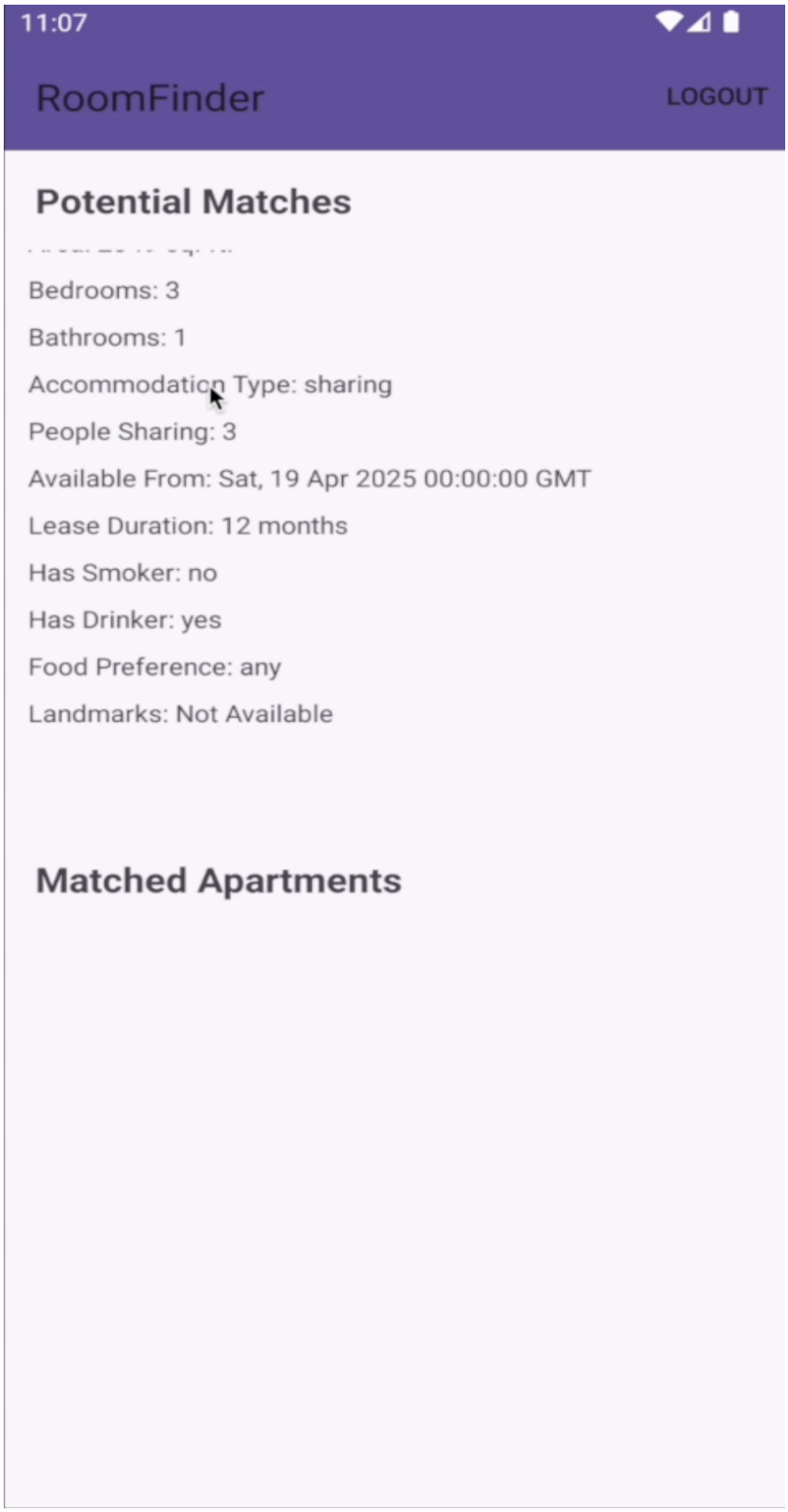
- More preferences like city, room-specific preferences.



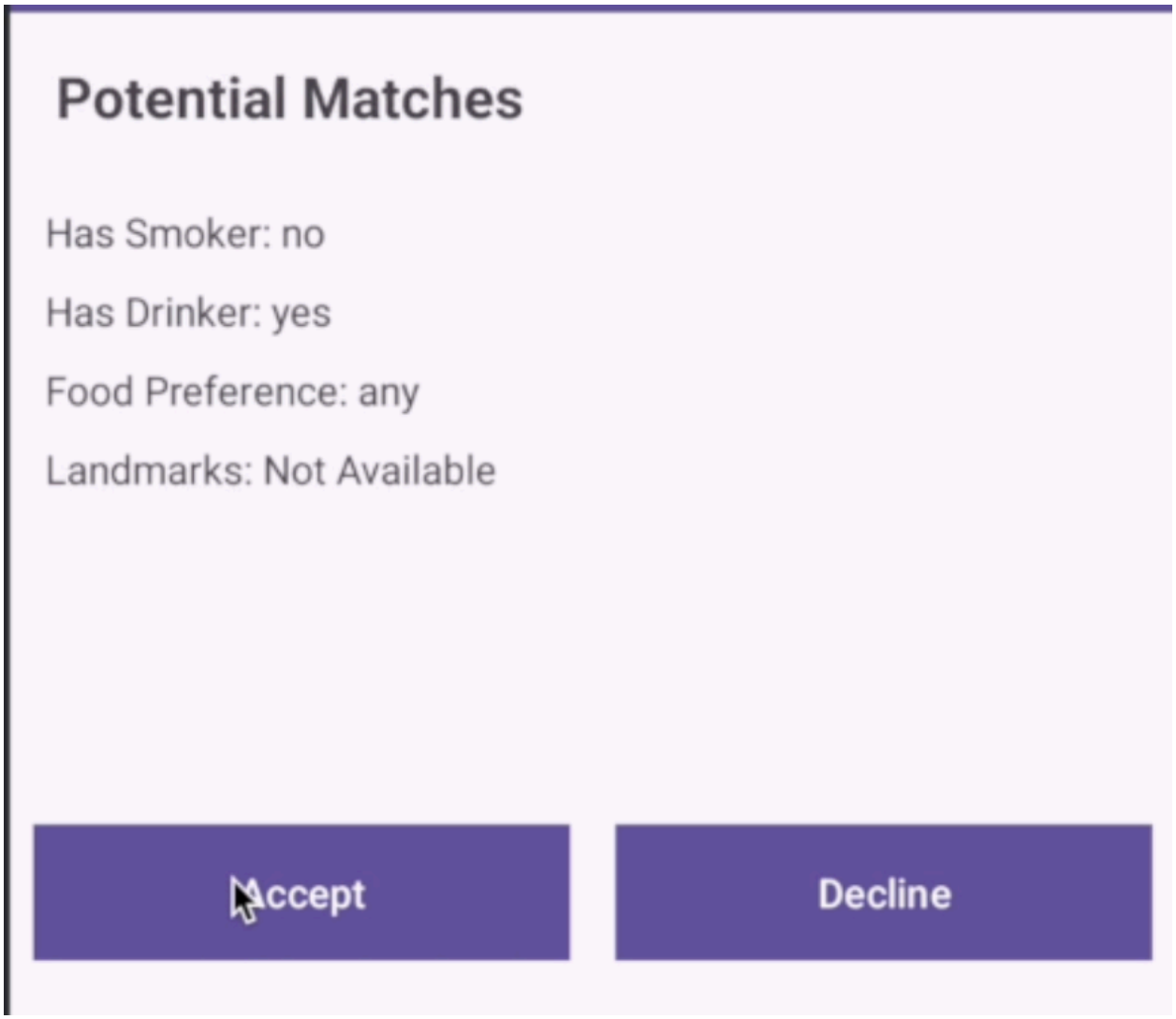
- Potential matches screen



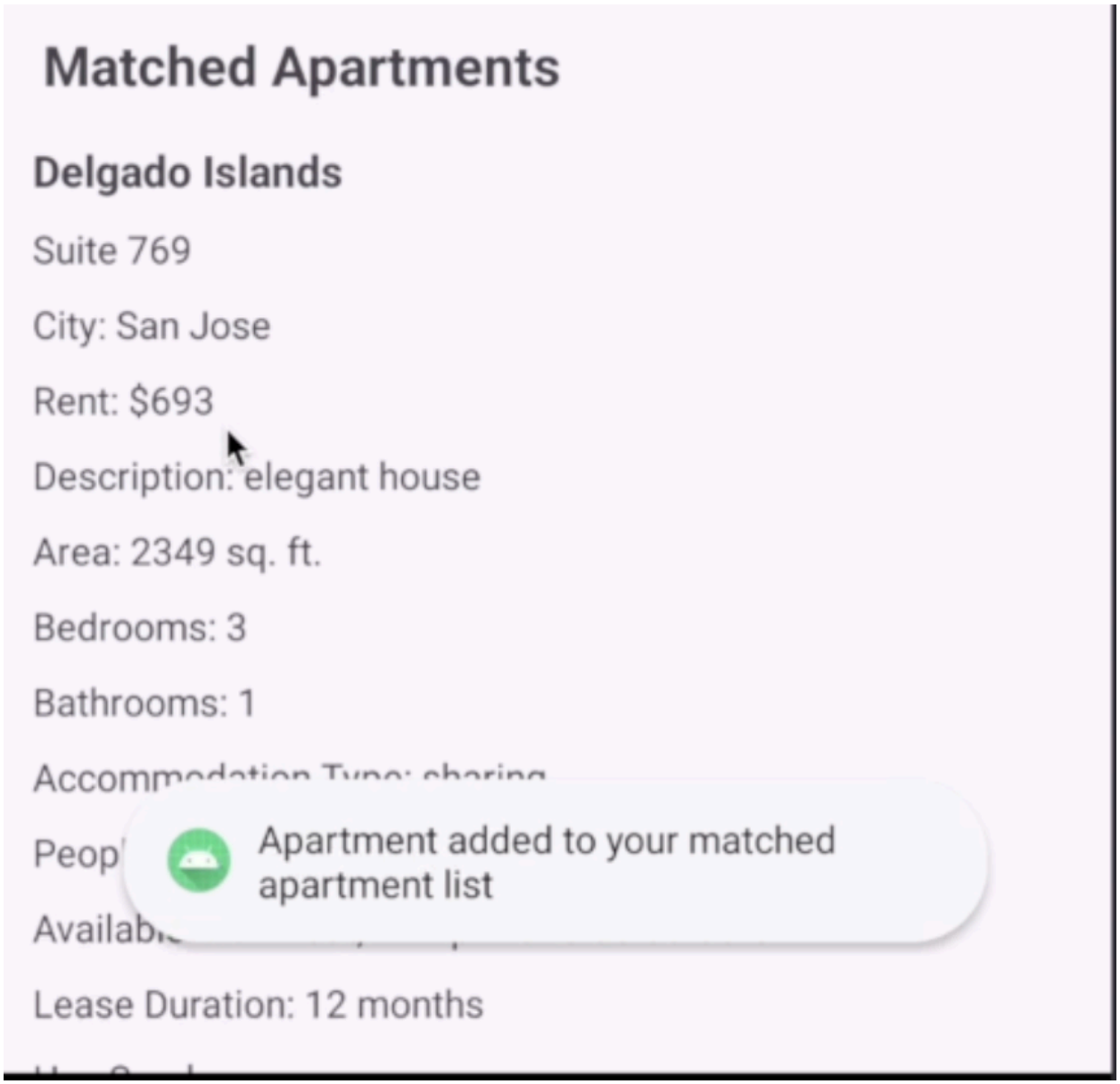
- Click on an apartment to view more details

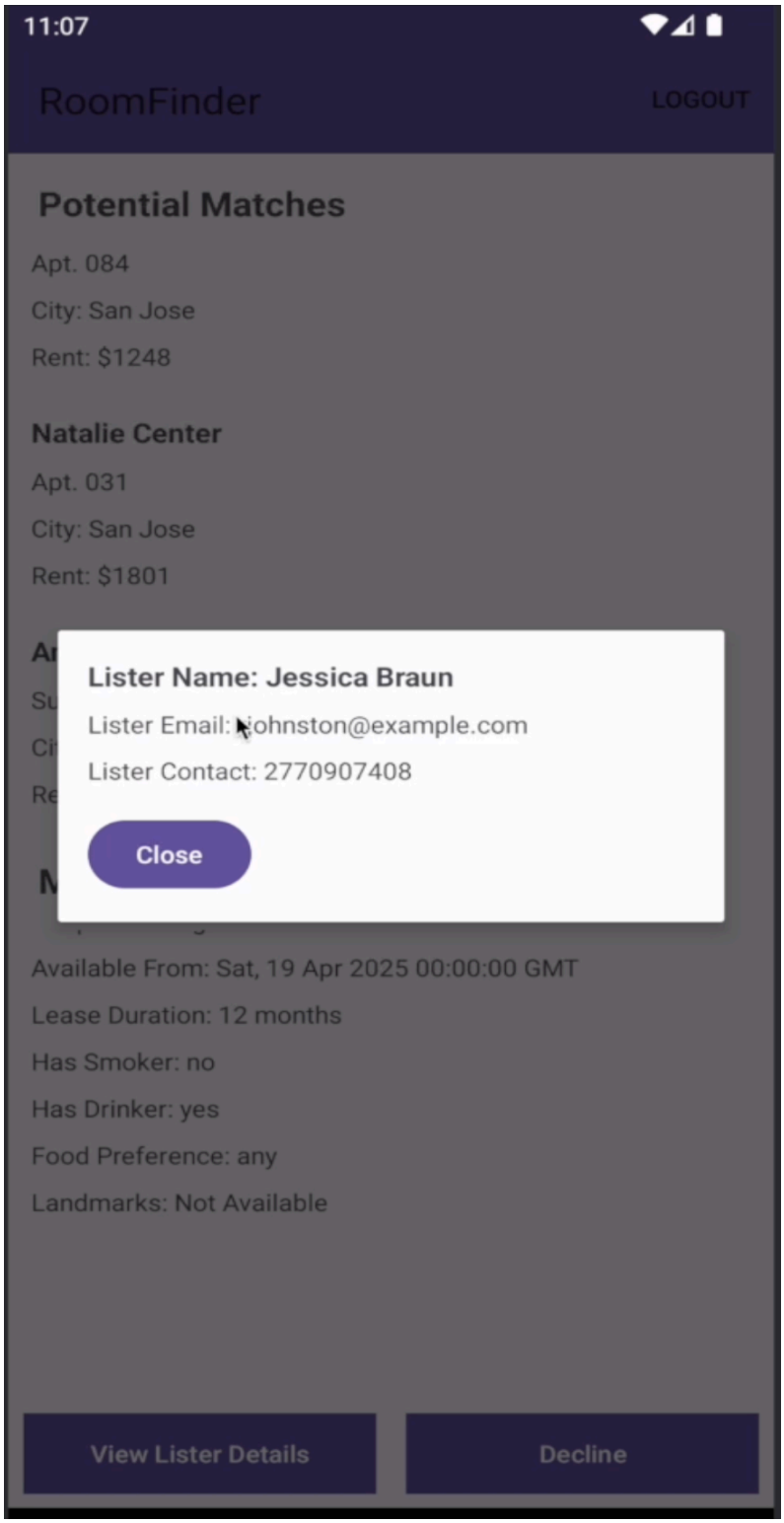


- User can Accept/ Decline the match

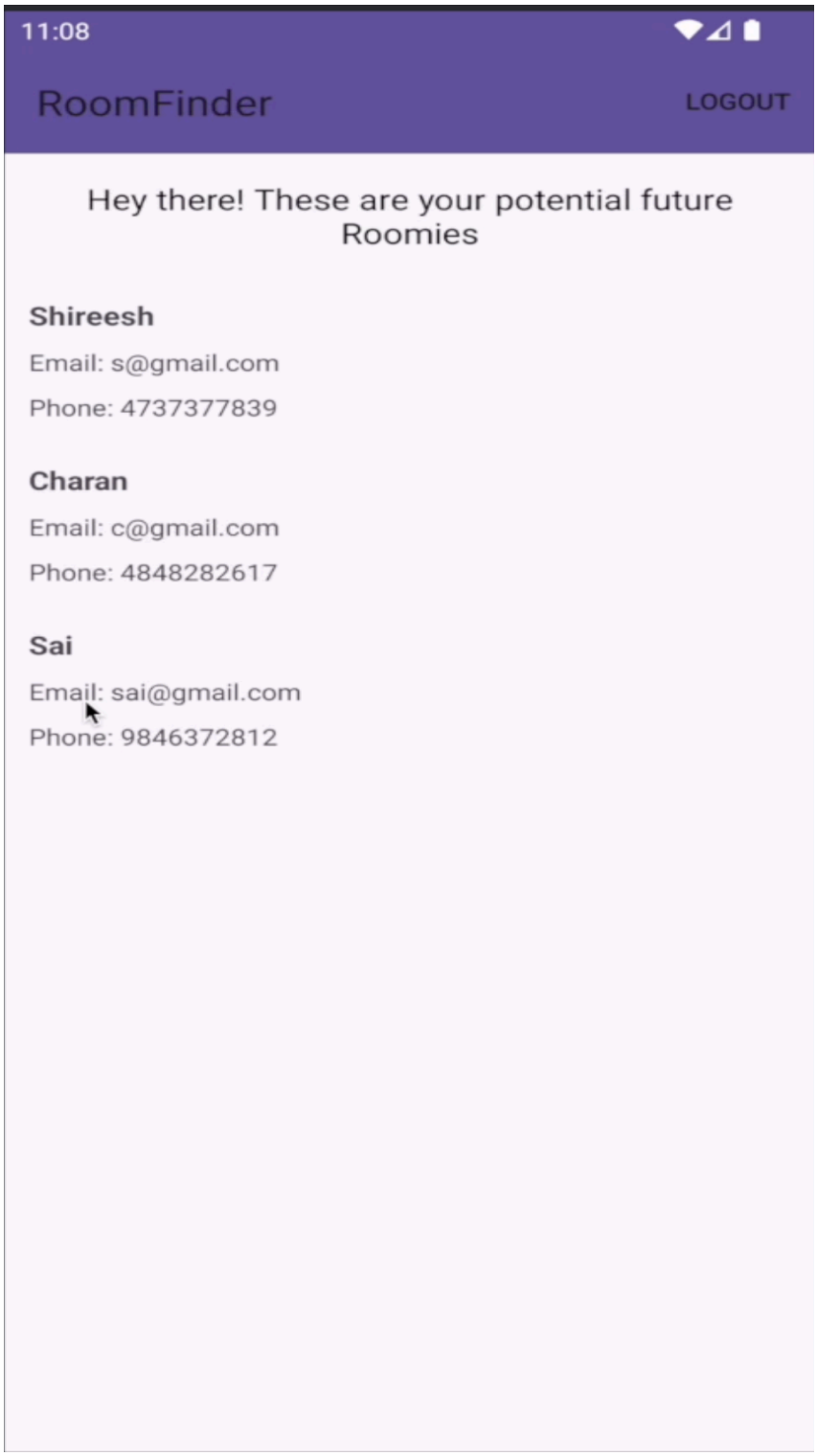


- By accepting, user will be able to see the contact details of the lister




11:07
RoomFinder
LOGOUT
Potential Matches
Apt. 084
City: San Jose
Rent: $1248
Natalie Center
Apt. 031
City: San Jose
Rent: $1801
Lister Name: Jessica Braun
Lister Email: johnston@example.com
Lister Contact: 2770907408
Close
Available From: Sat, 19 Apr 2025 00:00:00 GMT
Lease Duration: 12 months
Has Smoker: no
Has Drinker: yes
Food Preference: any
Landmarks: Not Available
View Lister Details
Decline

- The lister can see all the people who have accepted his listing



7. CONCLUSION

The Roommate Finder app project successfully demonstrates the application of advanced technologies such as GPT for natural language processing tasks and Singular Value Decomposition (SVD) for collaborative filtering to enhance the experience of finding roommates in a digital platform. Through meticulous development and comprehensive testing, the app has proven effective in matching users with compatible living arrangements based on detailed personal preferences and behavioral data.

The test cases outlined in the report highlight the app's robust functionality across various critical features, including user registration, preference management, interactive communication via a chatbot, and intelligent recommendation generation. The results show a high degree of reliability and user satisfaction, with the majority of test scenarios passing and confirming the system's readiness for real-world deployment.

However, the test results also indicate areas for improvement. The chatbot's performance with uncommon queries suggests the need for further refinement in its ability to handle a wider range of user interactions. Enhancing the chatbot's understanding and response capabilities will improve user engagement and support, contributing to a more accommodating user interface.

In conclusion, the Roommate Finder app stands as a strong testament to the potential of integrating artificial intelligence and machine learning technologies in everyday applications. Moving forward, continual updates and improvements, guided by user feedback and technological advancements, will be crucial in maintaining the relevance and efficiency of the app. This project not only meets its initial objectives but also sets a scalable foundation for future enhancements that could include more sophisticated data analysis, integration of real-time data, and broader social features to further enrich the user experience in finding the ideal living situation