ApartMate

Team 3
Group members - Adrian Raj, Akshay Srinivasan, Ian Rettig, Risheek Narayanadeverekere, Siddhant Patel

Design Document

Index

- Purpose 3
- Design Outline 4
    - Components
    - High-Level Overview
    - State/Activity Diagram
- Design Issues 6
    - Functional Requirements
    - Non-Functional Requirements
- Design Detail 9
    - Data Classes Diagram
    - Data Classes Description

Purpose

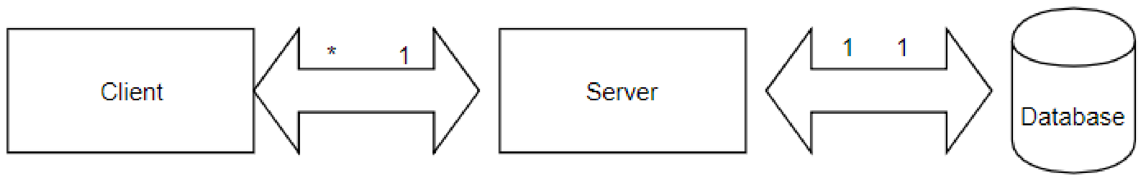
People moving to a new location and searching for an apartment need a place to go to find roommates to live with in an apartment. This app idea has been done a lot, so we decided to add more features to it. We also have a part of the app that is used after roommates are found. This part is intended for use throughout the time that the roommates live together in the apartment. Some features of that part include a group chat, schedules, chore lists with reminders, a bill splitter, and a shopping list with location based notifications. These features exist in an app already on the market, Roof, but this app has unfavorable reviews on the android market, mainly due to bugs in the system.

We intend to be a better choice than Roof by providing a bug free experience with a few additional features that that app does not have. On the other hand, we intend to be a better market for users searching than the other apps by having these extra features. Another additional feature that comes from the long nature of our app is ratings for individual users. This feature is intended to help with the future of an individual’s roommate search. Having these two parts together in one location makes our app the best choice when it comes to finding a roommate.

Design Outline

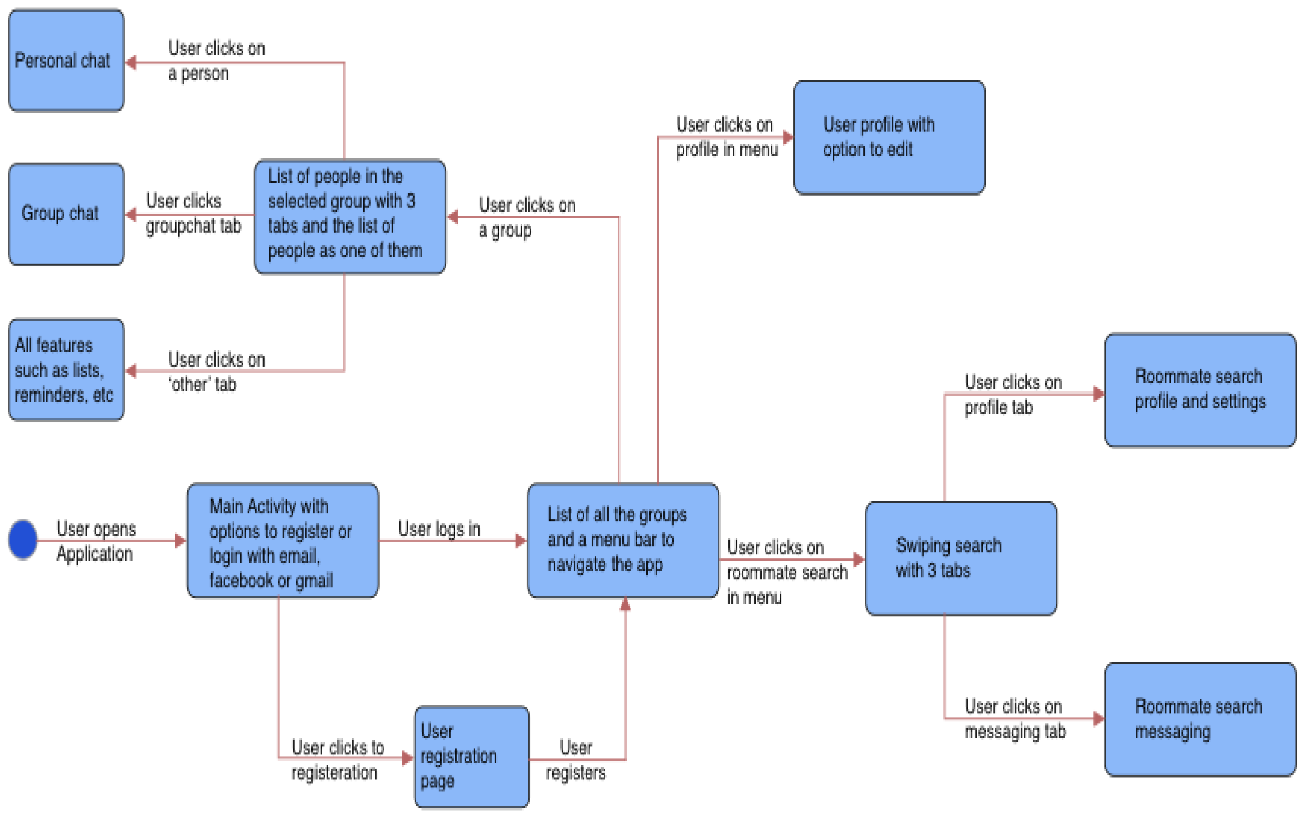
High Level Overview

The project will be an Android app that aims to have a clean, focused platform for all the needs of living in an apartment such as splitting bills, assigning chores, knowing each other's schedule, and having a grocery list along with a messaging feature. The application will follow a client-server architecture in which the server will be able to satisfy requests from multiple clients, query the database, and respond to the client with all the necessary information.



1. Client
    a. Provides a user interface for the user to interact with the backend.
    b. Sends XMLHttpRequests or AJAX requests to the server.
    c. Processes the response received from the server and updates the UI accordingly.
2. Server
    a. Receives, validates, and handles XMLHttpRequests or AJAX requests from the clients
    b. Queries the database for information and modifies the data stored if needed.
    c. Responds to the client with the appropriate information.
3. Database
    a. Firebase Cloud Firestore stores all the application data and user information.
    b. Responds to queries to gather or modify data and sends the information back to the server.

Activity/ State Diagram


Personal chat
User clicks on a person
Group chat
User clicks groupchat tab
List of people in the selected group with 3 tabs and the list of people as one of them
User clicks on a group
All features such as lists, reminders, etc
User clicks on 'other' tab
User clicks on profile in menu
User profile with option to edit
User clicks on profile tab
Roommate search profile and settings
User opens Application
Main Activity with options to register or login with email, facebook or gmail
User logs in
List of all the groups and a menu bar to navigate the app
User clicks on roommate search in menu
Swiping search with 3 tabs
User clicks on messaging tab
Roommate search messaging
User clicks to registeration
User registration page
User registers

Design Issues

Functional Issues

What information do we need for signing up

1: username and password
2: email, password
3: username, password, email, phone number

We decided to use email and password to sign up new users because we learned from reviews from the currently existing apps that the users did not like to share a lot of information with these apps and services, so that ruled out the third option for us. When comparing the first and the second options it made more sense for us to sign up a new user with their email so that we could have that on record as a means to communicate account-related information with them while having a way to validate the user. We did not consider it necessary to ask for a real name while signing up for the app however, they would be required to provide their name for the roommate search. For the sake of sign up simplicity, we decided that at sign up all the information we needed from the user was email and password.

How do we acquire the location of the user and when is it used?

1: manual input
2: GPS

We decided to use both manual input and GPS for location services. This is because there are two different features that you can use that are used in different contexts. One of these features is the location for where a user is looking for an apartment in a certain location. We do not want to use GPS for this feature due to the possibility of a user looking for an apartment in an area that is different from where they currently are. To solve that issue, we will have the ability to choose where to search for roommates. The second feature is the location based notifications for objects that are on the shopping list. This requires live location data, so GPS is the best option for it.

How do we allow users to chat with each other

1: instant messaging
2: real-time chat room

Due to one of the main focuses of the app being communication between its members, messaging is a concern. Instant messaging has a lot in common with a real-time chat room, however, you need to be in the chat room to receive these messages. We want the messages accessible at all times, so a user does not need to enter and leave a room. This is achieved by having instant messaging and storing the messages on a server, and possibly having them also stored locally to view while out of connection.

Non-Functional Issues

Deciding what framework to use

1: NativeScript
2: React
3. Android

Once we settled on an idea for our project, we needed to decide upon a framework to build it on. We decided to change our project to being just an android app after spending a week researching NativeScript and realizing that it is not as simple as it sounded. Some members of our group already were familiar with android development, so we changed in order to catch up to our deadline.

What database system to use

1: MySQL
2: MongoDB
3: FireBase

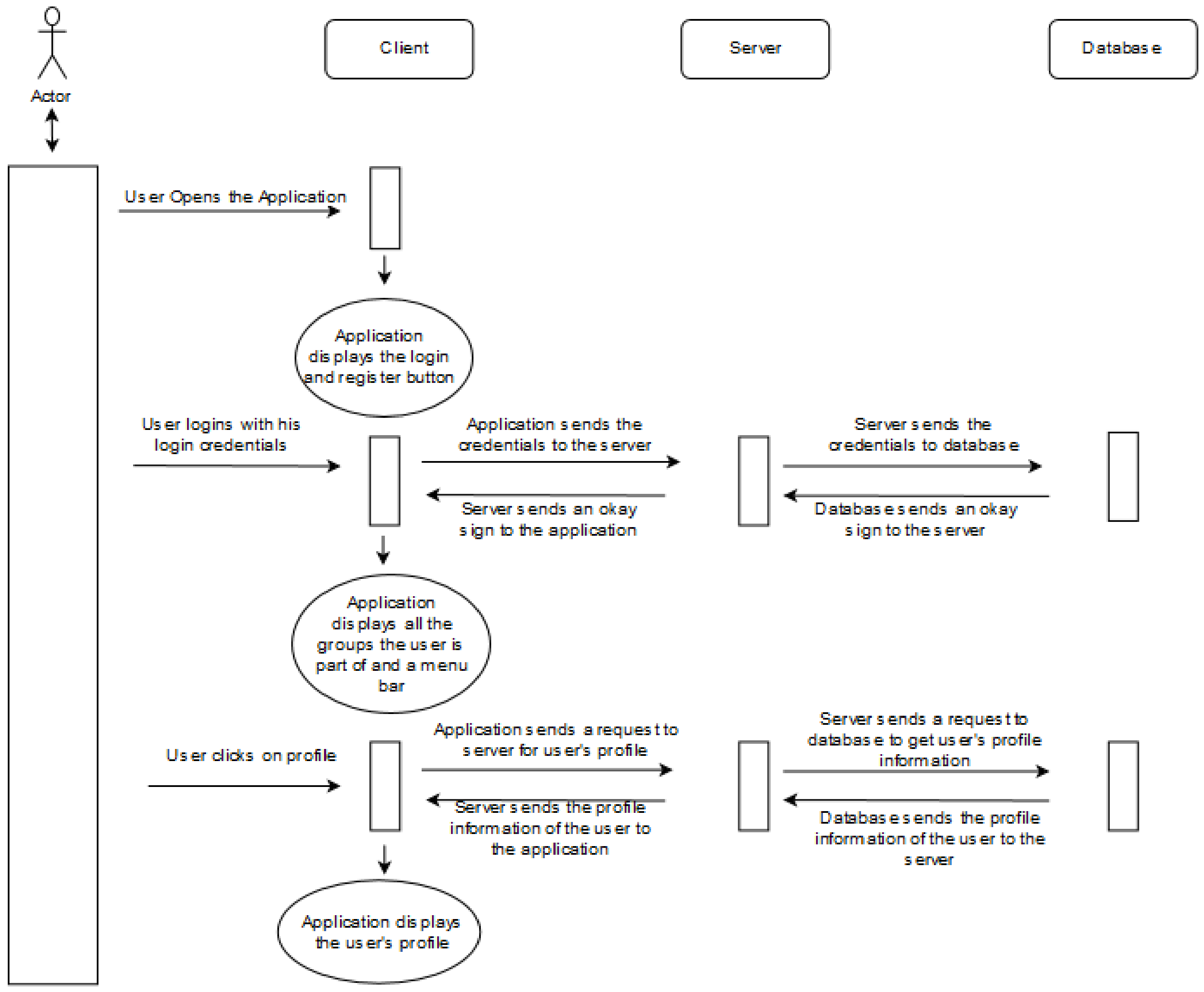
We were initially deliberating over the choice of our database but once the specifics of our project sinked in, FireBase was the obvious pick. It is cloud based which serves our purpose perfectly and hence the choice. Firebase provides certain algorithms that might prove useful. It also provides facebook and google login methods. Firebase also allows us, as maintainers to disable certain users for malpractice.

What API to use for location data
1: Google Maps API
2: HTML Geolocation API

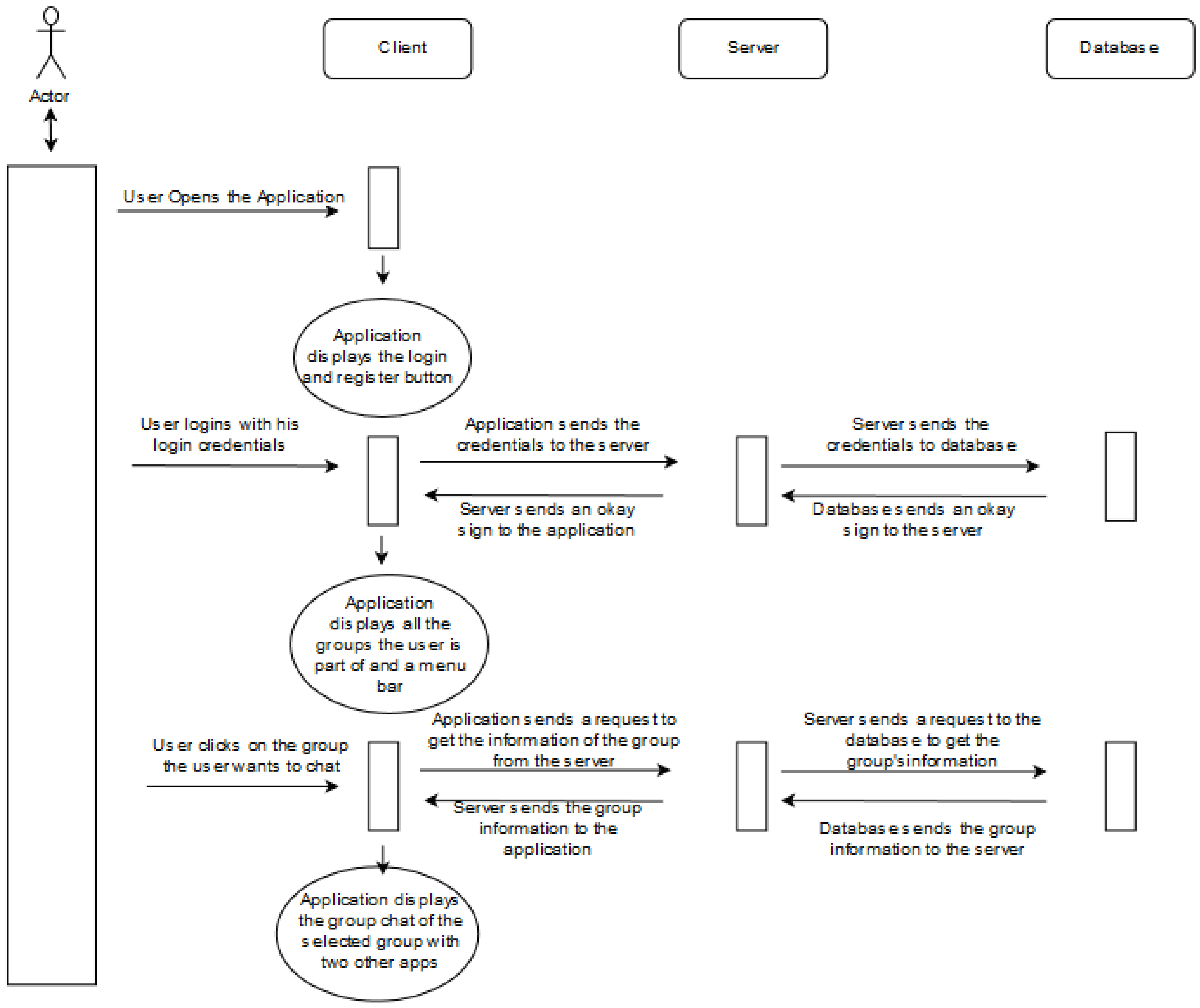
The HTML Geolocation API is obsolete and not used very often anymore. The Google Map API on the other hand is much more widely used, and is less intrusive about the data used to obtain the location. Google Maps API uses the wifi and cell tower data to find location whereas the HTML Geolocation API hands over control of the internet and physical inputs to the device in order to find the location.

Design Details

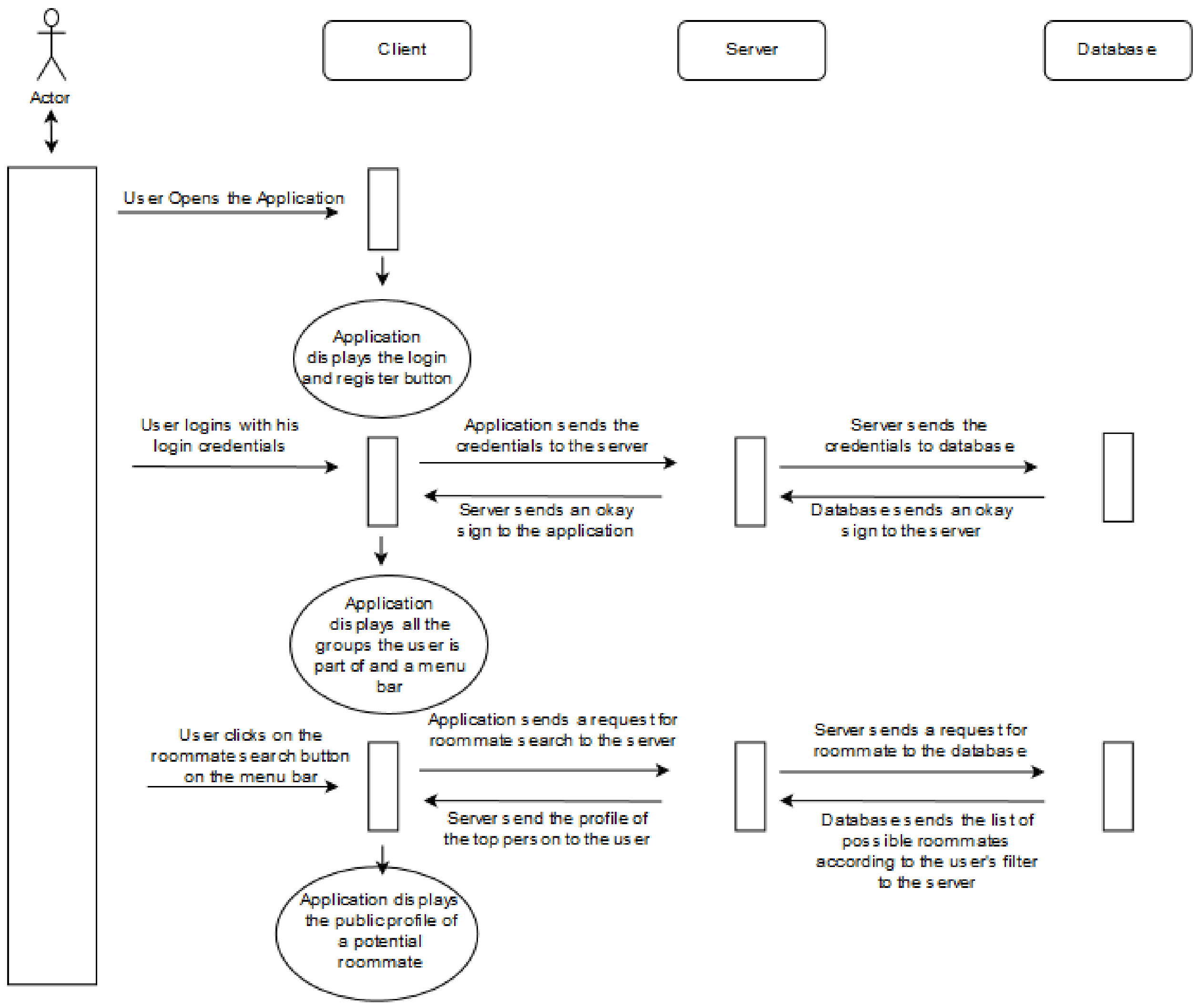
This sequence diagram shows the sequence of events when user wants to check his profile.


Actor
Client
Server
Database
User Opens the Application
Application displays the login and register button
User logins with his login credentials
Application sends the credentials to the server
Server sends an okay sign to the application
Server sends the credentials to database
Database sends an okay sign to the server
Application displays all the groups the user is part of and a menu bar
User clicks on profile
Application sends a request to server for user's profile
Server sends the profile information of the user to the application
Server sends a request to database to get user's profile information
Database sends the profile information of the user to the server
Application displays the user's profile

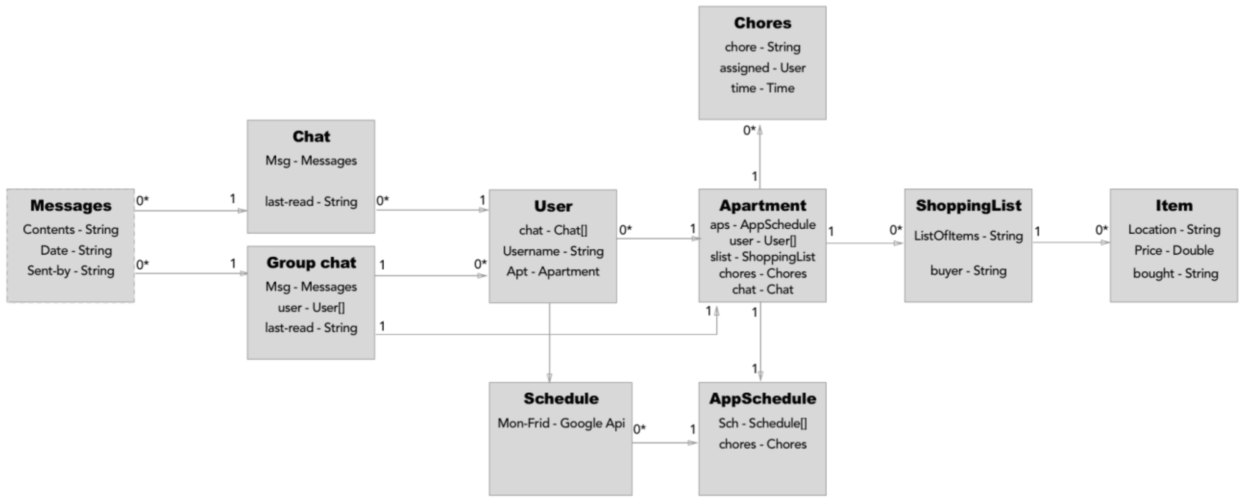
This sequence diagram shows the sequence of events when user wants to chat with a certain group the user is part of.


Actor
Client
Server
Database
User Opens the Application
Application displays the login and register button
User logins with his login credentials
Application sends the credentials to the server
Server sends the credentials to database
Server sends an okay sign to the application
Database sends an okay sign to the server
Application displays all the groups the user is part of and a menu bar
User clicks on the group the user wants to chat
Application sends a request to get the information of the group from the server
Server sends a request to the database to get the group's information
Server sends the group information to the application
Database sends the group information to the server
Application displays the group chat of the selected group with two other apps

This sequence diagram shows the sequence of events when user wants to see potential roommates.


Actor
Client
Server
Database
User Opens the Application
Application displays the login and register button
User logins with his login credentials
Application sends the credentials to the server
Server sends the credentials to database
Database sends an okay sign to the server
Server sends an okay sign to the application
Application displays all the groups the user is part of and a menu bar
User clicks on the roommate search button on the menu bar
Application sends a request for roommate search to the server
Server sends a request for roommate to the database
Database sends the list of possible roommates according to the user's filter to the server
Server send the profile of the top person to the user
Application displays the public profile of a potential roommate

Class Level Diagram



Description of Classes

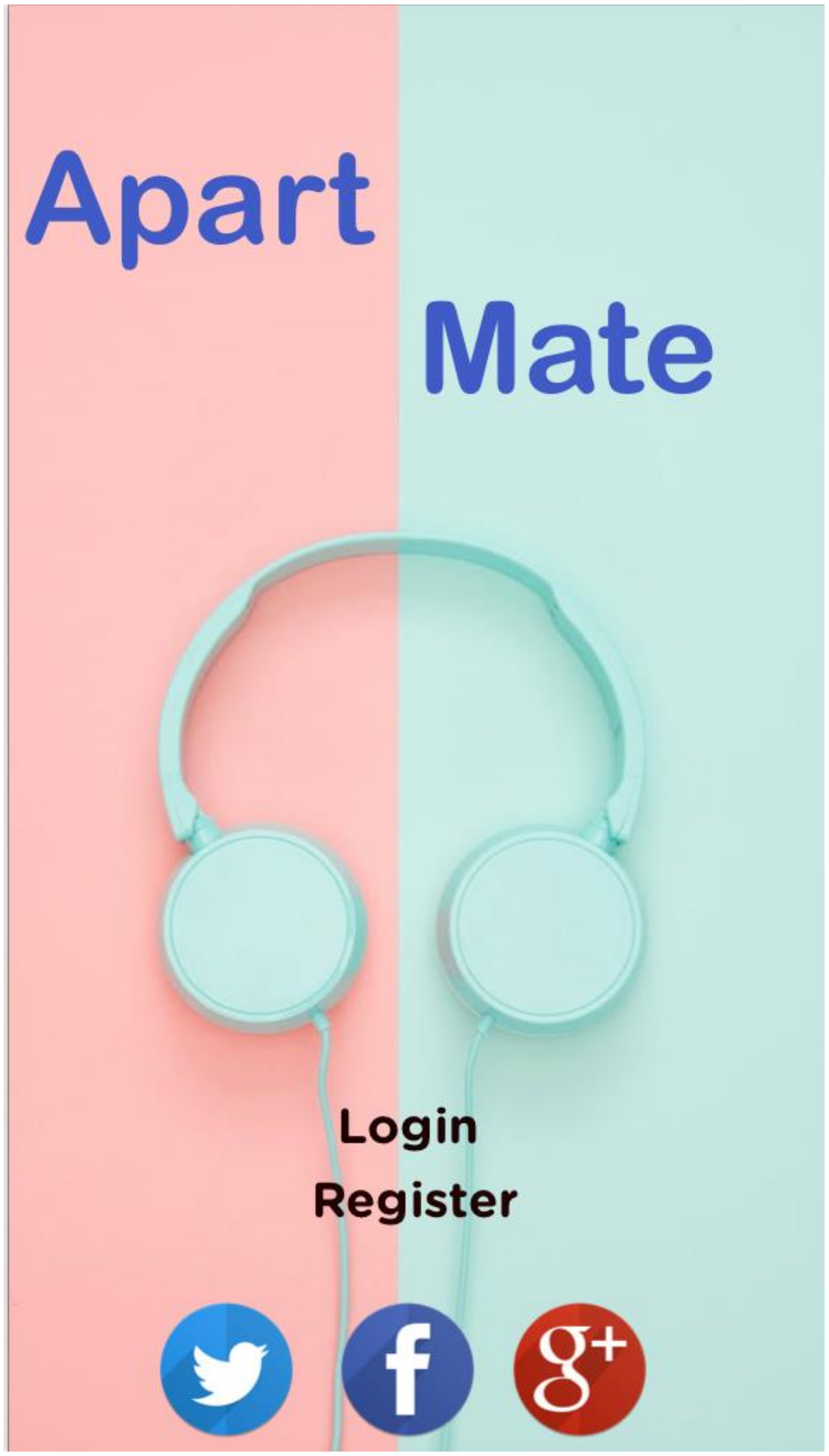
- Messages
  - Each message will have the content of the message which will be of the type String
  - Each message will have the date and time
  - Each message will show who sent it
- Chat
  - Each chat will have list of Messages
  - Each chat will have the user
  - Each chat will show the last-read by User
- GroupChat
  - Each GroupChat will have list of Messages
  - Each GroupChat will have a list of Users
  - Each GroupChat will show the last-read by Users
- User
  - Each User has a group of Chats
  - Each User has a username
- Schedule
  - The scheduling will be done using the Google API.

- Chores

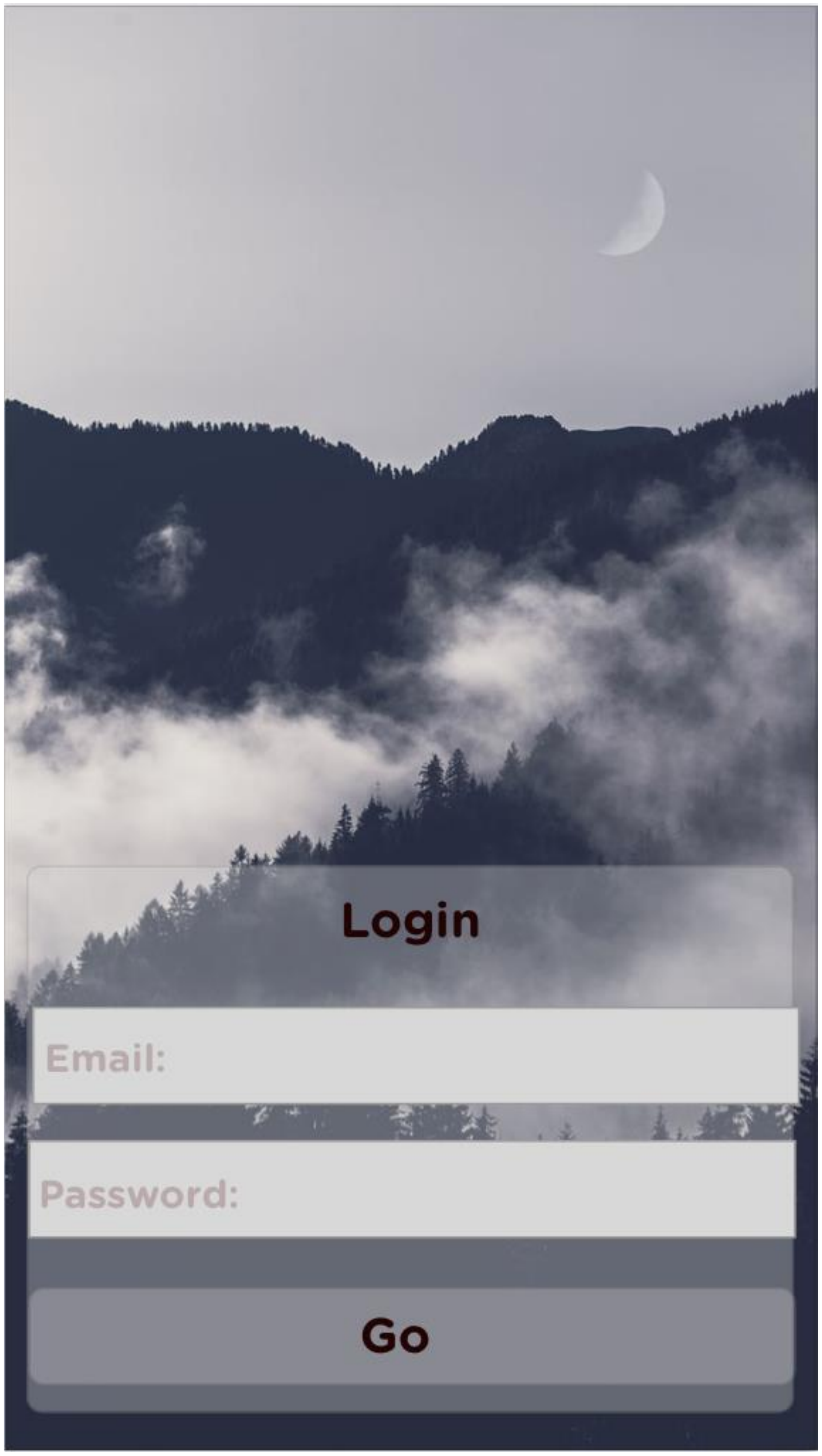
    - Each Chore has a string that contains what the chore is.
    - Each Chore has a assigned which tells us which user it is assigned to.
    - Each Chore has a time when the app sends a notification to the assigned person.
- Apartment
    - This is the overarching class for the apartment
    - Contains the apartment schedule, list of users, shopping list, chores, and group chat
- AppSchedule
    - Linked to the user's schedule to make an overarching schedule
    - Obtains chore information from the apartment in order to add the chores to the schedule
- Shopping List
    - Each shopping list has a list of items which is a string.
    - Each shopping list has a buyer which is of type string.
- Item
    - Each item has a location using Google Maps
    - Each item has a price which will be of int type.

UI Mockups

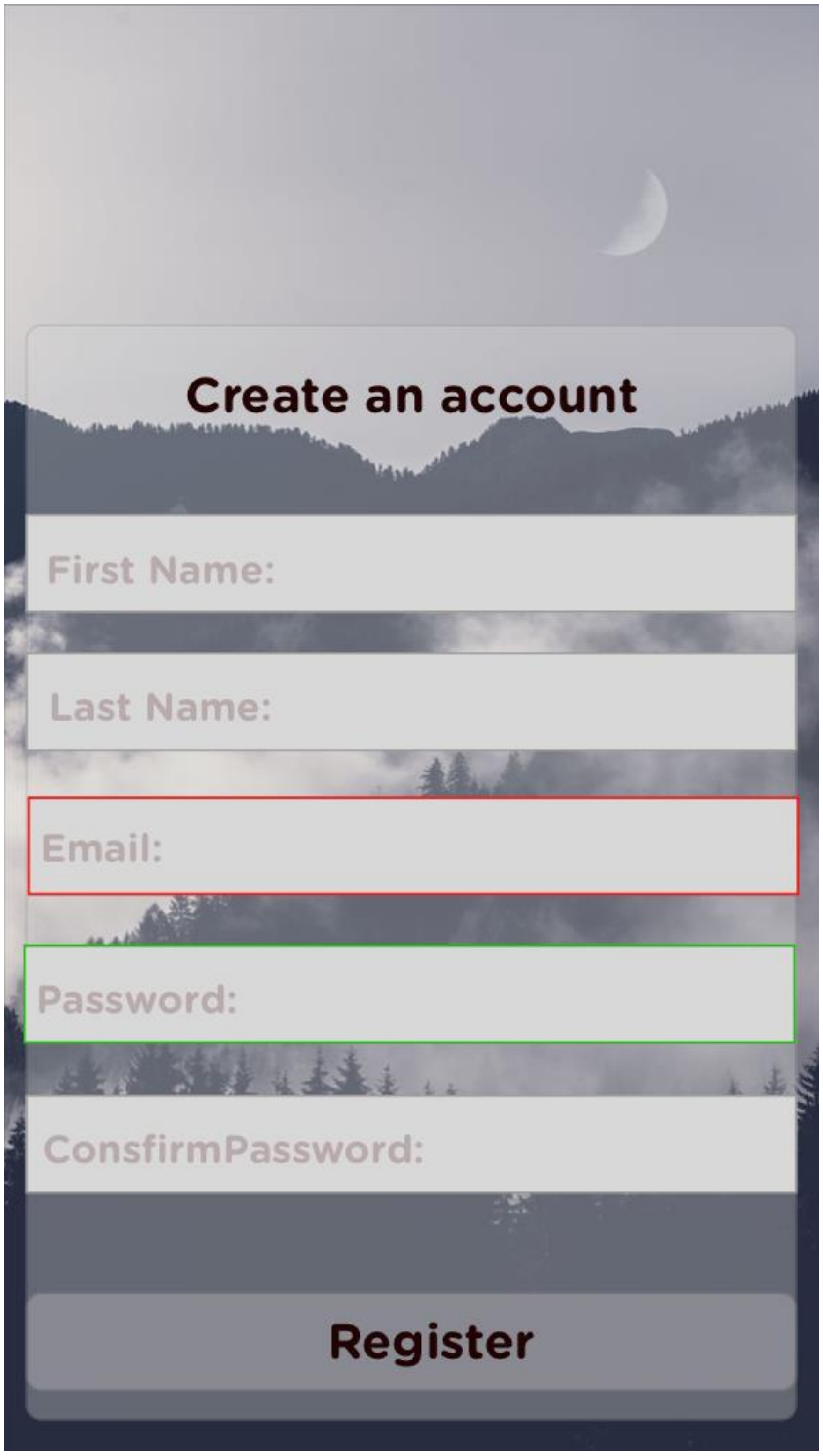
Home page - Users can login through their email, Facebook, Twitter or Google. Or they can register as new Users



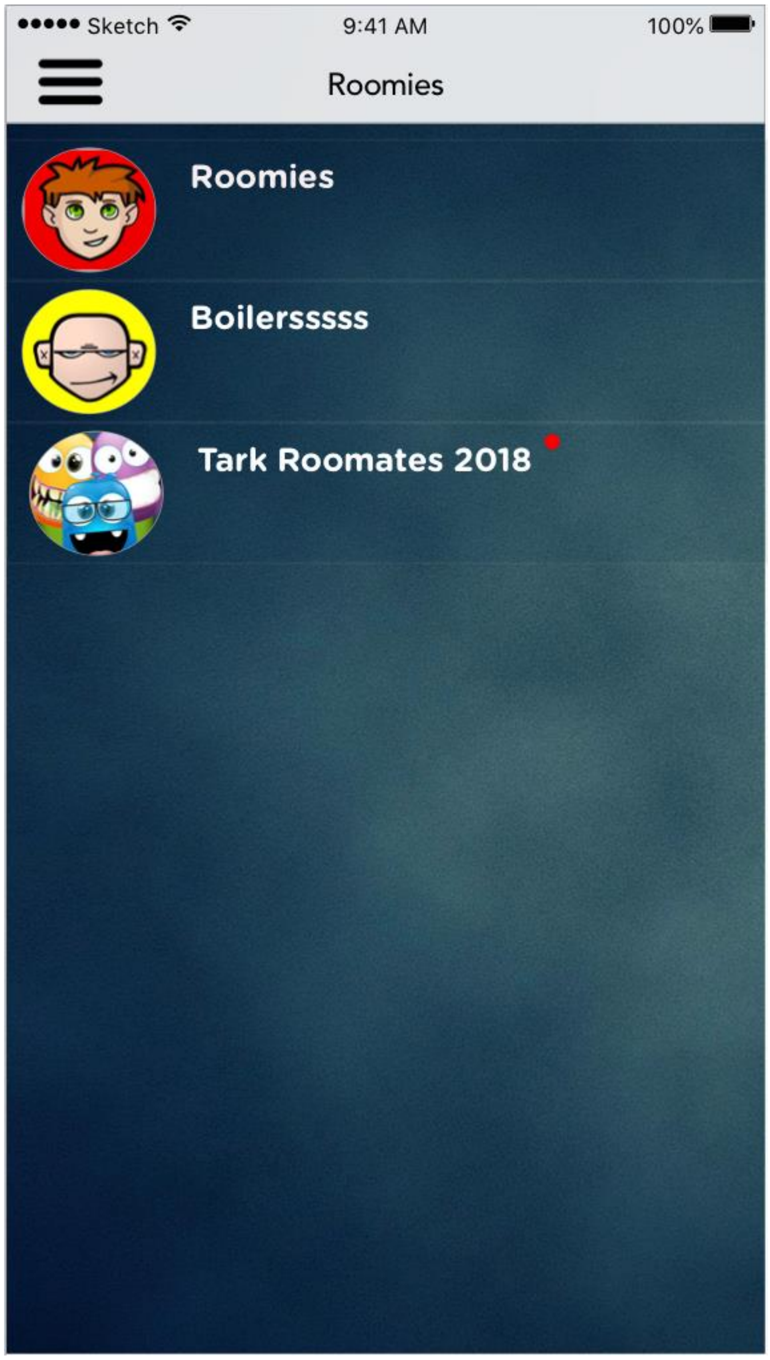
Login Page - Users can login through their email and password



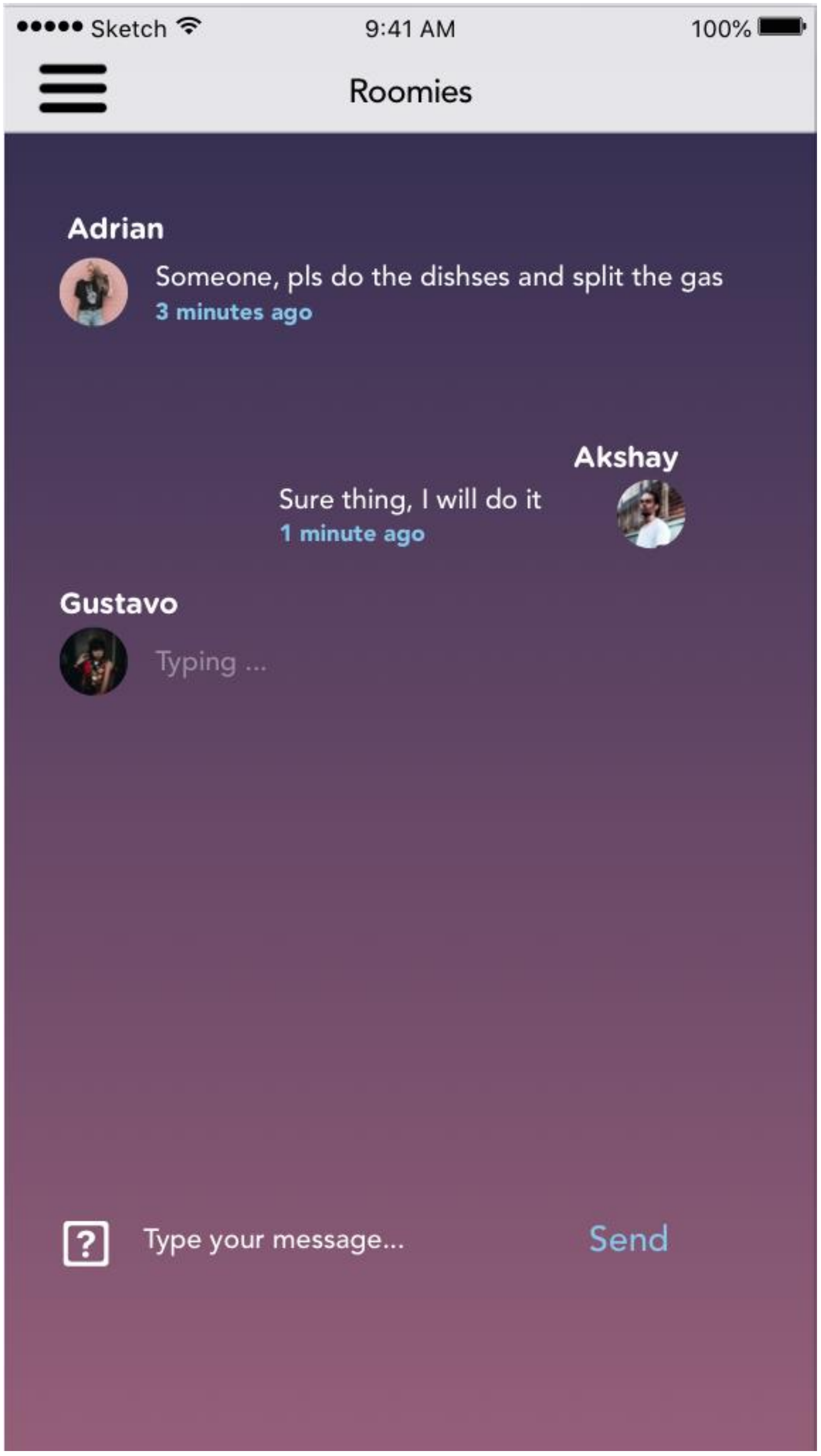
Register Page - Users can register through this page



Message Page - Users can see their group messages



Chat Page - Users can chat with each other in a group



Adrian

Someone, pls do the dishses and split the gas

3 minutes ago

Akshay

Sure thing, I will do it

1 minute ago

Gustavo

Typing ...