***Software Requirement and Design***

***Specifications***

***[Rent-A-Roo]***

***Version: [1.0.0]***

|  |  |
| --- | --- |
| *Course Code* | CS3004 |
| *Instructor* | Miss Nida Munawar |
| *Project Team* | 1. Ali Shah Naushad (20k-1078) 2. Muhammad Qasim Fuzail (20k-0157) 3. Syed Ahmed Mehmood (20k-0153) |
| *Submission Date* | 8-October-2022 |

***Table of Contents***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [*1.*](#page5) | [*INTRODUCTION.....................................................................................................................................................*](#page5) | | | [*4*](#page5) |
|  | [1.1.](#page5) |  | [Purpose of Document .........................................................................................................................](#page5) | [4](#page5) |
|  | [1.2.](#page5) |  | [Intended Audience ..............................................................................................................................](#page5) | [4](#page5) |
| [*2.*](#page6) | [*OVERALL SYSTEM DESCRIPTION ............................................................................................................................*](#page6) | | | [*5*](#page6) |
|  | [2.1.](#page6) |  | [Project Background.............................................................................................................................](#page6) | [5](#page6) |
|  | [2.2.](#page6) |  | [Project Scope ......................................................................................................................................](#page6) | [5](#page6) |
|  | [2.3.](#page6) |  | [Not In Scope ........................................................................................................................................](#page6) | [5](#page6) |
|  | [2.4.](#page6) |  | [Project Objectives ...............................................................................................................................](#page6) | [5](#page6) |
|  | [2.5.](#page6) |  | [Stakeholders ........................................................................................................................................](#page6) | [5](#page6) |
|  | [2.6.](#page6) |  | [Operating Environment ......................................................................................................................](#page6) | [5](#page6) |
|  | [2.7.](#page6) |  | [System Constraints .............................................................................................................................](#page6) | [5](#page6) |
|  | [2.8.](#page6) |  | [Assumptions & Dependencies ..........................................................................................................](#page6) | [6](#page6) |
| [*3.*](#page7) | [*EXTERNAL INTERFACE REQUIREMENTS..................................................................................................................*](#page7) | | | [*7*](#page7) |
|  | [3.1.](#page7) |  | [Hardware Interfaces ...........................................................................................................................](#page7) | [7](#page7) |
|  | [3.2.](#page7) |  | [Software Interfaces .............................................................................................................................](#page7) | [7](#page7) |
|  | [3.3.](#page7) |  | [Communications Interfaces ...............................................................................................................](#page7) | [7](#page7) |
| [*4.*](#page8) | [*FUNCTIONAL REQUIREMENTS................................................................................................................................*](#page8) | | | [*8*](#page8) |
| [*4.1.*](#page8) | | [*FUNCTIONAL HIERARCHY .............................................................................................................................*](#page8) | | [*8*](#page8) |
|  | [4.2.](#page8) |  | [Use Cases ............................................................................................................................................](#page8) | [9](#page8) |
|  | [4.2.1.](#page8) | | [[Use case Descript] ...........................................................................................................................](#page8) | 10 |
| [*5.*](#page9) | [*NON-FUNCTIONAL REQUIREMENTS........................................................................................................................*](#page9) | | | [*28*](#page9) |
|  | [5.1.](#page9) |  | [Performance Requirements ...............................................................................................................](#page9) | [28](#page9) |
|  | [5.2.](#page9) |  | [Safety Requirements ..........................................................................................................................](#page9) | [28](#page9) |
|  | [5.3.](#page9) |  | [Security Requirements .......................................................................................................................](#page9) | [28](#page9) |
|  |  |  |  |  |
|  | [5.4.](#page9) |  | [User Documentation ...........................................................................................................................](#page9) | [28](#page9) |
| [*SDS .........................................................................................................................................................................*](#page10) | | |  |  |
| [***6.***](#page11) | [***SYSTEM ARCHITECTURE ............................................................................................................................***](#page11) | | | [***30***](#page11) |
| [*6.1.*](#page11) | | [*SYSTEM LEVEL ARCHITECTURE ...................................................................................................................*](#page11) | | [*30*](#page11) |
| [*6.2.*](#page11) | | [*SOFTWARE ARCHITECTURE .........................................................................................................................*](#page11) | | [*31*](#page11) |
| [***7.***](#page12) | [***DESIGN STRATEGY .......................................................................................................................................***](#page12) | | | [***32***](#page12) |
| [***8.***](#page13) | [***DETAILED SYSTEM DESIGN .......................................................................................................................***](#page13) | | | [***33***](#page13) |
| [*8.1.*](#page13) | | [*DATABASE DESIGN .....................................................................................................................................*](#page13) | | [*33*](#page13) |
| [*9.*](#page15) | [*APPLICATION DESIGN .........................................................................................................................................*](#page15) | | | [*43*](#page15) |
| [*10. REFERENCES ......................................................................................................................................................*](#page15) | | | | [*47*](#page15) |
| [*11. APPENDICES .......................................................................................................................................................*](#page17) | | | | [*47*](#page17) |

1. ***Introduction***
   1. ***Purpose of Document:***

*The purpose of this document is to describe the Airbnb platform. This platform will prepare an accurate imitation to Airbnb, how person host his/her property for others to rent and whole procedure guest and host follows during the rental and booking process. In this document, the necessary explanations and models will be presented in detail for better understanding of the project. Additionally, in these models, the features of the users who can access the platform will be explained.*

* 1. ***Intended Audience:***

*This document is intended to be use by people who are involved in the creation of this project and its evaluation namely Project team members and Course Instructor.*

* 1. ***Definition of Terms, Acronyms and Abbreviations:***

|  |  |
| --- | --- |
| Terms, Acronyms and Abbreviations | Definition |
| Host | Person who list is property to be rented by others. |
| Guest | Person who rents the property listed on System. |
| Listing | Property which was added by Host and are to be rented. |
|  |  |

* 1. ***Document Conventions:***

*This document will be using Arial as its primary font. Main Heading will be of font size 16 with text being bold and Italic, All the sub-headings will be of font size 12 with text being bold and Italic and Content or body of the heading will be of font size 10 with text being Italic.*

1. ***Overall System Description***
   1. ***Project Background:***

*Airbnb is a two-sided marketplace that seeks to match people who own real estate properties with people interested in renting short-term lodging. Airbnb is one of the most successful booking services these days, reaching $110 billion in current valuation thanks to people's willingness to travel and live with locals in a new city. According to Statista, it's among the top 5 most downloaded travel apps worldwide, with 42 million downloads by 2020. Currently, a huge market in Pakistan doesn’t use apps like Airbnb; the norm is to book hotels, which are costly, so to capitalize on this huge market, we thought to build this project.*

* 1. ***Project Scope:***

*Scope of the Rent-A-Roo project is to provide user specifically of Pakistan a rental app, which allows user to host their property on to the app for other user to rent or they can rent any property which they want to rent. This app will provide easy and understandable graphical user interface. It will allow user to promote their listings and will also allow them to make any listing their favorite. Furthermore, it will allow user to change app language to what they desired. This app will provide a mean of easy and cost-effective accommodation.*

* 1. ***Not in Scope:***

*This app is not for the user outside Pakistan for now, it only caters customer who live within Pakistan.*

* 1. ***Project Objectives:***

*The objective of this project is to create an app which provides easy and cost-effective accommodation as compared to hotels which are quite expensive and are not always as shown on website. This app will allow users to find accommodations which are to their liking and will also show them reviews and rating so that they know what to expect from the property and its owner.*

* 1. ***Stakeholders:***

*Stakeholders in this project are team members, course instructors and expected stakeholders are people of Pakistan who will use this app.*

* 1. ***Operating Environment:***

*This software will run in hp aero 13 which contains AMD Ryzen 7 and 16gb of rams. This software will run on windows 11 on Vmware virtual machine containing Ubuntu 22.10. This software will run using fastapi local host and flutter host on google chrome web browser*

* 1. ***System Constraints:***

*This project is currently deployed using local host and is not up on a proper running server. Furthermore, the project is built as the project for Database Systems and Software Design and Analysis course so the scope of the project currently is small. The app is for Pakistanis so their will be language barrier for user who don’t speak Urdu and English.*

* 1. ***Assumptions and Dependencies:***

***Not Applicable***

1. ***External Interface Environment***
   1. ***Hardware Interface:***

*This software is executed on hp aero 13 and is an android application-based software which is made to be used on android smartphones, the app uses touchscreen of android smartphone to navigate, perform tasks which user wants to perform, to enter user or listing data and uses android notification manger for any notification and updates.*

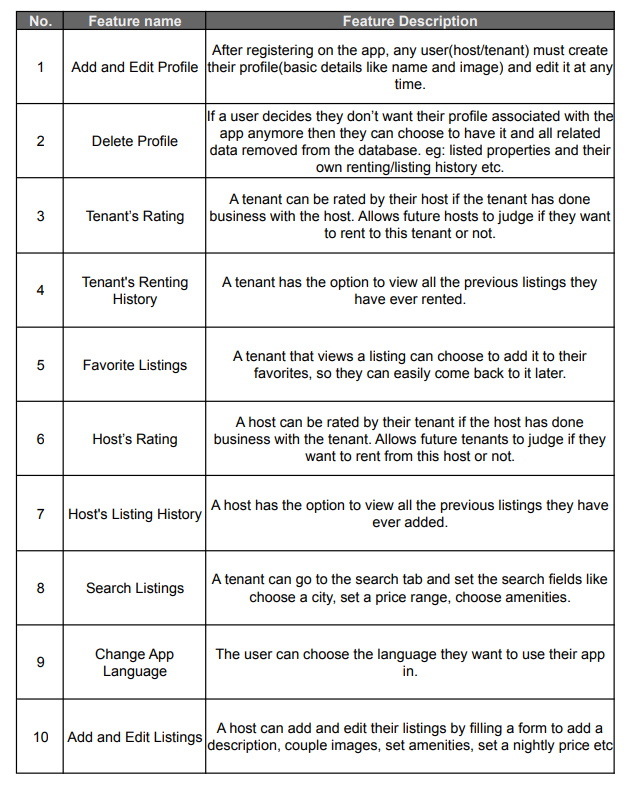
* 1. ***Software Interface:***

*This software uses FASTAPI local host server for the PostgreSQL database and uses flutter local host server to display fronted of the app on google chrome web browser. This app uses python libraries like sqlalchemy, pydantic for database and its schemas and FASTAPI for using API to communicate between PostgreSQL database and our frontend application.*

* 1. ***Communication Interface:***

*This software uses google chrome to execute on it. It uses https protocols to communicate between API’s and database.*

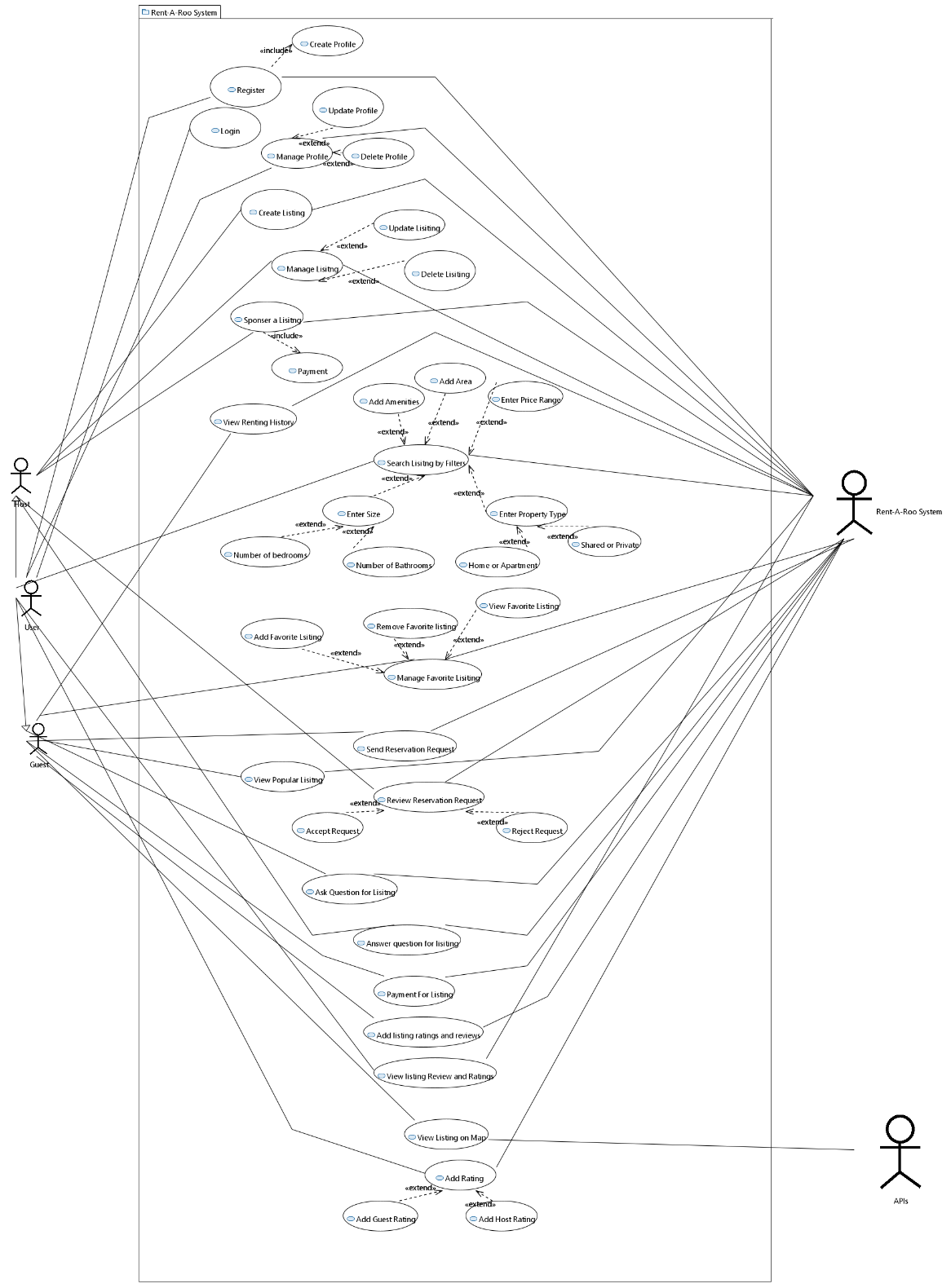
1. ***Functional Requirements***
   1. ***Functional Hierarchy:***



Table

Description automatically generated

* 1. ***Use Cases:***
     1. ***Use Case Diagram:***



* + 1. ***Use Case Description:***

|  |  |
| --- | --- |
| ID: | UC-1 |
| Use case name: | Add and Edit Profile |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | After registration on the app, any user(host/guest) can create his own profile (basic details like name and image) and edit it any time. |
| Primary Actor: | User |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User registered and logged on to app. |
| Postconditions: | User profile created and edited |
| Main  Success Scenario: | 1. User selects profile from menu. 2. User enters details which needs to be edited. 3. User selects submit 4. System registers the details and update it |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User enters wrong datatype in field.  2a1. User will be prompted about wrong datatype.  2a2. User either backs out or continue after entering correct details. |

|  |  |
| --- | --- |
| ID: | UC-2 |
| Use case name: | Delete Profile |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | If a user decides they don’t want their profile associated with the app anymore then they can choose to have it and all related data removed from the database e.g: listed properties and their own renting etc. |
| Primary Actor: | User |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | User deleted successfully |
| Main  Success Scenario: | 1. User selects delete profile 2. System prompts for confirmation 3. User confirms 4. System delete all the relevant details associated to user from database. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  1a. User choose to delete profile during renting or transaction period  1a1. System will prompt an error message and will back user out. |

|  |  |
| --- | --- |
| ID: | UC-3 |
| Use case name: | Guest Ratings |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A guest can be rated by their host if the guest has done business with the host. Allows future hosts to judge if they will allow this guest to rent their place. |
| Primary Actor: | User(Host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and is rating guest who has rented his place |
| Postconditions: | Guest rated |
| Main  Success Scenario: | 1. Host chose to rate guest. 2. Host selects stars he want to give to user 3. System saves guest rating and update it in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-4 |
| Use case name: | Guest Renting History |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A tenant has an option to view all the previous listings they have ever rented. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Renting history showed successfully |
| Main  Success Scenario: | 1)Guest selects to view all his previous renting history  2) System list all the previous renting from database having guest id same as current guest. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. No renting history found. |

|  |  |
| --- | --- |
| ID: | UC-5 |
| Use case name: | Favorite Listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A guest that views a listing can choose to add it to their favorites, so they can easily come back to it later. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and currently viewing a listing |
| Postconditions: | Listing added as favorite |
| Main  Success Scenario: | 1. User views a listing and likes it 2. User selects to add it as favorite 3. System will save user choice and update database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-6 |
| Use case name: | Host Rating |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A host can be rated by the guest if the host has done business with the guest. Allows future guest to judge if they want to rent from this host. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and has done business with the host |
| Postconditions: | Host Rated Successfully |
| Main  Success Scenario: | 1. Guest chose to rate host 2. Guest chose number of stars he wants to give to host 3. System saves user choice and update host rating in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-7 |
| Use case name: | Host’s Listings |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | Host will have option to view all the listing he/she have added. |
| Primary Actor: | User(Host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Listings viewed successfully |
| Main  Success Scenario: | 1. Host chose to view all his listings 2. System will get all the listings from database having host id equals to current host and display it. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-8 |
| Use case name: | Search Listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A guest can go to the search tab and set the filters to search desired listings |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Desired listing views successfully |
| Main  Success Scenario: | 1. Guest chose to search listing from menu. 2. Guest selects desired filters 3. Guest submits 4. System get all the listing with matching filters from database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  4a. No such listing found.  4a1. User backs out. |

|  |  |
| --- | --- |
| ID: | UC-9 |
| Use case name: | Change app language |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | The user can choose the language they want to use their app in. |
| Primary Actor: | User |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on. |
| Postconditions: | Language changed successfully. |
| Main  Success Scenario: | 1)User selects to change language from menu.  2) User selects the desired language.  3) User submits  4) System saves user choice and update itself |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. Cannot find desired language  2a1. User backs out |

|  |  |
| --- | --- |
| ID: | UC-10 |
| Use case name: | Add and Edit listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A host can add and edit their listings by filling form and add relevant detail to it. |
| Primary Actor: | User(Host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Listing added or edited successfully |
| Main  Success Scenario: | 1. Host choose to add or edit listing. 2. Host add relevant details for listing 3. Host selects submit. 4. System save user entered details and update it in database. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User enter wrong datatype in field or left required field out.  2a1. System prompt user by giving error message.  2a2. User continues by correcting his/her mistake or backs out |

|  |  |
| --- | --- |
| ID: | UC-11 |
| Use case name: | Delete Listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A host can unlist their property so that it doesn’t show up in search result but it will remain in listing history once added in database |
| Primary Actor: | User(host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Listing deleted Successfully |
| Main  Success Scenario: | 1. User select to delete listing from menu 2. User confirms to delete 3. System save user request and unlist the listing in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User chose to delete listing renting period.  2a1. System will show error message and will back user out. |

|  |  |
| --- | --- |
| ID: | UC-12 |
| Use case name: | Send a reservation request |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A guest can request to rent a property from its listing paper by specifying the date and duration of stay |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged in and is viewing listing he wants to rent |
| Postconditions: | Request send successfully |
| Main  Success Scenario: | 1. User selects to send request for reservation from menu. 2. User enters relevant details 3. User select submit 4. System saves user request in database and send it it listing host. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-13 |
| Use case name: | Review Request |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A host can review a request and choose to accept or decline the request |
| Primary Actor: | User(Host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Request accepted or declined |
| Main  Success Scenario: | 1. Host choose to review requests from menu. 2. Host choose to accept requests. 3. System save the user choice in database and also prompt the request sender about request being accepted |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User choose to reject the request.  2a1. System save the user choice in database and will also prompt the request sender about request being rejected. |

|  |  |
| --- | --- |
| ID: | UC-14 |
| Use case name: | Transaction |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | If a guest request is approved they will be prompt to confirm the reservation and pay for it using the provided options within a time limit. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and his request for reservation is accepted |
| Postconditions: | Transaction successful |
| Main  Success Scenario: | 1. User select transaction from menu. 2. User enter relevant details and amount for transaction 3. User select submit 4. Transaction get verified. 5. System save transaction in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User enter invalid details.  2a1. System prompt user of invalid credentials.  2a2. User will either continue after entering valid details or will back out.  2b. User enter invalid amount  2b1. System will prompt user of invalid amount.  2b2. User will either enter valid amount and continue or will back out. |

|  |  |
| --- | --- |
| ID: | UC-15 |
| Use case name: | Question about listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A question and answers section is provided on a listing page where guest can ask questions about a listing for host to answer. |
| Primary Actor: | User(guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and viewing a listing. |
| Postconditions: | Question asked successfully |
| Main  Success Scenario: | 1. Guest select to ask question from menu 2. Guest enter the question 3. Guest select submit 4. System save guest question in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. Entered question exceed word limit.  2a1. System will prompt user to enter question within word limit  2a2. User will continue after correcting his mistake or will back out. |

|  |  |
| --- | --- |
| ID: | UC-16 |
| Use case name: | Answer Question |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | In the question and answer section, host will provide guest questions with answers |
| Primary Actor: | User(host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and question is asked |
| Postconditions: | Question answered successfully |
| Main  Success Scenario: | 1)Host selects answer from menu  2) Host enters his answer  3) Host select submit  4) System save user answer for the question in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. Host answers word limit exceed.  2a1. System will prompt user about exceeding word limit.  2a2. User will correct his/her mistake and continue or will back out. |

|  |  |
| --- | --- |
| ID: | UC-17 |
| Use case name: | Add Review and Rating |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A guest that has stayed at a property can write a review and rate it on listing page. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and completed his renting period |
| Postconditions: | Review and Rating added successfully |
| Main  Success Scenario: | 1)User selects to add review and rating from menu.  2) User select stars to give to listing  3) User enters his review  4) User select submit  5) System saves user rating and reviews in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  3a. User review word limit exceeds.  3a1. System will prompt user about exceeding word limit  3a2. User will either correct his mistake and continue or will back out. |

|  |  |
| --- | --- |
| ID: | UC-18 |
| Use case name: | View reviews and ratings |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | Any user can view the reviews and rating of a property on its listing page. |
| Primary Actor: | User(Guest) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on and is viewing listing |
| Postconditions: | Ratings and reviews displat |
| Main  Success Scenario: | 1. User selects to view listing review and rating from menu 2. System will get review and rating of the listing from the database. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-19 |
| Use case name: | View property on a map |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A user can select option to view property on google maps |
| Primary Actor: | User |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Location viewed on map successfully |
| Main  Success Scenario: | 1. User selects to view listing on maps 2. System will redirect user to google maps and display listing on it. |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-20 |
| Use case name: | View Trending listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A page where the most viewed listings on the app are displayed |
| Primary Actor: | User |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged on |
| Postconditions: | Trending listing viewed successfully |
| Main  Success Scenario: | 1. User selects to view trending listing from menu 2. System will get all the listings with most view count from database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state. |

|  |  |
| --- | --- |
| ID: | UC-21 |
| Use case name: | Promote Listing |
| Author | Ali Shah, Ahmed, Qasim |
| Last revision | 11/7/2022 |
| Description: | A host can choose to pay the company, on a listing management page, to promote his/her listing |
| Primary Actor: | User(Host) |
| Supporting Actor | Rent-A-Roo System |
| Preconditions: | User logged in and is currently on listing he wants to promote |
| Postconditions: | Listing Promoted Successfully |
| Main  Success Scenario: | 1. User selects to promote listing from menu 2. User proceed to payment 3. User pays 4. System save user’s listing as promoted in database |
| Extensions: | \*a. At any time, System crashes:       In order to support recovery, ensure all sensitive state and events can be recovered at any step in the scenario.                1. User restarts the System, logs in, and      requests recovery of prior state. System reconstructs prior state.  2a. User enter invalid credentials for payment transaction or enter invalid amount  2a1. System prompt user with an error message.  2a2. User will either correct his mistake and continue or will back out |

1. ***Non-functional Requirements***
   1. ***Performance Requirements:***

*This system is currently running on a local host because of which it cannot be use by multiple use but when it will be properly deployed it around 1000 users can access application and its functionality concurrently. Average response time or latency of the app is 0.0275 ms. This app work with data in precise and concise manner.*

* 1. ***Safety Requirements:***

*There may be some property damage or maybe loss of credibility because of guest being aggressive to the property and adding fake reviews, so to ensure that this never happens only user who have rented the property are allowed to add review and rating of property and host and secondly, host will be able to rate its guest so this will allow other users to check the persons characteristic and credibility before the rent or give their property to him. The system must have way to dial emergency numbers if any illegal activity was found in property during the guest stay and a way to reimburse the host of potential damage.*

* 1. ***Security Requirements:***

*There may be privacy and integrity violation while renting the property as stranger is accessing private space of the user. While using the user privacy and integrity are safe and secure. Private details like passwords are hidden from others and this system uses JWT tokens and HS256 algorithm to encrypt user passwords and to ensure that only an authorize person can access or change the data.*

* 1. ***User Documentation:***

***NOT APPLICABLE***

***SDS***

1. ***System Architecture***
   1. ***System Level Architecture:***

Diagram

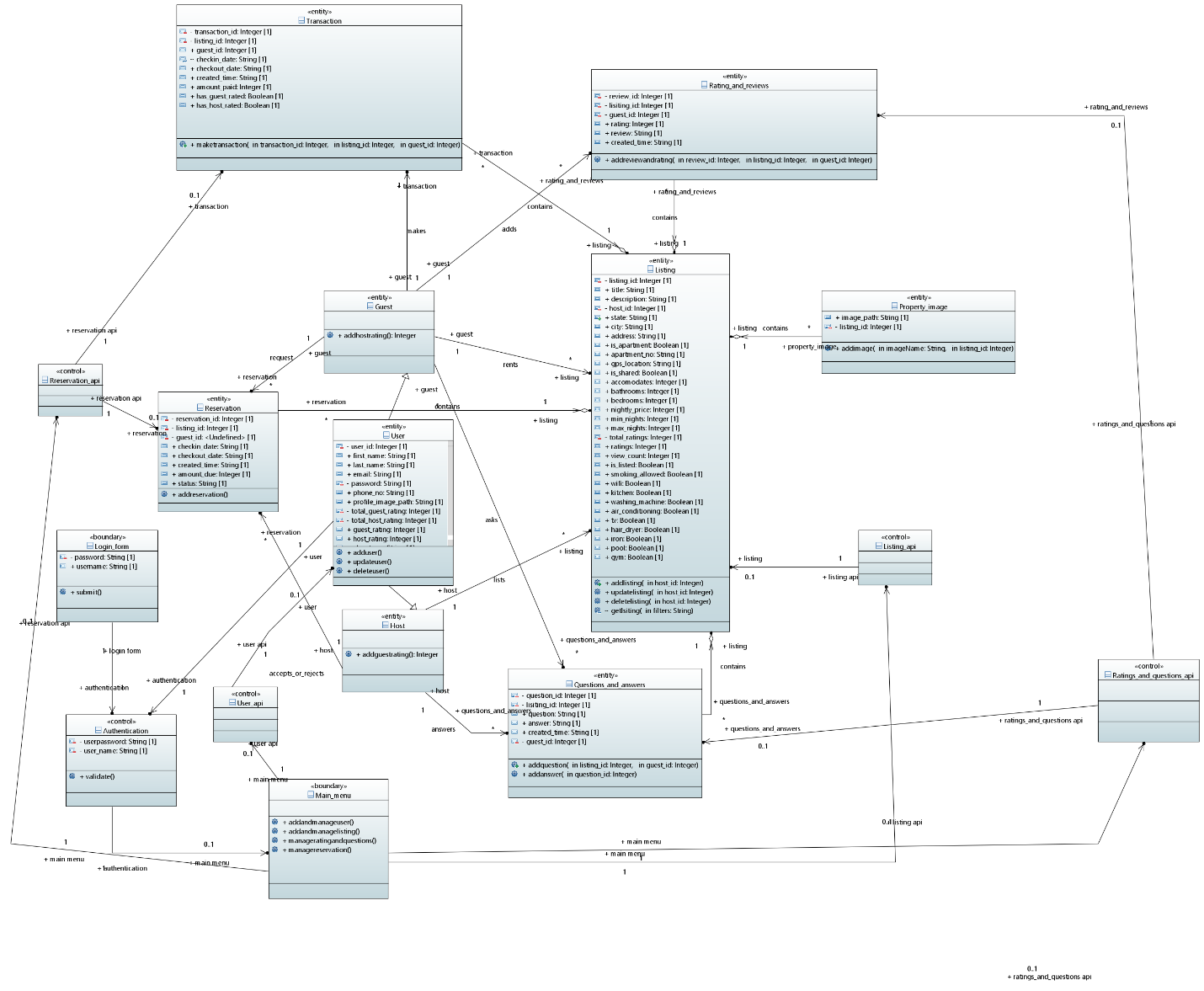
Description automatically generated

Diagram, engineering drawing

Description automatically generated

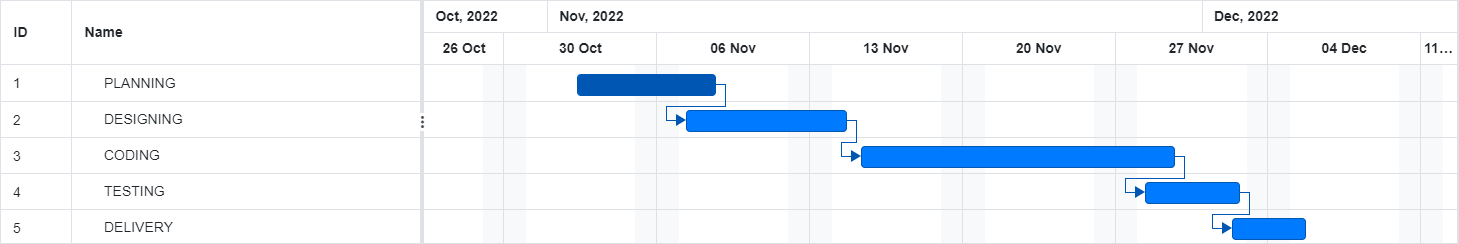
*The system use to execute the software is computer system consisting of keyboard , screen and trackpad to control or enter the data. The software is run using visual studio code containing the libraries which includes FastAPI a python framework, sqlalchemy, pydantic which are used to create APIs to manipulate database which in connected to our flutter frontend which user uses to interact. This software uses HTTP protocols and is executed on web browser using local host.*

* 1. ***Software Architecture:***



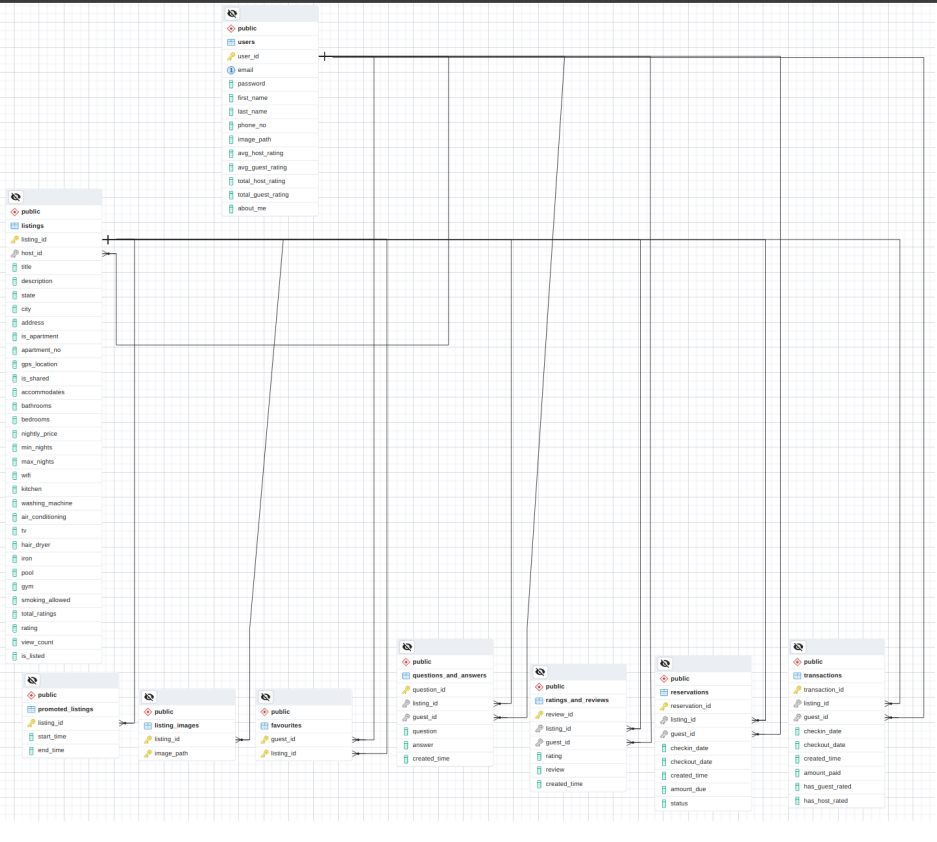
*The classes which are entities in our database contains the above mentioned data and perform above mentioned operations. The user land on login form where he enters his credentials which are verified against database records, after this he lands on main menu from where he can navigate to different pages like transaction, listing, reservations etc and can perform task on them using frontend buttons and bars which are connected to database using API’s.*

1. ***Design Strategy***



*During first 6 days we collected the information related to our project and problem statement and design a proposal which work as a ground work of what our project will do and what technology will be used. Then UML diagrams were generated to get an idea of how our system will iteract with different hardware, software and its user and what features it will provide. Next was the largest phase of coding where we setup work environments and database were built and connected using Api to the frontend and then testing was done with dummy data and next project was delivered.*

1. ***Detailed System Design***
   1. ***Database Design:***
      1. ***ER Diagram:***



* + 1. ***Data Dictionary:***
       1. *Data1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ***< Data 1>*** | | |  |  |
| ***Name*** |  |  | *listing* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  | *Write other names used for the first entry.* | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *This is use as an external entity which host user enters* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing content.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
| *Listing\_id* |  | | |  |  |  |  |  |  |
|  | *Unique id of listing to represent it* | |  | *Integer* |  | *-* | *No* | *-* | *Primary Key* |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *host\_id* |  | | |  |  |  |  |  |  |
|  | *Unique id of host who has listed this property* | |  | *Integer* |  |  |  | *NULL* | *Foreign Key* |
|  |  |  |  |  |  | *-* | *Yes* |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *title* | *Name for the listing* |  |  | *String* |  | *255* | *No* | *-* |  |
| *description* | *Details about the listing* |  |  | *string* |  | *255* | *No* |  |  |
| *State* | *State where property is* |  |  | *string* |  |  | *No* |  |  |
| *City* | *City where Property is* |  |  | *string* |  |  | *NO* |  |  |
| *address* | *Address of the property* |  |  | *string* |  |  | *No* |  |  |
| *Is\_apartment* | *Check whether it is apartment* |  |  | *bool* |  |  | *No* | *False* |  |
| *Apartment\_no* | *Number of apartment if it is* |  |  | *String* |  |  | *Yes* | *Null* |  |
| *Gps\_location* | *Location on Maps* |  |  | *String* |  |  | *Yes* | *Null* |  |
| *Is\_shared* | *Whether it is shared by others* |  |  | *bool* |  |  | *no* | *False* |  |
| *accommodates* | *How many people can stay* |  |  | *Integer* |  |  | *no* |  |  |
| *Bathrooms* | *Number of bathrooms* |  |  | *Integer* |  |  | *No* |  |  |
| *Bedrooms* | *Number of bedrooms* |  |  | *Integer* |  |  | *No* |  |  |
| *Nightly\_price* | *Price per night* |  |  | *Integer* |  |  | *No* |  |  |
| *Min\_night* | *Min night to stay* |  |  | *Integer* |  |  | *No* |  |  |
| *Max\_night* | *Max night to stay* |  |  | *Integer* |  |  | *no* |  |  |
| *Wifi* | *Wifi available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Kitchen* | *Kitchen available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Air\_conditioning* | *Air conditioned?* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Hair\_dryer* | *Hair dryer available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Iron* | *Iron available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Pool* | *Pool available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Gym* | *Gym available* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Smoking\_allowed* | *Smoking is allowed or not* |  |  | *Bool* |  |  | *No* | *False* |  |
| *Total\_rating* | *Sum of all rating* |  |  | *Integer* |  |  | *Yes* | *0* |  |
| *Rating* | *Avg Rating* |  |  | *Integer* |  |  | *Yes* | *0* |  |
| *View\_count* | *Number of Views* |  |  | *Integer* |  |  | *no* | *false* |  |
| *Is\_listed* | *Host has unlisted or not* |  |  | *Bool* |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 2>** | | |  |  |
| ***Name*** |  |  |  | | | | | | |
|  |  |  | *User* | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Use as an external entity representing host and guest* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing Guest and Host* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
| *User\_id* |  | |  |  |  |  |  |  |  |
|  |  |  |  | Integer |  |  |  |  | *Primary key* |
|  | Unique id of users |  |  |  |  |  | No |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *email* | *Username to login* | | | *String* |  |  | *No* |  |  |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Password | Password to login |  |  | String |  |  | No |  |  |
| Firsr\_name | Name of user |  |  | String |  |  | No |  |  |
| Last\_name | Name of user |  |  | String |  |  | No |  |  |
| Phone\_no | Contact details |  |  | String |  |  | No |  |  |
| Image\_path | Profile image location |  |  | String |  |  | Yes | Null |  |
| Avg\_host\_rating | Host rating as average |  |  | Integer |  |  | Yes | 0 |  |
| Avg\_guest\_rating | Guest rating as average |  |  | Integer |  |  | Yes | 0 |  |
| Total\_guest\_rating | Sum of all guest rating |  |  | Integer |  |  | Yes | 0 |  |
| Total\_host\_rating | Sum of all host rating |  |  | Integer |  |  | Yes | 0 |  |
| About\_me | Details about user |  |  | String |  |  | Yes | null |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 3>** | | |  |  |
| ***Name*** |  |  | *Listing Images* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Use to store images of listing* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing images of listing.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Listing\_id | Uniques id of listing |  |  | Integer |  |  | Yes |  | *Primary key* |
|  |  |  |  |  |  |  |  |  |  |
|  |  | | |  |  |  |  |  |  |
|  | *Image paths of images of listing* | |  | *String* |  |  | *No* |  | *Primary key* |
|  |  |  |  |  |  |  |  |  |  |
| Image\_path |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 4>** | | |  |  |
| ***Name*** |  |  | *Ratings and reviews* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Use to store ratings of listing and its review* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing reviews and rating of listing.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | | *Integer* |  |  |  |  |  |
|  |  | |  |  |  |  | *No* |  |  |
|  | Unique id of reviews |  |  |  |  |  |  |  | *Primary\_key* |
| Review\_id |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | *Unique id of listings* | | |  |  |  | *Yes* |  | *Foreign\_key* |
|  |  | |  | *Integer* |  |  |  |  |  |
| Listing\_id |  |  |  |  |  |  |  | Null |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Guest\_id | Unique id of Guest |  |  | Integer |  |  | Yes |  | Foreign\_key |
| Rating | Star rating for listing |  |  | Integer |  |  | No |  |  |
| Review | Review of listing |  |  | String |  |  | Yes | Null |  |
| Created\_time | Created time of rating and review |  |  | Data and time |  |  | no |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 5>** | | |  |  |
| ***Name*** |  |  |  | | | | | | |
|  |  |  | *Question and Answer* | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  | *Use ass an external entity to store user questions and it answer* | | | | | | |
| ***used*** |  |  |  | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing question and answers.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| Question\_id | Unique id of question |  |  | Integer |  |  | *No* |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *Listing\_id* | *Unique id of listings* | | | *Integer* |  |  | *Yes* |  |  |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | Null |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Question | String of question |  |  | String |  |  | No |  |  |
| Answer | Answer to question |  |  | String |  |  | Yes | null |  |
| Created\_time | Time created |  |  | Data and tiime |  |  | no |  |  |
| Guest\_id | Unique id of guest |  |  | Integer |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 6>** | | |  |  |
| ***Name*** |  |  |  | | | | | | |
|  |  |  | *Reservation* | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Use as an external entity to store user reservation request* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing reservations.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
|  |  | |  | *Integer* |  |  | *no* |  | *Primary key* |
| Reservation\_id | Unique id of reservation |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *Listing\_id* |  | | | *Integer* |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | *Foreign key* |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Yes |  |  |
|  | Unique id of listing |  |  |  |  |  |  |  |  |
| Guest\_id | Unique id of user |  |  | Integer |  |  | Yes |  | *Foreign key* |
| Checkin\_date | Renting start date |  |  | Date and time |  |  | No |  |  |
| Checkout\_date | Leaving date |  |  | Date and time |  |  | No |  |  |
| Created\_time | Time created |  |  | Date and time |  |  | No |  |  |
| Amount\_due | Amount to pay |  |  | Integer |  |  | No |  |  |
| status | Accepted or rejected |  |  | string |  |  | Yes |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 7>** | | |  |  |
| ***Name*** |  |  | *transaction* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Use to describe transaction between user and host* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing transaction.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
| *Transaction\_id* | *Unique id of transaction* | |  | *Integer* |  |  | *no* |  | *Primary key* |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | *Unique id of listing* | | | *Integer* |  |  |  |  | *Foreign key* |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Listing \_id |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Yes |  |  |
| Guest\_id | Unique of of user |  |  | Integer |  |  | Yes |  | *Foreign key* |
| Checkin\_date | Renting date |  |  | Date and time |  |  | No |  |  |
| Checkout\_date | Leaving date |  |  | Date and time |  |  | No |  |  |
| Created\_time | Time created |  |  | Date and time |  |  | No |  |  |
| Has\_guest\_rated | Guest rated or not |  |  | Bool |  |  | No |  |  |
| Has\_host\_rated | Host rated or not |  |  | Bool |  |  | no |  |  |
| Amount\_paid | Amount user paid |  |  | integer |  |  | No |  |  |

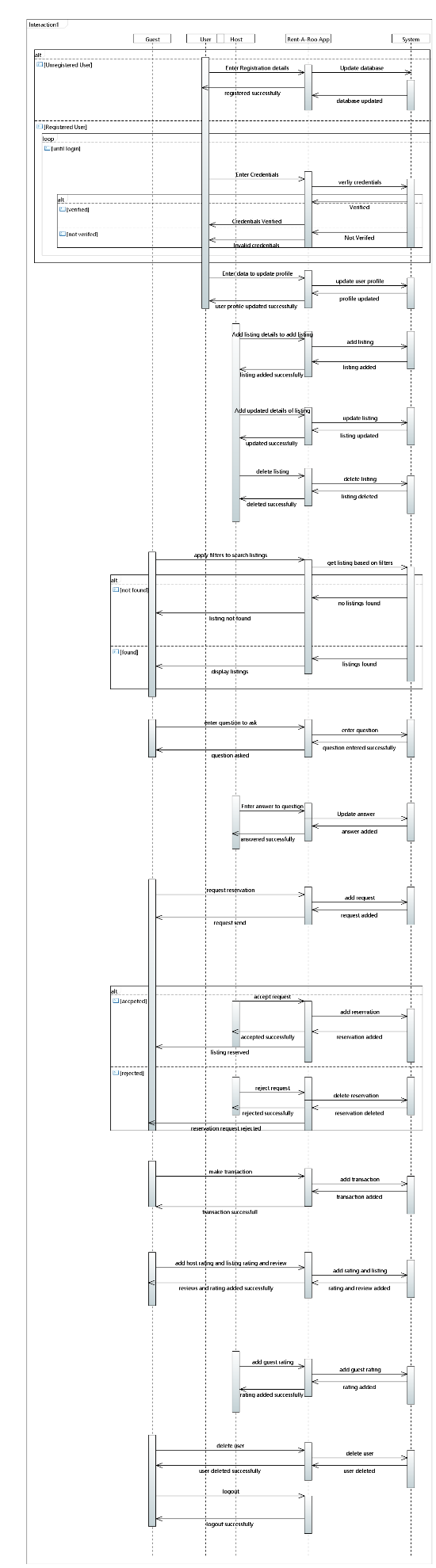
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 8>** | | |  |  |
| ***Name*** |  |  | *Promoted listing* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  |  | | | | | | |
| ***used*** |  |  | *Entity to show listing promoted by user* | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing promoted lisiting.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
| *Listing\_id* |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | Unique id of listing |  |  | Integer |  |  | no |  | *Foreign key, primary key* |
|  |  |  |  |  |  |  |  |  |  |
| *Start\_time* |  | | |  |  |  |  |  |  |
|  |  | |  | *Date and time* |  |  | *No* |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | Start of promotion |  |  |  |  |  |  |  |  |
| End\_time | End of promotion |  |  | Date and time |  |  | no |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **< Data 9>** | | |  |  |
| ***Name*** |  |  | *Favorites listing* | | | | | | |
|  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Alias*** |  |  |  | | | | |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Where-used/how-*** | |  | *Use as entity to store users favorite listing* | | | | | | |
| ***used*** |  |  |  | | | | | | |
|  |  |  |  | | | |  |  |  |
|  |  |  |  |  | |  |  |  |  |
| ***Content description*** | |  | *Notation for representing favorite listing.* | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| ***Column*** | ***Description*** | |  | ***Type*** |  | ***Length*** | ***Null able*** | ***Default*** | ***Key Type*** |
| ***Name*** |  |  |  |  |  |  |  | ***Value*** |  |
|  |  | | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | Unique Id of user |  |  | Integer |  |  |  |  | *Primary key, foreign key* |
| Guest\_id |  |  |  |  |  |  | no |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  | *Primary key, foreign key* |
|  |  |  |  |  |  |  | *no* |  |  |
| Listing\_id | Unique id of listing |  |  | Integer |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

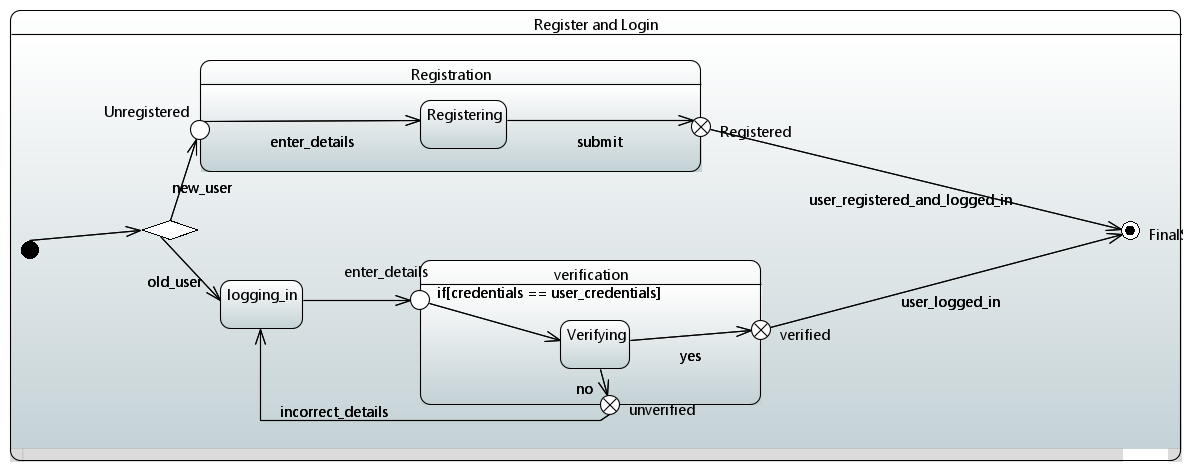
1. ***Application Design***

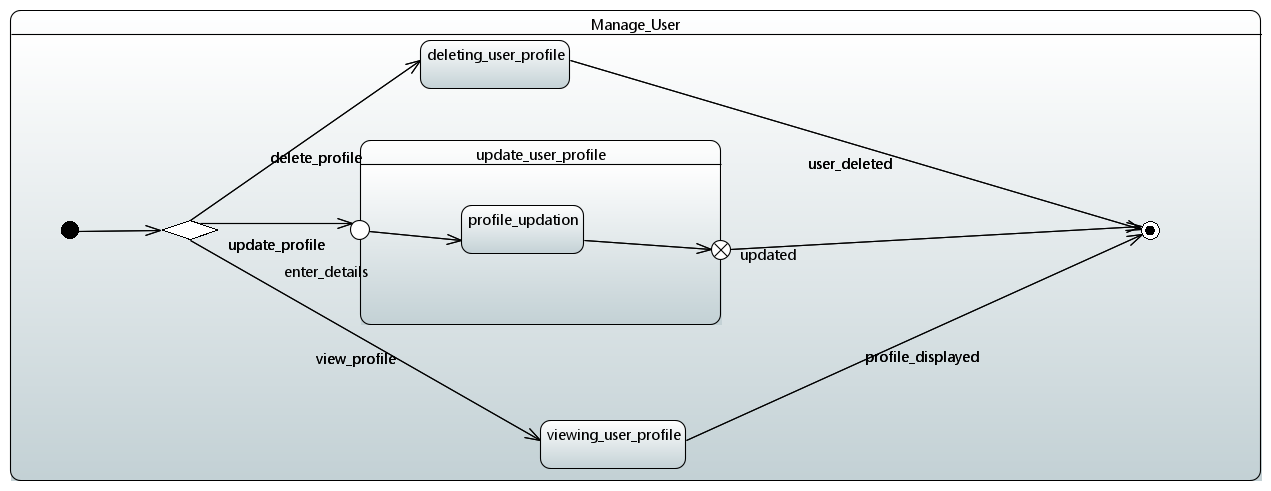
*A user can’t use the app unless they’re registered and logged in, the app takes the user’s details during registration such as email, password, name, phone number, about me etc. The user is free to edit their profile whenever they want and they can delete their profile too (only if a guest hasn’t already paid for a stay at one of their listings). After logging in the user can change their profile picture from the default image, the user can then play 2 roles at the same time in the app, a guest and a host. They can be a host and create listings with all of their property’s details such as an eye catching title, description, images, address and all the amenities(iron, washing machine, TV, WiFi, smoking allowed etc) they offer at their place as well as the no. of bedrooms and bathrooms, how many people can the place accommodate, the minimum and maximum nights a guest can stay and if the property is shared with other people or will the guest have the whole place to themselves and most importantly the nightly price to stay at this listing. The host is free to edit this listing whenever they want except after they delete the listing(it will be saved in the Database for transaction history purposes), the host can also pay to promote the listing to show up on the top of any relevant search results. The guest can search for listings and filter and sort their results too, they can also view popular listings(the top 3 most viewed listings on the app right now), they can press a listing to go to its listing page and add it to their favourites to view it later, or they can ask questions about the listing(which the host can later answer) and view previously asked and answered questions or rating and reviews about the listing. The guest can also press a link on the listing page which redirects them to Google Maps where the listing’s location will be displayed. The guest can reserve the listing and will be asked to choose a valid checkin and checkout date and they will be shown their Due Amount if the reservation is successfully sent to the host, the guest can then go to the guest’s reservations tab to view all their reservations and they have options to view the listing page the reservation was made to as well as check their reservation status(which may be Accepted/Rejected/Pending). The host can go to the host’s reservations tab to check any reservations they’ve recieved and go to the listing page related to the reservation or go to the guest’s page that sent the reservation and then accept or reject the reservation, the reservation will automatically be rejected in 24hours otherwise. If the host accepts it and the guest completes their payment for the confirmed reservation, then the transaction can be viewed in the host’s and guest’s transactions tab for the host and guest respectively. Here they can press the view listing button to view the listing related to the transaction, after the checkout date arrives the host can rate the guest and similarly the guest can rate the host as well as rate and review the listing. Every user has a guest rating and a host rating and they can change the app language whenever they want too*

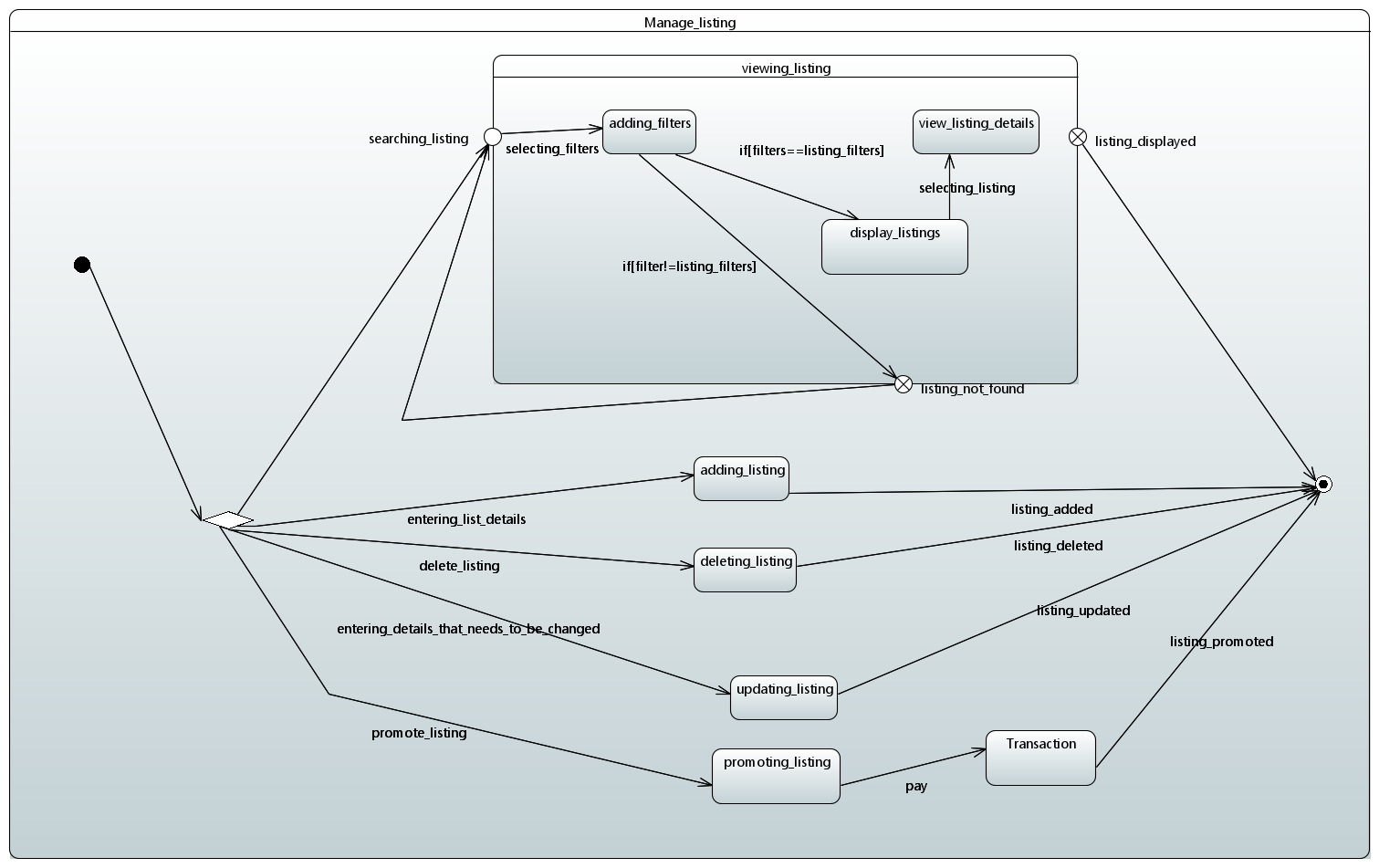
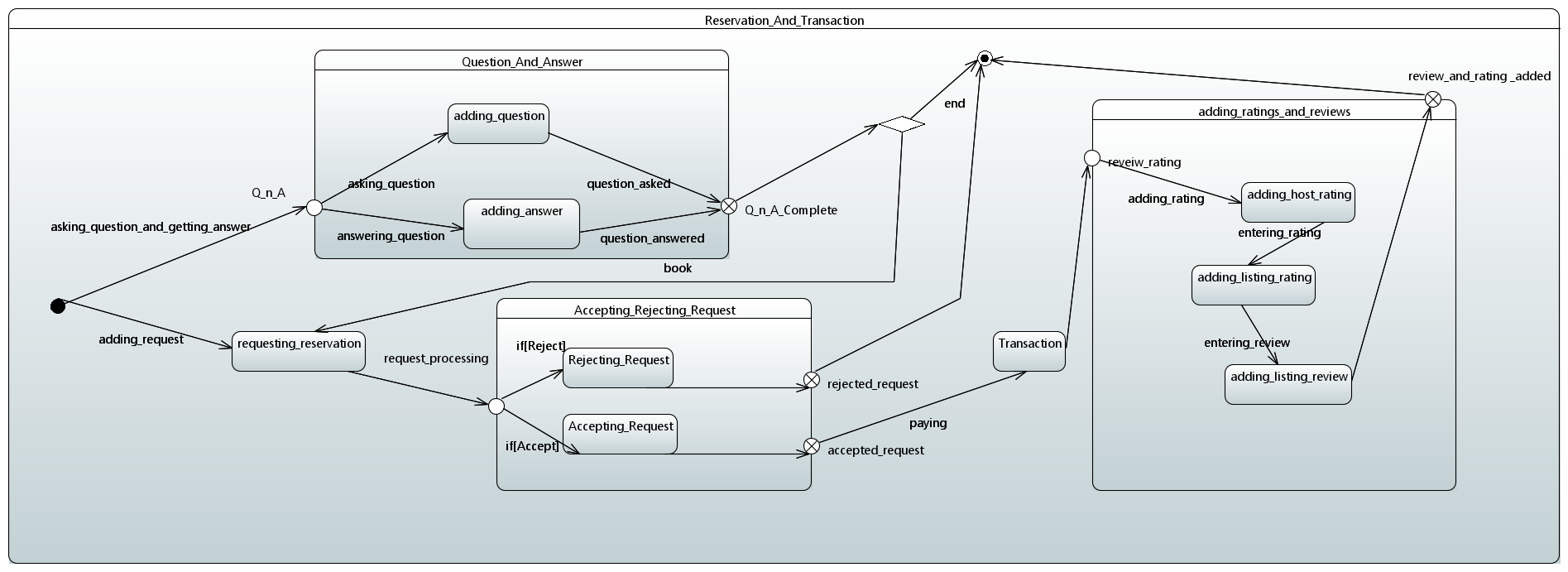
***9.1.1 Sequence Diagram***

******

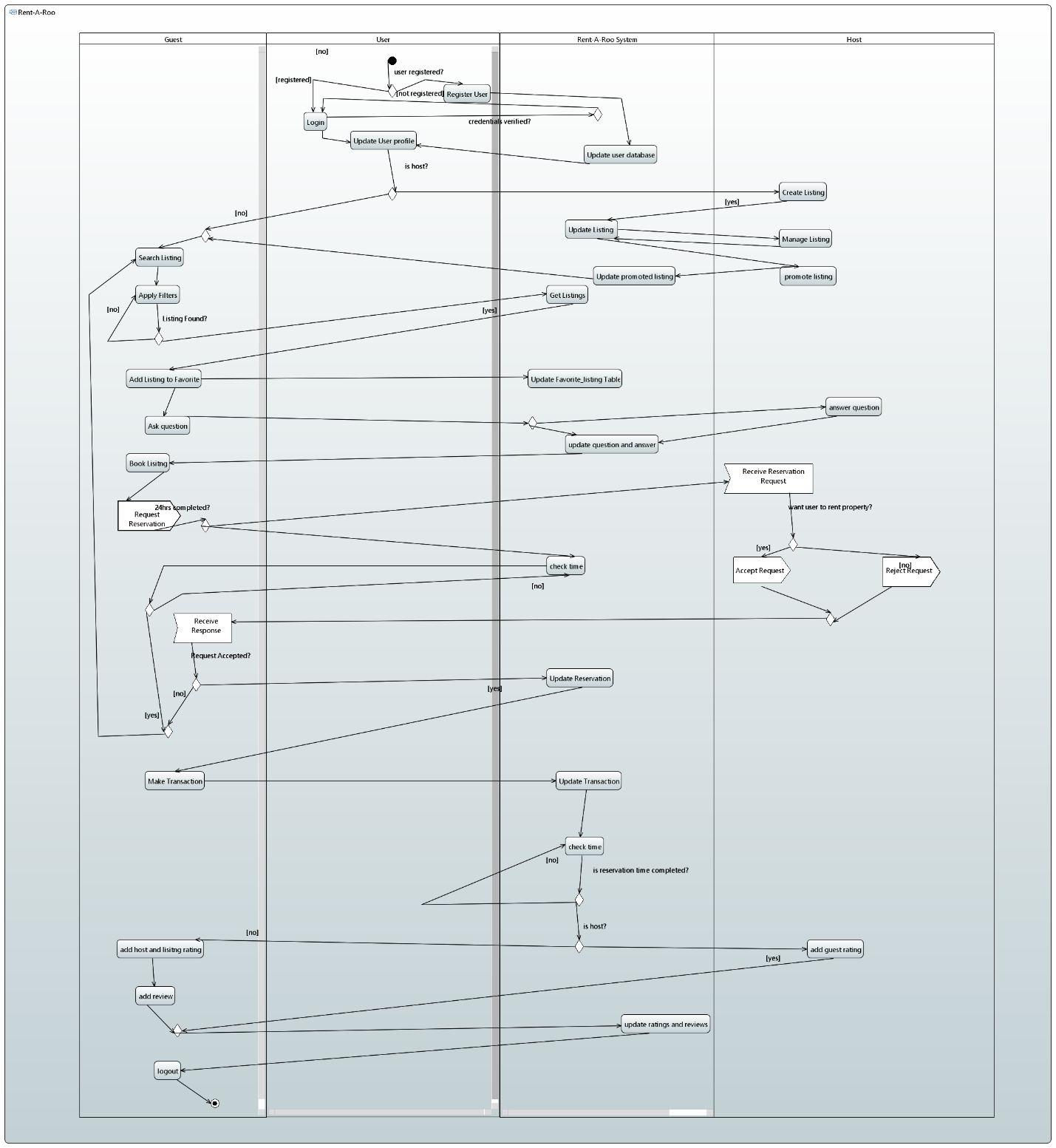
***9.1.2 State Diagram***

******

******

******

***9.1.3 Activity Diagram***



1. ***References***

*Not Applicable*

1. ***Appendix***

Not Applicable