# HOUSE RENTING SYSTEM

*By:*

*Team 01*

*CSC675-775 Fall 2019*

*Tejal Ghadge (Team Lead)*

*Gem Angelo  (Git Master)*

*Douglas Hebel*

*Zain Khan*

*Xinyu Zou*

*Milestone 1*

*Oct 08, 2019*

## 1. Executive Summary

Team 1 aims to develop an online marketplace for listing living spaces for rental. While hosts are looking to list a single room for a couple of days or their house for months, travelers needing a place to stay are able to easily browse the application to find the place that best suits their needs. Our application is an intermediary for those wanting to rent out spaces and those who want to rent those spaces. People in need a place to book simply create an account on out application and choose between a variety of options, once they have found a place that accommodates their traveling needs, we will assist in reservations for that space. Additionally, users wanting to list their available living space will be able to post information regarding that space such as prices, amenities, and features available.

## 2. Use Cases

### 1. Rentee

Jack is a backpacker from Florida. He wants to have a vacation in San Francisco. Since he just finished high school, he doesn't have a lot of money to spend. All the hotels in San Francisco are too expensive for him to afford. He needs to find a good place for him to stay in, and also with good public transport service.

### 2. Renter

Jane was a History teacher of high school. She retired last year and her grandson went to another city for university. So she has an extra room that she wants to rent it out.

### 3. Admin

John is an administrator who has access to the database and has the right to manage that data. He opens a platform for people to publish their rooms/ houses for renting. And he is willing to help users who are having problem for using the database.

## 3. Entities Glossary

**Users**

- Rentee - The people who want to rent the house, studio or apartment.
- Renter - The people who have extra house, studio or apartment to rent out.
- Admin - The people who have access to the database, and also has the manage the data.

**Listing**

- Studio - One of the listings that offered for the rentee.
- House - One of the listings that offered for the rentee.
- Apartment - One of the listings that offered for the rentee.

**Amenities**

Whether the studio, house or apartment has pool, parking, WIFI or AC etc. to offer for the rentee.

**Charges**

The fee charged by the renter for the renter using their studio, house or apartment, may also have extra charge for cleaning and damaged stuff.

**Booking**

Rentee can book studio, apartment or house. Rentee can have at least one booking.

**Account**

Account is created by the user, it can be rentee and renter. Users need password and username to create account.

## 4. Business Rules

**Users**

1. User creates an account  (Account needs a password and username)
2. User can be either a renter or rentee.

**Renter**

3. Renters can post many listing .
4. Renter can manage many listings .

**Rentee**

5. Rentees can have at least one booking.
6. Rentee pays one booking at a time.

**Listing**

7. Listing can be a house, apartment and studio. (Listing attributes are location/address)
8. Listing belongs to many bookings.
9. Listing can have many amenities.
10. Listing can have one or more charges.

## 5. List of Entities, Relationships, and Attributes

1.1. Entity: User (strong)
  - Relations (if any) : creates, is_a
  - Attributes:
    - Id (key)
    - Name (composite)
    - Dob
    - Age (derived)
    - Email
    - Phone Number

1.2. Entity: Account (weak)
  - Relations (if any) : created_by
  - Attributes:
    - Id (key)
    - Uid (FK)
    - Username (alphabet)
    - Password (composite)

1.3. Entity: Renter (strong)
  - Relations (if any): is_a, posts, manages
  - Attributes:
    - Id (key)
    - Name (composite)
    - Dob
    - Age (derived)
    - Email
    - Phone Number

1.4. Entity: Rentee (strong)
  - Relations (if any): have, pay
  - Attributes:
    - Id (key)
    - Name (composite)
    - Dob
    - Age (derived)
    - Email
    - Phone Number

1.5. Entity: Listing (weak)
  - ■ Relations (if any): get_posted, managed_by, have
  - ■ Attributes:
    - Id (key)
    - Type
    - Rooms
    - Location
    - Price
    - RenterID (FK)
    - RenteeID (FK)

1.6. Entity: House (weak)
  - ■ Relations (if any): is_a
  - ■ Attributes:
    - Id (key)
    - Rooms
    - Location
    - Price
    - RenterID (FK)
    - RenteeID (FK)

1.7. Entity: Apartment (weak)
  - ■ Relations (if any): is_a
  - ■ Attributes:
    - Id (key)
    - Rooms
    - Location
    - Price
    - RenterID (FK)
    - RenteeID (FK)

1.8. Entity: Studio (weak)
  - ■ Relations (if any): is_a
  - ■ Attributes:
    - Id (key)
    - Rooms
    - Location
    - Price
    - RenterID (FK)
    - RenteeID (FK)

1.9. Entity: Booking (weak)

- Relations (if any): pays, have, belongs_to
- Attributes:
  - Id (key)
  - RenteeID (FK)
  - Start date
  - End date
  - Number of guests
  - Total price (derived)

1.10. Entity: Amenities (weak)

- Relations (if any): have
- Attributes:
  - hasPool
  - hasParking
  - hasWifi
  - hasAC
  - ListingID (FK)

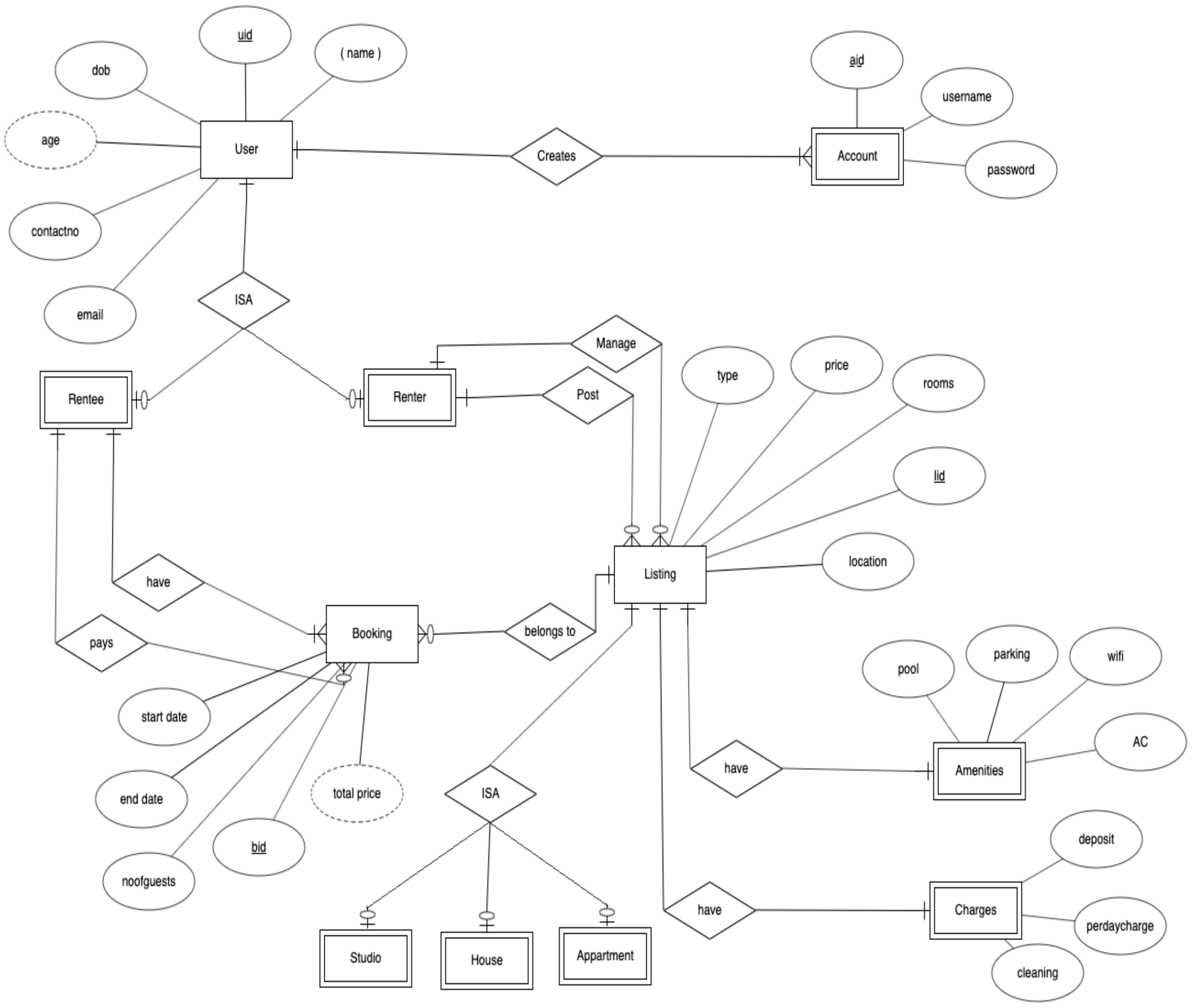
1.11. Entity: Renting Fee (weak)

- Relations (if any): is_a
- Attributes:
  - Price
  - ListingID (FK)

1.12. Entity: Charges (weak)

- Relations (if any): have
- Attributes:
  - Deposit
  - Perday Charge
  - Cleaning Fee
  - ListingID (FK)

## 6. Entity Relationship Diagram -ERD



## 7. ERD Test

| Business Rule | Entity 1 | Relationship | Type Rel | Entity2 | Pass/Fail | Modify |
|---|---|---|---|---|---|---|
| 1 | User | creates | 1 to M | Account | Failed | Marked account as weak entity |
| 2 | User | ISA | 1 to 1 | Rentee | Pass | |
| | User | ISA | 1 to 1 | Renter | Pass | |
| 3 | Renter | post | 1 to M | Listing | Pass | |
| 4 | Renter | manage | 1 to M | Listing | Pass | |
| 5 | Rentee | have | 1 to M | Booking | Failed | Added partial participation |
| 6 | Rentee | pay | 1 to M | Booking | Pass | |
| 7 | Listing | ISA | 1 to 1 | House | Pass | |
| | Listing | ISA | 1 to 1 | Apartment | Pass | |
| | Listing | ISA | 1 to 1 | Studio | Pass | |
| 8 | Listing | belongs | 1 to M | Booking | Pass | |
| 9 | Listing | have | 1 to 1 | Amenities | Pass | |
| 10 | Listing | have | 1 to 1 | Charges | Pass | |

## 8. List of Non-Functional Requirements

**Performance**

1. The system should respond quickly for all requests.
2. Database queries should not consume unnecessary memory and should execute in minimal time.

**Scalability**

1. System should handle the additional workload and should maintain the expected throughput with the given resources.

**Maintainability**

1. In a total failure case, the whole system should be put down to revision.
2. If broken, the mean time to recovery shall not excess of one day.

**Capacity**

Database should be able to handle the expected growth in data storage.

**Availability**

The data should be always accessible when and where needed to the authorized users.

**Security**

1. Database should secure the sensitive data like password.
2. Database should not be prone to SQL Injection attack.

## 9. Work Done by Team

1. Tejal Ghadge (Team Lead) - 10

Worked on ERD, ERD test cases and non-functional requirements.

2. Gem Angelo (Git Master) - 10

Worked on Entities, Relationships and attributes and document submission to git repository.

3. Douglas Hebel - 10

Worked on executive summary and non-functional requirements.

4. Zain Khan - 10

Created Business rules and document review.

5. Xinyu Zou -10

Worked on use cases and entity glossary and document review.