



# INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation On

## “SHARESPACE”

PG-e-DAC MARCH 23

**Submitted By**

**Group No: 88**

**Roll No.**  
**233186**  
**233213**

**Name:**  
**Prasad Ashok Taware.**  
**Shubham Mohan Sabane.**

**Mrs. Rupali Thorat**  
**Project Guide**

**Mr. Rohit Puranik**  
**Centre Coordinator**

## **ABSTRACT**

The main aim of the project is to build an interaction between property owner and property seeker; a system that will be able to manage listed properties data and provide easy access to the same. The project titled "Sharespace" is a comprehensive web application developed to emulate the functionalities of the online flat renting management system. This application offers users a one-stop solution for property listings and searches, making it easier to find rental properties without involving middlemen. The project involves the development of both front-end and back-end components, implementing features such as property listings, user registration, and messaging between property owners and seekers.

## ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, **Mrs. Rupali Thorat** for providing me with the right guidance and advice at the crucial junctures and for showing me the right way. I extend my sincere thanks to our respected **Centre Co-Ordinator Mr. Rohit Puranik**, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

**Prasad Taware (233186)**

**Shubham Sabane (233213)**

## Table of Contents

<b>ABSTRACT.....</b>	<b>2</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>3</b>
<b>1. INTRODUCTION .....</b>	<b>7</b>
<b>1.1 PROJECT OBJECTIVE .....</b>	<b>7</b>
<b>1.2 PROJECT OVERVIEW.....</b>	<b>7</b>
<b>1.3 PROJECT SCOPE .....</b>	<b>8</b>
<b>1.4 STUDY OF SYSTEM.....</b>	<b>8</b>
<b>1.4.1 MODULES:.....</b>	<b>8</b>
<b>2. SYSTEM ANALYSIS.....</b>	<b>10</b>
<b>2.1 EXISTING SYSTEM.....</b>	<b>10</b>
<b>2.2 PROPOSED SYSTEM.....</b>	<b>10</b>
<b>2.3 SYSTEM REQUIREMENT SPECIFICATION.....</b>	<b>11</b>
<b>2.3.1 GENERAL DESCRIPTION.....</b>	<b>11</b>
<b>2.3.2 SYSTEM OBJECTIVE.....</b>	<b>12</b>
<b>2.3.3 SYSTEM REQUIREMENTS .....</b>	<b>12</b>
<b>3. SYSTEM DESIGN .....</b>	<b>14</b>
<b>3.1 INPUT AND OUTPUT DESIGN .....</b>	<b>14</b>
<b>3.1.1 INPUT DESIGN.....</b>	<b>14</b>
<b>3.1.2 OUTPUT DESIGN.....</b>	<b>15</b>
<b>3.2 DATABASE DESIGN.....</b>	<b>16</b>
<b>3.3 SYSTM TOOLS.....</b>	<b>16</b>
<b>3.3.1 SOFTWARE REQUIREMENT .....</b>	<b>16</b>
<b>3.3.2 HARDWARE REQUIREMENT .....</b>	<b>16</b>
<b>4. SYSTEM DIAGRAMS.....</b>	<b>17</b>
<b>4.1 ACTIVITY DIAGRAM.....</b>	<b>17</b>
<b>4.2 DATA FLOW DIAGRAM.....</b>	<b>20</b>
<b>4.3 ER DIAGRAM (SYSTEM GENERATED) .....</b>	<b>21</b>

<b>4.4 USE CASE DIAGRAM .....</b>	<b>22</b>
<b>4.5 ER DIAGRAM.....</b>	<b>24</b>
<b>4.6 SEQUENCE DIAGRAM.....</b>	<b>25</b>
<b>4.7 CLASS DIAGRAM.....</b>	<b>26</b>
<b>5. TABLE STRUCTURE .....</b>	<b>27</b>
<b>6. PROJECT PHOTOS .....</b>	<b>30</b>
<b>7. FUTURE SCOPE .....</b>	<b>39</b>
<b>8. CONCLUSION.....</b>	<b>39</b>
<b>9. REFERENCES .....</b>	<b>39</b>

## LIST OF FIGURES

<b>FIGURE 1: L O G I N ACTIVITY DIAGRAM-----</b>	<b>15</b>
<b>FIGURE 2: LISTER ACTIVITY DIAGRAM -----</b>	<b>16</b>
<b>FIGURE 3: SEEKER ACTIVITY DIAGRAM-----</b>	<b>17</b>
<b>FIGURE 4: DATAFLOW DIAGRAM -----</b>	<b>18</b>
<b>FIGURE 5: ER DIAGRAM (SYSTEM GENERATED) -----</b>	<b>19</b>
<b>FIGURE 6: ADMIN USE CASE DIAGRAM -----</b>	<b>20</b>
<b>FIGURE 7: LISTER USE CASE DIAGRAM -----</b>	<b>20</b>
<b>FIGURE 8: SEEKER USE CASE DIAGRAM -----</b>	<b>21</b>
<b>FIGURE 9: ER DIAGRAM -----</b>	<b>22</b>
<b>FIGURE 10: SEQUENCE DIAGRAM-----</b>	<b>23</b>
<b>FIGURE 11: CLASS DIAGRAM -----</b>	<b>24</b>

# 1. INTRODUCTION

The advent of the digital age has revolutionized various industries, including real estate. With the increasing dependency on the internet for information and services, there has been a growing demand for online platforms that simplify the process of property search, rental, and purchase. In response to this demand, the project "Sharespace" aims to replicate the success and functionalities of the Online flat renting management system application catering to the modern needs of property seekers and owners.

The digital platform's significance lies in its potential to streamline the property search process. With user-friendly interfaces, advanced search options, and secure communication channels, the application intends to create a bridge between property seekers and owners, fostering a direct and hassle-free interaction. Through this, the project aims to address common challenges associated with property hunting, such as the lack of reliable information, high costs due to intermediaries, and the time-consuming nature of the search process.

## 1.1 PROJECT OBJECTIVE

The project's core objectives are to develop a feature-rich web application that replicates the functionalities of Sharespace, enhancing the property rental experience. This includes creating a user-friendly interface for property listings, implementing secure user authentication, enabling direct communication between users, and ensuring cross-device compatibility. The project aims to streamline property searches, reduce dependency on intermediaries, lower brokerage fees, and improve transparency in the real estate industry, contributing to the digital transformation of this sector.

## 1.2 PROJECT OVERVIEW

The primary purpose of our project is to provide a user-friendly, efficient, and transparent online platform for property rental searches. This platform seeks to revolutionize the traditional real estate industry by eliminating the dependency on intermediaries, reducing brokerage fees, and simplifying the property rental process.

Our project aims to address the following key purposes:

**Eliminate Middlemen:** To reduce the involvement of intermediaries, thereby lowering brokerage fees and improving transparency in property transactions.

**Streamline Property Searches:** To provide advanced search features, including filters and recommendations, that make it easier for users to find properties that match their preferences.

**Reduce Time and Effort:** To save users time and effort in their property search, making it a

more convenient and efficient process.

**Promote Transparency:** To promote transparency by providing detailed property listings with images and descriptions, reducing uncertainties associated with property rentals.

## 1.3 PROJECT SCOPE

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

Detail the scope of your project by listing the specific functionalities and features it includes. This may include:

- User registration and authentication
- Property listing and searching
- Property details with images and descriptions
- Messaging system for communication between property owners and seekers
- User profiles
- Admin panel for managing listings and users
- Search filters and recommendations

## 1.4 STUDY OF THE SYSTEM

### 1.4.1 MODULES:

The system after careful analysis has been identified to be presented with the following modules and roles.

The modules involved are:

- Administrator
- Lister (Property Owner)
- Seeker ( Property Seeker)



#### 1.4.1.1 Administrator:

The administrator is the super user of this application. Only admin have access into this admin page.

- The system will allow the administrator to select, add, delete ,update listers and seekers.
- The system will allow the administrator to select "listers, seekers" and check their details.
- The system will allow administrators to select, add, delete, update properties/ short-listings / matches/ bookings.

#### 1.4.1.2 Lister

- The system will allow the lister to add his own details and create a profile.
- The system will allow the lister to enlist properties with terms and conditions.
- Lister can view shortlisted properties and make a match if interested.
- Lister can fill booking details after contact has been made with seeker.

#### 1.4.1.3 Seeker

- The system will allow the seeker to add his own details and create a profile.
- The system will allow the seeker to search properties and view its details.
- The system will allow the seeker to shortlist properties.
- After match has been made, seeker can see lister's contact details. Further negotiations and renting property will takes place through contact

## **SYSTEM ANALYSIS**

System analysis is the process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified, and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

### **2.1 EXISTING SYSTEM**

The traditional process of finding rental properties is fraught with inefficiencies, inconveniences, and a lack of transparency. Prospective renters often encounter numerous challenges, including the excessive involvement of intermediaries, high brokerage fees, incomplete or outdated property information, and the absence of secure channels for direct communication with property owners. Moreover, the real estate industry has been slow to embrace the digital age, leaving many potential renters frustrated by the archaic methods of property searching and transactions.

### **2.2 PROPOSED SYSTEM**

In this context, our project seeks to address these pressing issues by developing a comprehensive full-stack web application and maintaining all the records in online systems database which makes it very easy to access and retrieve data from the database. By doing so, we aim to eliminate the problems associated with traditional property searches and transactions. We endeavor to create a digital ecosystem that empowers property seekers to find their ideal rental properties easily, communicate directly with property owners, reduce brokerage fees, and enhance the overall rental experience. Furthermore, the project intends to contribute to the

ongoing evolution of the real estate industry by embracing technology and offering a contemporary alternative to the age-old practices that have long been a source of frustration for both renters and property owners.

## **2.3 SYSTEM REQUIREMENT SPECIFICATION**

### **2.3.1 GENERAL DESCRIPTION**

#### **Product Description:**

"ShareSpace" is a dynamic and user-centric Full Stack Web Application designed to simplify the property rental experience. Inspired by the success of platforms like NoBroker, ShareSpace offers a seamless ecosystem comprising three essential modules: Admin, Property Owner, and Property Seeker. Property Owners can effortlessly list and manage properties with rich details, while Property Seekers can explore listings tailored to their preferences and directly connect with property owners. The platform aims to eliminate intermediaries, reduce brokerage fees, and promote transparency. While ShareSpace may not replicate every feature of established platforms, it represents a significant leap towards modernizing the property rental process.

#### **Problem Statement:**

The traditional process of finding rental properties is fraught with inefficiencies, inconveniences, and a lack of transparency. Prospective renters often encounter numerous challenges, including the excessive involvement of intermediaries, high brokerage fees, incomplete or outdated property information, and the absence of secure channels for direct communication with property owners. Moreover, the real estate industry has been slow to embrace the digital age, leaving many potential renters frustrated by the archaic methods of property searching and transactions.

The project aims to provide a solution that revolutionizes the property rental market by introducing a user-friendly, transparent, and efficient platform. By doing so, we aim to eliminate the problems associated with traditional property searches and transactions. Through the development of this Sharespace, we endeavor to create a digital ecosystem that empowers property seekers to find their ideal rental properties easily, communicate directly with property owners, reduce brokerage fees, and enhance the overall rental experience.

## **2.3.2 SYSTEM OBJECTIVES**

- To provide a Web site for online Flat renting and Management System.

## **2.3.3 SYSTEM REQUIREMENTS**

### **2.3.3.1 NON-FUNCTIONAL REQUIREMENTS**

Following non-functional requirements will be there in the insurance to the internet:

- (i) Secure access to consumer's confidential data.
- (ii) 24X7 availability.
- (iii) Better component design to get better performance at peak time.
- (iv) Flexible service-based architecture will be highly desirable for future extension.
- (v) Nonfunctional requirements define system properties and constraints.

Various other nonfunctional requirements are:

- Security
- Reliability
- Maintainability
- Portability
- Extensibility
- Reusability
- Compatibility
- Resource Utilization

### 2.3.3.2 FUNCTIONAL REQUIREMENTS

Any anonymous guests (visitors) can visit the application; can view the listings (list of properties) with their basic information such as City, Region, Type, Pricing, Availability etc.

Users can create accounts with email or social media authentication. Users can log in securely.

Password reset and account recovery options are available. Users can create and edit their profiles with personal information, contact details, and profile pictures. Users can set privacy settings for their profiles.

Property owners can create listings with details such as title, description, location, price, property type, and photos. Listings can be edited, updated, or removed by property owners. Property owners can specify availability dates for rentals.

Users can search for properties based on location, price range, property type, number of bedrooms, and other criteria. User should get properties within mentioned area. Advanced filters are available for refining search results. Users can save properties as favorites i.e. shortlist them and can remove them later from favorites if not interested.

Detailed property pages display all relevant information, including photos, description, owner details, and contact information. Users can request more information or schedule viewings from the property page. Users can send inquiries or booking requests for properties. Property owners/agents can accept or reject inquiries and bookings.

If request get accepted by owner, then property seeker will get necessary contact details of property owner. Further process of viewing property physically and booking it will happen with the help of that contact details.

Booking calendars are available for property owners to manage availability.

## **SYSTEM DESIGN**

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. Its emphasis on translating design. Specifications to performance specification. System design has two phases of development.

- Logical Design
- Physical Design

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

### **3.1 INPUT AND OUTPUT DESIGN**

#### **3.1.1 INPUT DESIGN:**

Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer-based format in the input design. Input data are collected and organized into groups of similar data. Once identified, the appropriate input media are selected for processing. All the input data are validated and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project the student details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

### **3.1.2 OUTPUT DESIGN:**

Computer output is the most important and direct source of information to the user. Output design

is a very important phase since the output needs to be in an efficient manner.

Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

## DATABASE DESIGN

Databases are the storehouses of data used in the software systems. The data is stored in tables inside the database. Several tables are created for the manipulation of the data for the system. Two essential settings for a database are

- Primary key - the field that is unique for all the record occurrences
- Foreign key - the field used to set relation between tables

Normalization is a technique to avoid redundancy in the tables.

### 3.1 SYSTEM TOOLS

#### 3.1.1 SOFTWARE REQUIREMENT

Technology	:	J2SE and J2EE, Hibernate, Spring Boot
Web-Technologies	:	React 18 , CSS, Javascript
Web Server	:	Tomcat 9.0
Java Version	:	JAVA Version 11, latest
Backend Database	:	MySQL 8.0
IDE	:	Spring Tool Suite

#### 3.1.2 HARDWARE REQUIREMENT

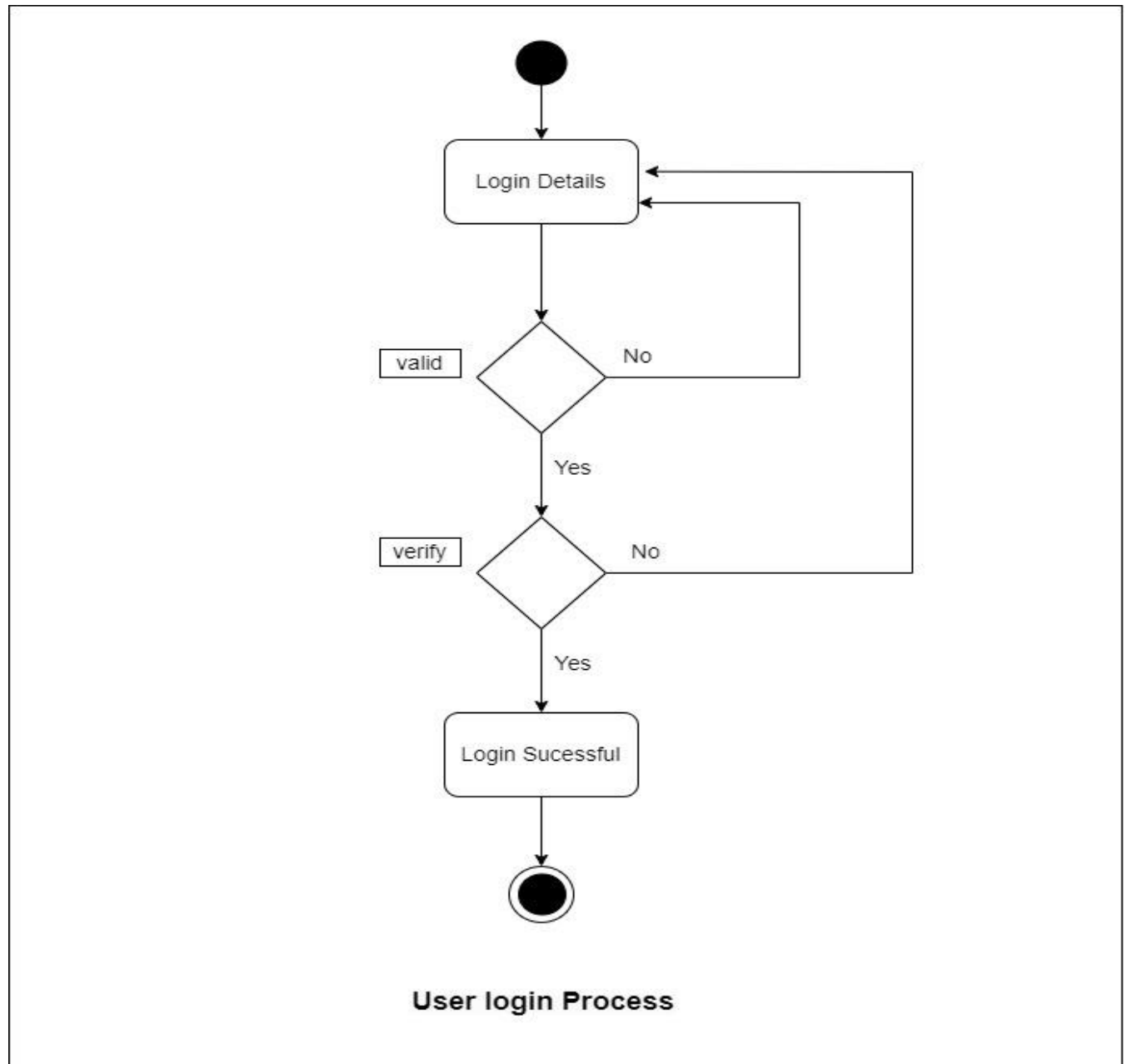
Hardware requirements for insurance on internet will be same for both parties which are as follows:

- Processor – i3 or above
- RAM - 2 GB
- Hard Disk – 160 GB
- NIC – For each party 4.3 Communication Interface

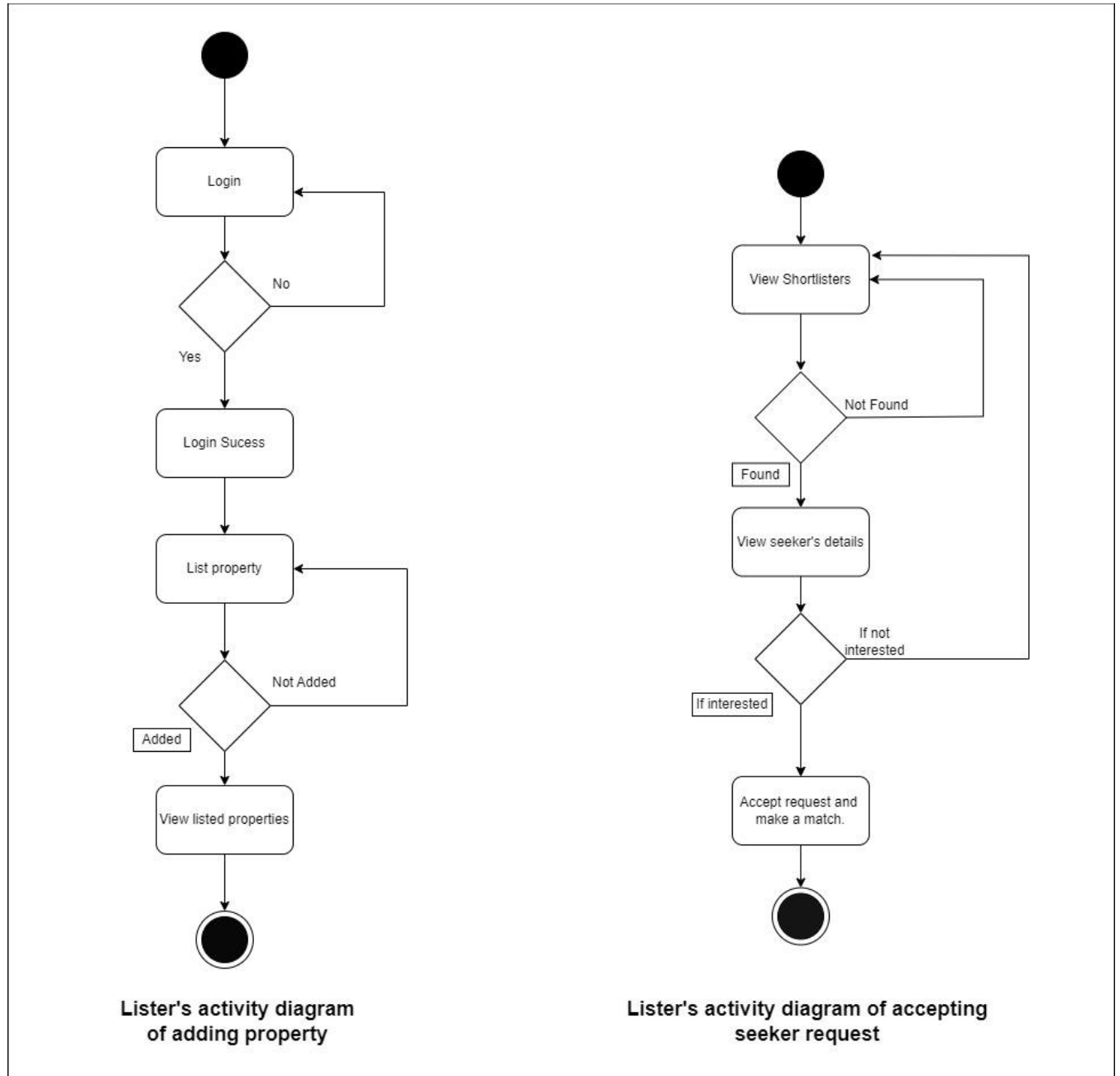


## 4 SYSTEM DIAGRAM

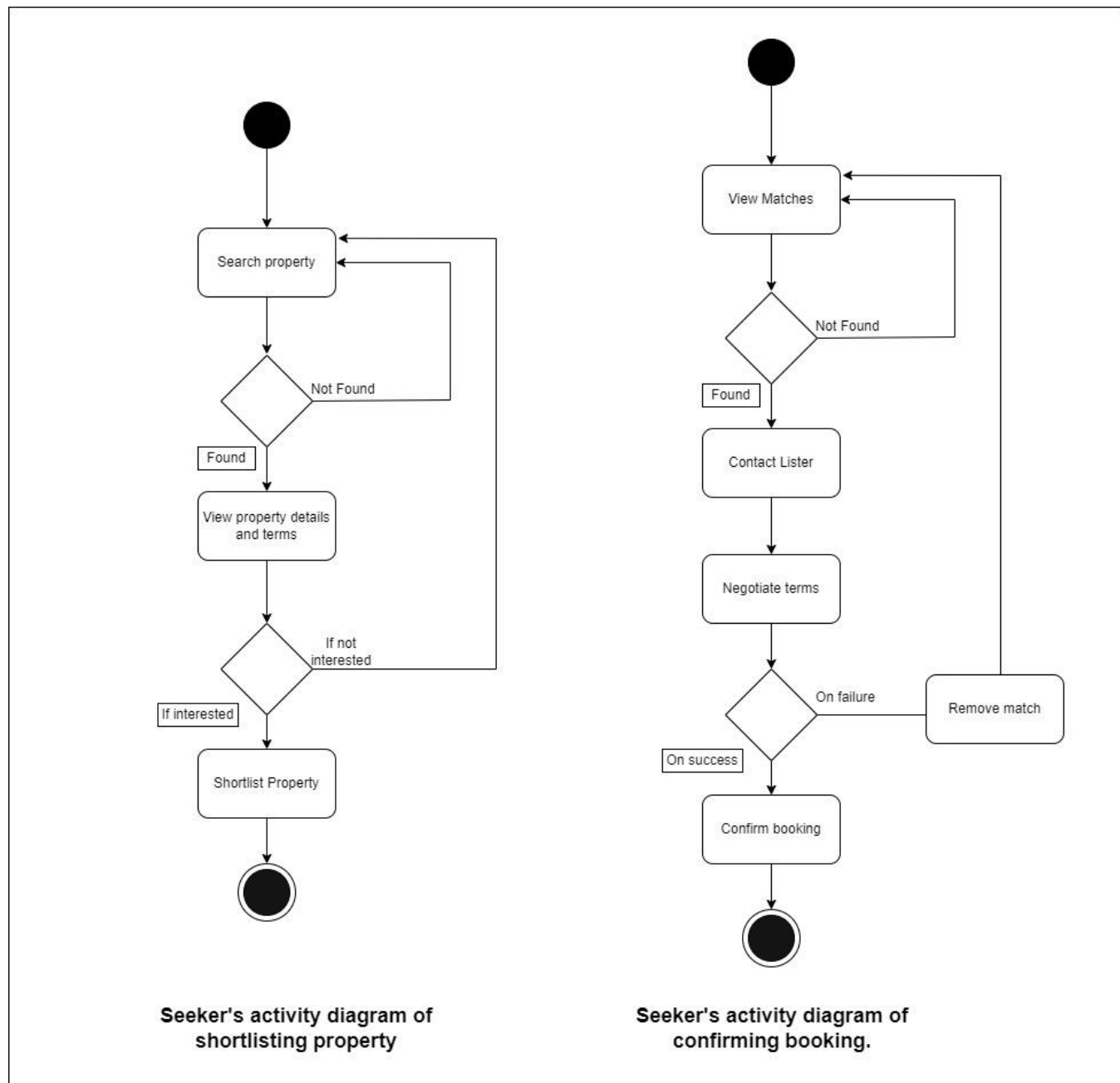
### 4.1 ACTIVITY DIAGRAM



*Fig.1: Login Activity diagram*

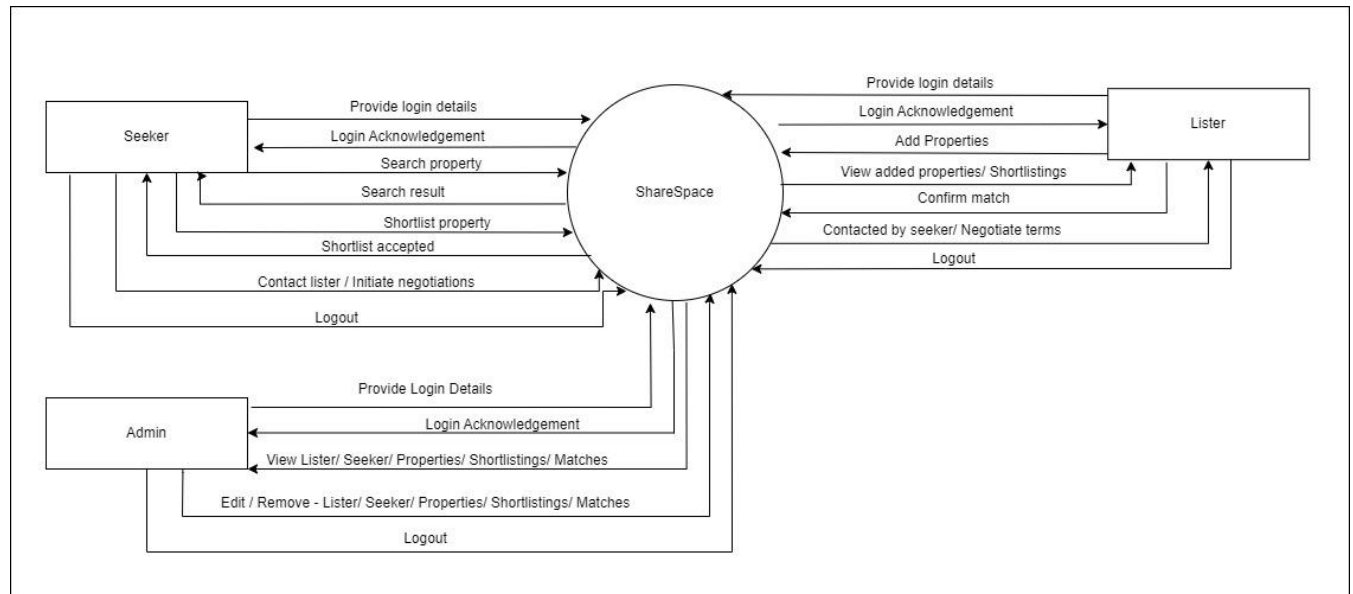


***Fig.2: Lister Activity diagram***



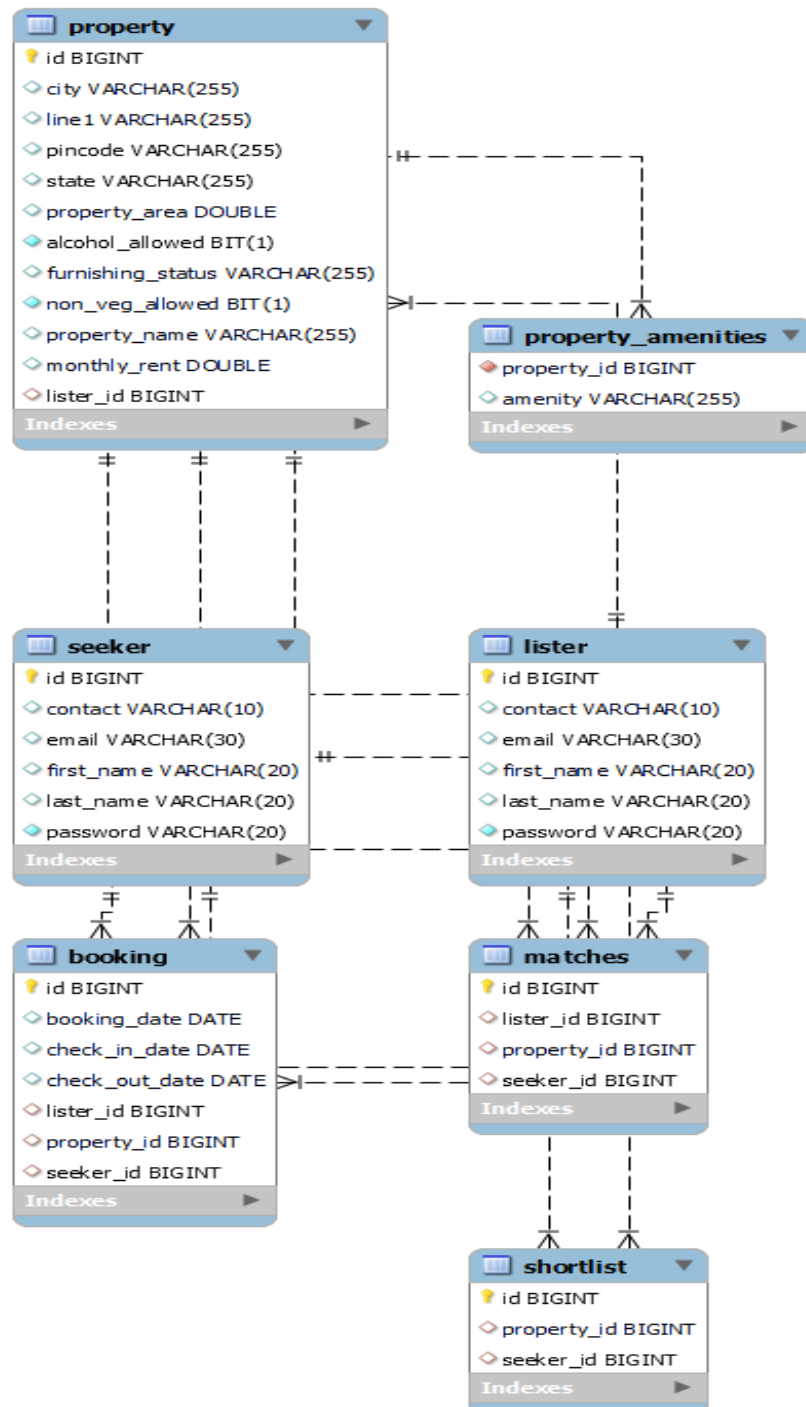
***Fig.3: Seeker Activity diagram***

## 4.2 DATA FLOW DIAGRAM



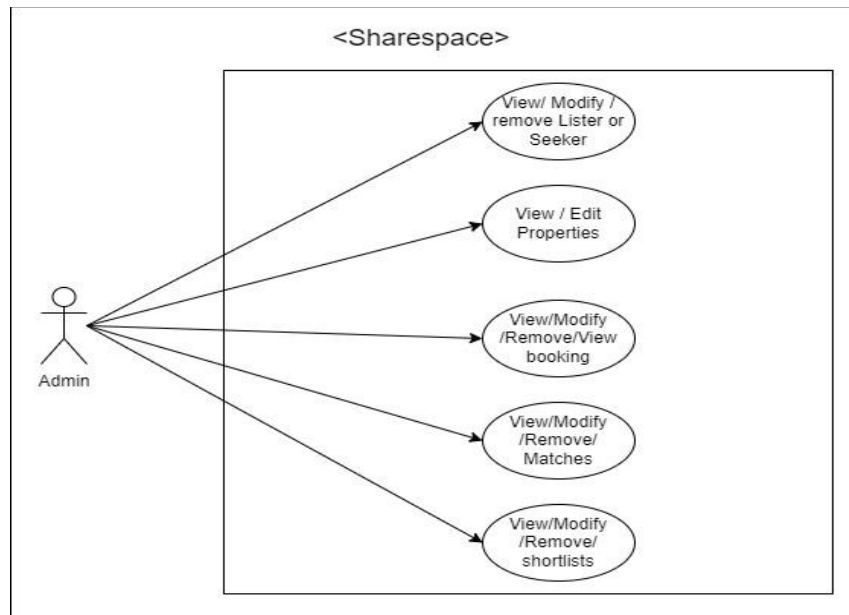
*Fig.4: Dataflow diagram*

### 4.3 E-R DIAGRAM (System generated)

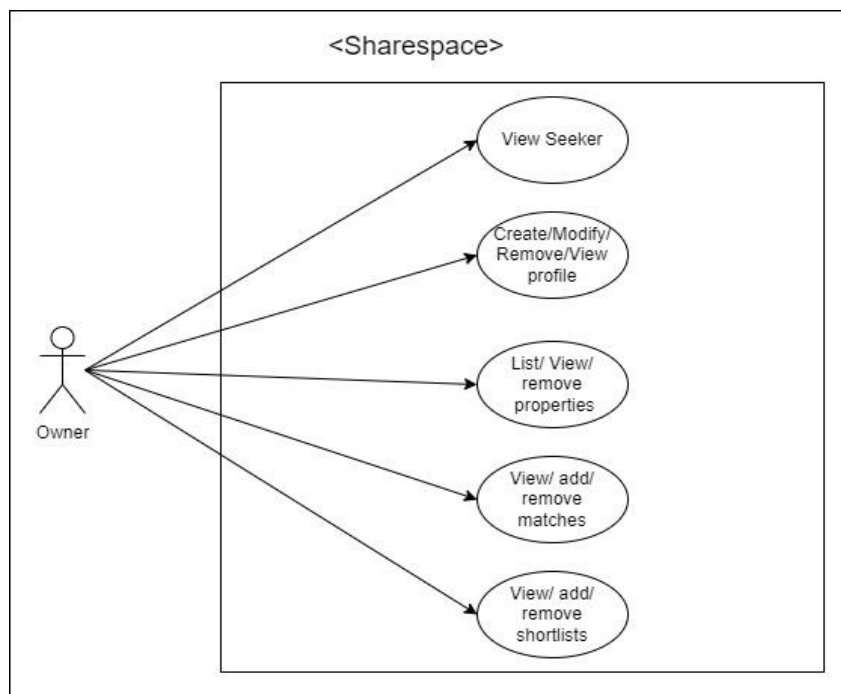


*Fig.5: Class diagram*

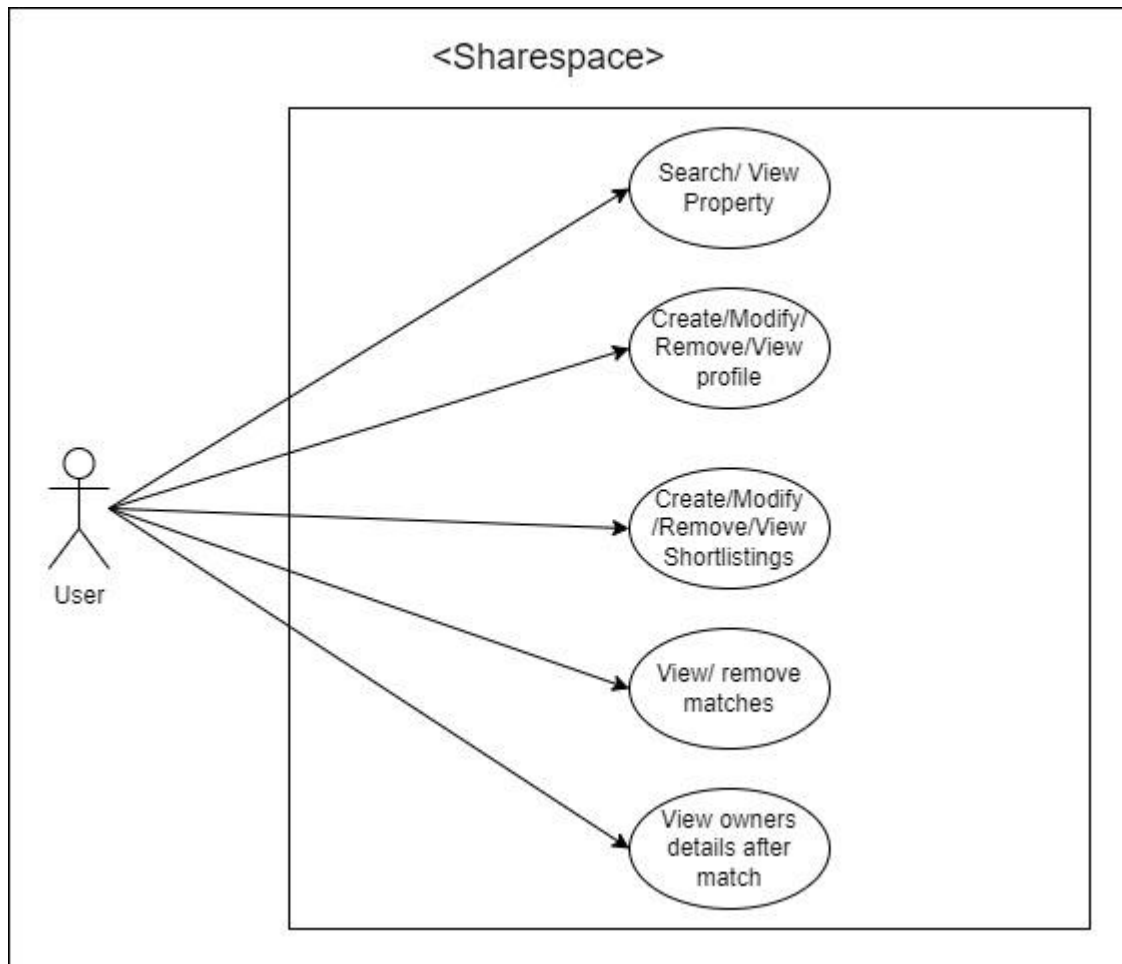
## 4.4 USE CASE DIAGRAM



*Fig.6: Admin Use Case diagram*

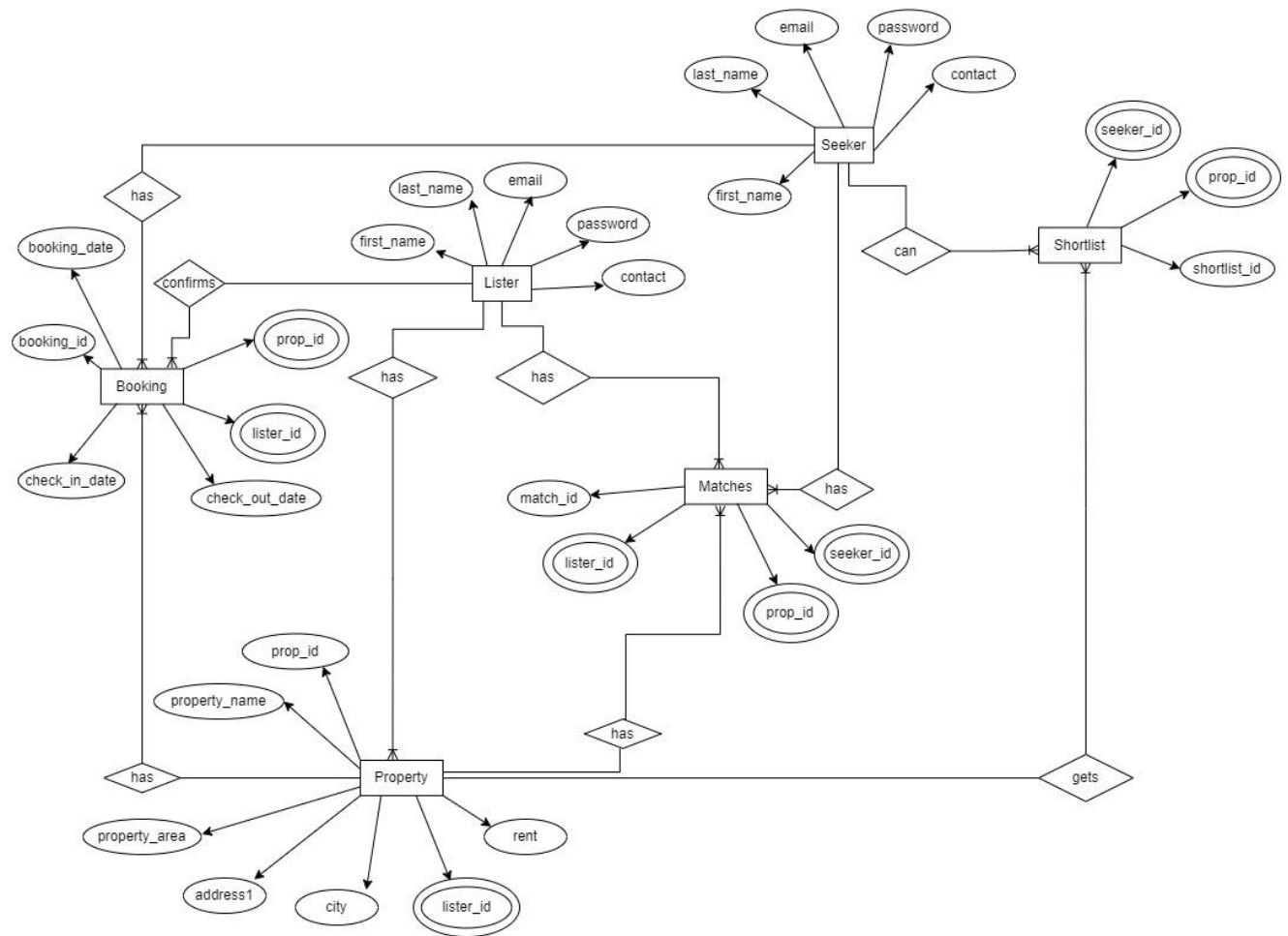


*Fig.7: Lister Use Case diagram*



***Fig.8: Seeker Use Case diagram***

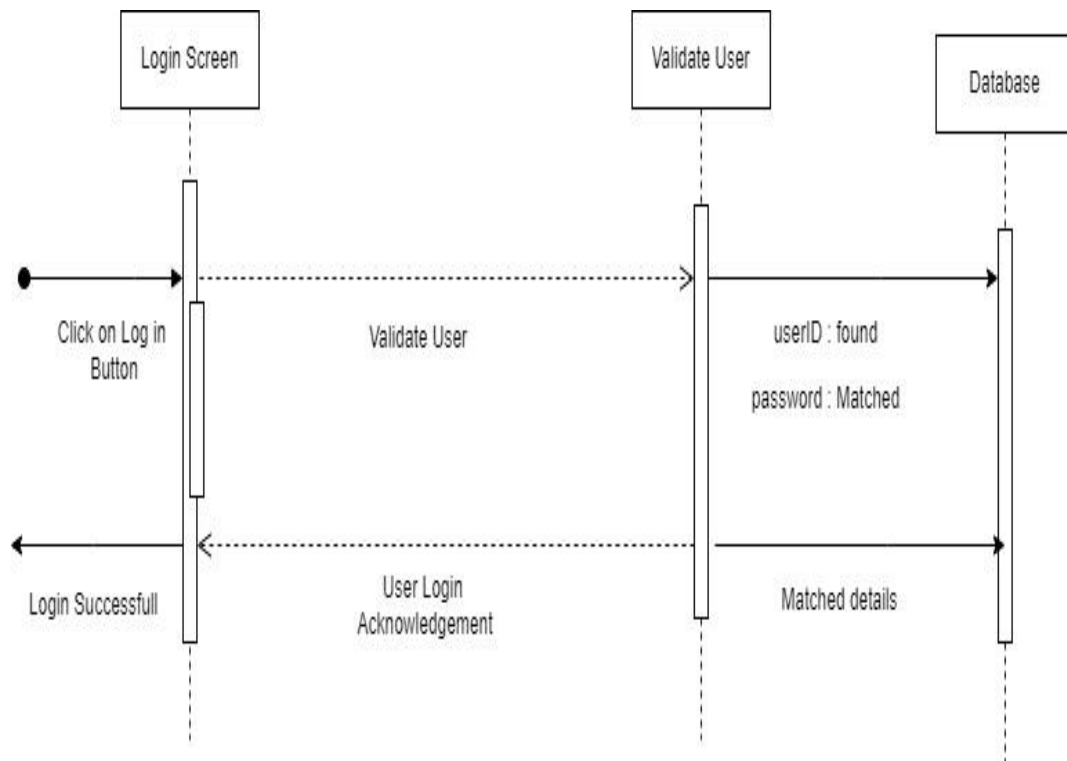
## 4.5 ER DIAGRAM



**Fig.9: ER diagram**

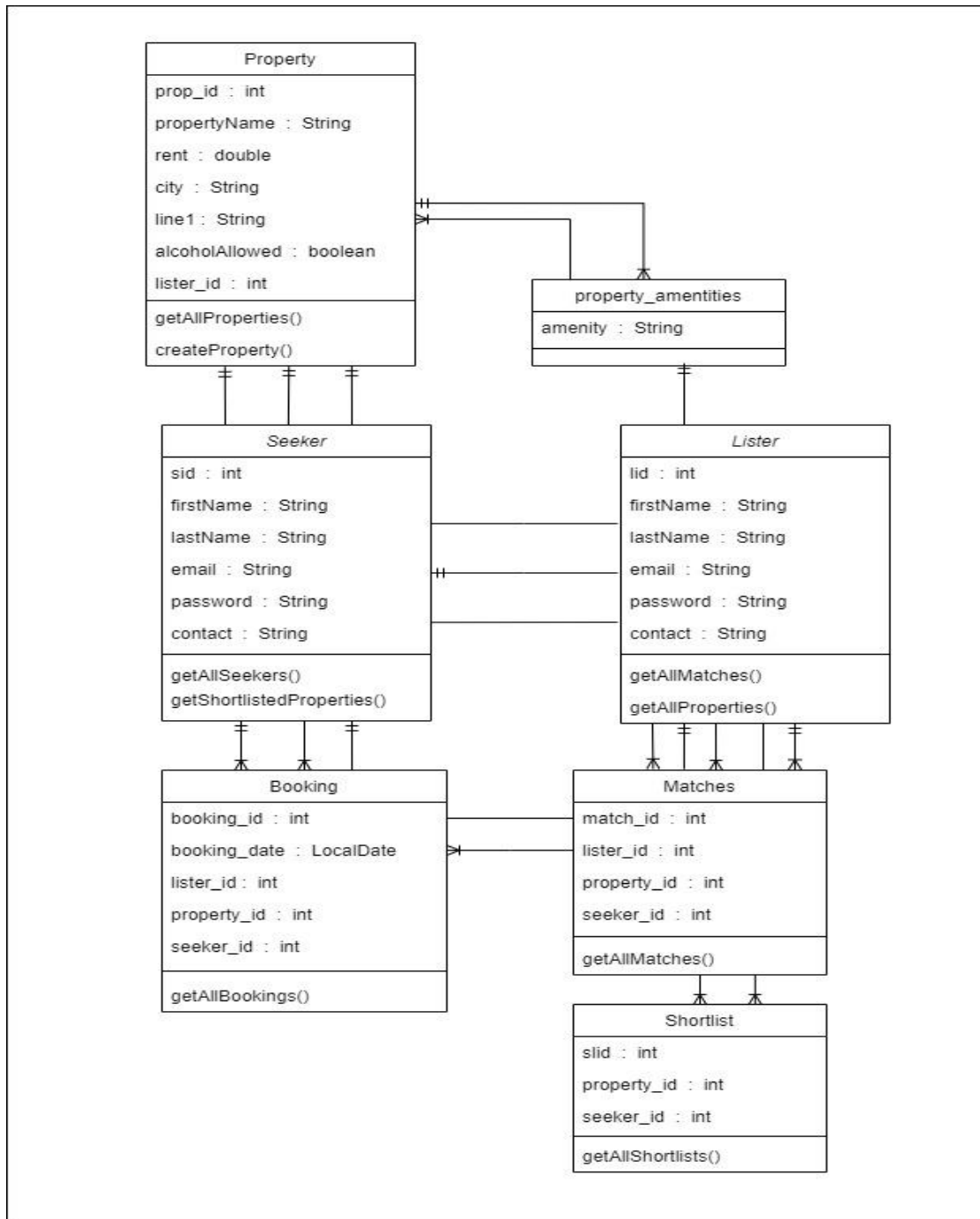


## 4.6 SEQUENCE DIAGRAM



*Fig.10: Sequence Diagram*

## 4.7 CLASS DIAGRAM



*Fig.11: Class Diagram*

## 5 TABLE STRUCTURE

### Tables:

```
mysql> use sharespace;
Database changed
mysql> show tables;
+-----+
| Tables_in_sharespace |
+-----+
| booking               |
| lister                 |
| matches               |
| property               |
| property_amenities    |
| seeker                |
| shortlist              |
+-----+
7 rows in set (0.00 sec)
```

### Seeker schema:

```
mysql> desc seeker;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | bigint        | NO   | PRI | NULL    | auto_increment |
| contact    | varchar(10)   | YES  | UNI | NULL    |                |
| email      | varchar(30)   | YES  | UNI | NULL    |                |
| first_name | varchar(20)   | YES  |     | NULL    |                |
| last_name  | varchar(20)   | YES  |     | NULL    |                |
| password   | varchar(20)   | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

### Shortlist schema:

```
mysql> desc shortlist;
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id             | bigint    | NO   | PRI | NULL    | auto_increment |
| property_id    | bigint    | YES  | MUL | NULL    |                |
| seeker_id      | bigint    | YES  | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

### Lister schema:

```
mysql> desc lister;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
contact	varchar(10)	YES	UNI	NULL	
email	varchar(30)	YES	UNI	NULL	
first_name	varchar(20)	YES		NULL	
last_name	varchar(20)	YES		NULL	
password	varchar(20)	NO		NULL	

```
6 rows in set (0.00 sec)
```

### Matches schema:

```
mysql> desc matches;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
lister_id	bigint	YES	MUL	NULL	
property_id	bigint	YES	MUL	NULL	
seeker_id	bigint	YES	MUL	NULL	

```
4 rows in set (0.00 sec)
```

### Booking schema:

```
mysql> desc booking;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
booking_date	date	YES		NULL	
check_in_date	date	YES		NULL	
check_out_date	date	YES		NULL	
lister_id	bigint	YES	MUL	NULL	
property_id	bigint	YES	MUL	NULL	
seeker_id	bigint	YES	MUL	NULL	

```
7 rows in set (0.00 sec)
```

**Property schema:**

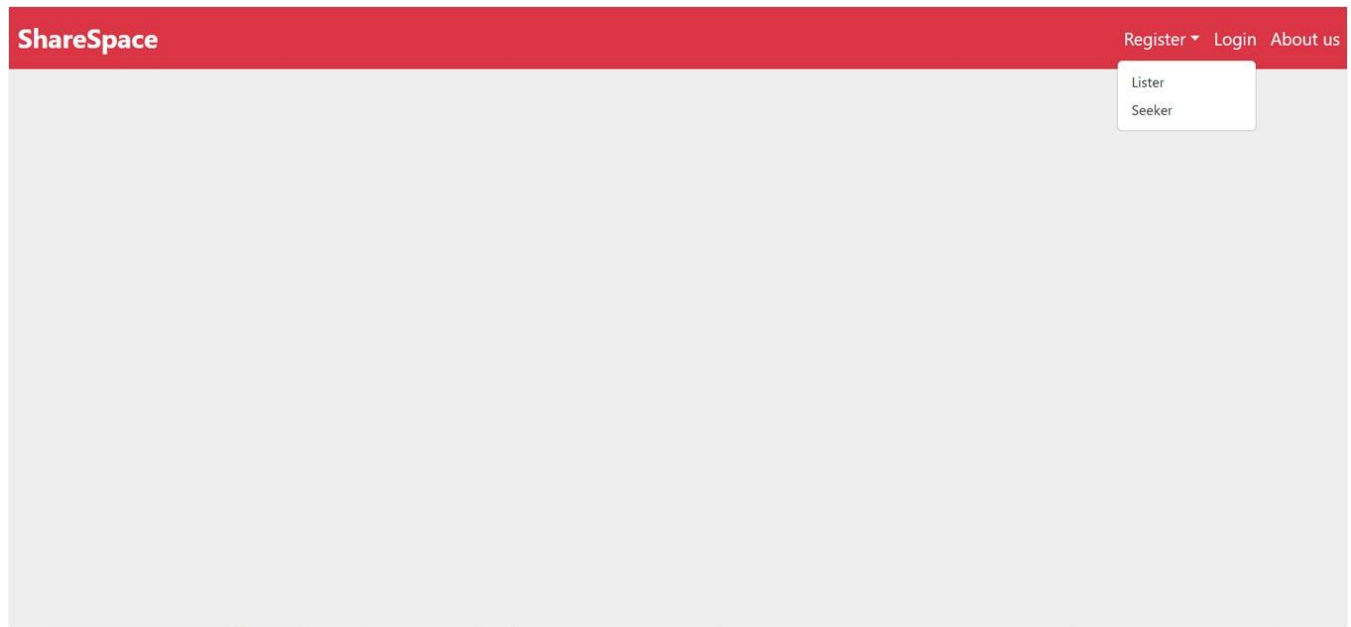
```
mysql> desc property;
```

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI	NULL	auto_increment
city	varchar(255)	YES		NULL	
line1	varchar(255)	YES		NULL	
pincode	varchar(255)	YES		NULL	
state	varchar(255)	YES		NULL	
property_area	double	YES		NULL	
alcohol_allowed	bit(1)	NO		NULL	
furnishing_status	varchar(255)	YES		NULL	
non_veg_allowed	bit(1)	NO		NULL	
property_name	varchar(255)	YES		NULL	
monthly_rent	double	YES		NULL	
lister_id	bigint	YES	MUL	NULL	

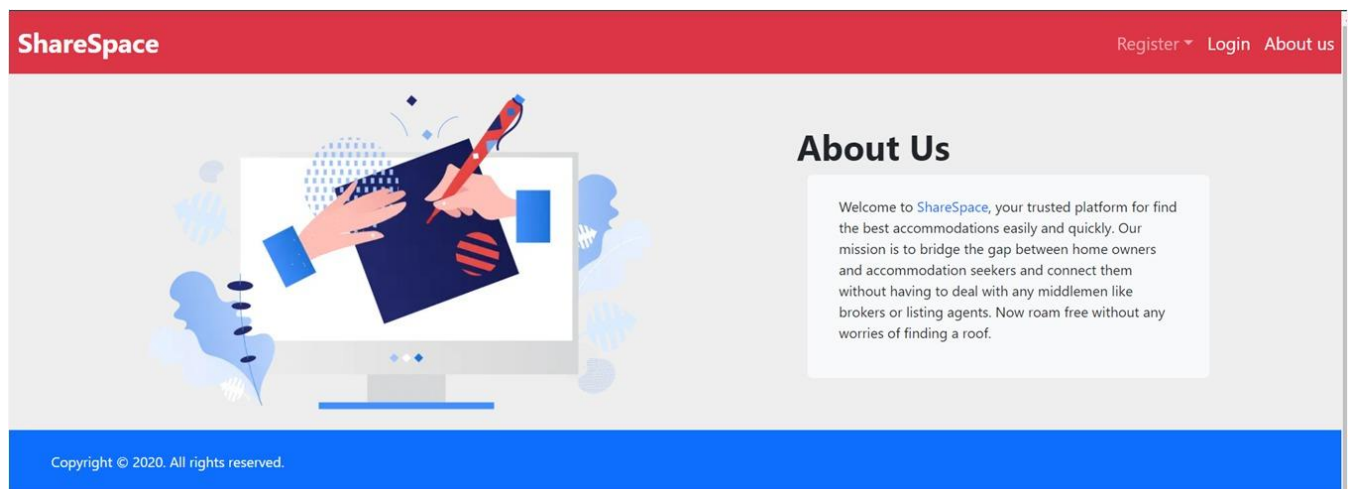
```
12 rows in set (0.00 sec)
```

## 6 PROJECT PHOTOS

### 6.1 WELCOME PAGE



### 6.2 ABOUT US



### 6.3 Registration Page :

ShareSpace

Register ▾

## Sign up


First Name

Last Name

Your Email

Contact number


Password



### 6.4 LOGIN PAGE :

ShareSpace

Register ▾ Login About us



## Welcome

Email address

Password

Role

☐ Remember me [Forgot password?](#)

Login

Don't have an account? [Register](#)

Copyright © 2020. All rights reserved.

## 6.5 RestAPI endpoint results (Using Swagger):

### 6.5.1 Seeker's Registration : (Adding seeker's details using POST method)

POST

/seeker

Parameters

No parameters

Request body required

application/json

```
{  "firstName": "Tanisha",  "lastName": "Priya",  "email": "tan@gmail.com",  "password": "tan@123",  "confirmPassword": "tan@123",  "contact": "1234567895"}  
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  'http://localhost:8080/seeker' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "firstName": "Tanisha",  "lastName": "Priya",  "email": "tan@gmail.com",  "password": "tan@123",  "confirmPassword": "tan@123",  "contact": "1234567895"}  '
```

Request URL

```
http://localhost:8080/seeker
```

Server response

Code	Details
201	<div>Response body</div> <div><pre>{  "message": "Seeker added successfully",  "timestamp": "2023-08-30T12:55:35.995208"}  </pre></div> <div>Download</div>



### 6.5.2 Viewing all seeker's details : (Using GET method)

GET

/seeker

Parameters

Cancel

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/seeker' \
-H 'accept: */*'

```

Request URL

```
http://localhost:8080/seeker

```

Server response

Code

Details

200

Response body

Code

Details

200

Response body

```
{
  "id": 5,
  "firstName": "Prasad",
  "lastName": "Laware",
  "email": "pat@gmail.com",
  "contact": "1234567890"
},
{
  "id": 6,
  "firstName": "Shubham",
  "lastName": "Sabane",
  "email": "sma@gmail.com",
  "contact": "1234567891"
},
{
  "id": 7,
  "firstName": "Vaibhav",
  "lastName": "Bisen",
  "email": "vai@gmail.com",
  "contact": "1234567892"
},
{
  "id": 8,
  "firstName": "Aman",
  "lastName": "Masham",
  "email": "aman@gmail.com",
  "contact": "1234567893"
}

```

Download

### 6.5.3 Viewing individual seeker's details:

seeker-controller

GET

/seeker/{id}

Parameters

Cancel

Name	Description
id * required	
integer(\$int64)	9
(path)	

ExecuteClear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/seeker/9' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/seeker/9
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "id": 9,   "firstName": "Sharayu",   "lastName": "Shinde",   "email": "ss@gmail.com",   "contact": "1234567894" }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Wed, 30 Aug 2023 07:32:43 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Responses

### 6.5.4 Updating seeker's details: (Using PUT method)

**PUT** /seeker/{id}

Parameters

Cancel Reset

Name	Description
id <span>★ required</span>	
integer(\$int64)	9
(path)	

Request body required

application/json

```
{
  "firstName": "Sharayu",
  "lastName": "Shinde",
  "email": "sha@gmail.com",
  "password": "sha@123",
  "contact": "1234567894"
}
```

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/seeker/9' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "firstName": "Sharayu",
    "lastName": "Shinde",
    "email": "sha@gmail.com",
    "password": "sha@123",
    "contact": "1234567894"
  }'
```

Request URL

http://localhost:8080/seeker/9

Server response

Code	Details
200	<p>Response body</p> <pre>{   "message": "Seeker details updated",   "timestamp": "2023-08-30T13:07:02.953799" }</pre> <p>Response headers</p>

### 6.5.5 Deleting seeker : (Using DELETE method)

**DELETE** /seeker/{id}

Parameters Cancel

Name	Description
id * required integer(\$int64) (path)	<input type="text" value="8"/>

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/seeker/8' \
  -H 'accept: */*' \
```

Request URL

http://localhost:8080/seeker/8

Server response

Code	Details
200	<p>Response body</p> <pre>{   "message": "Seeker deleted successfully",   "timestamp": "2023-08-30T13:12:28.7439626" }</pre> <p>Response headers</p>

### 6.5.5 Viewing property details : (Using GET method)

The screenshot shows a REST client interface with the following sections:

- GET /property**: The endpoint being tested.
- Parameters**: A section with a "Cancel" button and the text "No parameters".
- Execute**: A blue button to execute the request.
- Clear**: A button to clear the request.
- Responses**: A section containing:
  - Curl**: A text box with the command: `curl -X 'GET' \ 'http://localhost:8080/property' \ -H 'accept: */*'`
  - Request URL**: A text box with the URL: `http://localhost:8080/property`
  - Server response**: A table with two columns: "Code" and "Details".
    - Code**: 200
    - Details**: Response body

This screenshot provides a detailed view of the server response, showing the following:

- Server response**: A section with a "Details" tab.
- Code**: 200
- Response body**: A large text area containing the JSON response:

```
{
  "id": 3,
  "propertyName": "Krushnai",
  "propertyArea": 500,
  "rent": 5000,
  "address": {
    "line1": "Hinjewadi",
    "city": "Pune",
    "state": "Maharashtra",
    "pincode": "444001"
  },
  "propertyFeatures": {
    "alcoholAllowed": true,
    "nonVegAllowed": false,
    "amenities": [
      "LIFT"
    ]
  },
  "furnishingStatus": "FULLY_FURNISHED"
},
{
  "id": 3,
  "propertyName": "Krushnai",
  "propertyArea": 500,
  "rent": 5000,
  "address": {
    "line1": "Hinjewadi",
    "city": "Pune",
    "state": "Maharashtra",
    "pincode": "444001"
  },
  "propertyFeatures": {
    "alcoholAllowed": true,
    "nonVegAllowed": false,
    "amenities": [
      "LIFT"
    ]
  },
  "furnishingStatus": "FULLY_FURNISHED"
},
{
  "id": 3,
  "propertyName": "Krushnai",
  "propertyArea": 500,
  "rent": 5000,
  "address": {
    "line1": "Hinjewadi",
    "city": "Pune",
    "state": "Maharashtra",
    "pincode": "444001"
  },
  "propertyFeatures": {
    "alcoholAllowed": true,
    "nonVegAllowed": false,
    "amenities": [
      "LIFT"
    ]
  },
  "furnishingStatus": "FULLY_FURNISHED"
}
]
```
- Response headers**: A section at the bottom for viewing response headers.
- Download**: A button to download the response body.

## 6.6 Controllers :

### seeker-controller

GET	/seeker/{id}	✓
PUT	/seeker/{id}	✓
DELETE	/seeker/{id}	✓
GET	/seeker	✓
POST	/seeker	✓

### property-controller

GET	/property/{id}	✓
PUT	/property/{id}	✓
DELETE	/property/{id}	✓
GET	/property	✓
POST	/property	✓

### lister-controller

GET	/lister/{id}	✓
PUT	/lister/{id}	✓
DELETE	/lister/{id}	✓
GET	/lister	✓
POST	/lister	✓

### shortlist-controller

GET	/shortlist	✓
POST	/shortlist	✓
GET	/shortlist/{id}	✓
DELETE	/shortlist/{id}	✓

**matches-controller**

GET /matches

POST /matches

GET /matches/{id}

DELETE /matches/{id}

**booking-controller**

GET /booking

POST /booking

GET /booking/{id}

DELETE /booking/{id}

## 6.6 DTOs :

**Schemas**

SeekerUpdateDto &gt;

AddressRequestDto &gt;

PropertyFeaturesRequestDto &gt;

PropertyUpdateDto &gt;

ListerUpdateDto &gt;

ShortlistRequestDto &gt;

SeekerRequestDto &gt;

PropertyRequestDto &gt;

## 7. FUTURE SCOPE

Developmental changes in the pipeline:

1. Payment and E-wallet services.
2. Searching and Map functionalities using geo-location.
3. Contract negotiations and other documentation services.

## 8. CONCLUSION

In conclusion, our Flat Renting Management system web application – “Sharespace”, inspired by the NoBroker platform, represents a significant step towards modernizing and simplifying the property rental process. While we have made significant progress in replicating key features and functionalities, it's essential to acknowledge that we may not have implemented all aspects present in the NoBroker application. Our project has successfully addressed several critical objectives, including enhancing user experience, eliminating middlemen, streamlining property searches, and fostering direct communication between property owners and seekers. These achievements have contributed to a more user-friendly and efficient platform that alleviates many of the longstanding challenges associated with traditional property rentals. Our project serves as a foundation upon which we can build, expand, and contribute further to the digital transformation in real estate sector.

## 9. REFERENCES

- <https://bootstrapmade.com/mentor-free-education-bootstrap-theme/>
- <https://www.javatpoint.com/java-mail-api-tutorial>
- <https://www.w3schools.com/>
- <https://reactjs.org/docs/getting-started.html>
- <https://javaee.github.io/javaee-spec/javadocs/>