



# **HOUSE RENTAL SYSTEM**



## **A PROJECT REPORT**

*Submitted by*

**MEERA B (8115U23EC062)**

*in partial fulfillment of requirements for the award of the course*

**EGB1201 - JAVA PROGRAMMING**

*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

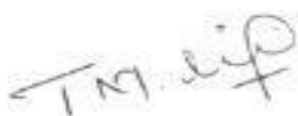
**DECEMBER - 2024**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**HOUSE RENTAL SYSTEM** ” is the bonafide work of **MEERA B (8115U23EC062)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



**SIGNATURE**

Dr. T. M. NITHYA, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

**ASSOCIATE PROFESSOR**

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.



**SIGNATURE**

Mr V KUMARARAJA, M.E., (Ph.D.),

**SUPERVISOR**

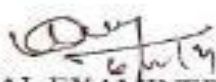
**ASSISTANT PROFESSOR**

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 06/12/24



**INTERNAL EXAMINER**



**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**HOUSE RENTAL SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **EGB1201 - JAVA PROGRAMMING**.

**Signature**

A rectangular box containing a handwritten signature in blue ink, which appears to read 'Meera B'.

---

**MEERA B**

Place: Samayapuram

Date: 06/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Engineering (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. V. KUMARARAJA, M.E., (Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To achieve a prominent position among the top technical institutions.

## **MISSION OF THE INSTITUTION**

- M1: To bestow standard technical education par excellence through state of the art  
  
infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

## **VISION OF DEPARTMENT**

To create eminent professionals of Computer Science and Engineering by imparting quality education.

## **MISSION OF DEPARTMENT**

**M1:** To provide technical exposure in the field of Computer Science and Engineering through

state of the art infrastructure and ethical standards.

**M2:** To engage the students in research and development activities in the field of Computer

Science and Engineering.

**M3:** To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

## **PROGRAM EDUCATIONAL OBJECTIVES**

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

## **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## ABSTRACT

A **House Rental System** is a digital platform designed to streamline the process of renting properties for both landlords and tenants. It offers a centralized system to manage property listings, tenant applications, lease agreements, payments, and communication. The system can cater to diverse property types, including apartments, houses, and commercial spaces. For landlords, the platform provides tools to list properties with details such as rent amount, amenities, and availability, while tracking tenant applications and payment history. Tenants can search for properties using filters such as location, budget, and property size, view detailed descriptions, and initiate the rental process seamlessly. Modern rental systems often integrate with payment gateways for secure online transactions, offer document management for lease agreements, and include features like automated reminders for rent due dates. Advanced systems may also leverage AI to suggest properties based on user preferences, enhance fraud detection, and support dynamic pricing. The primary benefits of such systems include increased efficiency, reduced administrative burden, enhanced transparency, and improved user experience. With growing urbanization and demand for rental accommodations, a robust House Rental System is essential for simplifying the rental process and fostering trust between landlords and tenants.



**ABSTRACT WITH POs AND PSOs MAPPING**  
**CO 5: BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME**  
**PROBLEMS.**

<b>ABSTRACT</b>	<b>POs MAPPED</b>	<b>PSOs MAPPED</b>
A house rental system connects landlords and tenants, simplifying property listing, search, booking, and rent management. It streamlines the rental process for both parties efficiently.	<b>PO1 -3</b> <b>PO2-3</b> <b>PO3-3</b> <b>PO4-3</b> <b>PO5-3</b> <b>PO6-3</b> <b>PO7-3</b> <b>PO8-3</b> <b>PO9-3</b> <b>PO10-3</b> <b>PO11-3</b> <b>PO12-3</b>	<b>PSO1-3</b> <b>PSO2-3</b>

Note: 1- Low, 2-Medium, 3- High

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Objective	1
	1.2 Overview	2
	1.3 Java Programming concepts	2
<b>2</b>	<b>PROJECT METHODOLOGY</b>	
	2.1 Proposed Work	5
	2.2 Block Diagram	6
<b>3</b>	<b>MODULE DESCRIPTION</b>	
	3.1 User Management Module	7
	3.2 Property Listing Module	7
	3.3 Search And Filter Module	7
	4. Payment Module	8
	5. Lease Management Module	8
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	
	4.1 Conclusion	9
	4.2 Future Scope	10
	<b>APPENDIX A (SOURCE CODE)</b>	11
	<b>APPENDIX B (SCREENSHOTS)</b>	21
	<b>REFERENCES</b>	25

# CHAPTER 1

## INTRODUCTION

### 1.1Objective

The objective of a **House Rental System** is to provide a seamless, efficient, and user-friendly platform for managing rental properties and tenant interactions. It aims to streamline the rental process by digitizing property listings, applications, payments, and lease management, reducing the time and effort required for both landlords and tenants. By offering an organized and transparent process, the system enhances trust and reliability in property rentals. For landlords, the system simplifies property management by enabling them to list properties with detailed descriptions, track tenant applications, and manage rent payments. It provides tools for maintaining financial records, sending automated reminders, and digitally handling lease agreements. These features minimize manual tasks and ensure better monitoring of their rental portfolio.

Tenants benefit from a convenient interface to search for properties, filter options based on preferences, and apply online. Secure payment gateways and timely reminders for rent deadlines add to the convenience, while detailed property information ensures transparency. Overall, the objective is to create a reliable and efficient rental ecosystem that fosters smoother interactions between landlords and tenants.

## 1.2 Overview

A **House Rental System** is a comprehensive digital platform designed to facilitate and streamline the property rental process for landlords, tenants, and property managers. It centralizes various tasks such as property listings, tenant applications, lease agreements, and payment management, making the entire rental journey efficient and user-friendly. Landlords can use the system to list properties with details such as rent, location, and amenities while tracking tenant inquiries, applications, and rental payments. Tenants benefit from an intuitive interface to search, compare, and apply for properties that match their preferences. Features like secure online payment options, automated reminders for due dates, and digital lease management enhance convenience and transparency for all parties involved.

The system also incorporates advanced tools, such as data analytics and AI-driven recommendations, to optimize rental processes. It fosters better communication between landlords and tenants by enabling direct messaging and providing updates on the rental process. By reducing manual tasks, ensuring accuracy, and enhancing accessibility, a House Rental System serves as a valuable solution in the growing rental market, meeting the needs of modern property management.

## 1.3 Java Programming Concepts

### 1.Object-Oriented Programming (OOP):

- **Classes and Objects:** Used to represent entities like Property, Landlord, Tenant, and Lease.
- **Inheritance:** For example, a Residential Property class might inherit from a base Property class, sharing common attributes like address and rent amount.

- **Polymorphism:** Enables methods like `calculate Rent()` to behave differently for different property types (e.g., residential vs. commercial).
- **Encapsulation:** Ensures sensitive data, such as payment details or tenant information, is protected by controlling access through getters and setters.

## 2. Collections Framework:

- **List, Set, and Map:** Used for managing lists of properties, tenants, and payments. For instance, a `Map` might store payment records with keys as dates and values as amounts.

## 3. File Handling and Serialization:

- Used to store and retrieve data like property details, tenant records, or payment history in files or databases. Serialization can save the state of objects for persistence.

## 4. Database Connectivity (JDBC):

- Java Database Connectivity (JDBC) is used to connect the system to a database (e.g., MySQL, PostgreSQL) for storing and managing rental data.

## 5. Exception Handling:

- Ensures robustness by handling errors such as invalid user input, database connection issues, or file access problems.

## 6. Java FX or Swing for GUI:

- If the system includes a graphical user interface, Java FX or Swing can be used to create interactive forms, dashboards, and property search pages.

## 7. Multithreading and Concurrency:

- Allows simultaneous handling of multiple users accessing the system. For instance, one thread might process a payment while another manages property updates.

## 8. APIs and Networking:

- APIs can be used to integrate third-party services like payment gateways. Networking concepts enable functionalities like email notifications or property searches across locations.

## 9. Design Patterns:

- **Singleton Pattern:** Used for managing shared resources like database connections.
- **Factory Pattern:** Helps in creating instances of different property types dynamically.
- **MVC (Model-View-Controller):** Ensures separation of concerns in the application, especially for GUI-based systems.

## **CHAPTER 2**

### **PROJECT METHODOLOGY**

#### **2.1 Proposed Work**

The proposed house rental system aims to create an efficient platform that bridges the gap between property owners and tenants, simplifying the rental process. Property owners will be able to list their rental spaces with detailed descriptions, including location, price, amenities, and images, making it easier for tenants to browse and find suitable options. Tenants will have access to a robust search functionality, allowing them to filter properties based on criteria such as budget, location, and property type, enhancing their experience and saving time.

Key features include secure user registration and profile management for both landlords and tenants. Integrated payment options will enable seamless rent transactions, while automated notifications and reminders will keep users informed about important dates, such as rent due or lease expiry. The system will also include digital lease management, allowing for electronic agreements and renewal tracking, along with a rating and feedback mechanism to ensure transparency and accountability between users. To further enhance functionality, the system will integrate mapping services for property location visualization and include an admin dashboard for platform monitoring and management. Scalability and robust security measures will ensure that the platform can accommodate growth and protect user data effectively. This system is designed to make the rental process more convenient, transparent, and accessible for all users.

## 2.2 Block Diagram

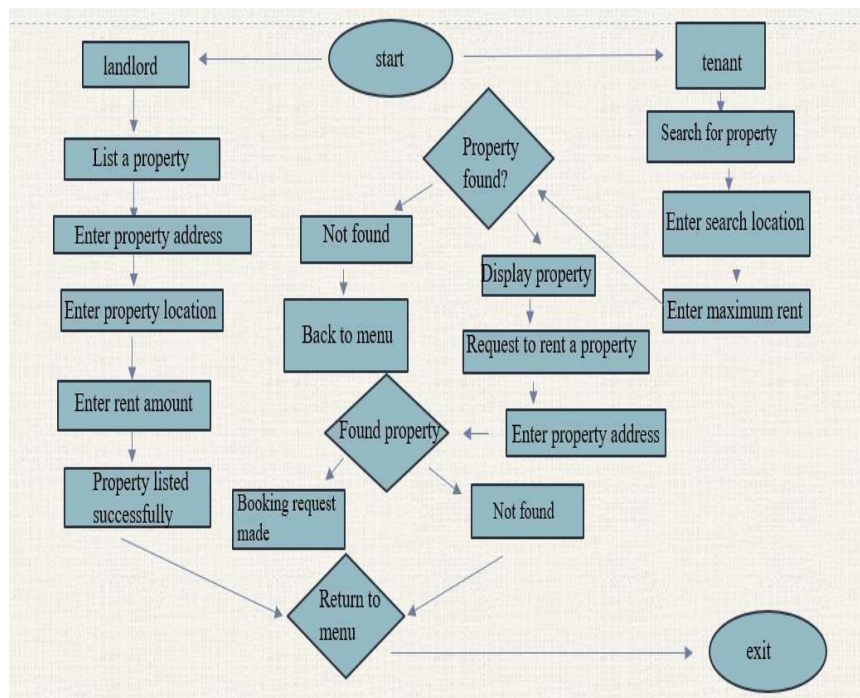


Fig 2.1 : Architecture Diagram

This flowchart illustrates the architecture of a property management system designed for two user roles: landlords and tenants. Landlords can list properties by entering details like the property address, location, and rent amount, after which the property is successfully added to the system. Tenants, on the other hand, can search for properties by specifying their preferred location and maximum rent. If a suitable property is found, tenants can view its details and send a rental request; otherwise, they can modify their search or return to the main menu. The system provides a simple and organized process for property listing and renting..



## CHAPTER 3

### MODULE DESCRIPTION

#### 1. User Management Module

- **Description:** Manages user accounts and authentication for landlords, tenants, and administrators.
- **Key Features:**
  - User registration, login, and role-based access control.
  - Password encryption and secure authentication using Java libraries like Spring Security.
  - Profile management for users to update their details.

#### 1. Property Listing Module

- **Description:** Allows landlords to add and manage property details and tenants to view available properties.
- **Key Features:**
  - CRUD (Create, Read, Update, Delete) operations for property listings using frameworks like Hibernate.
  - Upload and display property images.
  - Search functionality to display properties dynamically.

#### 1. Search and Filter Module

- **Description:** Enables tenants to search and filter properties based on various criteria.
- **Key Features:**
  - Search properties by location, budget, and type using database queries (e.g., SQL or JPQL).
  - Sorting options like price, size, or availability.

### 3.4 Payment Module

- **Description:** Facilitates rent payments securely and tracks payment history.
- **Key Features:**
  - Integration with payment gateways using APIs (e.g., Stripe or Razorpay).
  - Rent receipt generation and email notifications.
  - View and manage payment history for both landlords and tenants.

## 5. Lease Management Module

- **Description:** Handles lease agreements, renewals, and tracking.
- **Key Features:**
  - Digital lease creation with tenant and landlord approvals.
  - Automatic reminders for lease expiry or renewal.
  - Store and manage signed agreements securely.

## **CHAPTER 4**

### **CONCLUSION & FUTURE SCOPE**

#### **4.1 CONCLUSION**

In conclusion, the **House Rental System** provides an innovative and efficient solution for managing the rental process in a digital era. By automating key tasks like property listing, tenant application processing, and rent payments, the system significantly reduces the administrative burden on landlords and enhances the experience for tenants. The integration of secure payment methods, AI-driven property recommendations, and seamless communication between landlords and tenants ensures a smooth, transparent, and user-friendly platform. This makes the rental process faster, more reliable, and accessible to all parties involved.

Moreover, the system's scalability and modular design allow for easy customization and adaptation to different types of properties, rental businesses, and user requirements. The use of modern technologies ensures high security, performance, and user satisfaction, making it an essential tool for property managers and real estate owners. As the real estate industry continues to embrace digital transformation, the House Rental System stands out as a comprehensive and forward-thinking solution that will help meet the growing demands of both landlords and tenants, while fostering a more efficient and streamlined rental experience.

## **4.2 FUTURE SCOPE**

The future scope of a house rental system lies in leveraging advanced technologies to create a more seamless and innovative user experience. Features like AI-powered recommendations can provide personalized property suggestions based on user behavior and preferences, making the search process more efficient. Virtual reality (VR) or 360-degree tours can enable tenants to explore properties remotely, saving time and enhancing convenience. Block chain technology can revolutionize the system by offering secure and tamper-proof digital lease agreements and payment records, ensuring transparency and trust between users. Additionally, dynamic pricing algorithms can help landlords set optimal rental rates by analyzing market trends and demand.

Expanding the system's accessibility through mobile applications and multi-language, multi-currency support will open opportunities for global use. Integrating smart home technology can add features like keyless entry and smart device management, enhancing property appeal. Advanced analytics can provide landlords with insights into rental trends and tenant preferences, enabling better decision-making. Fraud prevention mechanisms and automated tenant screening can improve security and reliability. With these enhancements, the house rental system can evolve into a comprehensive, future-ready platform that caters to the diverse needs of landlords and tenants.

## APPENDIX A (SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.List;

// Abstract User class representing a common user
abstract class User { String name; String email;
public User(String name, String email) {
this.name = name; this.email = email;
}
public abstract void showMenu(JFrame frame, List<Property>
properties, Runnable returnToMainMenu);
// Method to view all properties
public void viewAllProperties(JFrame frame, List<Property>
globalProperties, Runnable returnToMainMenu) {
StringBuilder propertyList = new StringBuilder("All Properties:\n");
if (globalProperties.isEmpty ()) {

propertyList.append("No properties available.");
} else {
```

```

for (Property property : globalProperties) {
    propertyList.append(property.toString()).append("\n");
}
}
JOptionPane.showMessageDialog(frame, propertyList.toString());
}
}
// Landlord class to list properties
class Landlord extends User {
    List<Property> properties;
    public Landlord(String name, String email) {
        super(name, email);
        properties = new ArrayList<>();
    }

    public void listProperty(JFrame frame, List<Property> globalProperties,
        Runnable returnToMainMenu) {
        JPanel panel = new JPanel(new GridLayout(4, 2));
        JTextField addressField = new JTextField();
        JTextField locationField = new JTextField();
        JTextField rentField
        = new JTextField();
        JButton submitButton = new JButton("List Property");
        JButton backButton
        = new JButton("Back to Main Menu");
    }
}

```

```

panel.add(new JLabel("Property Address: "));
panel.add(addressField);
panel.add(new JLabel("Property Location: "));
panel.add(locationField); panel.add(new
JLabel("Rent Amount: "));
panel.add(rentField);
panel.add(submitButton);
panel.add(backButton);
frame.setContentPane(pa nel);
frame.revalidate();
submitButton.addActionListener(e -> {
String address = addressField.getText();
String location = locationField.getText();

try {
double rent = Double.parseDouble(rent Field.getText());
Property newProperty = new Property(address, location, rent);

properties.add(newProper ty);
globalProperties.add(new Property);

JOptionPane.showMessageDialog(frame, "Property listed successfully:\n"
+ newProperty);
} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Invalid rent amount. Please
enter a valid number.");
}
});
backButton.addActionListener(e -> returnToMainMenu.run()
);
}

```

```

@Override public void
showMenu(JFrame frame, List<Property> properties, Runnable
returnToMainMenu) {
    JPanel panel = new JPanel(new GridLayout(3, 1));
    JButton listPropertyButton = new JButton("List a new property");
    geDialog(frame, "Invalid rent amount. Please enter a valid number.");
}
});

```

```

backButton.addActionListener(e -> returnToMainMenu.run()
);
}

```

```

@Override public void
showMenu(JFrame frame, List<Property> properties, Runnable
returnToMainMenu) {
    JPanel panel = new JPanel(new GridLayout(3, 1));
    JButton listPropertyButton = new JButton("List a new property");
    ton);
    panel.add(viewAllButton);
    panel.add(backButton);
    frame.setContentPane(pa nel);
    frame.revalidate();
}
}

```

```

// Tenant class to search properties and request to rent
class Tenant extends User
{
    public Tenant(String name, String email) {
        super(name, email);
    }
}

```



```

public void
searchProperty(JFrame frame, List<Property> properties, Runnable
returnToMainMenu) {
    JPanel panel = new JPanel(new GridLayout(3, 2));
    JTextField locationField = new JTextField();
    JTextField maxRentField = new JTextField();
    JButton searchButton = new JButton("Search");
    JButton backButton
= new JButton("Back to Main Menu");
    panel.add(new JLabel("Enter Location: "));
    panel.add(locationField);
    panel.add(new JLabel("Enter Maximum Rent: "));
    panel.add(maxRentField);
    panel.add(searchButton);
    panel.add(backButton);
    frame.setContentPane(panel);
    frame.revalidate();
    searchButton.addActionListener(e -> {
        String location = locationField.getText();
        try {
            double maxRent = Double.parseDouble(maxRentField.getText());
            StringBuilder result = new StringBuilder("SearchResults:\n");
            boolean found = false;

            for (Property property : properties) {
                if (property.location.equalsIgnoreCase(location) && property.rent <=
maxRent) {

```

```

result.append(property).append("\n");
found = true;
}
}
if (!found) {
    result.append("No properties found matching your criteria.");
}
JOptionPane.showMessageDialog(frame, result.toString());
} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(frame, "Invalid rent amount. Please
enter a valid number.");
}
});
    backButton.addActionListener(e -> returnToMainMenu.run()
);
}
    public void requestToRent(JFrame frame, List<Property> properties,
Runnable returnToMainMenu) {
        JPanel panel = new JPanel(new GridLayout(2, 2));
        JTextField addressField = new JTextField();
        JButton requestButton = new JButton("Request to Rent");
        JButton backButton
= new JButton("Back to Main Menu");
        panel.add(new JLabel("Enter Property Address: "));

```

```

panel.add(addressField);
panel.add(requestButton);
panel.add(backButton);
frame.setContentPane(panel);
frame.revalidate();
requestButton.addActionListener(e -> {
String address = addressField.getText();
for (Property property : properties) {
if (property.address.equalsIgnoreCase(address)) {
JOptionPane.showMessageDialog(frame, "Request made for property: " +
property.address + "\nBooked Successfully!");

Return;
}
}
JOptionPane.showMessageDialog(frame, "Property not found.");
});
backButton.addActionListener(e -> returnToMainMenu.run()
);
}

@Override public void
showMenu(JFrame frame, List<Property> properties, Runnable
returnToMainMenu) {
JPanel panel = new JPanel(new GridLayout(4, 1));
JButton searchButton = new JButton("Search Properties");
JButton requestButton = new JButton("Request to Rent");
JButton viewAllButton = new

```

```

JButton("View All Properties");
JButton backButton
= new JButton("Back to Main Menu");
searchButton.addActionListener(e -> searchProperty(frame, properties,
returnToMainMenu));

requestButton.addActionListener(e -> requestToRent(frame, properties,
returnToMainMenu));

viewAllButton.addActionListener(e -> viewAllProperties(frame, properties,
returnToMainMenu));

backButton.addActionListener(e -> returnToMainMenu.run()
);
panel.add(searchButton);
panel.add(requestButton);
panel.add(viewAllButton);
panel.add(backButton);
frame.setContentPane(panel);
frame.revalidate();
}
}
// Property class representing a rental property
class Property { String address;
String location;
double rent;

```

```

public Property(String address, String location, double rent) {
    this.address = address;
    this.location = location;
    this.rent = rent;
}

@Override
public String toString()
{
    return "Property at "
    + address + " in " + location + " with rent " + rent;
}
} // Main class with GUI public class Main {
    public static void
    main(String[] args) {
        JFrame frame = new JFrame("Rental System");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        List<Property> properties = new ArrayList<>()
        final Runnable[] showMainMenu = new Runnable[1];
        showMainMenu[0]
        = () -> {

        JPanel panel = new JPanel(new GridLayout(3, 1));
        JButton landlordButton = new
        JButton("Landlord");

        JButton tenantButton = new JButton("Tenant");
        JButton exitButton = new JButton("Exit");
        landlordButton.addActionListener(e -> {

```

```

String name = JOptionPane.showInputDialog(frame, "Enter your name:");
String email = JOptionPane.showInputDialog(frame, "Enter your email:");
if (name != null && email != null) {
    Landlord landlord = new Landlord(name, email);
    landlord.showMenu(frame, properties, showMainMenu[0]);
}
});
tenantButton.addActionListener(e -> {
    String name = JOptionPane.showInputDialog(frame, "Enter your name:");
    String email = JOptionPane.showInputDialog(frame, "Enter your email:");
    if (name != null && email != null) {
        Tenant tenant
= new Tenant(name, email);
        tenant.showMenu(frame, properties, showMainMenu[0]);
    }
});
exitButton.addActionListener(e -> System.exit(0));
panel.add(landlordButton
);
panel.add(tenantButton);
panel.add(exitButton);
frame.setContentPane(panel);
frame.revalidate();

};
showMainMenu[0].run();
frame.setVisible(true);
}
}

```

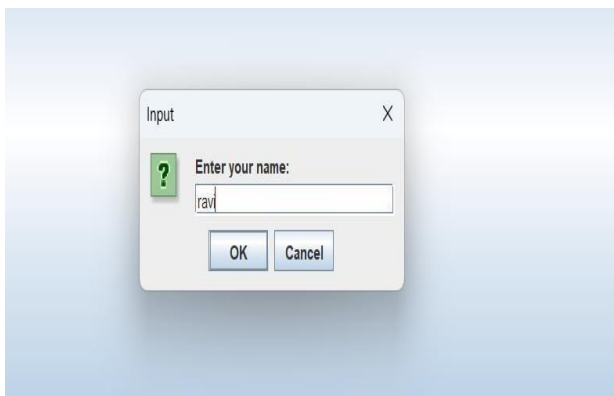
# APPENDIX B

## (SCREENSHOTS)

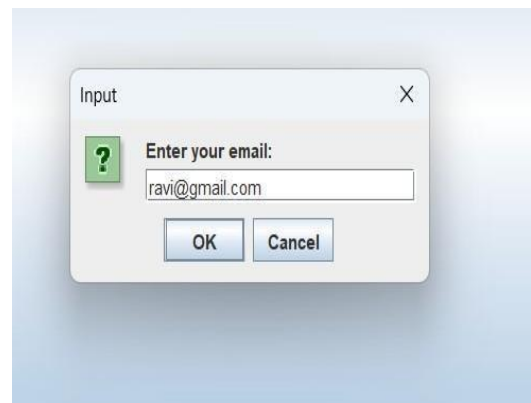
STEP 1:



STEP 2:



STEP 3:



## STEP 4:

<a href="#">List a new property</a>
<a href="#">View All Properties</a>
<a href="#">Back to Main Menu</a>

## STEP 5:

Property Address:	121/229,east street,katpadi
Property Location:	vellore
Rent Amount:	3000
<a href="#">List Property</a>	<a href="#">Back to Main Menu</a>

Message

Property listed successfully:  
Property at 121/229,east street,katpadi in vellore with rent 3000.0

OK



STEP 6:

<div>Search Properties</div>
<div>Request to Rent</div>
<div>View All Properties</div>
<div>Back to Main Menu</div>

STEP 7:

Enter Location:

Enter Maximum Rent:

vellore

Search

Back to Main Menu

Message

Search Results:  
Property at 121/229,east street,katpadi in vellore with rent 3000.0

OK

STEP 8:

Enter Property Address:

121/229,east street,katpadi

Request to Rent

Back to Main Menu

Message

Request made for property: 121/229,east street,katpadi  
Booked Successfully!

OK

STEP 9:

List a new property

Message

All Properties:  
Property at 121/229,east street,katpadi in vellore with rent 3000.0  
Property at 12/17,1st ward,bharathi nagar in chennai with rent 4000.0

OK

Back to Main Menu

## REFERENCES

### GITHUB PROJECTS :

<https://github.com/piyushwani004/OnlineHouseRentingSystem>

<https://github.com/GowthamBabu25/HouseRentalSystem-using-JAVA>

<https://github.com/Rishabhsaini0204/House-Rental-Portal-on-Java> JSP-

MySQL-and Servlet-A-Comprehensive-Guide

### PROJECT :

<https://www.buyprojectcode.in/product/online-house-rental-application>

<https://1000projects.org/online-house-rental-management-java-project.html>

### YOUTUBE LINKS :

<https://www.youtube.com/watch?v=5JgFR6CO7yU>

<https://www.youtube.com/watch?v=MDGQt7tWj1k>

### BOOKS :

<https://www.oreilly.com/openbook/javawt/book>

- Deitel, P. J., & Deitel, H. M., “Java: How to Program,” 11th Edition, Pearson, 2017, ISBN-13: 978-0134743356
- Schildt, H., “Java: The Complete Reference,” 11th Edition, McGraw- Hill Education, 2018, ISBN-13: 978-1259589310
- GeeksforGeeks, "JavaProgramming,"  
<https://www.geeksforgeeks.org/java/>