# DormNest: Student Accommodation Finder

**Prepared by:** Parameshwaran K - 2022503509

Pavithran S - 2022503705

Inbavanan K - 2022503507

**Department :** Computer Technology

**Institution :** Madras Institute of Technology, Anna University.

**Date :** 13-11-2024

**Mentor :** Dr.R.Kathiroli

Computer Technology

Madras Institute of Technology, Anna University.

**Executive Summary**

DormNest is a student accommodation finder application designed to simplify the search for housing tailored to students' needs. Recognizing the challenges students face when looking for affordable, accessible, and suitable accommodations, DormNest serves as a digital platform where students can browse, filter, and select housing options from listings provided by property owners. By bringing both students and owners onto a unified platform, DormNest seeks to streamline the accommodation search and selection process, providing an efficient, user-friendly solution.

**Objectives:**

* To connect students with verified, student-friendly housing options.
* To provide property owners with a dedicated space to list and manage their properties aimed at the student demographic.
* To facilitate an efficient and secure booking process, minimizing the typical challenges associated with finding and securing student housing.

**Key Features and Technical Highlights:**

* User Registration: Tailored registration forms for students and property owners ensure that users provide relevant information for their roles within the application.
* Owner Dashboard: A comprehensive dashboard for property owners where they can view and manage their listings, update property details, and monitor inquiries from students.
* Property Listings: A searchable and filterable database of properties that includes details such as location, pricing, amenities, and property images.
* Database Integration: DormNest is backed by a PostgreSQL database that securely stores user and property data, ensuring smooth retrieval and scalability as the application grows.
* User Interface (UI): A clean, intuitive interface designed to enhance user experience, making it easy for students to navigate listings and for owners to manage properties.
* Security and Data Integrity: Implemented measures to protect user data and prevent unauthorized access, adhering to data privacy standards.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sno** | **Sections** | **Description** |
| **1** | Title Page | Project title, author, institution, date |
| **2** | Executive Summary | Brief overview, objectives, key features |
| **3** | Introduction | Background, purpose, and scope |
| **4** | Requirements Analysis | Functional and non-functional requirements |
| **5** | Implementation | Technologies, database connection, code structure |
| **6** | Testing and Evaluation | Testing approach, test cases, performance evaluation |
| **7** | Conclusion | Summary of contributions and project outcomes |

**Problem Statement**

Finding suitable accommodation is one of the most significant challenges faced by students, especially when they are moving to new cities or countries for their studies. The lack of a centralized platform specifically designed for students leads to inefficient search processes, with students often relying on generic housing platforms that do not cater to their unique needs. Furthermore, property owners who wish to rent their properties to students often face difficulties reaching a targeted audience, managing listings, and responding to inquiries efficiently.

**Key Challenges:**

* **For Students:**
  + Difficulty in finding affordable, reliable, and student-friendly accommodation.
  + Lack of a streamlined process to search and filter properties based on specific student needs (e.g., budget, proximity to educational institutions, amenities).
  + Time-consuming and inefficient communication with property owners.
* **For Property Owners:**
  + Difficulty in reaching students looking for accommodations.
  + The challenge of managing multiple property listings across different platforms.
  + Lack of a system to directly interact with potential student tenants, making it difficult to manage inquiries and respond promptly.

**Introduction**

**Background**

Finding suitable accommodation is a common challenge for students, especially those moving to new cities or countries for their studies. Many face issues such as limited budget options, lack of reliable sources, and time constraints in finding a place to live. Traditional housing search platforms often don't cater specifically to student needs, leading to frustration and inefficiency. DormNest addresses this gap by providing a dedicated platform focused on student accommodation, simplifying the search for affordable, verified, and accessible housing options tailored to students.

**Purpose**

DormNest aims to streamline the student housing search process by connecting students with property owners who offer accommodations suited to student requirements. By centralizing accommodation listings, DormNest provides students with a user-friendly platform to browse, filter, and book housing options, thereby reducing the time and effort needed to find suitable accommodation. For property owners, DormNest provides an interface to manage listings, reach a targeted audience, and communicate directly with prospective student tenants.

**Requirements Overview:**

DormNest encompasses both functional and non-functional requirements to ensure it meets the needs of users efficiently:

* **Functional Requirements:**
  + User Registration and Authentication: Allows students and property owners to create accounts, ensuring secure access.
  + Owner Dashboard: Provides a portal for property owners to manage property listings, add details, and update availability.
  + Property Listings with Search and Filters: Displays property details, including location, pricing, and amenities, allowing students to search and filter based on their preferences.
  + Database Integration: Uses PostgreSQL to manage and store all user and property information securely.
  + Communication Interface: Enables direct interaction between students and property owners to arrange property viewings or negotiate terms.
* **Non-Functional Requirements:**
  + Usability: Designed for ease of use, with a straightforward interface that ensures seamless navigation for both students and property owners.
  + Performance: Ensures quick response times for loading property listings and dashboard actions.
  + Security: Protects user data through secure authentication and data handling practices.
  + Scalability: Built to handle an increasing number of users and properties as the application grows in popularity.

**Requirements Analysis**

**Functional Requirements**

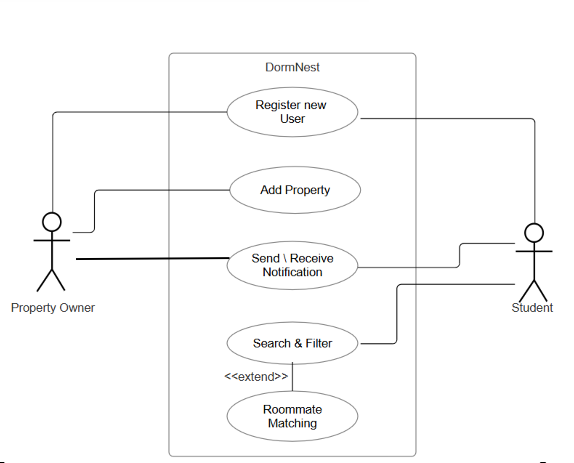
DormNest provides several core functions to facilitate efficient user interaction and property management. The main functional requirements include:

* **User Registration and Authentication:**
  + Allows students and property owners to create and securely access accounts.
  + Authentication protocols ensure that only registered users can access specific functionalities.
* **Property Listing Creation and Management:**
  + Property owners can create listings, including details such as location, price, property type, amenities, and upload images.
  + Owners can edit or update listings to reflect current availability or pricing changes.
* **Search and Filtering of Properties:**
  + Students can search for properties based on specific criteria, including location, budget range, and preferred amenities.
  + Filter options provide students with targeted search results based on their individual needs.
* **Owner Dashboard:**
  + A dedicated interface for property owners to view and manage their listings.
  + Owners can monitor engagement (such as views or inquiries) and respond to messages from interested students.
* **Student/Owner Communication:**
  + Students can send requests directly to property owners, facilitating a streamlined communication process.
* **Database Integration with PostgreSQL:**
  + Stores and manages all user data (students and owners) and property listing details in a PostgreSQL database.
  + Ensures efficient retrieval and updates of information while maintaining data integrity.

**Non-Functional Requirements**

To ensure DormNest operates effectively, it adheres to the following non-functional requirements:

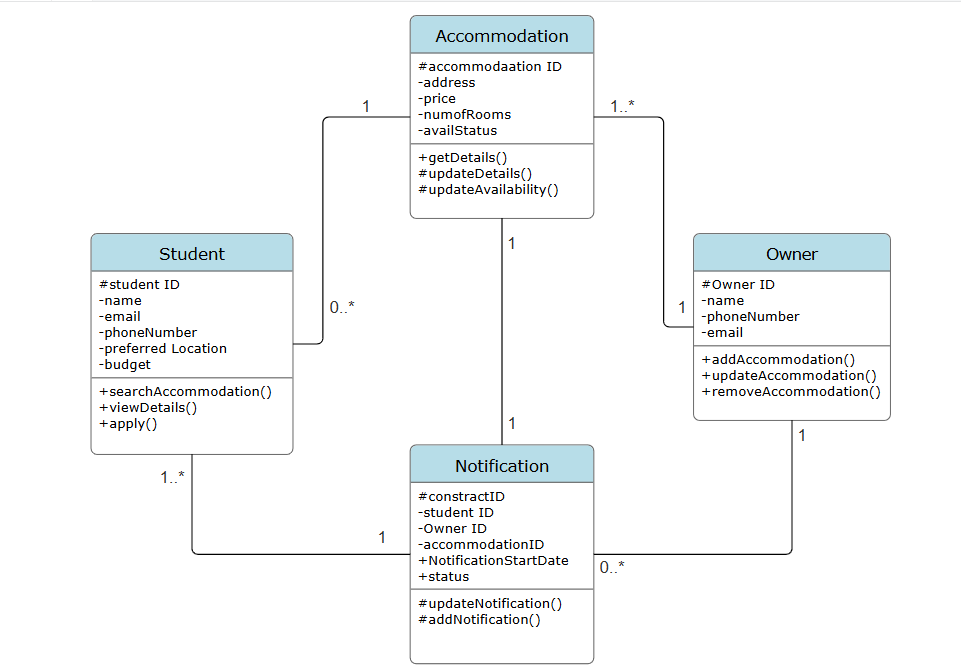
* **Usability:**
  + Designed with a simple and intuitive user interface, making it easy for students and property owners to navigate.
  + Provides a smooth user experience with straightforward access to search, filter, and dashboard functionalities.
* **Performance:**
  + Optimized to load property listings quickly and handle simultaneous user actions without delays.
* **Scalability:**
  + Built with scalability in mind, allowing for increased listings and user activity as the application’s popularity grows.
  + Database and application architecture support expansion to accommodate additional functionalities or higher traffic.
* **Security:**
  + Incorporates secure authentication and authorization mechanisms to protect user data and restrict unauthorized access.
  + Follows data protection standards, ensuring sensitive user information is encrypted and stored securely.

**Use Case Diagram: **

**Requirement Traceability Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. ID** | **Description** | **Design** | **Status** |
| **REQ-001** | User registration | Swing forms | Complete |
| **REQ-002** | Property listing management | Owner dashboards | Complete |
| **REQ-003** | Property search and filter | Search in UI | Complete |
| **REQ-004** | User login/logout | Login Form | Complete |
| **REQ-005** | Database integration | PostgreSQL Schema | Complete |

**Class Diagram**

****

**Class Responsibility-Collaboration cards** **CRC**

|  |  |
| --- | --- |
| **Student** | |
| student ID | Notification |
| Name | Accommodation |
| Email |  |
| Phone number |  |
| Preferred Location |  |
| Budget |  |
| Search for Accommodation |  |
| View Accommodation Details |  |
| Apply for Accommodation |  |

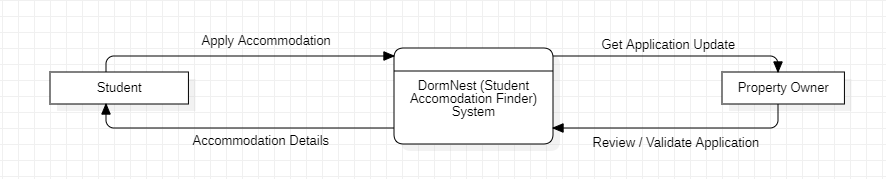
|  |  |
| --- | --- |
| **Accommodation** | |
| Accommodation ID | Student |
| Accommodation Address | Owner |
| Price | Notification |
| Number of rooms |  |
| Availability Status |  |
| Get details of accommodation |  |
| Update details of Accomodation |  |
| Update availability status |  |

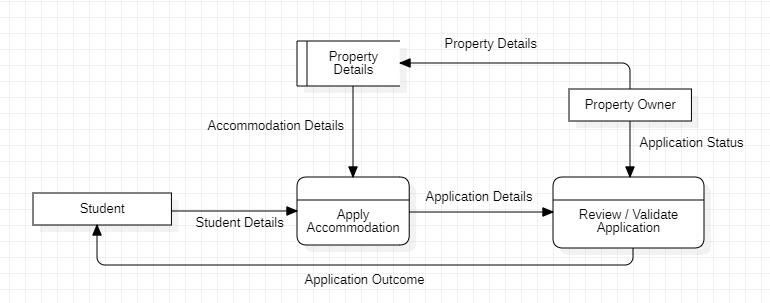
|  |  |
| --- | --- |
| **Owner** | |
| Owner ID | Notification |
| Owner Name | Accommodation |
| Email |  |
| Phone number |  |
| Add an Accommodation |  |
| Update Details of accommodation |  |
| Remove the accommodation |  |

|  |  |
| --- | --- |
| **Notification** | |
| Notification ID | Student |
| Student ID | Accommodation |
| Owner ID | Owner |
| Accommodation ID |  |
| Notification start date |  |
| Notification Status |  |
| Update Status |  |
| Add Notification |  |
| Delete Notification |  |

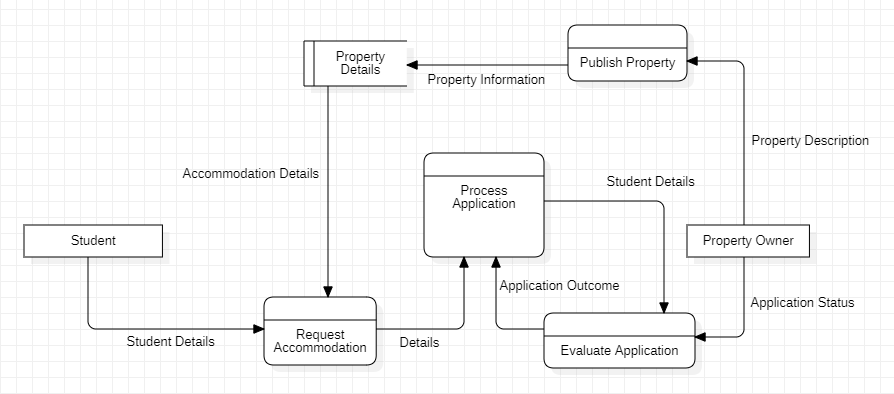
**Data Flow Diagarm DFD**

**Level 0:**

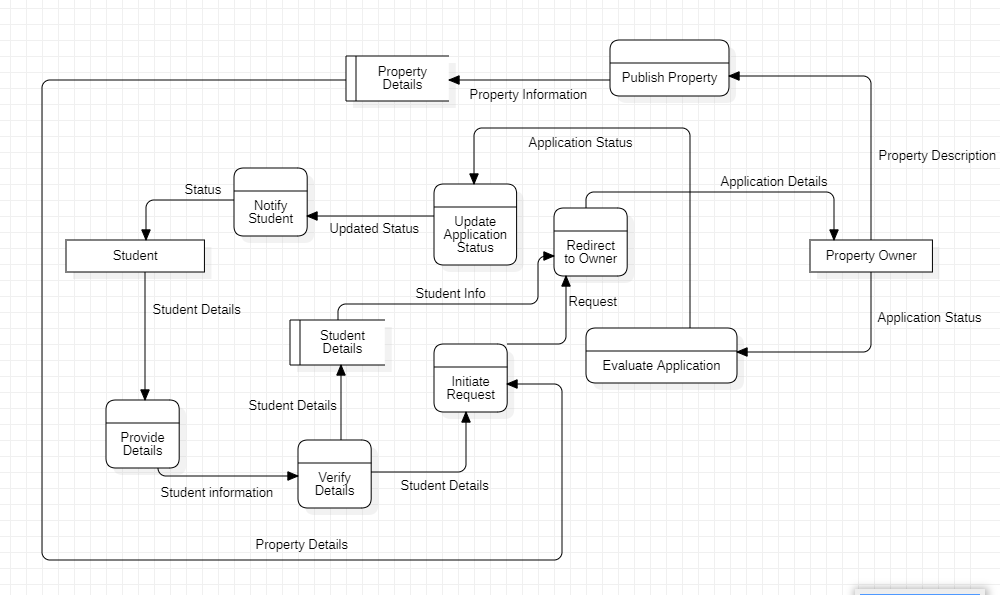


**Level 1:** 

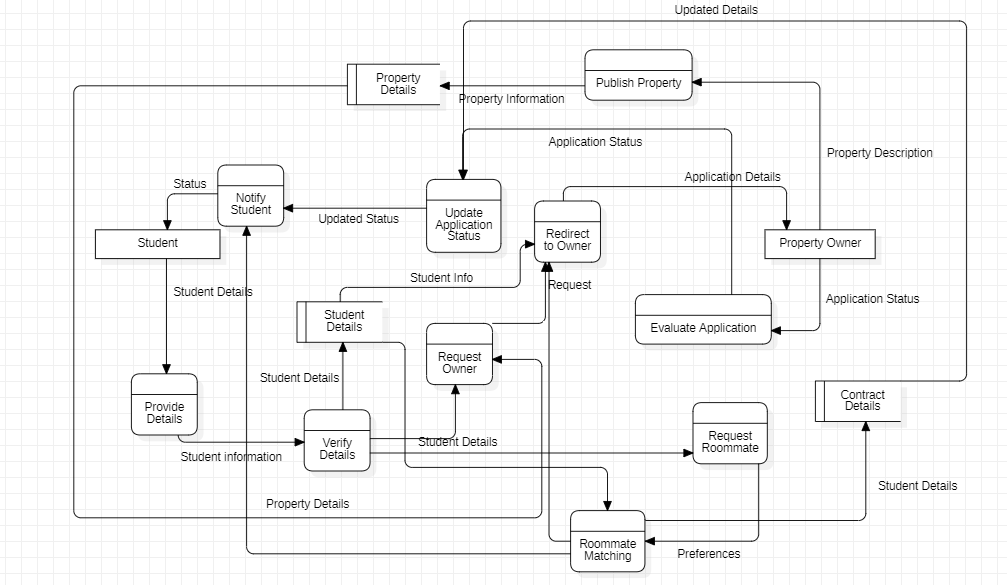
**Level 2:**



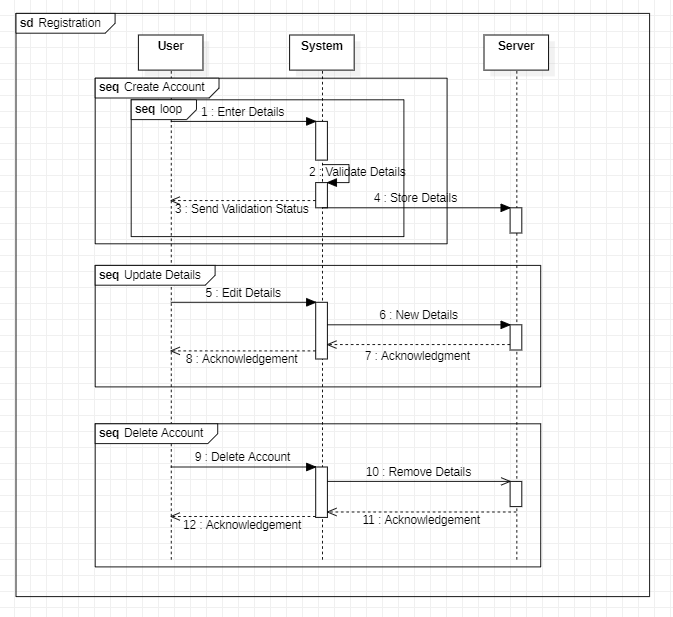
**Level 3:**



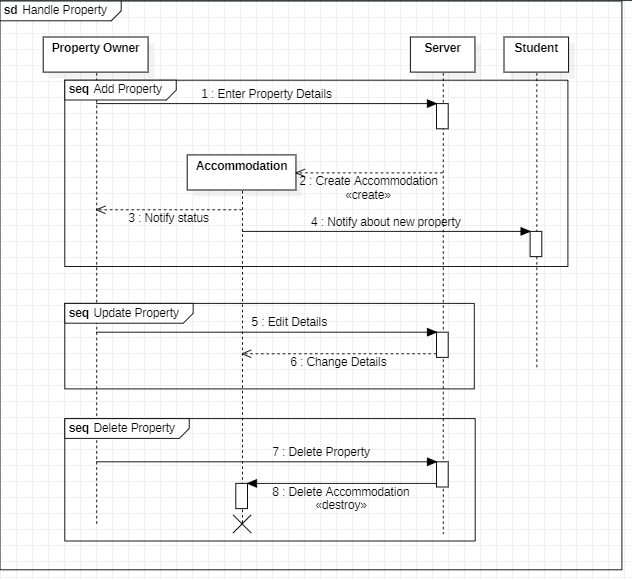
**Level 4:**

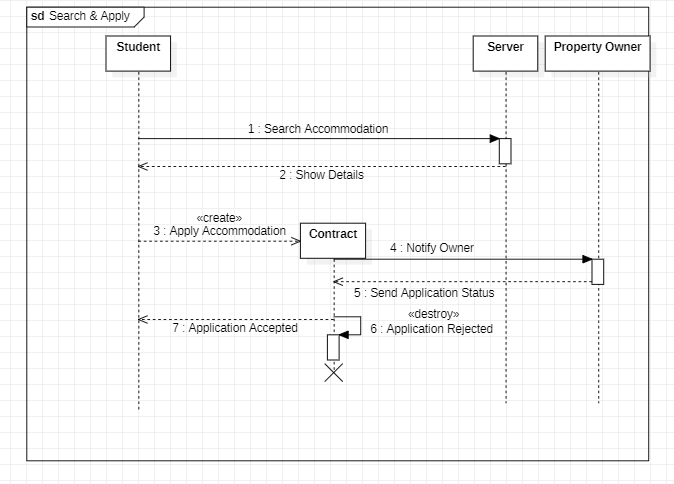


**Registration:**

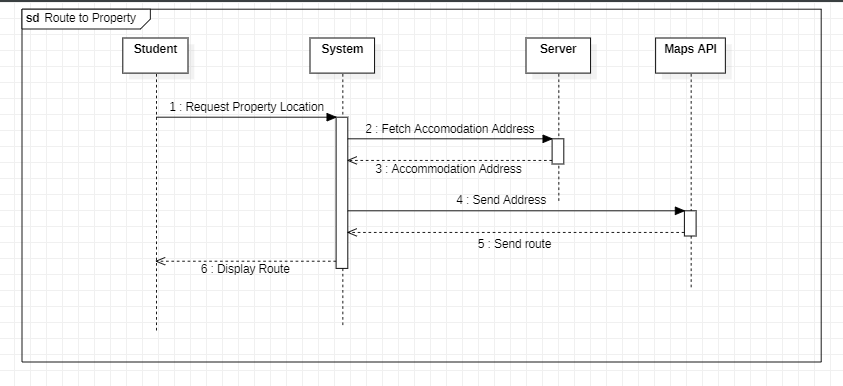


**Handle Property:**

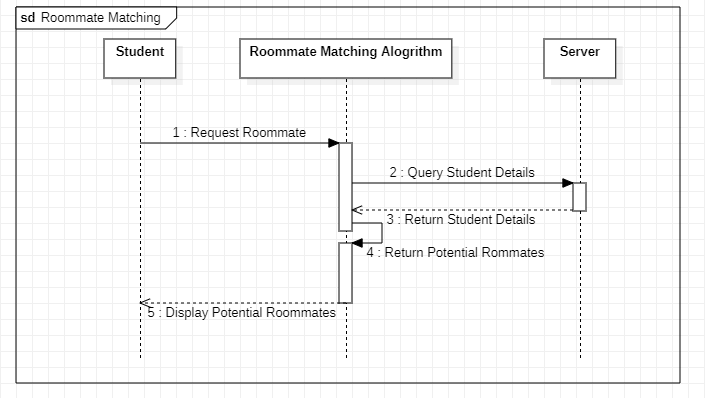


**Search & Apply:** 

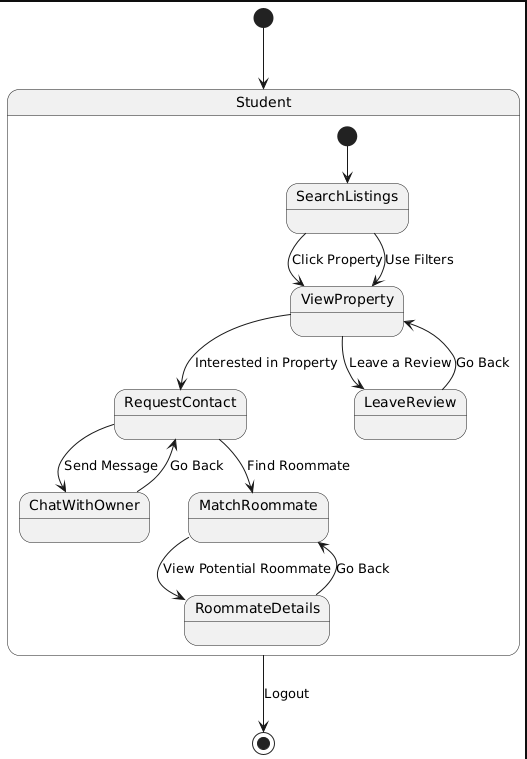
**Route to Property:**

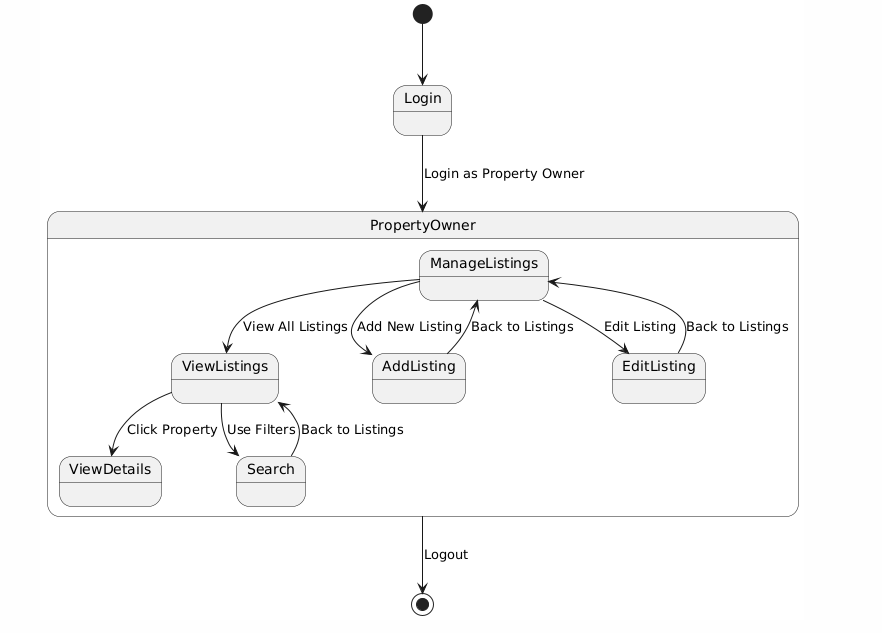


**Roommate Matching:**



**State Machine Diagram**



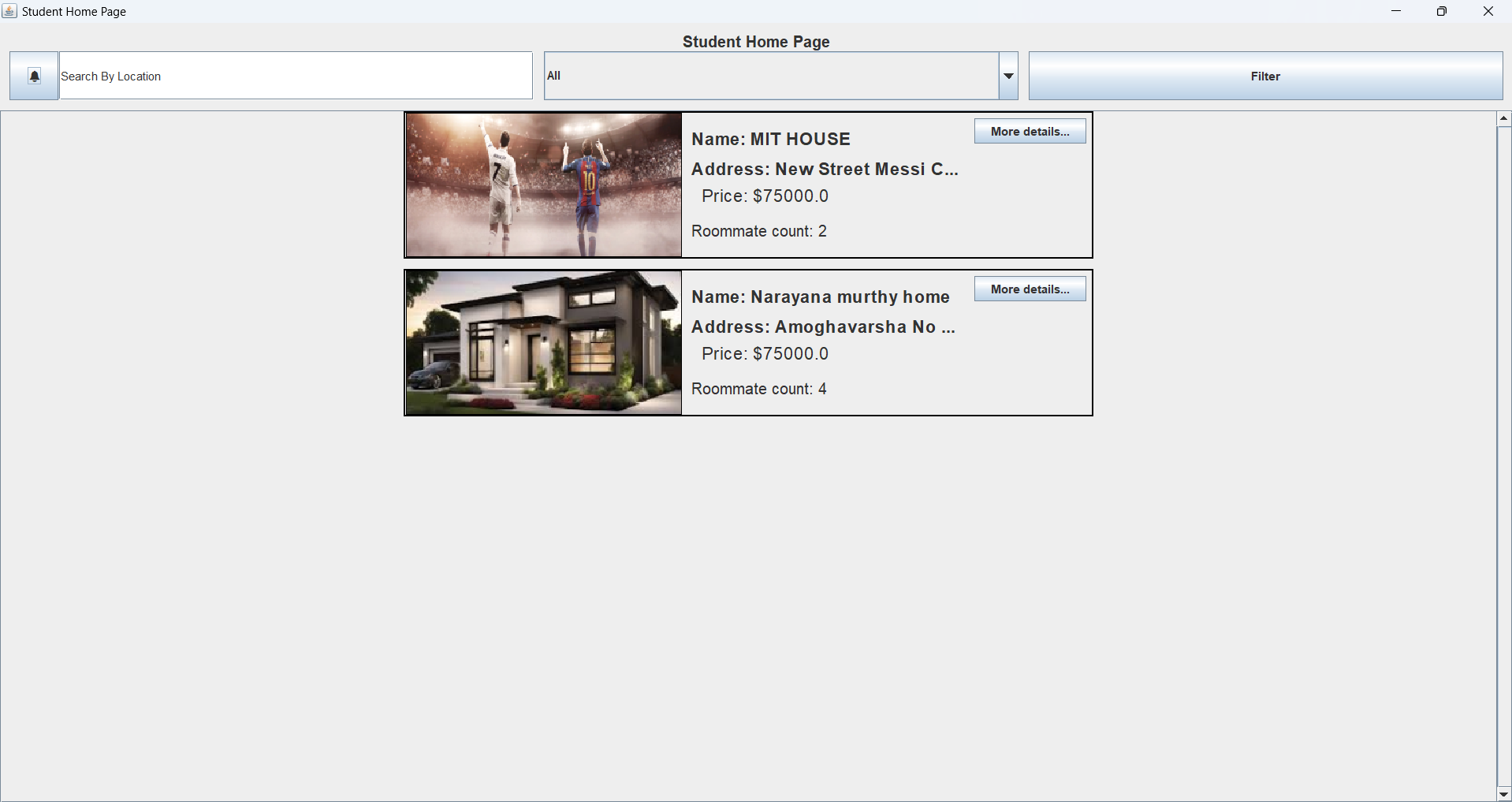


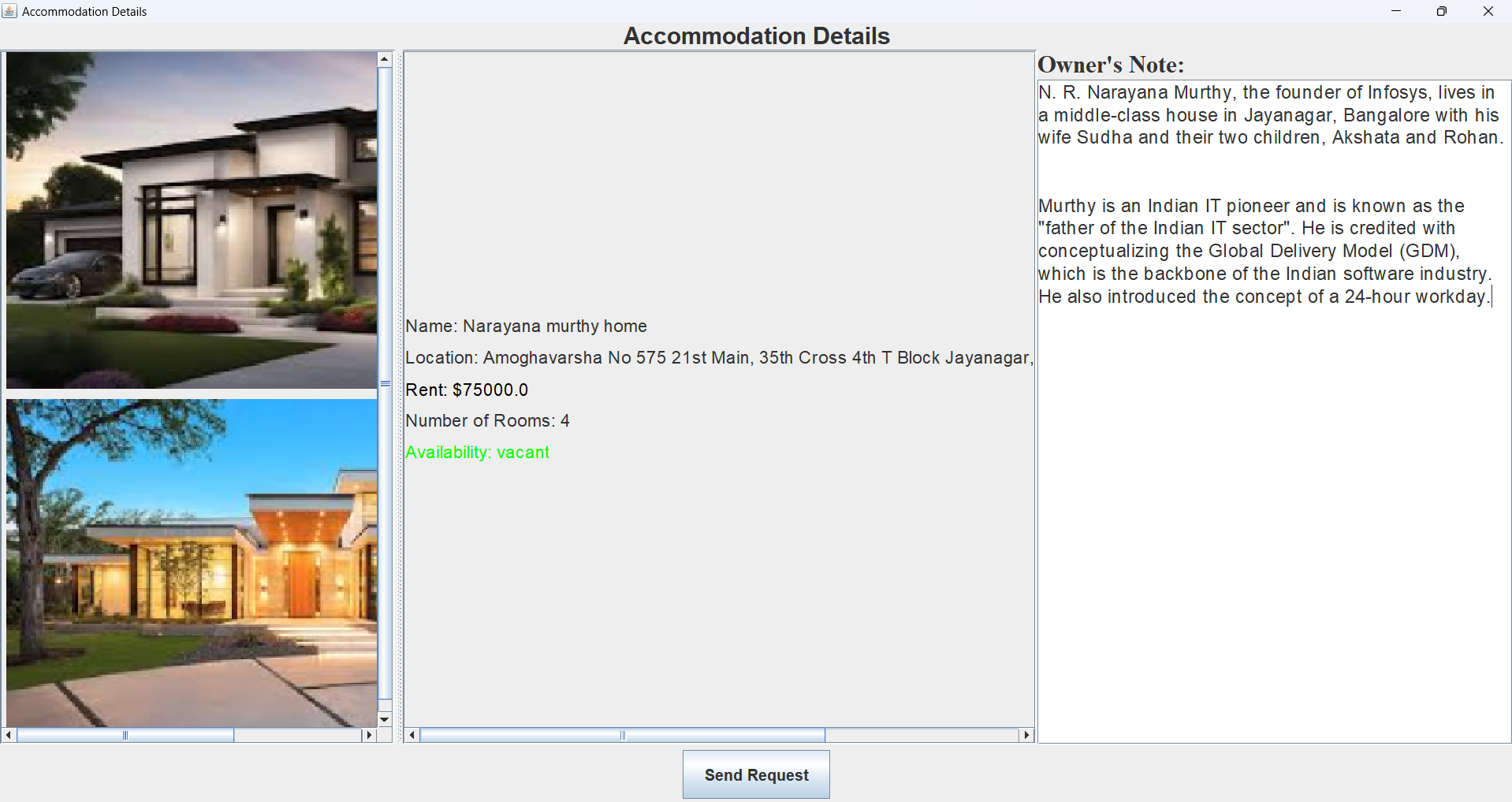
**Sample Implementation of Student Home Page**

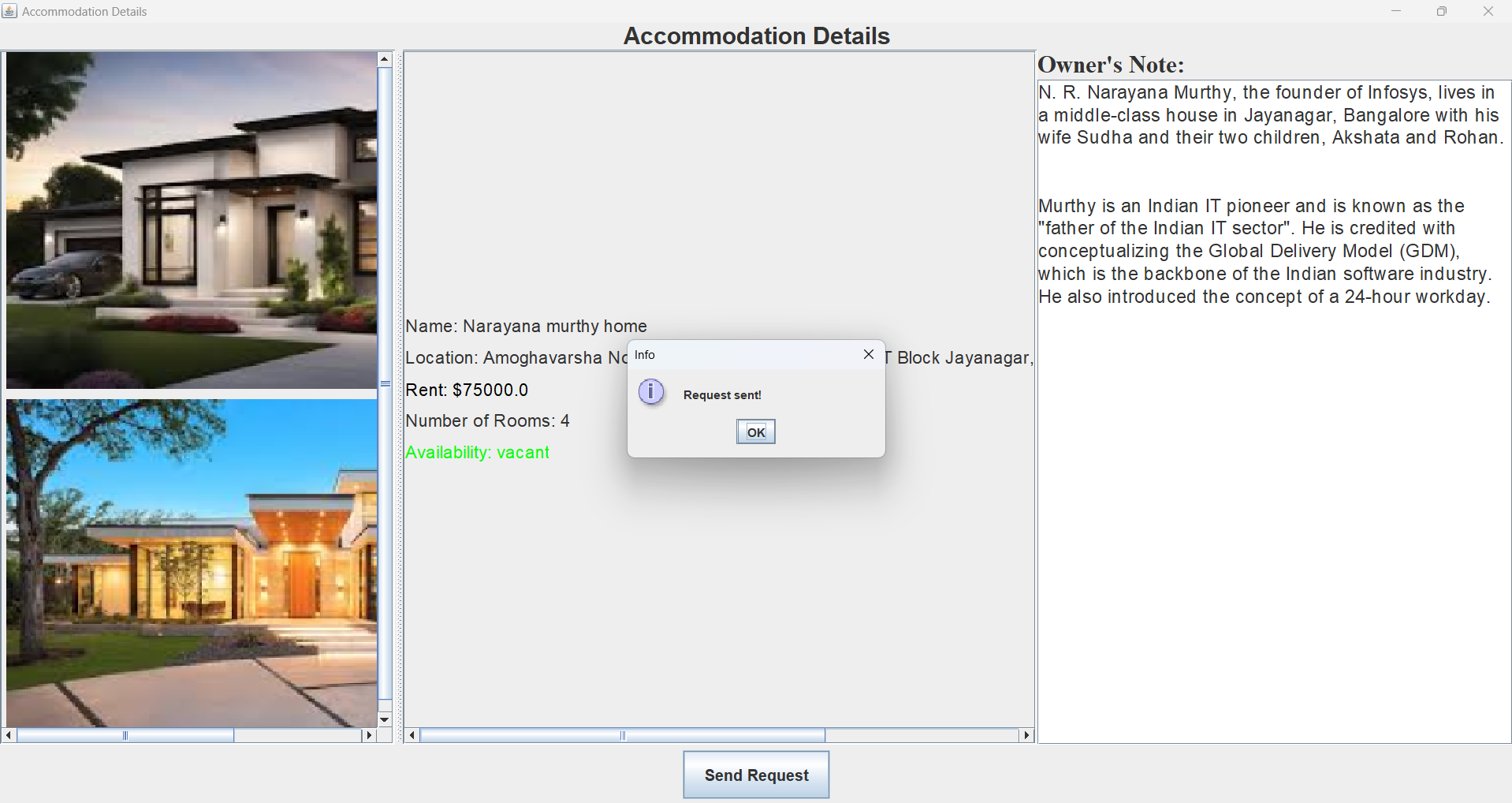
**Code:**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.\*;  
  
public class StudentHomePageGUI extends JFrame {  
  
 int userID;  
 private JPanel contentPanel;  
 private JComboBox<String> rentFilter;  
 private JTextField locationField;  
  
 public StudentHomePageGUI( int userID ) {  
  
 this.userID = userID;  
  
 setTitle("Student Home Page");  
 setSize(800, 600);   
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setLayout(new BorderLayout());  
 setExtendedState(JFrame.*MAXIMIZED\_BOTH*);  
  
 JPanel mainHeaderPanel = new JPanel(new BorderLayout());  
 mainHeaderPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));  
  
 JLabel notificationHeading = new JLabel("Student Home Page", SwingConstants.*CENTER*);  
 notificationHeading.setFont(new Font("Arial", Font.*BOLD*, 16));  
 mainHeaderPanel.add(notificationHeading, BorderLayout.*NORTH*);  
  
 JButton notificationButton = new JButton("🔔");  
 notificationButton.setPreferredSize(new Dimension(50, 50));  
 mainHeaderPanel.add(notificationButton, BorderLayout.*WEST*);  
 notificationButton.addActionListener(e -> {  
 StudentNotificationGUI sn = new StudentNotificationGUI();  
 });  
  
 JPanel searchPanel = new JPanel(new GridLayout(1, 3, 10, 10));  
 locationField = new JTextField("Search By Location");  
  
 rentFilter = new JComboBox<>(new String[]{"All", "0-100", "100-200", "200-300", "300-400", "400+"});  
 JButton filterButton = new JButton("Filter");  
 filterButton.addActionListener(new FilterAction());  
  
 searchPanel.add(locationField);  
 searchPanel.add(rentFilter);  
 searchPanel.add(filterButton);  
  
 mainHeaderPanel.add(searchPanel, BorderLayout.*CENTER*);  
 add(mainHeaderPanel, BorderLayout.*NORTH*);  
  
 contentPanel = new JPanel();  
 contentPanel.setLayout(new BoxLayout(contentPanel, BoxLayout.*Y\_AXIS*));  
 JScrollPane scrollPane = new JScrollPane(contentPanel);  
 scrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  
  
 loadAccommodations(null, null);  
 add(scrollPane, BorderLayout.*CENTER*);  
 setVisible(true);  
 }  
  
 private void loadAccommodations(String location, String rentRange) {  
 contentPanel.removeAll();  
  
 DB\_Functions db = new DB\_Functions();  
 try (Connection conn = db.connect\_to\_db()) {  
 String query = String.*format*("SELECT \* FROM accommodation WHERE user\_id=%d",userID);  
  
 if (location != null && !location.isEmpty()) {  
 query += " AND accommodation\_address ILIKE ?";  
 }  
  
 if (rentRange != null) {  
 if (rentRange.equals("0-100")) {  
 query += " AND rent <= 100";  
 } else if (rentRange.equals("100-200")) {  
 query += " AND rent > 100 AND rent <= 200";  
 } else if (rentRange.equals("200-300")) {  
 query += " AND rent > 200 AND rent <= 300";  
 } else if (rentRange.equals("300-400")) {  
 query += " AND rent > 300 AND rent <= 400";  
 } else if (rentRange.equals("400+")) {  
 query += " AND rent > 400";  
 }  
 }  
  
 PreparedStatement stmt = conn.prepareStatement(query);  
 int paramIndex = 1;  
  
 if (location != null && !location.isEmpty()) {  
 stmt.setString(paramIndex++, "%" + location + "%");  
 }  
  
 ResultSet rs = stmt.executeQuery();  
 while (rs.next()) {  
 String accName = rs.getString("accommodation\_name");  
 String address = rs.getString("accommodation\_address");  
 String price = "$" + rs.getDouble("rent");  
 int roommateCount = rs.getInt("numRooms");  
 int accId = rs.getInt("accommodation\_id");  
  
 String query2 = "SELECT image\_data FROM accommodation\_images WHERE accommodation\_id = ?";  
 PreparedStatement stmt2 = conn.prepareStatement(query2);  
 stmt2.setInt(1, accId);  
 ResultSet forImage = stmt2.executeQuery();  
  
 ImageIcon accImage = null;  
 ImageIcon scaledAccImage = null;  
 if (forImage.next()) {  
  
 byte[] imageBytes = forImage.getBytes("image\_data");  
 if (imageBytes != null) {  
 accImage = new ImageIcon(imageBytes);  
  
 Image scaledImage = accImage.getImage().getScaledInstance(280, 200, Image.*SCALE\_SMOOTH*);  
 scaledAccImage = new ImageIcon(scaledImage);  
 }  
 }  
 forImage.close();  
 stmt2.close();  
  
 JPanel accommodationCard = createAccommodationCard(accId, accName, scaledAccImage, address, price, roommateCount);  
 contentPanel.add(accommodationCard);  
 contentPanel.add(Box.*createVerticalStrut*(10));   
 }  
 rs.close();  
 stmt.close();  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(null, e.getMessage());  
 }  
  
 contentPanel.revalidate();  
 contentPanel.repaint();  
 }  
  
 private JPanel createAccommodationCard(int accoID,String accName,ImageIcon accImage ,String address, String price, int roommateCount) {  
 JPanel card = new JPanel(new BorderLayout());  
 card.setPreferredSize(new Dimension(700, 150));   
 card.setMaximumSize(new Dimension(700, 150));   
 card.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 card.setBackground(Color.*WHITE*);  
  
 JLabel photoLabel = new JLabel(accImage); //"Photo", SwingConstants.CENTER  
 photoLabel.setPreferredSize(new Dimension(280, 200));   
 photoLabel.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  
 card.add(photoLabel, BorderLayout.*WEST*);  
  
 JPanel infoPanel = new JPanel(new GridLayout(4, 1));  
 infoPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));  
  
 JLabel accNameLabel = new JLabel("Name: "+ accName);  
 accNameLabel.setFont(new Font("Arial", Font.*BOLD*, 18));  
 infoPanel.add(accNameLabel);  
  
 JLabel addressLabel = new JLabel("Address: " + address);  
 addressLabel.setFont(new Font("Arial", Font.*BOLD*, 18));  
 infoPanel.add(addressLabel);  
  
 JPanel detailsPanel = new JPanel(new FlowLayout(FlowLayout.*LEFT*, 10, 0));  
 JLabel priceLabel = new JLabel("Price: " + price);  
 priceLabel.setFont(new Font("Arial", Font.*PLAIN*, 18));  
 JLabel roommateCountLabel = new JLabel("Roommate count: " + roommateCount);  
 roommateCountLabel.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 detailsPanel.add(priceLabel);  
 infoPanel.add(detailsPanel);  
 infoPanel.add(roommateCountLabel);  
  
 card.add(infoPanel, BorderLayout.*CENTER*);  
  
   
 JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.*RIGHT*));  
 JButton detailsButton = new JButton("More details...");  
 detailsButton.addActionListener(e -> {  
  
  
 AccommodationDetailsSwingGUI accDetailedPage = new AccommodationDetailsSwingGUI(2, accoID , true );  
 });  
 buttonPanel.add(detailsButton);  
  
 card.add(buttonPanel, BorderLayout.*EAST*);  
  
 return card;  
 }  
  
 private class FilterAction implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String location = locationField.getText().equals("Search By Location") ? "" : locationField.getText();  
 String rentRange = (String) rentFilter.getSelectedItem();  
 loadAccommodations(location, rentRange.equals("All") ? null : rentRange);  
 }  
 }  
  
 public static void main(String[] args) {  
 new StudentHomePageGUI(1);  
 }  
}

**Result:**

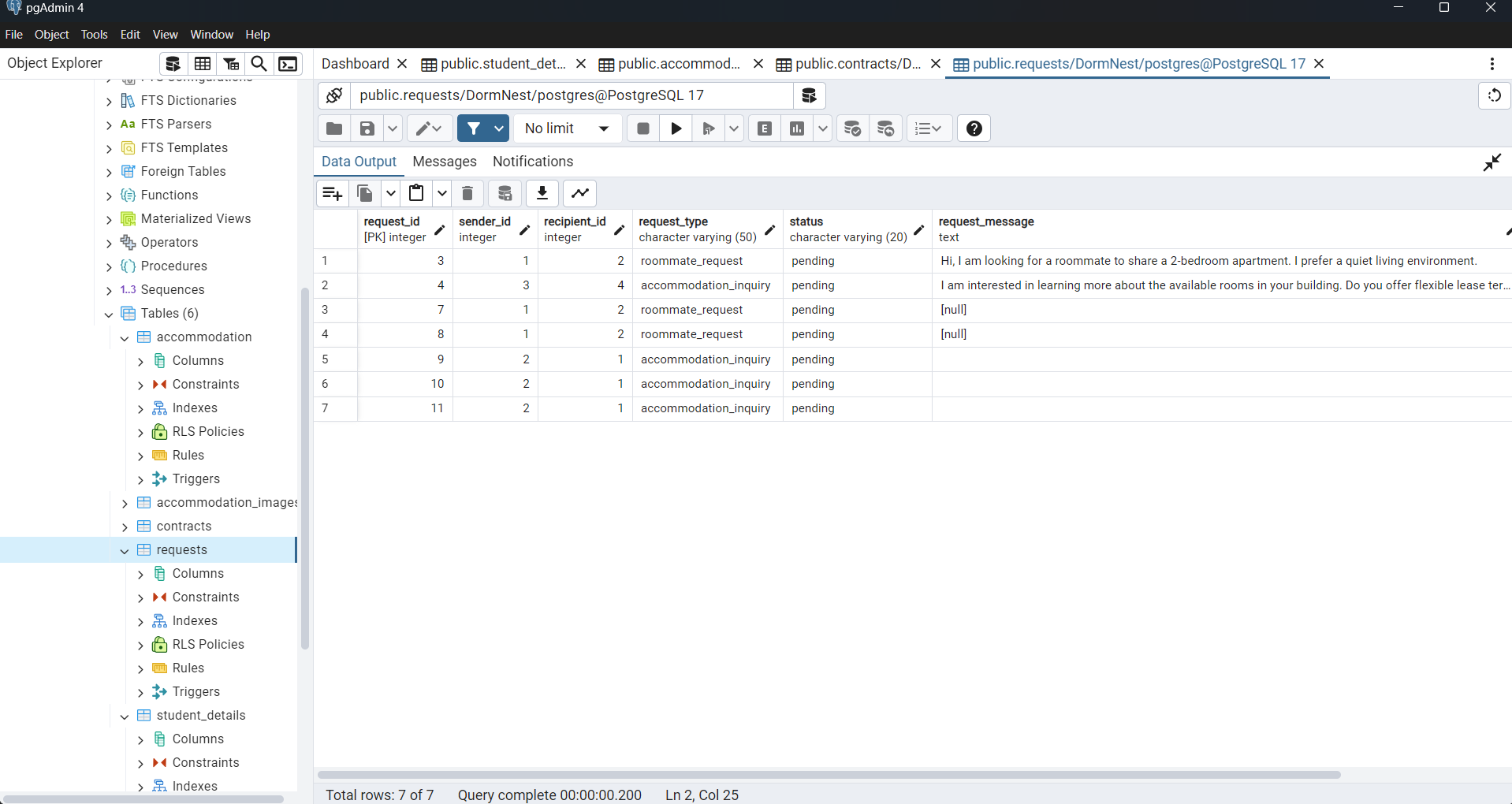








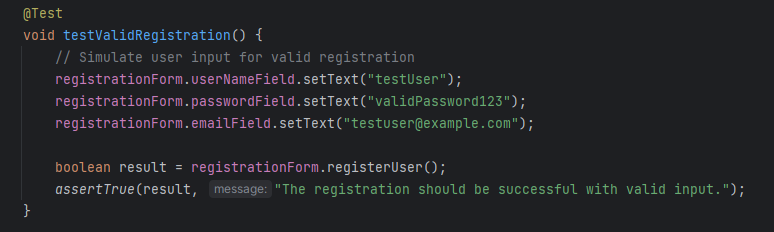
**DataBase:**

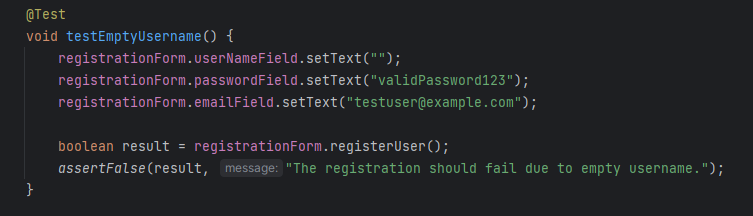


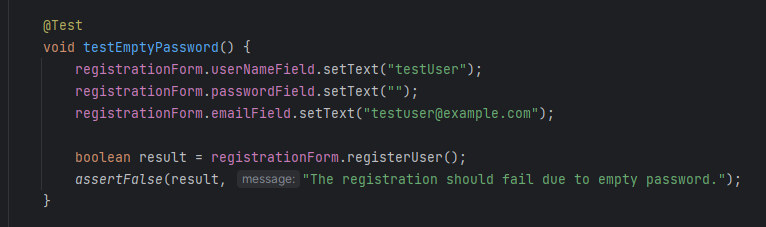
**Testing and Evaluation**

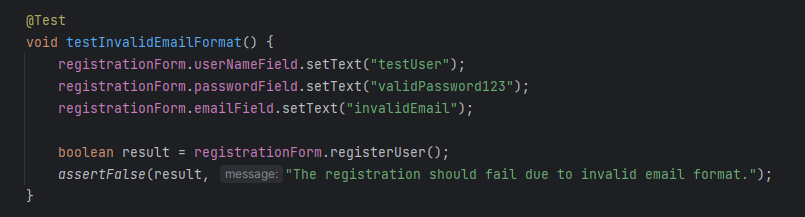
**Creating a Tester Class to test Register Module:**

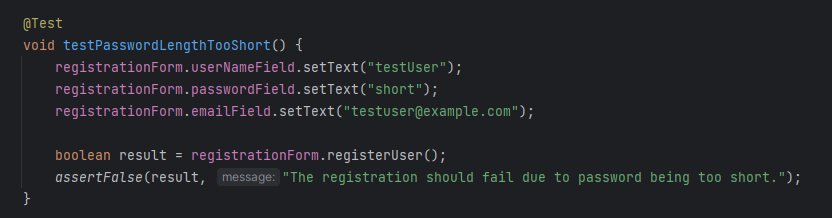
**Register Page contains Several Testers (Listed Below) :**

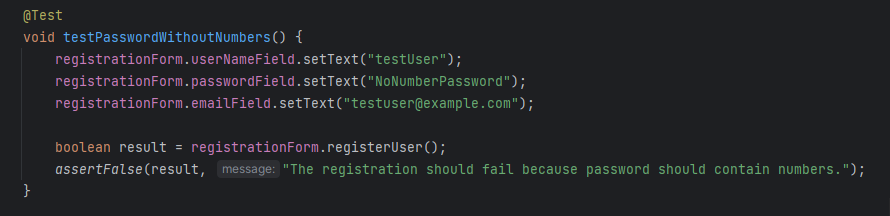
****

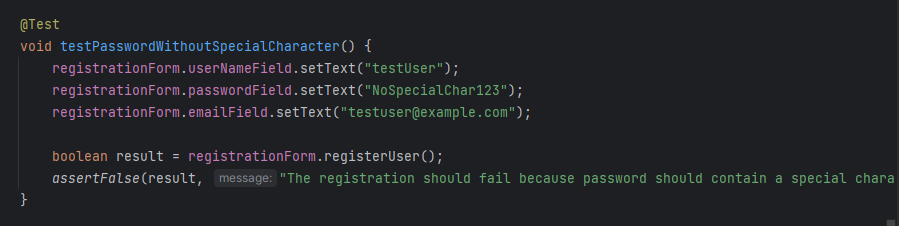
****

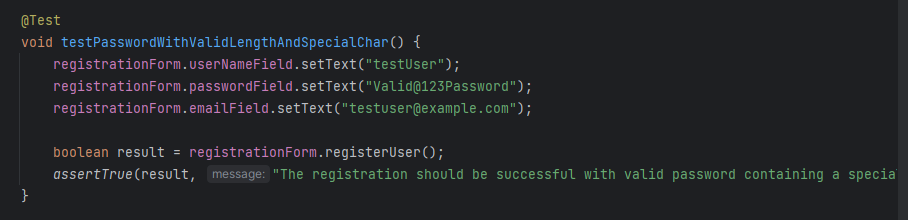
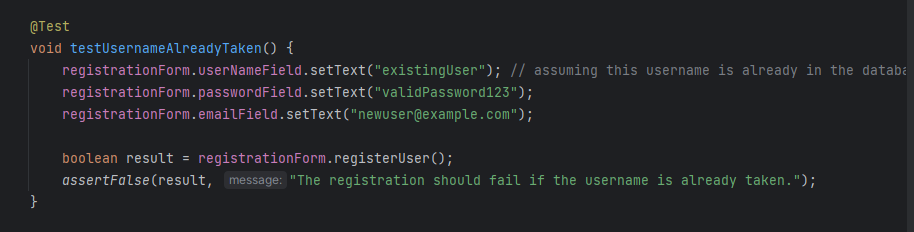
****

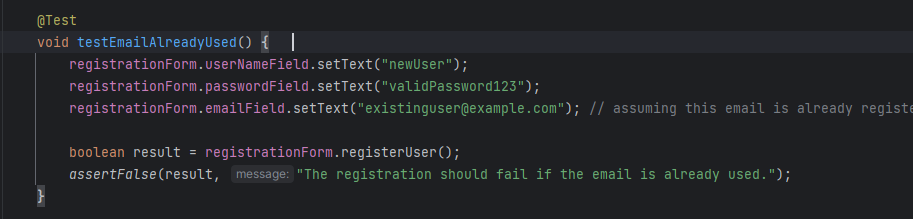
** **

****

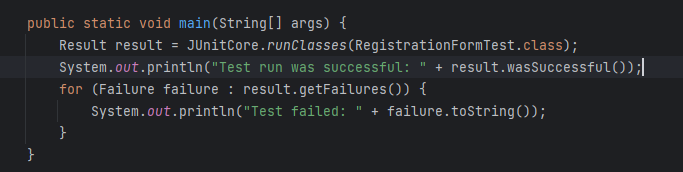
****

****

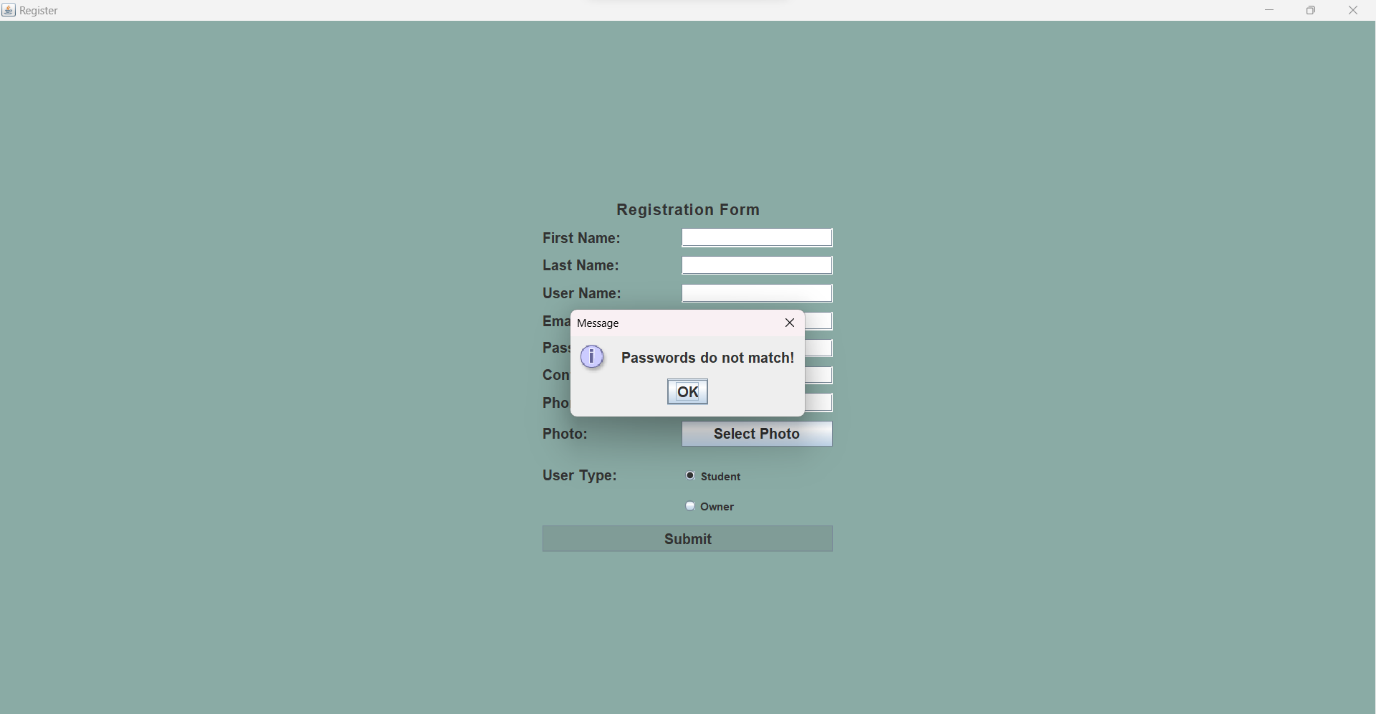
** **

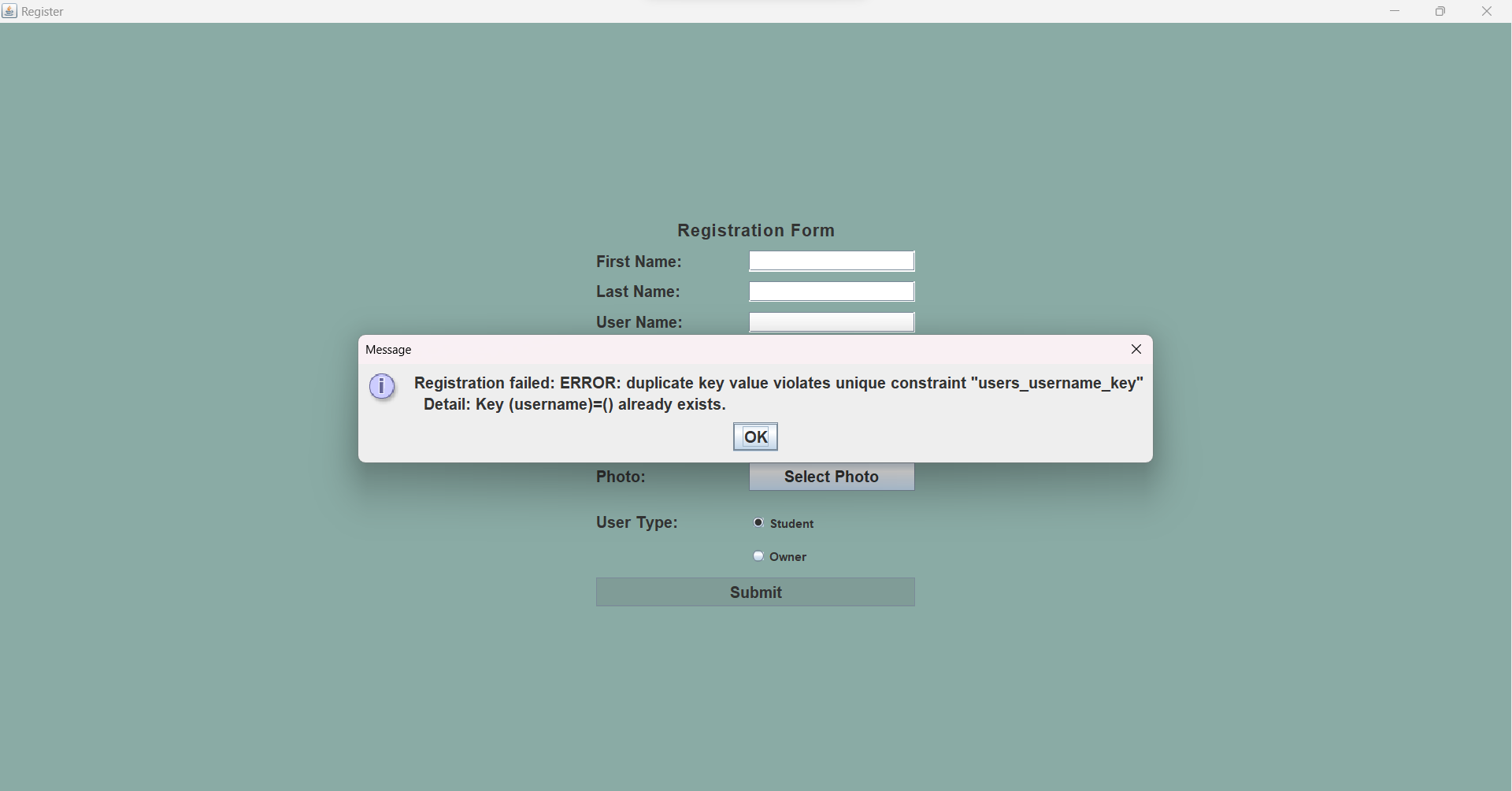
****

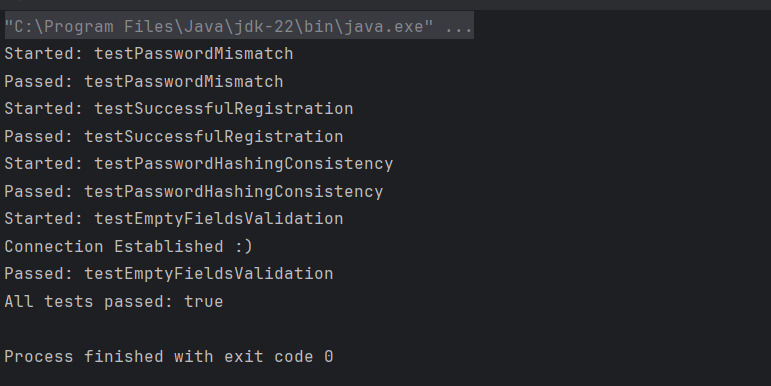
**Driver to run the Test Methods:**

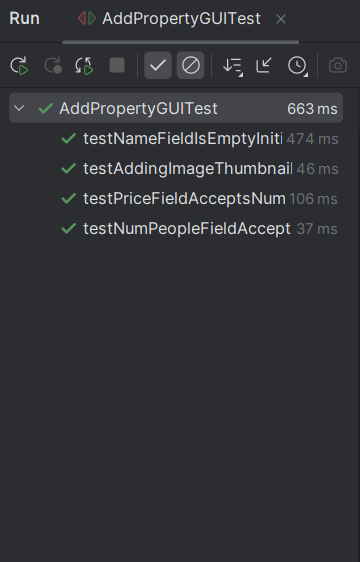
****

**Final Result of All the Test Cases:**

****

****

****

****

**Conclusion:**

The development of DormNest has addressed the pressing need for a streamlined, student-focused accommodation finder. By providing a dedicated platform that connects students with property owners, DormNest simplifies the housing search process for students while helping owners reach their target audience efficiently. The platform’s essential features—including user registration, property listing management, search and filter functionalities, and a communication system—contribute to creating a user-friendly and effective tool for both students and property owners.

Throughout this project, I have gained invaluable skills, particularly in Java Swing for UI design, PostgreSQL database integration, and building a user-centered application. This experience has strengthened my knowledge of application architecture, data management, and secure coding practices.