REAL ESTATE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

HARIRAJ G	**230701104**
DIVYADHARSHINI K	**230701081**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report "**REAL ESTATE MANAGEMENT SYSTEM**" is the bonafide work of **"HARIRAJ G (230701104), DIVYADHARSHINI K (230701081)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

Mrs. Divya M
ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

SIGNATURE

Mr. Ragu
ASSISTANT PROFESSOR
Dept. of Computer science and Engg,
Rajalakshmi Engineering College
Chennai

EXTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT:

The Real Estate Management System is a streamlined, user-friendly platform for managing real estate specifically focused on buying, selling, and renting properties. By centralizing property listings and user data, the system enables users to efficiently connect with property owners or agents. It offers advanced real-time updates to help buyers and renters find properties that suit their criteria while empowering owners and agents to reach a broad audience. The system organizes listings with essential details-location, price, and property type making information easily accessible. This centralization reduces search time, while property owners can manage listings, update details, and track inquiries efficiently. By bridging the gap between owners and buyers.The system enhances communication, ensures data accuracy, and enables faster transactions, thus streamlining the overall real estate process.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 INTRODUCTION

The Real Estate Management System is an advanced, web-based platform built using Java and SQL, tailored to revolutionize the real estate experience for property buyers, sellers, renters. This system is designed to centralize and streamline property transactions by offering a comprehensive suite of tools for managing property listings, handling user inquiries, and facilitating efficient communication across all parties involved.

They can create and update property listings, adjust property details such as location, price, and type, and track interest and inquiries in real time, ensuring data remains current and accurate. Prospective buyers and renters can access a user-friendly interface that simplifies property search based on specific criteria, helping them find homes or commercial spaces that best meet their needs.

The Real Estate Management System's real-time updates reduce search time and improve decision-making by allowing buyers and renters to stay informed on property availability and price changes instantly.

1.2 OBJECTIVES

1. Develop a secure, accessible platform with distinct logins for property managers and clients.
2. Enhance the organization, speed, and accuracy of property management tasks, allowing real estate agents and property owners to efficiently connect with potential buyers or renters.
3. Support clients in making well-informed decisions by offering clear, real-time information on property details, availability, and pricing.
4. Improve transaction transparency and communication, enabling all users to have a smooth, reliable experience throughout the real estate process.

1.3 MODULES

1. Login Page

The **Login Page** is the entry point for users of the Real Estate Management System. It allows registered users to securely log in to their accounts using their credentials (username and password). The page includes basic authentication features to ensure secure access and protect user information. Once authenticated, users can proceed to the **Home Page** to access property-related features.

2. Homepage

The Dashboard serves as the central hub of the Real Estate Management System, offering a user-friendly interface for buyers, sellers, renters. The dashboard provides easy navigation.

3.Main Page

After selecting one or more options on the Home Page (Buy, Sell, Rent), users are directed to the Property Page.

4.New Property Listing

The New Property Listing module enables owners to add new properties to the system. They can input important details such as the property's address, type (e.g., house, apartment), price, description, images, and availability (for sale or rent). This module ensures that properties are categorized correctly and stored in the database for future reference. The system validates inputs, including pricing and availability, to ensure consistency and accuracy. This page streamlines the process of listing new properties and enhances data management.

6.View Property Details

This page provides customers with a detailed view of a specific property. Information such as the property's price, features, images, agent or owner details, and contact information are displayed clearly. This module allows users to get a thorough understanding of the property before making further inquiries or expressing interest.

7. Database Connection

The **Database Connection** class handles the connection to the MySQL database. The connection details such as the database URL, username, and password are stored in this class. The `getConnection()` method establishes a connection with the database using **JDBC**. This class ensures that the application can interact with the database to manage data, such as user profiles, property listings, and transactions.

Chapter 2 SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

The Real Estate Management System is built with a clear separation between the frontend and backend for streamlined property management. Java is used to develop the user interface, ensuring an intuitive and responsive experience for property managers and clients. SQL is employed as the backend database management system, securely handling property details, user data, and transactions. This combination ensures efficient data processing, seamless interactions, and high system reliability for all users.

2.2 LANGUAGES

The Real Estate Management System utilizes Java for frontend development and SQL for backend data management, each chosen for their specific roles in enhancing system functionality and performance.

2.2.1 Java

Role: Java is used for the frontend, developing a dynamic and user-friendly interface for both property managers and clients.

Usage: Java drives the graphical user interface (GUI), allowing users to search for properties, view details, communicate with property managers, and manage their profiles. The Java-based frontend ensures a responsive, smooth experience, simplifying navigation and interaction with the system.

Advantages:

- **Cross-Platform Compatibility**: Java's "write once, run anywhere" feature ensures the frontend works across various operating systems without requiring major changes.
- **User-Friendly GUI Components**: Java's libraries like Swing and JavaFX offer an array of customizable components to create a visually appealing and organized interface.
- **Stability and Reliability**: Java is known for its robust performance, ensuring a consistent user experience even as the system scales.

2.2.2 SQL

Role: SQL (Structured Query Language) is used for backend operations, managing

the relational database that stores all property listings, user data, transactions, and inquiries.

Usage: SQL handles data storage, retrieval, and manipulation, ensuring that property details, user information, and transaction records are securely stored and easily accessed. The relational database structure allows efficient querying and updating of data, keeping everything synchronized and accurate.

Advantages:

- **Efficient Data Management**: SQL's advanced query capabilities allow quick retrieval and processing of large datasets, ensuring smooth data handling for real estate listings and transactions.
- **Data Security**: SQL offers strong data security features, protecting sensitive user and property data with robust access control mechanisms.
- **Reliability**: SQL's relational database ensures data integrity and consistency, which is critical for managing property details, user interactions, and transaction records.

Chapter 3 REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

User Authentication and Authorization

The system meets the functional requirement for user authentication by allowing users to register, log in, and access the platform with unique accounts. The login process ensures secure access, preventing unauthorized use of the system. Users can create accounts and securely access the homepage and property listings based on their role, such as a property owner, buyer, or admin.

Property Management for Buy, Sell, and Rent

The system allows users to manage their property listings by providing options to buy, sell, or rent properties. The "Buy" and "Rent" functionalities enable users to view properties with details such as price, size, and location, while "Sell" options allow users to add property listings with all necessary details. This fully satisfies the functional requirement for property management in all three categories—buy, sell, and rent.

Property Details Viewing

The "Buy" and "Rent" pages allow users to view detailed property information. This feature satisfies the need for users to access all property details in a clear and straightforward manner, eliminating the need for a search function but ensuring all information is easily accessible and viewable in a user-friendly format.

3.1.2 Non-Functional Requirements

Security

The system ensures the security of sensitive user information by implementing data encryption during storage and transmission. A secure authentication mechanism is provided to ensure that only authorized users can access the system, protecting user accounts and property details from unauthorized access.

Performance

The system is designed to handle an increasing number of users and property listings as the platform grows, ensuring scalability. It also ensures quick response times for all user actions, such as navigating to the homepage or viewing property details, to deliver a seamless user experience.

Reliability

To ensure reliability, the platform guarantees high availability with minimal downtime, so users can access the system at any time. Regular data backups are performed to safeguard property and user information, allowing quick recovery in case of system failures or data loss.

Usability

The interface is designed to be user-friendly and intuitive, enabling users to log in and navigate effortlessly to the homepage. All property details are displayed clearly, eliminating the need for search functionality, so users can quickly view, add, or manage properties for buying, selling, or renting.

Maintainability

The system adopts a modular architecture, allowing developers to update or expand specific components, such as the buy, sell, or rent sections, without affecting the entire platform. Comprehensive documentation is provided to assist both users and developers, ensuring smooth usage and future updates.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

Laptop: A reliable desktop PC or laptop to host the Real Estate Management System

Processor: Intel® Core™ i7
RAM: Minimum 4 GB RAM to handle concurrent user requests and database operations.

System Architecture: 64-bit operating system, x64-based processor for optimal performance.

Monitor Resolution: 1920 x 1080 monitor resolution for clear display of the system interface.

Input Devices: Keyboard and Mouse for user interaction.

Server: High-processing power server with ample storage capacity to host the live system.

Reliable Network Infrastructure: Stable network connectivity to support remote access and ensure uninterrupted service.

Software Requirements

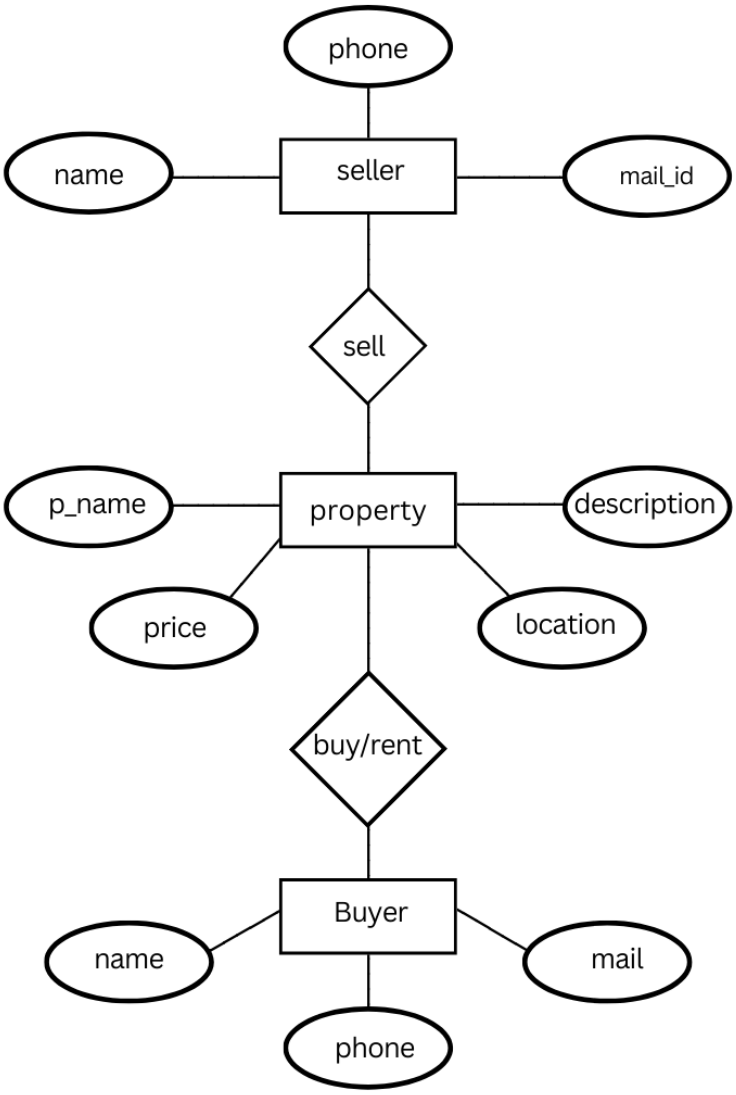
Operating System: Windows 11

Code Editor:Visual Studio Code (VSCode**)**

Front End: JAVA

Back End: MySQL for the SQL

3.3 ER DIAGRAM



Chapter 4 **PROGRAM CODE**

1. LOGIN PAGE

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.RenderingHints;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

public class LoginPage {
    public static void main(String[] args) {
        // Show loading screen for 3 seconds
        LoadingScreen loadingScreen = new LoadingScreen();
        loadingScreen.showLoadingScreen();
```

```
        // Delay showing the login screen until the loading
screen is closed
        SwingUtilities.invokeLater(() -> {
            try {
                Thread.sleep(3000); // Ensure loading screen
shows for 3 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            showLoginScreen();
        });
    }

    private static void showLoginScreen() {
        JFrame frame = new JFrame("Real Estate Management -
Login");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        frame.setLocationRelativeTo(null);

        JPanel gradientPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D) g;
                g2d.setPaint(new GradientPaint(0, 0, new
Color(30, 35, 45), 0, getHeight(), new Color(0, 173, 181)));
                g2d.fillRect(0, 0, getWidth(), getHeight());
            }
        };
        gradientPanel.setLayout(new BoxLayout(gradientPanel,
BoxLayout.Y_AXIS));
```

```
        frame.add(gradientPanel);

        JLabel title = new JLabel("Real Estate Management");
        title.setFont(new Font("SansSerif", Font.BOLD, 36));
        title.setForeground(new Color(0, 173, 181));
        title.setAlignmentX(Component.CENTER_ALIGNMENT);
        gradientPanel.add(title);

        JPanel loginBox = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D) g;

g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2d.setColor(new Color(57, 62, 70));
                g2d.fillRoundRect(0, 0, getWidth(),
getHeight(), 50, 50);
            }
        };
        loginBox.setOpaque(false);
        loginBox.setPreferredSize(new Dimension(350, 220));
        gradientPanel.add(Box.createVerticalStrut(20));
        gradientPanel.add(loginBox);

        loginBox.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setForeground(Color.WHITE);
```

```
        JTextField usernameField = new JTextField(12);
        usernameField.setBackground(new Color(57, 62, 70));
        usernameField.setForeground(Color.WHITE);
        usernameField.setCaretColor(Color.WHITE);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setForeground(Color.WHITE);
        JPasswordField passwordField = new
JPasswordField(12);
        passwordField.setBackground(new Color(57, 62, 70));
        passwordField.setForeground(Color.WHITE);
        passwordField.setCaretColor(Color.WHITE);

        gbc.gridx = 0; gbc.gridy = 0;
        loginBox.add(usernameLabel, gbc);
        gbc.gridx = 1;
        loginBox.add(usernameField, gbc);

        gbc.gridx = 0; gbc.gridy = 1;
        loginBox.add(passwordLabel, gbc);
        gbc.gridx = 1;
        loginBox.add(passwordField, gbc);

        JButton loginButton = new JButton("Login");
        loginButton.setBackground(new Color(0, 173, 181));
        loginButton.setForeground(Color.WHITE);
        loginButton.setPreferredSize(new Dimension(120, 30));
        loginButton.addActionListener(e -> {
            frame.dispose();
            // Placeholder for home page or next step
            System.out.println("Logged in");
        });
```

```
        gbc.gridx = 1; gbc.gridy = 2;
        loginBox.add(loginButton, gbc);

        frame.setVisible(true);
    }
}
```

2. HOME PAGE

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.util.ArrayList;
import java.util.List;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JToggleButton;

public class HomePage {
    private ArrayList<JToggleButton> toggleButtons;
    public HomePage() {
        JFrame frame = new JFrame("Real Estate Management -
Welcome");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 500);
        frame.setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel() {
            @Override
```

```
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D) g;
                g2d.setPaint(new GradientPaint(0, 0, new
Color(30, 35, 45), 0, getHeight(), new Color(0, 173, 181)));
                g2d.fillRect(0, 0, getWidth(), getHeight());
            }
        };
        mainPanel.setLayout(new BoxLayout(mainPanel,
BoxLayout.Y_AXIS));
        frame.add(mainPanel);

        JLabel welcomeLabel = new JLabel("Real Estate
Management");
        welcomeLabel.setFont(new Font("SansSerif", Font.BOLD,
30));
        welcomeLabel.setForeground(new Color(0, 173, 181));

welcomeLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        mainPanel.add(Box.createVerticalStrut(20));
        mainPanel.add(welcomeLabel);

        JLabel promptLabel = new JLabel("How can we help
you?");
        promptLabel.setFont(new Font("SansSerif", Font.PLAIN,
16));
        promptLabel.setForeground(Color.WHITE);

promptLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        mainPanel.add(promptLabel);

        JLabel instructionLabel = new JLabel("Choose a few
```

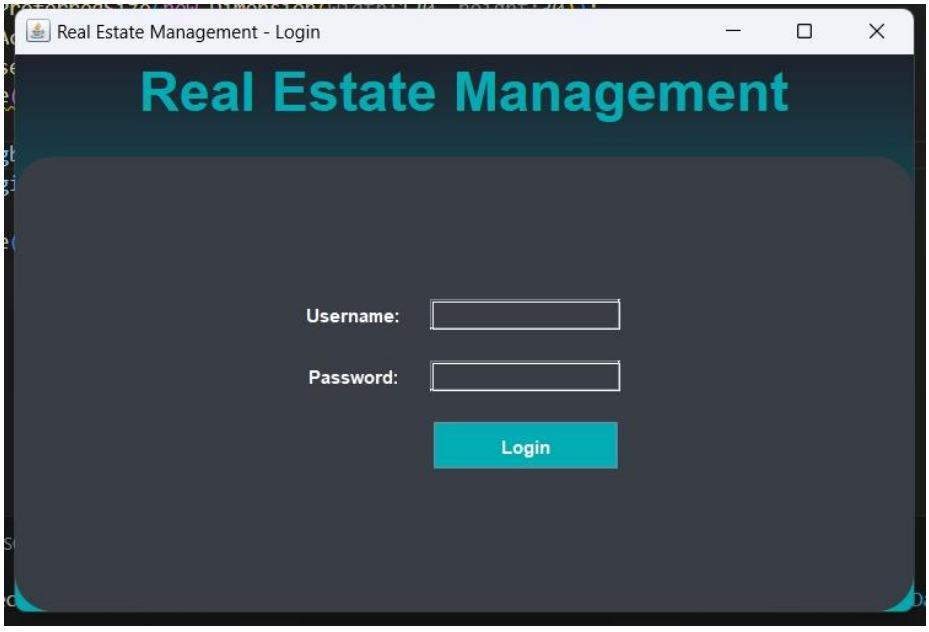
```
options if you need to:");
        instructionLabel.setFont(new Font("SansSerif",
Font.PLAIN, 14));
        instructionLabel.setForeground(Color.WHITE);

instructionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        mainPanel.add(instructionLabel);

        JPanel optionsPanel = new JPanel();
        optionsPanel.setOpaque(false);
        optionsPanel.setLayout(new BoxLayout(optionsPanel,
BoxLayout.Y_AXIS));
        mainPanel.add(optionsPanel);

        String[] options = {"Buy", "Sell", "Rent"};
        toggleButtons = new ArrayList<>();

        for (String option : options) {
            JToggleButton toggleButton = new
JToggleButton(option);
            toggleButton.setMaximumSize(new Dimension(200,
40));

toggleButton.setAlignmentX(Component.CENTER_ALIGNMENT);
            toggleButton.setBackground(new Color(57, 62,
70));
            toggleButton.setForeground(Color.WHITE);
            toggleButton.setFont(new Font("SansSerif",
Font.PLAIN, 16));
            toggleButton.setFocusPainted(false);
            toggleButtons.add(toggleButton);
            optionsPanel.add(Box.createVerticalStrut(10));
```

```
            optionsPanel.add(toggleButton);
        }

        JButton nextButton = new JButton("Next");
        nextButton.setPreferredSize(new Dimension(100, 30));
        nextButton.setBackground(new Color(0, 173, 181));
        nextButton.setForeground(Color.WHITE);
        nextButton.setFont(new Font("SansSerif", Font.BOLD,
14));
        nextButton.setFocusPainted(false);
        nextButton.setAlignmentX(Component.CENTER_ALIGNMENT);
        mainPanel.add(Box.createVerticalStrut(20));
        mainPanel.add(nextButton);

        nextButton.addActionListener(e -> {
            List<String> selectedOptions = new ArrayList<>();
            for (JToggleButton toggleButton : toggleButtons)
{
                if (toggleButton.isSelected()) {

selectedOptions.add(toggleButton.getText());
                }
            }
            if (selectedOptions.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "Please
select at least one option.");
            } else {
                frame.dispose();
                new PropertyPage(selectedOptions);
            }
        });
```

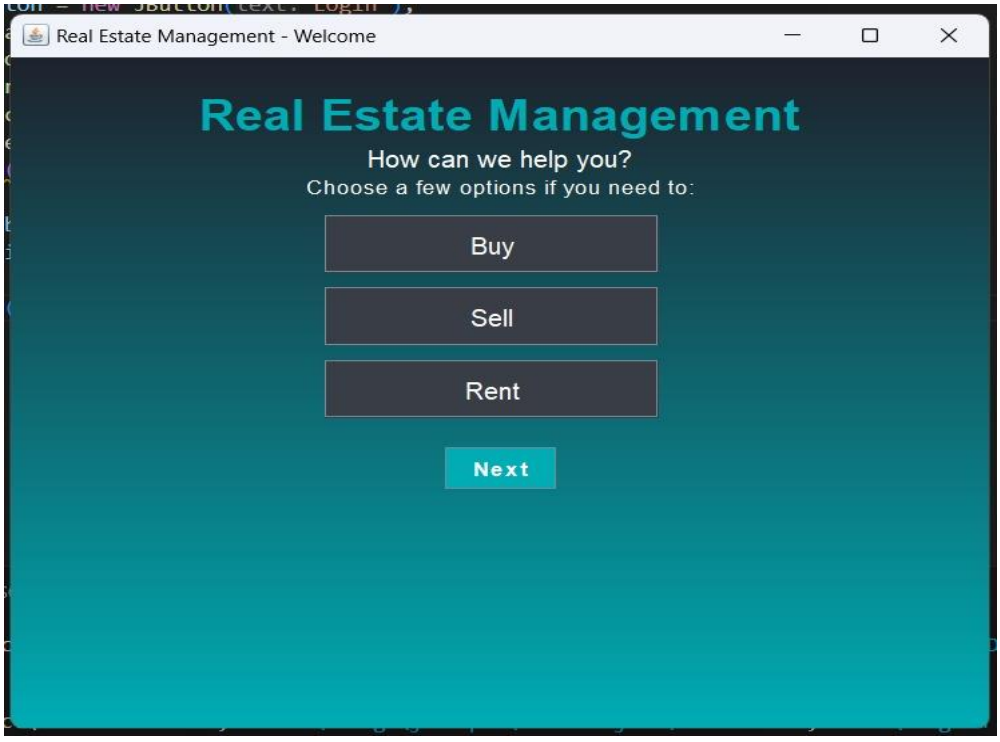
```
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        new HomePage();
    }
}
```

Chapter 5 RESULTS AND DISCUSSION

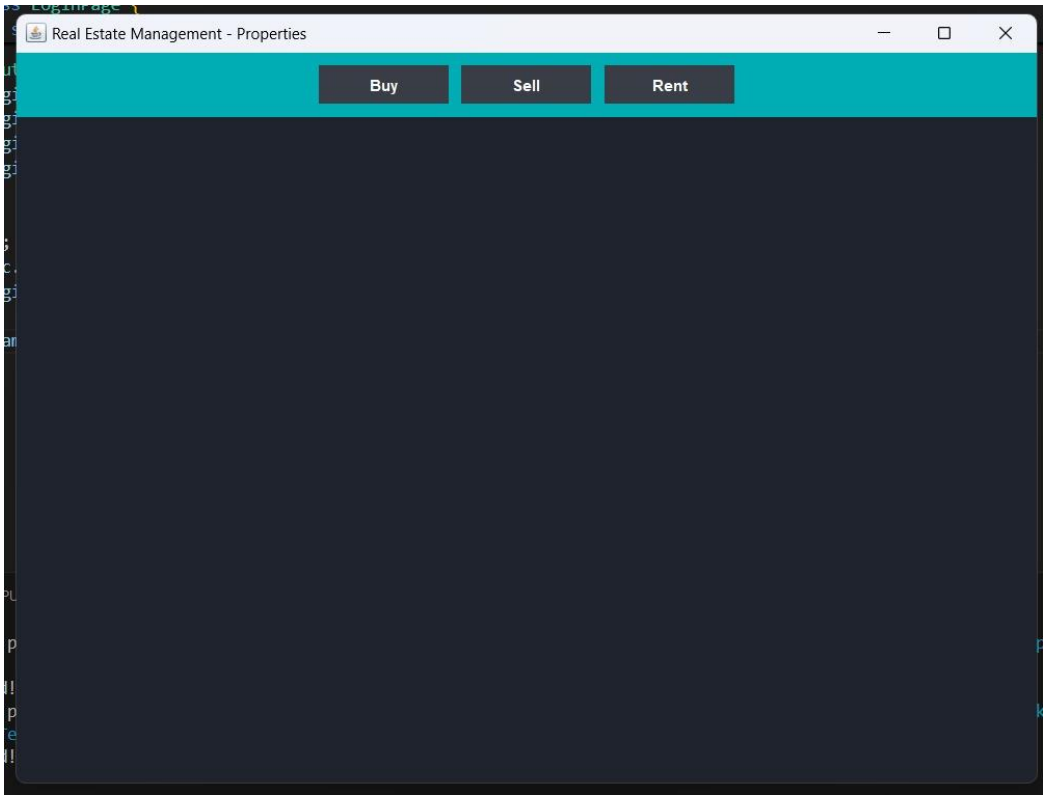
LOGIN PAGE



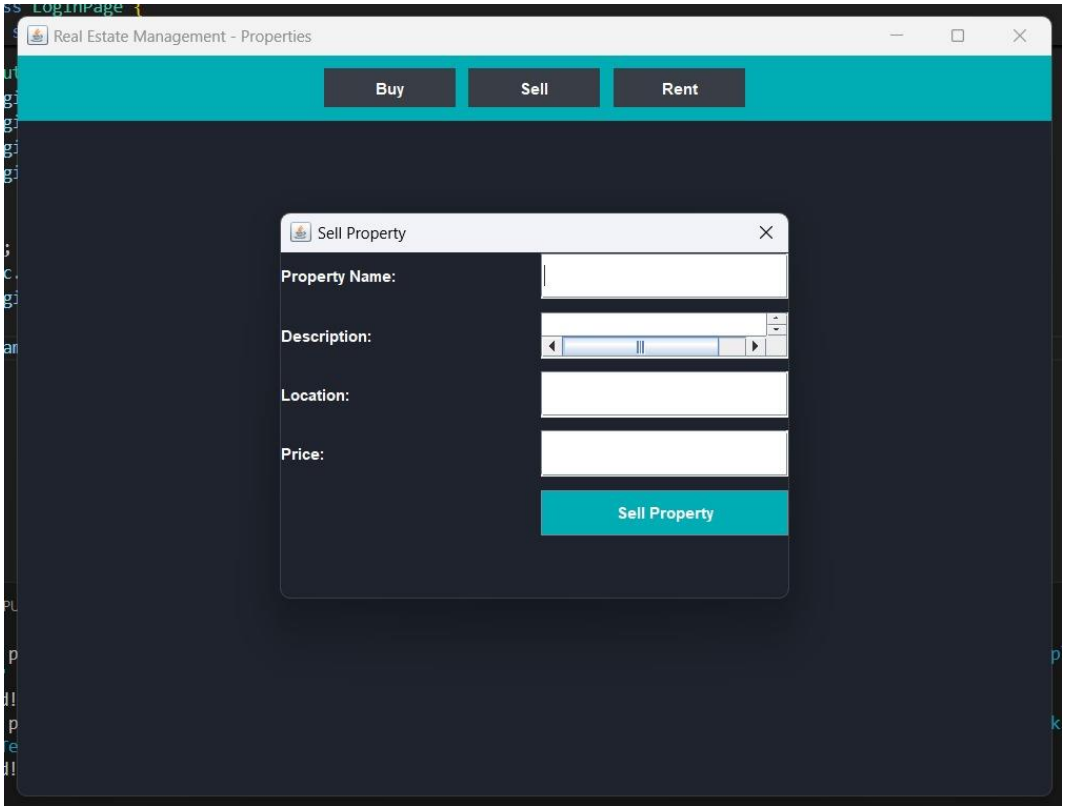
HOME PAGE



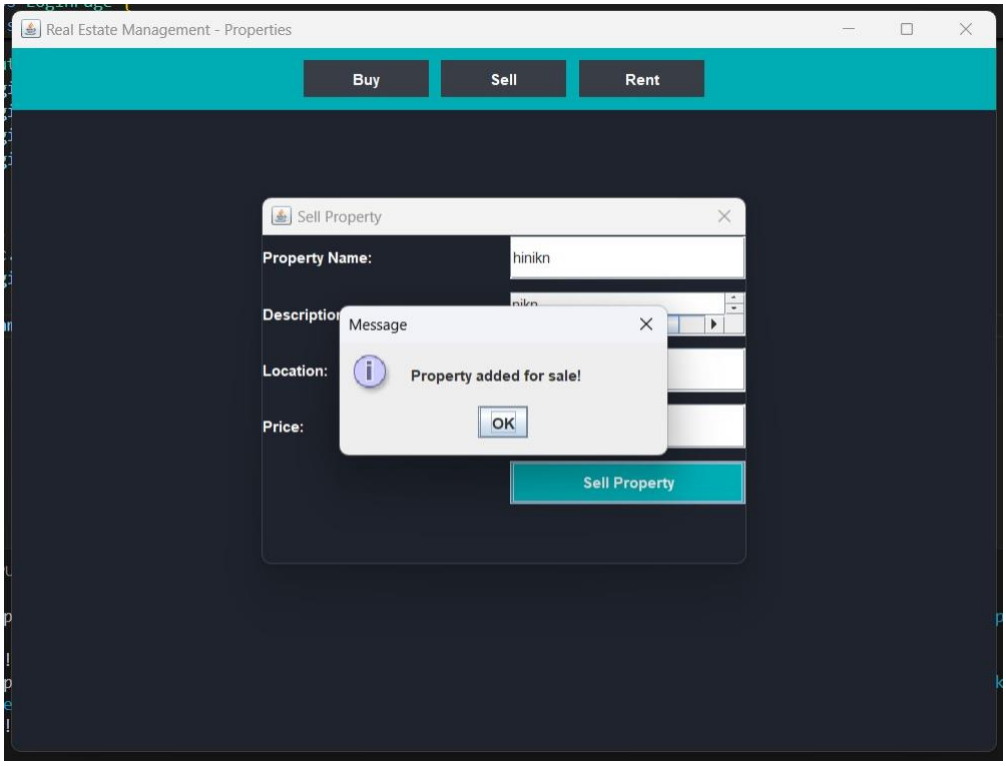
MAIN PAGE



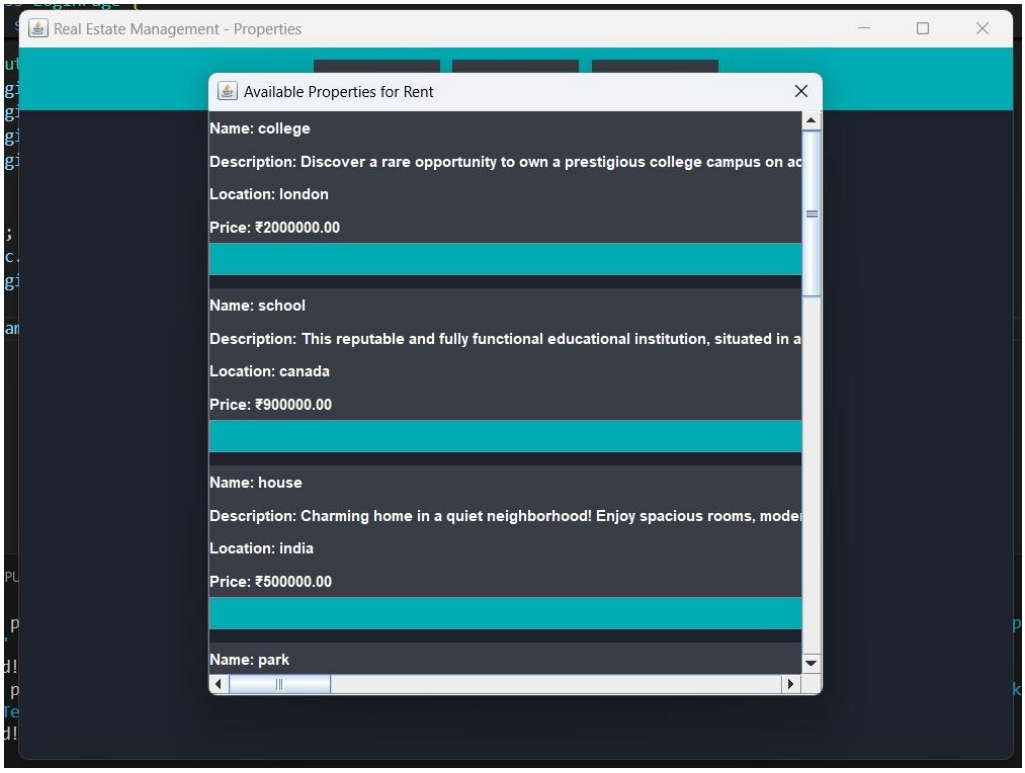
ADD NEW PROPERTY

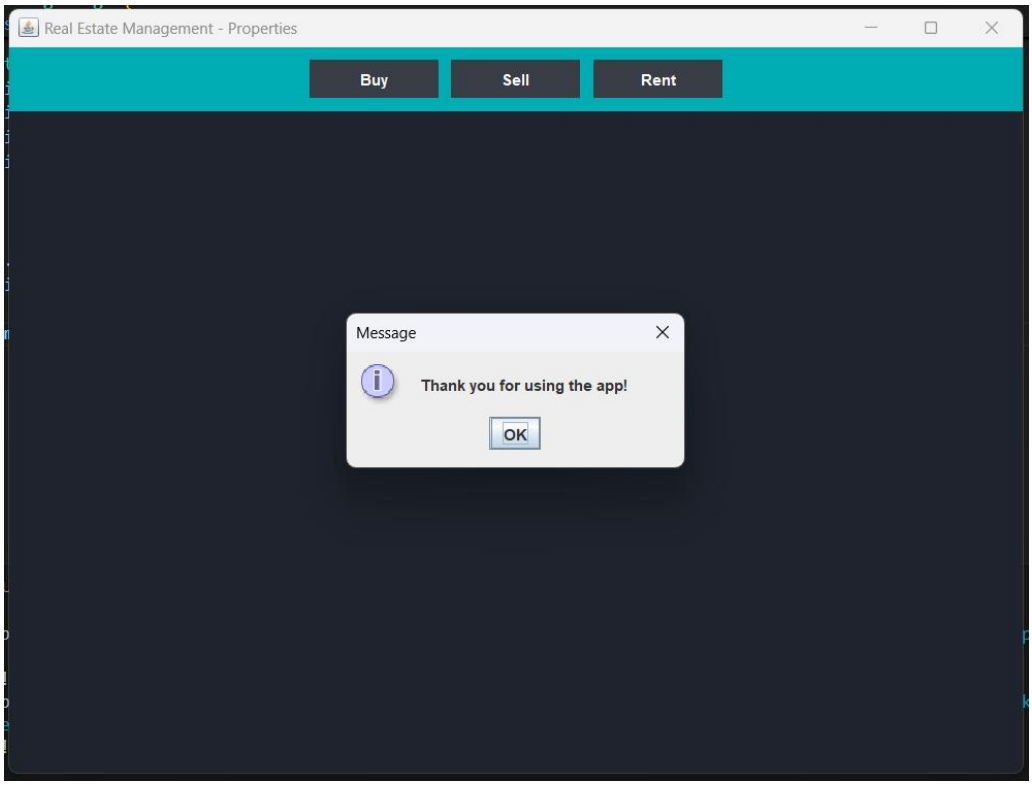


PROPERTY ADDED



PROPERTY DETAILS VIEW




Real Estate Management - Properties
Buy
Sell
Rent
Message
Thank you for using the app!
OK

Chapter 6 CONCLUSION

6.1 Conclusion

In conclusion, our Real Estate Management System serves as an efficient platform for managing property-related data and interactions among buy,sell,rent. With its intuitive interface and streamlined processes, the system enhances transparency, simplifies property management, and ensures accurate record-keeping.

By leveraging SQL for data handling and Java for frontend functionality, the system provides a reliable and scalable foundation for real estate operations. While the current implementation focuses on core features, future improvements could include adding advanced analytics,

Overall, this project lays a solid groundwork for a comprehensive and modern approach to real estate management, benefiting stakeholders with improved efficiency, accessibility, and reliability.

Chapter 7 REFERENCES

7.1 REFERENCES

1) https://www.geeksforgeeks.org/introduction-to-java-swing/

2) https://www.geeksforgeeks.org/how-to-connect-to-mysql-server-using-vs-code-and-fix-errors/

3) https://www.geeksforgeeks.org/introduction-to-jdbc/