RAJALAKSHMI
ENGINEERING COLLEGE

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

Laboratory Record Note Book

NAME .....

BRANCH .....

UNIVERSITY REGISTER No. .....

COLLEGE ROLL No. .....

SEMESTER .....

ACADEMIC YEAR .....

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BONAFIDE CERTIFICATE

NAME .....

ACADEMIC YEAR ..... SEMESTER .....BRANCH.....

UNIVERSITY REGISTER No. [ ]

Certified that this is the bonafide record of work done by the above student in the

..... *Laboratory during the year* 20 - 20

Signature of Faculty - in - Charge

Submitted for the Practical Examination held on .....

Internal Examiner **External Examiner**

INDEX

Name : ______________________ **Branch :** __________ **Sec :** ________ **Roll No :** ____________

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks

EX NO:1	WRITE THE COMPLETE PROBLEM STATEMENT
DATE	

AIM:

To prepare PROBLEM STATEMENT for any project.

ALGORITHM:

1. The problem statement is the initial starting point for a project.
2. A problem statement describes what needs to be done without describing how.
3. It is basically a one-to-three-page statement that everyone on theproject agrees with that describes what will be done at a high level.
4. The problem statement is intended for a broad audience and shouldbe written in non- technical terms.
5. It helps the non-technical and technical personnel communicate byproviding a description of a problem.
6. It doesn't describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by customer.
2. It may give an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements.
4. Here, requirements are identified with the help of customer and existing system processes.

Problem:

Real estate management is becoming increasingly complex as property owners, tenants, and real estate agents face challenges in tracking property details, managing rental agreements, handling payments, and maintaining communication. Currently, many real estate businesses rely on manual processes or outdated systems that result in inefficiencies, errors, and poor customer experiences. The management of properties, including leasing, maintenance requests, and financial tracking, requires a more streamlined and automated approach to save time, reduce mistakes, and enhance overall business operations.

For instance, property managers often face issues related to tenant communication, timely maintenance, and accurate financial reporting. They may also struggle with updating lease agreements, tracking rent payments, and managing property availability in real-time. This leads to operational delays, frustrated tenants, and missed revenue opportunities. With the growing number of properties and tenants to manage, these issues are only expected to worsen unless a comprehensive solution is implemented.

Background:

The real estate industry has seen rapid growth, leading to an increasing demand for more efficient ways to manage properties. However, many businesses are still using manual systems, spreadsheets, or

outdated software to manage property listings, tenant information, and financial records. These systems lack integration and automation, creating numerous challenges for property managers, agents, and tenants alike.

A central challenge for real estate management is the need to handle multiple tasks—such as leasing, payment processing, and property maintenance—across diverse properties. Furthermore, clients (both property owners and tenants) expect transparent communication and quick resolutions for their issues. An effective real estate management system should streamline these processes, offering an intuitive interface for users and automated functions for routine tasks.

Relevance:

A comprehensive and automated real estate management system is crucial for improving operational efficiency, reducing administrative overhead, and enhancing customer satisfaction. By addressing these issues, businesses can improve tenant retention, reduce manual errors, streamline property management workflows, and improve overall profitability. For property owners and managers, it's essential to have a unified platform that integrates leasing, maintenance, communication, and financial tracking.

For tenants, a modern management system can provide better visibility into their rental agreements, make rent payments easier, and facilitate quicker responses to maintenance requests. The end goal is to foster a more organized and efficient real estate operation, benefiting all stakeholders—property owners, managers, tenants, and real estate agents.

Objectives:

The primary objective of this project is to develop a Real Estate Management System that streamlines and automates property management tasks, improving the efficiency and effectiveness of real estate operations. The specific objectives include:

1. Property Listing and Management : Create a user-friendly platform for property owners and real estate agents to list, update, and track properties in real-time, including available units, rental prices, and status updates.

2. Tenant and Lease Management:Automate the process of tracking tenants, lease agreements, rent due dates, and payment history. Provide tenants with easy access to their rental agreements and payment records.

3. Maintenance Request Tracking : Implement a system for tenants to submit maintenance requests and track their status, allowing property managers to efficiently assign, schedule, and resolve maintenance tasks.

4. Financial Management and Reporting :Automate rent collection, payment reminders, and generate detailed financial reports for property owners and managers, including revenue tracking, expenses, and profitability analysis.

5. Communication Channels : Develop integrated communication tools for tenants and property managers to ensure timely and effective interaction regarding property issues, lease renewals, and maintenance needs.

6. Real-Time Availability Updates : Integrate a real-time property availability feature to keep potential tenants informed of open units and rental prices, minimizing vacant periods.

7. User Access Control and Security : Provide secure user access for property managers, tenants, and property owners, ensuring sensitive data (lease agreements, financial transactions, etc.) is protected.

8. Mobile Compatibility :Ensure the system is accessible through both web and mobile platforms, allowing stakeholders to manage properties and communicate on-the-go.

9. Scalability and Flexibility : Design the system to be scalable, accommodating a growing portfolio of properties and users as businesses expand.

10. Ongoing Support and Updates : Provide ongoing updates, user support, and training to ensure the system remains functional, secure, and aligned with evolving business needs.

By addressing these objectives, the Real Estate Management System will improve operational efficiency, reduce administrative workload, and enhance tenant and property owner satisfaction, ultimately leading to a more profitable and sustainable real estate business.

Result:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for any Project.

ALGORITHM:

SRS shall address are the following:

a) **Functionality.** What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system‘s hardware, other hardware, and other software?

c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What is the portability, correctness, maintainability, security, etc. considerations?

e) **Design constraints imposed on an implementation.** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

1. Introduction

1.1 PURPOSE

The purpose of this document is to define the requirements for an online Real Estate Management System that will simplify the management of real estate properties, clients, transactions, and related processes. This system aims to streamline property management for agents, property owners, and prospective tenants or buyers by providing a digital platform to manage property listings, client information, bookings, and financial transactions.

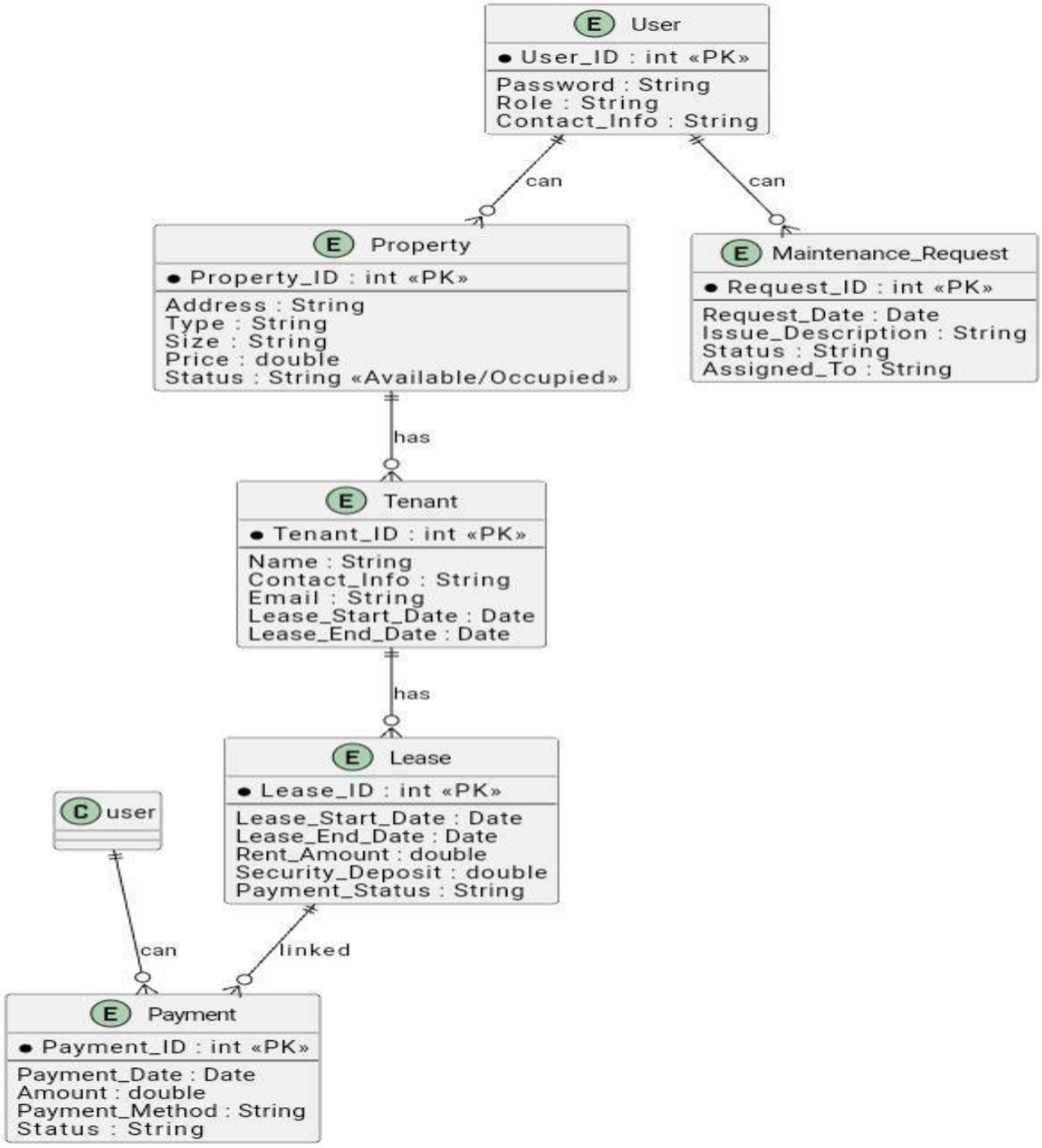
1.2 DOCUMENT CONVENTIONS

This document uses the following conventions:

- DB: Database

- ER: Entity Relationship

Entity relationship diagram :



1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This document is intended for project managers, software developers, real estate agencies, and system testers involved in the project. The Real Estate Management System will be useful for property managers, real estate agents, tenants, and buyers. The document provides a clear outline of system features, user requirements, and technical specifications needed for development.

1.4 PROJECT SCOPE

The Real Estate Management System aims to facilitate the management of property listings, customer relationships, and transaction records within a centralized digital platform. By automating key processes, the system is expected to improve efficiency and provide a seamless user experience. This platform will allow agents to manage properties, customers to view listings and book visits, and landlords to track their properties' performance and occupancy rates. The system will support a variety of property types, including residential, commercial, and rental properties.

1.5 REFERENCES

- "Database Management Systems" by Raghu Ramakrishnan

- Real estate management articles from [RealEstateTech.com](https://realestatetech.com)

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The Real Estate Management System will act as a central repository of property, client, and transaction data. It will provide an interface for property managers, agents, and potential buyers or tenants to access relevant data. Key components include:

Property Listings: Includes property details, pricing, availability, and location information.

Client Information: Stores client data such as name, contact information, preferences, and transaction history.

Transaction Management: Records rental payments, property sales, bookings, and appointment details.

2.2 PRODUCT FEATURES

The main features of the Real Estate Management System are illustrated in the below Entity Relationship diagram :

2.3 USER CLASS AND CHARACTERISTICS

The Real Estate Management System will cater to various user classes:

Admin: Manages user roles, property listings, and system settings.

Agents: Manages property listings, interacts with clients, and schedules property visits.

Property Owners: Views property performance and financial reports.

Customers: Searches properties, views listings, and books property visits.

2.4 OPERATING ENVIRONMENT

The Real Estate Management System will operate in a web environment and should support mobile and desktop devices. The operating environment includes:

- Web server: Apache or Nginx

- Database: MySQL or PostgreSQL

- Platform: PHP/JavaScript (React or Angular for frontend)

2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

- Must ensure data security for client information and property details.

- Real-time data updates for property availability and transaction status.

- Compliance with industry regulations for data protection and privacy.

2.6 ASSUMPTION DEPENDENCIES

- Users have internet access to view listings and book appointments.

- The system is designed to handle a high volume of users and transactions.

- Data backups are performed regularly to prevent data loss.

3. SYSTEM FEATURES

DESCRIPTION AND PRIORITY

The Real Estate Management System will prioritize property listing management, client relationship management, and transaction handling. It is a high-priority project, as it will streamline property management and improve customer experience in the real estate industry.

STIMULUS/RESPONSE SEQUENCES

- Search for properties by location, price range, and other filters.

- View detailed property listings, photos, and availability.

- Book property visits online.

FUNCTIONAL REQUIREMENTS

The system will have the following key features:

1. Property Management:

- Add/edit property listings.
- Upload property images and documents.
- View and filter properties by type, location, price, and availability.

2. Client Management:
- Add/edit client information.
- Track inquiries and schedule appointments.

3. Transaction Management:
- Record payments and generate receipts.
- Maintain financial records and reports.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

Frontend: Developed using HTML, CSS, and JavaScript frameworks (e.g., React or Angular).

Backend: PHP or Node.js for server-side logic.

4.2 HARDWARE INTERFACES

Server: Requires a server with sufficient storage and processing capacity to handle multiple simultaneous users.

Client Device: Any device with internet connectivity and a modern web browser.

4.3 SOFTWARE INTERFACES

The system will interact with external systems and APIs for:

- Payment processing (Stripe, PayPal)
- Location services (Google Maps API)

4.4 COMMUNICATION INTERFACES

The system will support HTTP/HTTPS protocols for secure data transmission.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The system must support concurrent users and handle transactions without delay. Server response time should be under 2 seconds for any user action.

5.2 SAFETY REQUIREMENTS

Regular data backups and disaster recovery plans will be in place. The system should automatically log out inactive users after a set period for security.

5.3 SECURITY REQUIREMENTS

User authentication will include role-based access control. Sensitive data, including client and financial records, will be encrypted.

5.4 SOFTWARE QUALITY ATTRIBUTES

Availability: The system should be available 24/7.

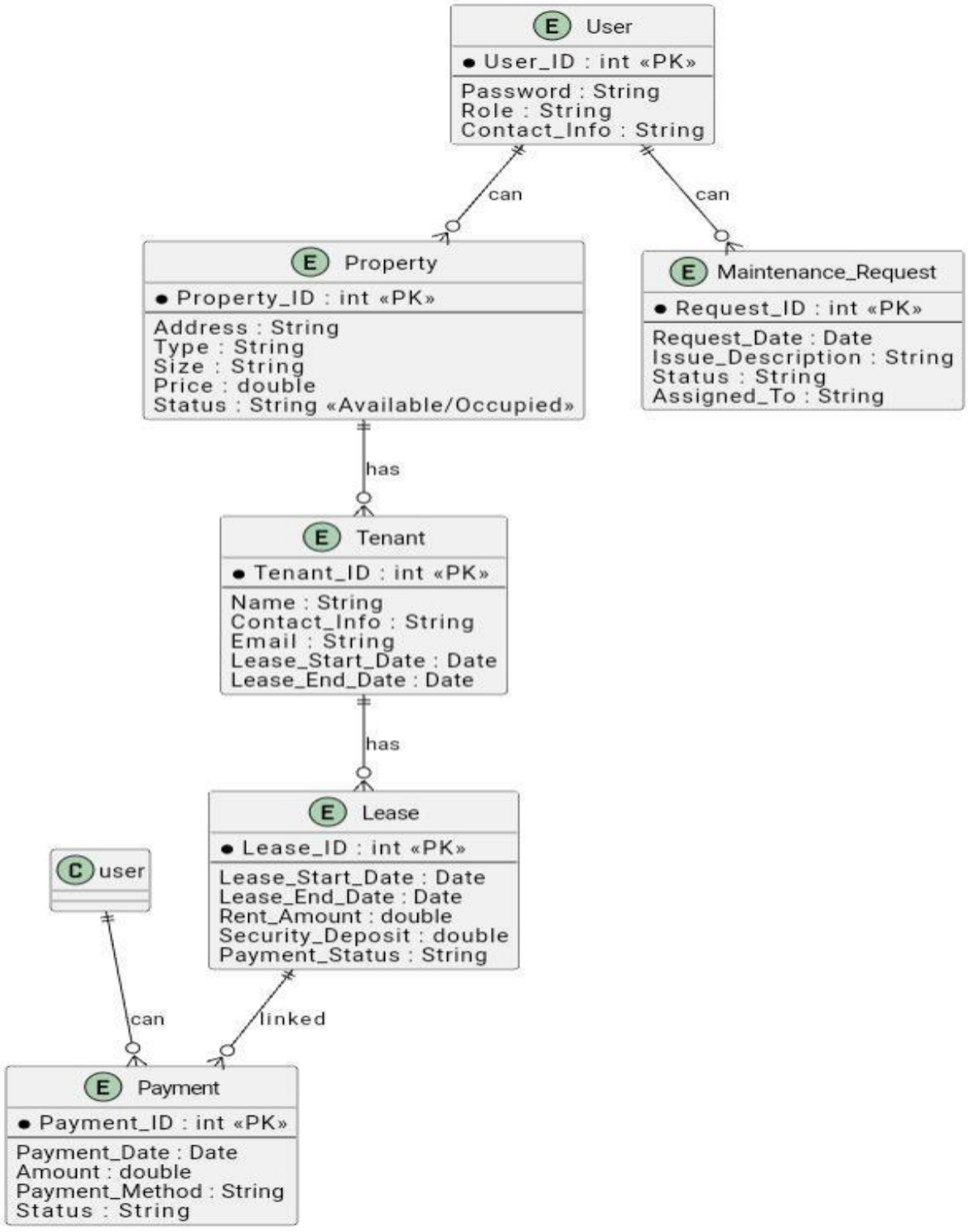
Usability: Designed for ease of use, with an intuitive interface accessible from both desktop and mobile devices.

Maintainability: The system should support easy updates for bug fixes, feature enhancements, and security patches.

Scalability: The system must be scalable to accommodate growth in the number of users and transactions.

Result:

Diagram :


User
User_ID : int «PK»
Password : String
Role : String
Contact_Info : String
can
can
Property
Property_ID : int «PK»
Address : String
Type : String
Size : String
Price : double
Status : String «Available/Occupied»
Maintenance_Request
Request_ID : int «PK»
Request_Date : Date
Issue_Description : String
Status : String
Assigned_To : String
has
Tenant
Tenant_ID : int «PK»
Name : String
Contact_Info : String
Email : String
Lease_Start_Date : Date
Lease_End_Date : Date
has
Lease
Lease_ID : int «PK»
Lease_Start_Date : Date
Lease_End_Date : Date
Rent_Amount : double
Security_Deposit : double
Payment_Status : String
user
can
linked
Payment
Payment_ID : int «PK»
Payment_Date : Date
Amount : double
Payment_Method : String
Status : String

EX NO:3	DRAW THE ENTITY RELATIONSHIP DIAGRAM
DATE	

AIM:

To Draw the Entity Relationship Diagram for any project.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

Entity Relationship Matrix

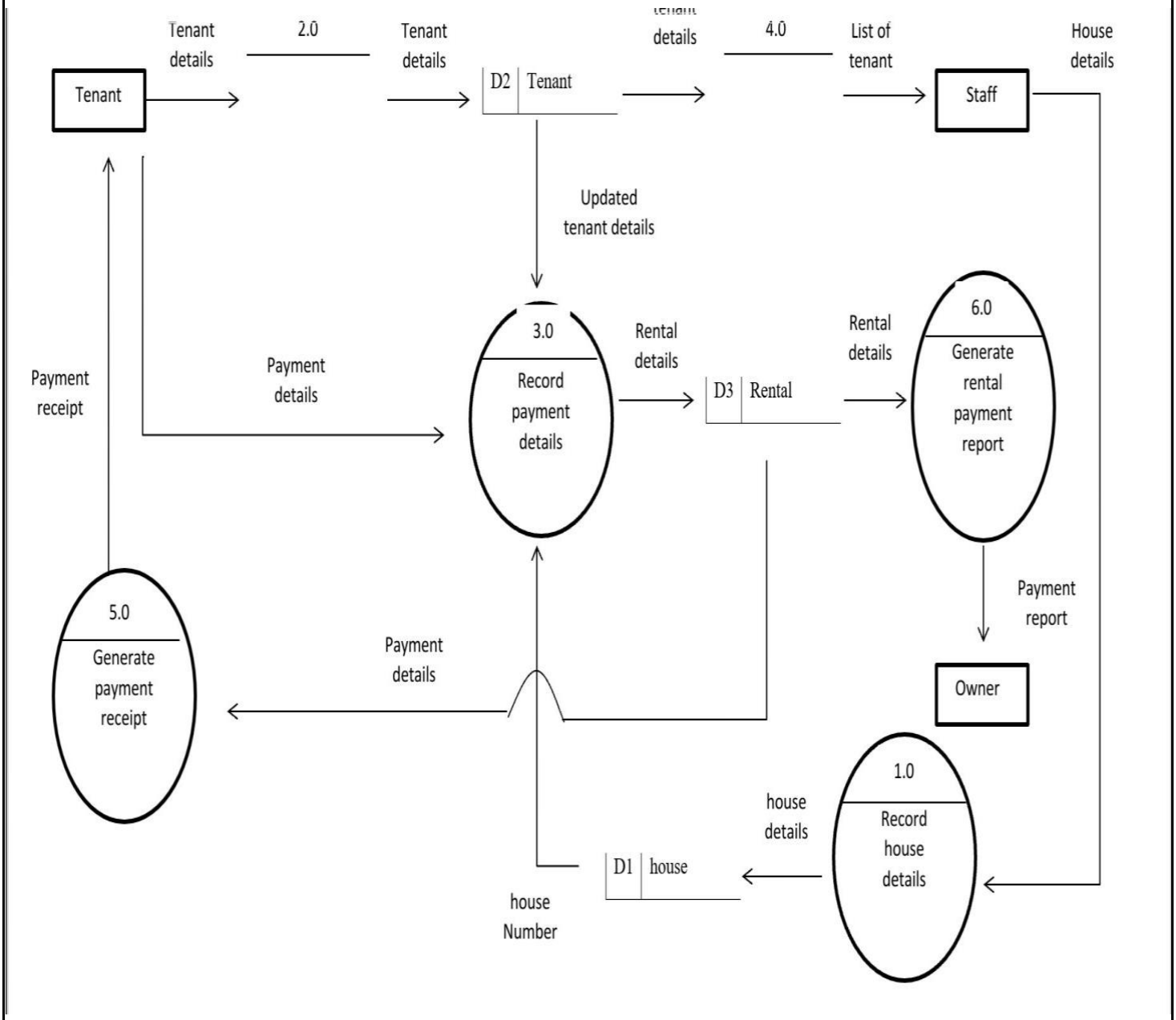
Primary Keys

Attributes

Mapping of Attributes with Entities

Result:

Diagram :


Tenant
Tenant details
2.0
Tenant details
D2 Tenant
tenant details
4.0
List of tenant
Staff
House details
Updated tenant details
3.0
Record payment details
Payment details
Payment receipt
Rental details
D3 Rental
Rental details
6.0
Generate rental payment report
Payment report
Owner
5.0
Generate payment receipt
Payment details
1.0
Record house details
house details
D1 house
house Number

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE	

AIM:

To Draw the Data Flow Diagram for any project and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

INPUT:

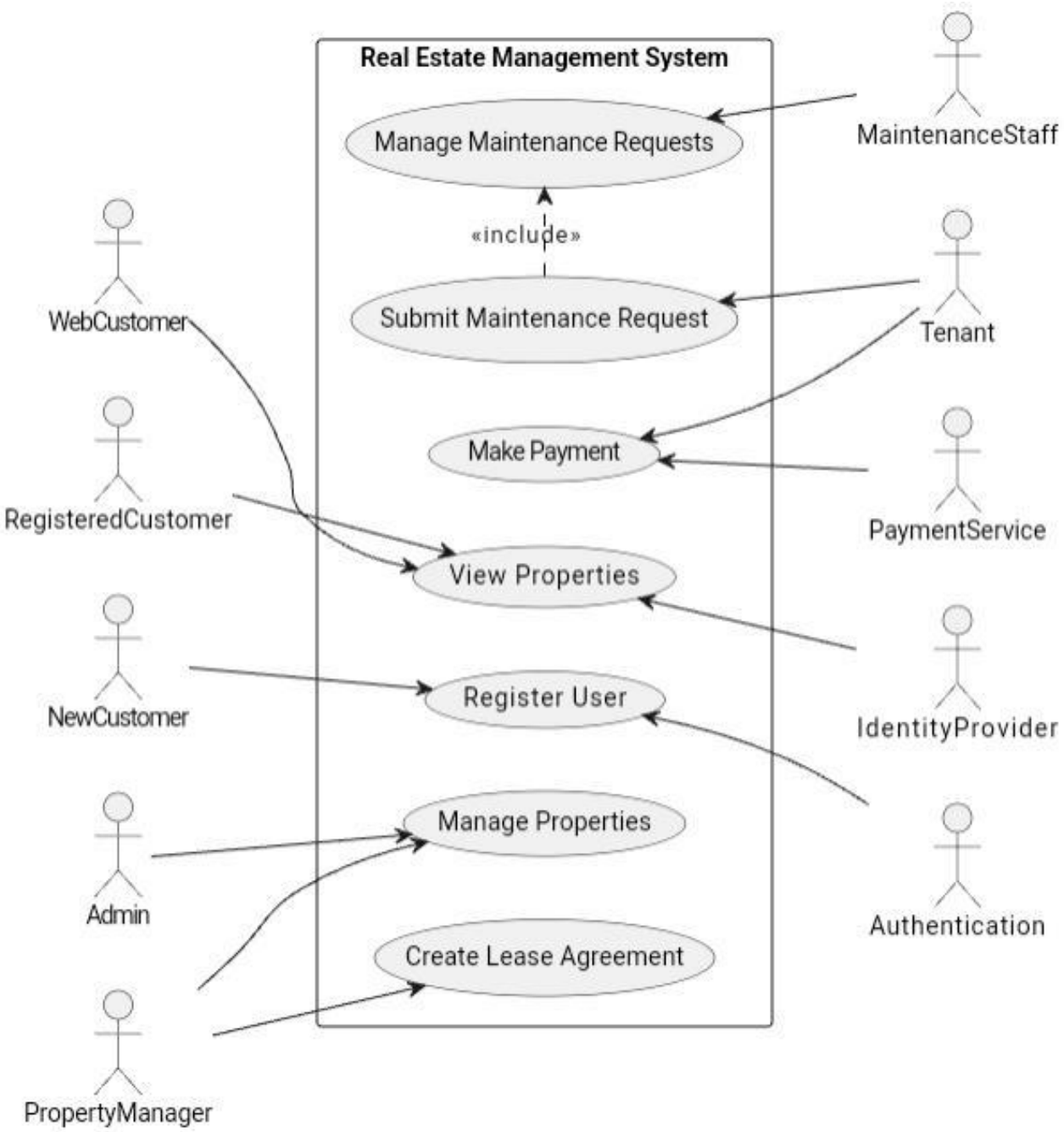
Processes

Datastores

External Entities

Result:

Diagram:


Real Estate Management System
Manage Maintenance Requests
«include»
Submit Maintenance Request
Make Payment
View Properties
Register User
Manage Properties
Create Lease Agreement
MaintenanceStaff
Tenant
PaymentService
IdentityProvider
Authentication
WebCustomer
RegisteredCustomer
NewCustomer
Admin
PropertyManager

EX NO:5	DRAW USE CASE DIAGRAM
DATE	

AIM:

To Draw the Use Case Diagram for any project

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

INPUTS:

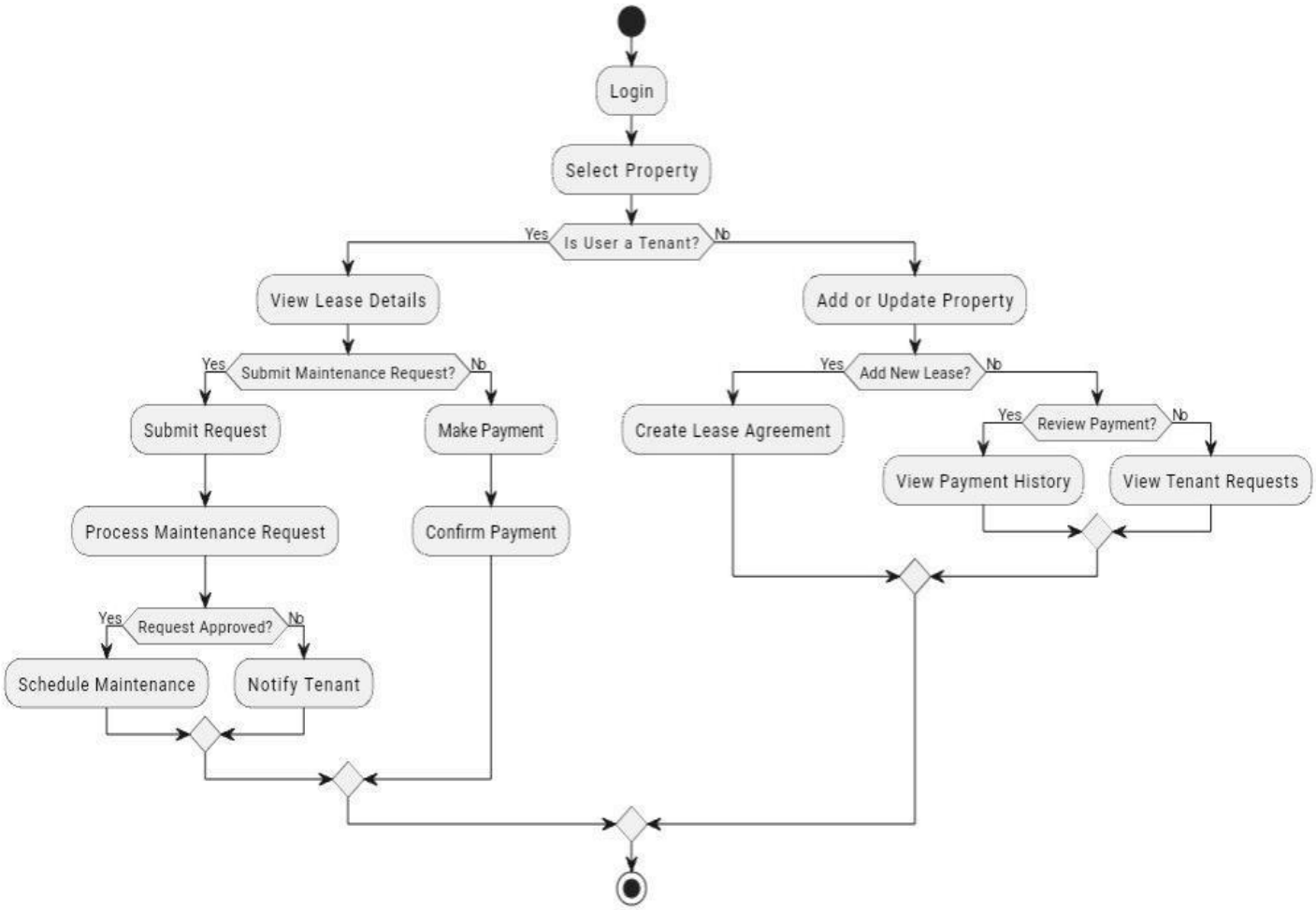
Actors

Use Cases

Relations

Result:

Diagram:


Login
Select Property
Is User a Tenant?
Yes
No
View Lease Details
Add or Update Property
Submit Maintenance Request?
Yes
No
Add New Lease?
Yes
No
Submit Request
Make Payment
Create Lease Agreement
Review Payment?
Yes
No
View Payment History
View Tenant Requests
Process Maintenance Request
Confirm Payment
Request Approved?
Yes
No
Schedule Maintenance
Notify Tenant

EX NO:6	**DRAW ACTIVITY DIAGRAM OF ALL USE CASES.**
DATE	

AIM:

To Draw the activity Diagram for any project

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

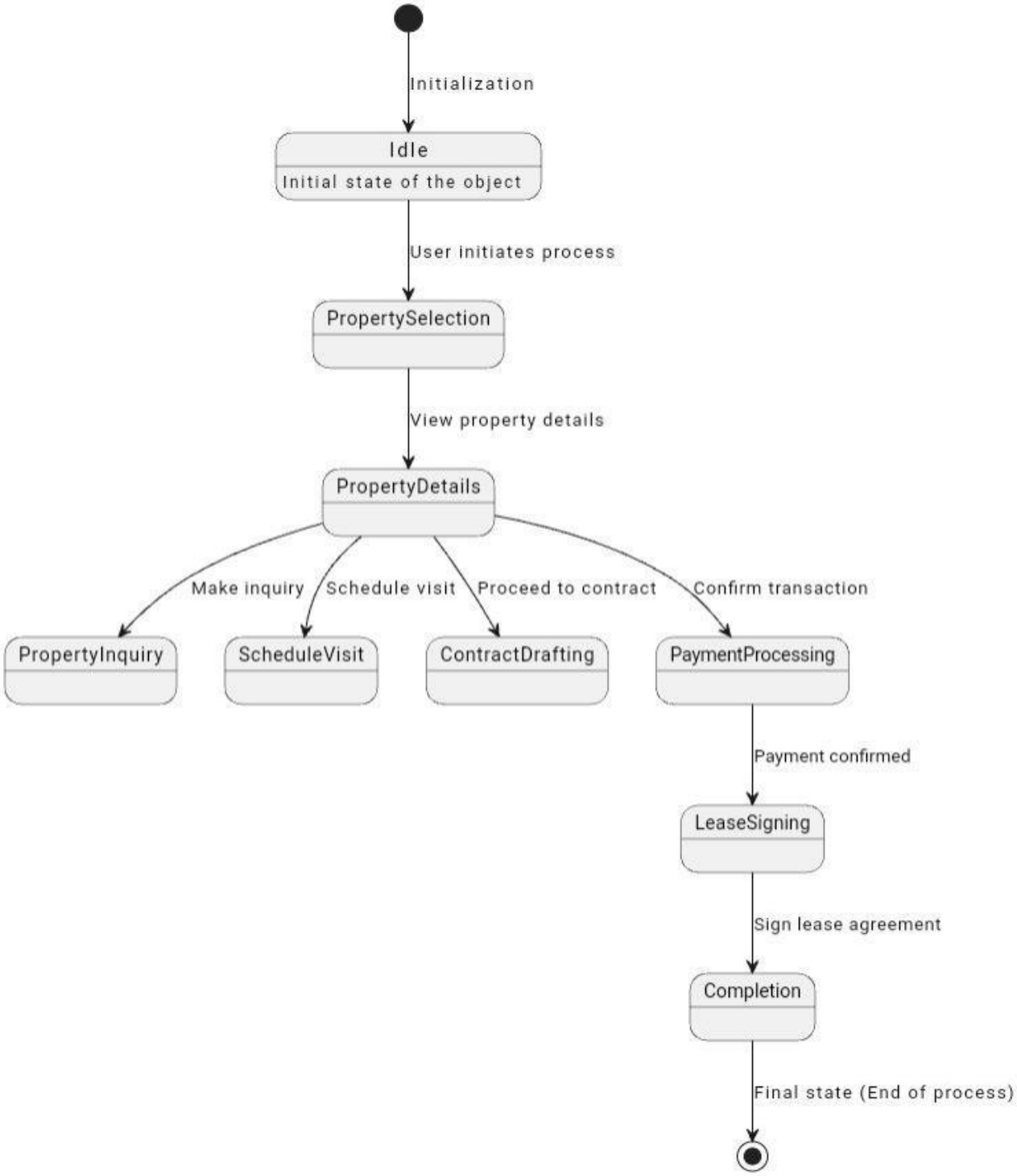
Guards

Parallel Activities

Conditions

Result:

Diagram :


Initialization
Idle
Initial state of the object
User initiates process
PropertySelection
View property details
PropertyDetails
Make inquiry
Schedule visit
Proceed to contract
Confirm transaction
PropertyInquiry
ScheduleVisit
ContractDrafting
PaymentProcessing
Payment confirmed
LeaseSigning
Sign lease agreement
Completion
Final state (End of process)

EX NO:7	DRAW STATE CHART DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the State Chart Diagram for any project

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

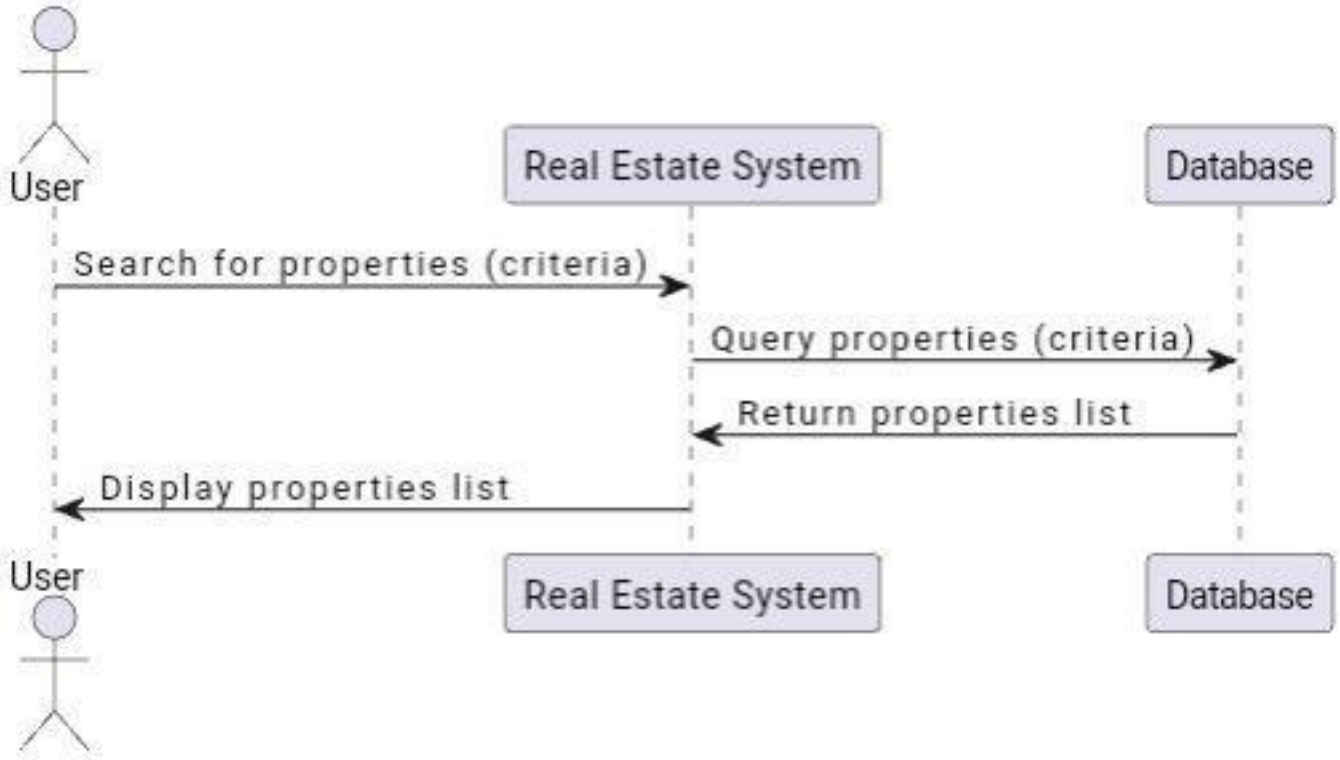
Objects

States

Events

Result:

Diagram :


User
Real Estate System
Database
Search for properties (criteria)
Query properties (criteria)
Return properties list
Display properties list
User
Real Estate System
Database

EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the Sequence Diagram for any project

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

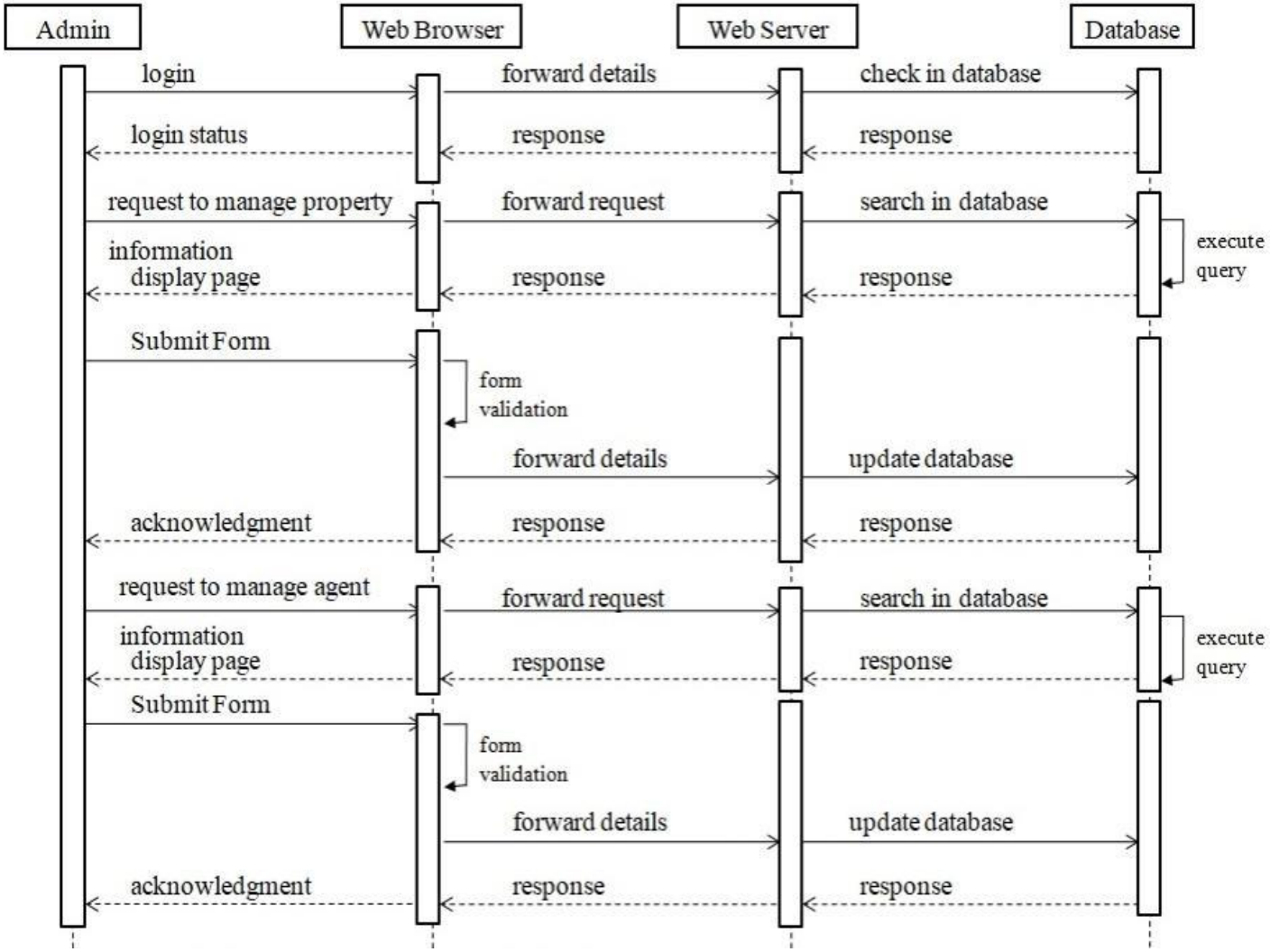
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Diagram:


Admin
Web Browser
Web Server
Database
login
forward details
check in database
login status
response
response
request to manage property
forward request
search in database
execute query
information display page
response
response
Submit Form
form validation
forward details
update database
acknowledgment
response
response
request to manage agent
forward request
search in database
execute query
information display page
response
response
Submit Form
form validation
forward details
update database
acknowledgment
response
response

EX NO:9	DRAW COLLABORATION DIAGRAM OF ALL USE CASES
DATE	

AIM:

To Draw the Collaboration Diagram for any project

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

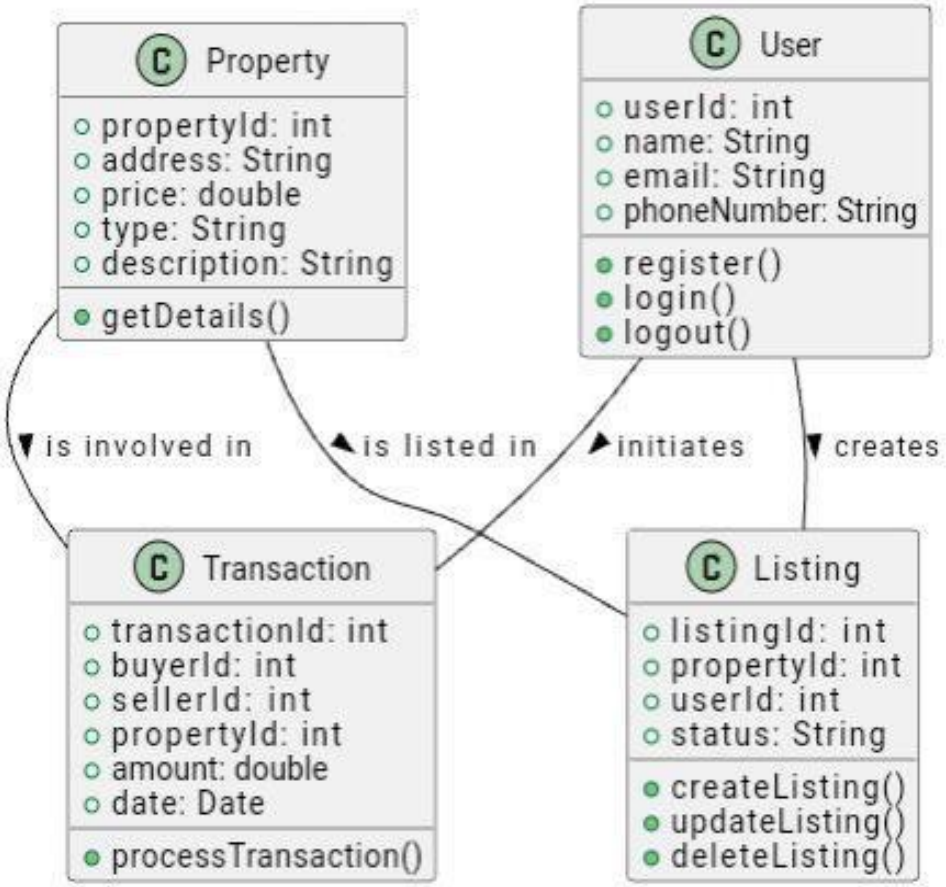
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

Diagram :


Property
propertyId: int
address: String
price: double
type: String
description: String
getDetails()
User
userId: int
name: String
email: String
phoneNumber: String
register()
login()
logout()
is involved in
is listed in
initiates
creates
Transaction
transactionId: int
buyerId: int
sellerId: int
propertyId: int
amount: double
date: Date
processTransaction()
Listing
listingId: int
propertyId: int
userId: int
status: String
createListing()
updateListing()
deleteListing()

EX NO:10	ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.
DATE	

AIM:

To Draw the Class Diagram for any project

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

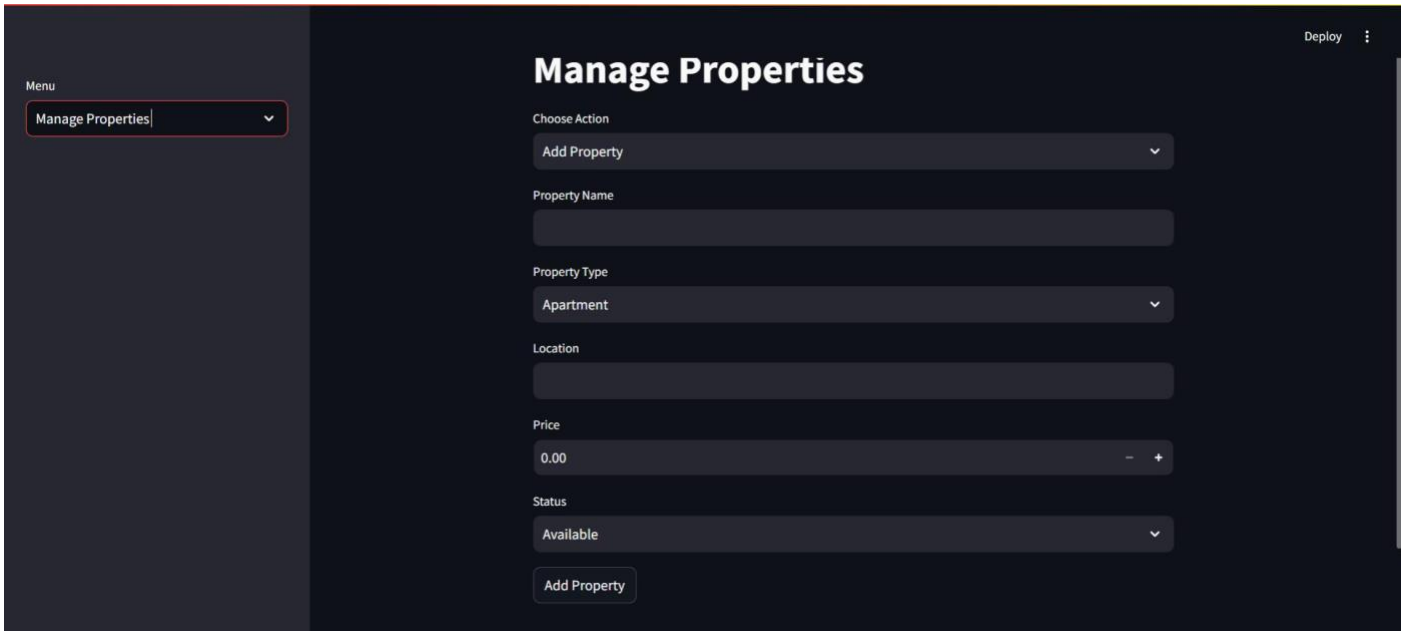
INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

OUTPUT :


Deploy
Menu
Home
Real Estate Management System
Welcome to the Real Estate Management System!


Deploy
Menu
Manage Properties
Manage Properties
Choose Action
Add Property
Property Name
Property Type
Apartment
Location
Price
0.00
Status
Available
Add Property

EX NO:11	MINI PROJECT- STUDENT RESULT MANAGEMENT & AUTOMATED NOTIFICATION SYSTEM
DATE	

AIM:

The aim of the Real Estate Management System project is to develop an intuitive and efficient web application to manage various real estate operations. The system focuses on automating tasks like property listings, client management, and property bookings. By utilizing Streamlit for the user interface and MySQL for data storage, the project seeks to enhance operational efficiency, minimize errors, and provide a seamless user experience for real estate professionals

ALGORITHM:

1. Initialize the application and set up navigation using Streamlit.

2. Connect to the MySQL database using a dedicated function.

3. Create Navigation Menu:

- Options include "Home", "Manage Properties", "Manage Clients", "Property Bookings", and "Booking History".

4. Implement Functionalities:

- Home: Display a welcome message.

- Manage Properties: Add, remove, or view property details.

- Manage Clients: Add, remove, or view client information.

- Property Bookings: Book a property or cancel a booking.

- Booking History: View booking history for a specific client.

5. Execute Database Operations for each action, including SQL queries for data management.

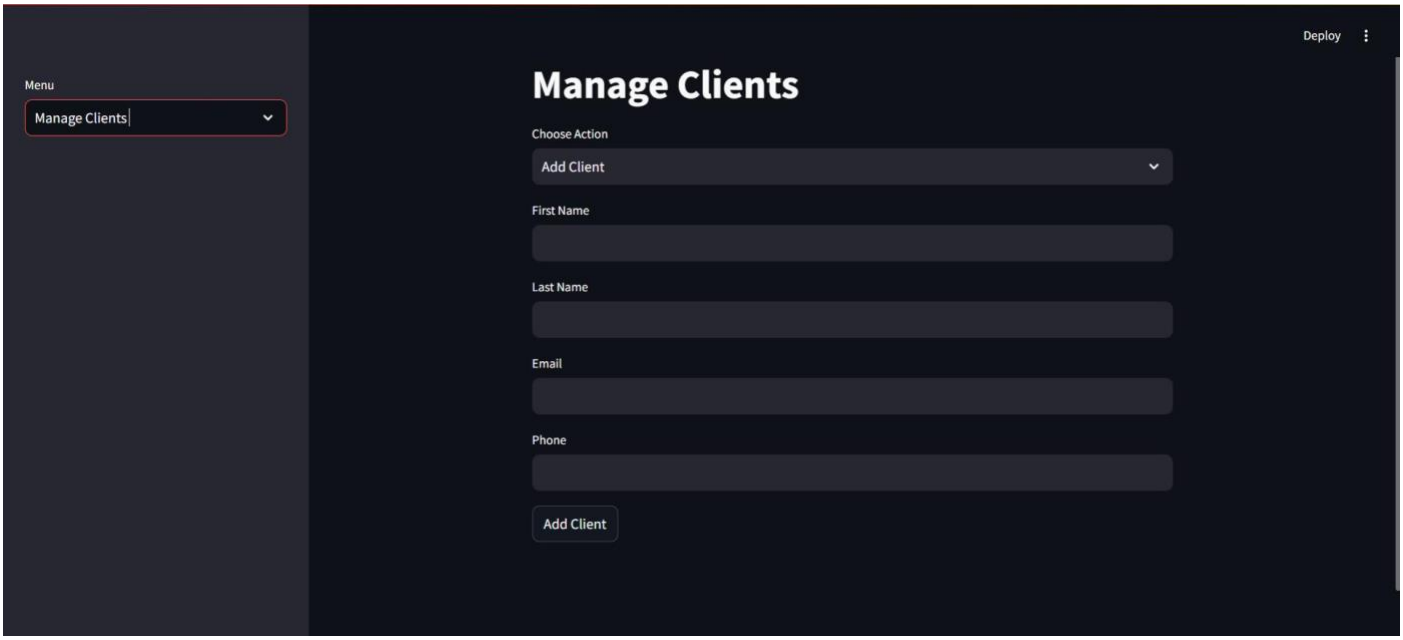
6. Handle User Input and display feedback (e.g., success or error messages).

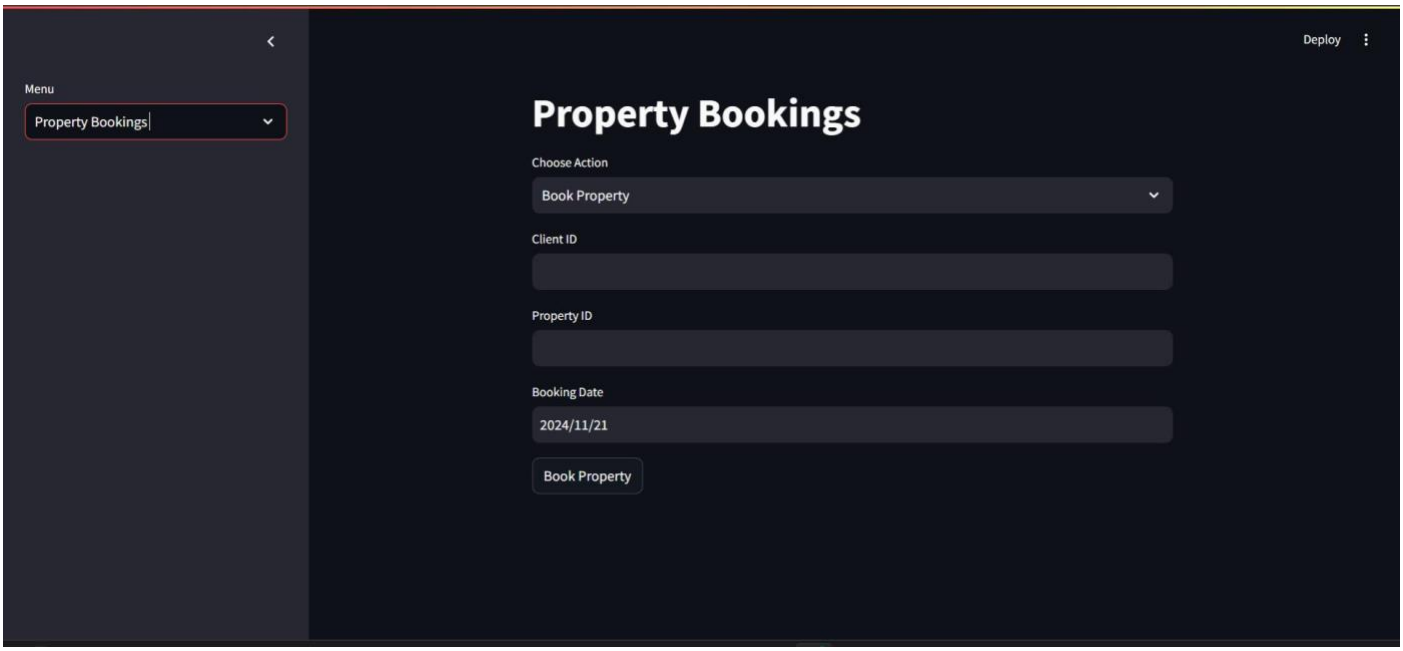
7. End the application and close the database connection to ensure efficiency.

PROGRAM:

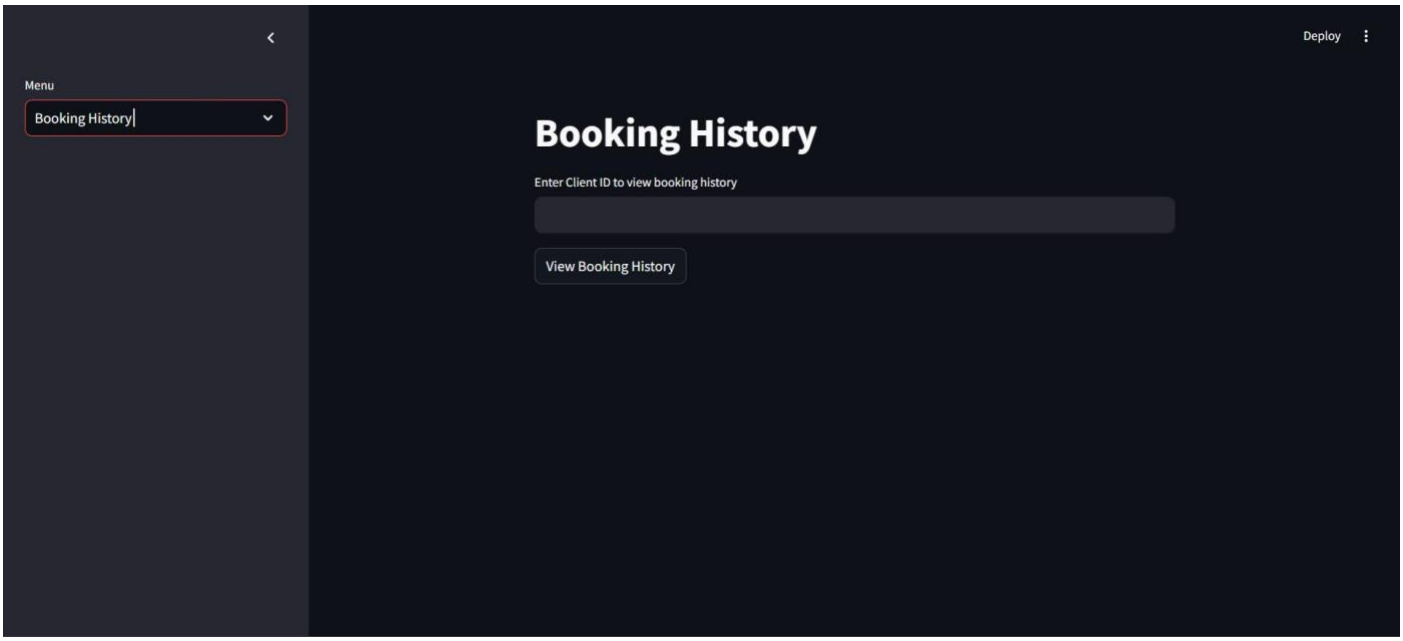
```
Import streamlit as st

Import mysql.connector

Def create_connection():
```

Return mysql.connector.connect(





```
        Host=”127.0.0.1”,
        Port=2129,
        User=”root”,
        Password=”2129”,
        Database=”real_estate”
    )
Menu = [“Home”, “Manage Properties”, “Manage Clients”, “Property Bookings”, “Booking History”]
Choice = st.sidebar.selectbox(“Menu”, menu)
If choice == “Home”:
    St.title(“Real Estate Management System”)
    St.write(“Welcome to the Real Estate Management System!”)
Elif choice == “Manage Properties”:
    St.title(“Manage Properties”)
    Action = st.selectbox(“Choose Action”, [“Add Property”, “Remove Property”, “View All Properties”])
    If action == “Add Property”:
        Property_name = st.text_input(“Property Name”)
        Property_type = st.selectbox(“Property Type”, [“Apartment”, “House”, “Commercial”])
        Location = st.text_input(“Location”)
        Price = st.number_input(“Price”, min_value=0.0)
        Status = st.selectbox(“Status”, [“Available”, “Sold”, “Rented”]
        If st.button(“Add Property”):
            Conn = create_connection()
            Cursor = conn.cursor()
            Cursor.execute(f”INSERT INTO Properties (PropertyName, PropertyType, Location, Price, Status) VALUES (‘{property_name}’, ‘{property_type}’, ‘{location}’, {price}, ‘{status}’)”)
            Conn.commit()
            St.success(“Property added successfully!”)
    Elif action == “Remove Property”:
```


Deploy
Menu
Booking History
Booking History
Enter Client ID to view booking history
View Booking History

```
        Property_id = st.text_input(“Enter Property ID to remove”)
        If st.button(“Remove Property”):
            Conn = create_connection()
            Cursor = conn.cursor()
            Cursor.execute(f”DELETE FROM Properties WHERE PropertyID = {property_id}”)
            Conn.commit()
            St.success(“Property removed successfully!”)
    Elif action == “View All Properties”:
        Conn = create_connection()
        Cursor = conn.cursor()
        Cursor.execute(“SELECT * FROM Properties”)
        Properties = cursor.fetchall()
        St.write(properties)
# Manage Clients
Elif choice == “Manage Clients”:
    St.title(“Manage Clients”)
    Action = st.selectbox(“Choose Action”, [“Add Client”, “Remove Client”, “View Client Details”])
    If action == “Add Client”:
        First_name = st.text_input(“First Name”)
        Last_name = st.text_input(“Last Name”)
        Email = st.text_input(“Email”)
        Phone = st.text_input(“Phone”)
        If st.button(“Add Client”):
            Conn = create_connection()
            Cursor = conn.cursor()
            Cursor.execute(f”INSERT INTO Clients (FirstName, LastName, Email, Phone) VALUES
(‘{first_name}’, ‘{last_name}’, ‘{email}’, ‘{phone}’)”)
            Conn.commit()
```

```
            St.success(“Client added successfully!”)
        Elif action == “Remove Client”:
            Client_id = st.text_input(“Enter Client ID to remove”)
            If st.button(“Remove Client”):
                Conn = create_connection()
                Cursor = conn.cursor()
                Cursor.execute(f”DELETE FROM Clients WHERE ClientID = {client_id}”)
                Conn.commit()
                St.success(“Client removed successfully!”)
        Elif action == “View Client Details”:
            Client_id = st.text_input(“Enter Client ID to view details”)
            If st.button(“View Details”):
                Conn = create_connection()
                Cursor = conn.cursor()
                Cursor.execute(f”SELECT * FROM Clients WHERE ClientID = {client_id}”)
                Client = cursor.fetchone()
                St.write(client)
    # Property Bookings
    Elif choice == “Property Bookings”:
        St.title(“Property Bookings”)
        Action = st.selectbox(“Choose Action”, [“Book Property”, “Cancel Booking”])
        If action == “Book Property”:
            Client_id = st.text_input(“Client ID”)
            Property_id = st.text_input(“Property ID”)
            Booking_date = st.date_input(“Booking Date”)
            If st.button(“Book Property”):
                Conn = create_connection()
                Cursor = conn.cursor()
```

```
        Cursor.execute(f'INSERT INTO Bookings (ClientID, PropertyID, BookingDate) VALUES ('{client_id}', '{property_id}', '{booking_date}')")

        Conn.commit()

        St.success("Property booked successfully!")

    Elif action == "Cancel Booking":

        Booking_id = st.text_input("Enter Booking ID to cancel")

        If st.button("Cancel Booking"):

            Conn = create_connection()

            Cursor = conn.cursor()

            Cursor.execute(f'DELETE FROM Bookings WHERE BookingID = {booking_id}")

            Conn.commit()

            St.success("Booking cancelled successfully!")

# Booking History

Elif choice == "Booking History":

    St.title("Booking History")

    Client_id = st.text_input("Enter Client ID to view booking history")

    If st.button("View Booking History"):

        Conn = create_connection()

        Cursor = conn.cursor()

        Cursor.execute(f'SELECT * FROM Bookings WHERE ClientID = {client_id}")

        Bookings = cursor.fetchall()

        St.write(bookings)
```

SQL code :

```
CREATE TABLE Properties (

    PropertyID INT AUTO_INCREMENT PRIMARY KEY,

    PropertyName VARCHAR(100) NOT NULL,

    PropertyType VARCHAR(50) NOT NULL,

    Location VARCHAR(100) NOT NULL,
```

```
    Price FLOAT NOT NULL,
    Status VARCHAR(20) NOT NULL
);
CREATE TABLE Clients (
    ClientID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15) UNIQUE NOT NULL
);
CREATE TABLE Bookings (
    BookingID INT AUTO_INCREMENT PRIMARY KEY,
    ClientID INT,
    PropertyID INT,
    BookingDate DATE NOT NULL,
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),
    FOREIGN KEY (PropertyID) REFERENCES Properties(PropertyID)
);
```

Conclusion :

The Real Estate Management System developed using Streamlit and MySQL provides a user-friendly platform for real estate professionals to efficiently manage operations.This system centralizes key functionalities such as adding, updating, and viewing property details, managing client information, and tracking property bookings and history, ensuring seamless and organized real estate management.