

RAJALAKSHMI ENGINEERING COLLEGE

**RAJALAKSHMI NAGAR, THANDALAM – 602 105**

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**

**RAJALAKSHMI NAGAR, THANDALAM – 602-105 BONAFIDE CERTIFICATE**

**NAME: REGISTER NO.:**

**ACADEMIC YEAR**: 2024-25 **SEMESTER:** III **BRANCH:** B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

**CB23332-SOFTWARE ENGINEERING -** Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on

Internal Examiner External Examiner



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1. | Preparing Problem Statement |  |  |
| 2. | Software Requirement Specification (SRS) |  |  |
| 3. | Entity-Relational Diagram |  |  |
| 4. | Data Flow Diagram |  |  |
| 5. | Use Case Diagram |  |  |
| 6. | Activity Diagram |  |  |
| 7. | State Chart Diagram |  |  |
| 8. | Sequence Diagram |  |  |
| 9. | Collaboration Diagramt |  |  |
| 10. | Class Diagram |  |  |

**1. PREPARING PROBLEM STATEMENT**

**Aim :**

**1. PREPARING PROBLEM STATEMENT**

To prepare PROBLEM STATEMENT for real estate booking system using smart contracts.

**Algorithm** :

1. **Initialize System**
   * Set up the blockchain environment and deploy the smart contract for the booking system.
   * Initialize the database for properties, users (buyers/sellers), and transaction records.
2. **User Registration**
   * Input: User details (e.g., ID, name, wallet address).
   * Process: Validate user details and register the user on the platform.
   * Output: User registration confirmation.
3. **Property Listing**
   * Input: Seller inputs property details (e.g., property ID, location, price, description).
   * Process: Verify property ownership and validity. Store property data on the blockchain through the smart contract.
   * Output: Property successfully listed with a unique identifier.
4. **Search and View Properties**
   * Input: Buyer inputs search criteria (e.g., location, price range).
   * Process: Filter properties based on search criteria and display relevant listings.
   * Output: Display of properties matching the search criteria.
5. **Book Property**
   * Input: Buyer selects a property and requests booking.
   * Process:
     + Lock the property to prevent duplicate bookings.
     + Initialize a smart contract for the booking transaction, defining terms such as payment conditions, property transfer conditions, and cancellation policies.
   * Output: Property reserved for the buyer, awaiting payment.
6. **Payment Processing**
   * Input: Buyer initiates payment through the platform.
   * Process: Verify payment details and update the smart contract with the payment status.
   * Output: Payment confirmed, funds held in escrow (or directly transferred if conditions are met).
7. **Ownership Transfer**
   * Input: Payment confirmation and fulfillment of smart contract terms.
   * Process:
     + Check if all conditions (e.g., payment, documentation) are satisfied.
     + Trigger the transfer of property ownership in the smart contract.
     + Update property status on the blockchain.
   * Output: Ownership transferred to the buyer; transaction recorded on the blockchain.
8. **Transaction Record and Confirmation**
   * Process: Record all details of the transaction (e.g., buyer ID, seller ID, property ID, payment) on the blockchain.
   * Output: Transaction record stored; confirmation sent to both buyer and seller.
9. **Handle Disputes (if applicable)**
   * Input: Either party initiates a dispute.
   * Process: Trigger the dispute resolution mechanism outlined in the smart contract (e.g., arbitration, refunds if applicable).
   * Output: Dispute resolved per contract terms.
10. **End Process**
    * Process: Update system and close the transaction. Release any locked property if booking is canceled.
    * Output: System ready for the next transaction.

**INPUTS:**

**1. User Registration Inputs**

* **User ID**: Unique identifier for each user.
* **User Type**: Type of user (buyer, seller, or agent).
* **Personal Details**: Name, contact information, and identification number.
* **Digital Wallet Address**: Blockchain wallet for transactions.

**2. Property Listing Inputs (entered by sellers)**

* **Property ID**: Unique identifier for each property.
* **Property Details**: Address, type (e.g., residential, commercial), and description.
* **Property Size**: Area in square feet or meters.
* **Price**: Listed price of the property.
* **Ownership Documents**: Digital proof of ownership, stored securely.
* **Images**: Photos of the property.
* **Additional Features**: Number of bedrooms, bathrooms, parking spaces, etc.

**3. Search and Filter Inputs (entered by buyers)**

* **Location**: Desired city or neighborhood.
* **Budget Range**: Minimum and maximum price.
* **Property Type**: Type of property (e.g., apartment, villa).
* **Property Size**: Size range if applicable.
* **Additional Filters**: Amenities, number of rooms, etc.

**4. Booking Request Inputs**

* **Buyer ID**: ID of the buyer making the booking request.
* **Selected Property ID**: ID of the property the buyer wants to book.
* **Booking Terms**: Any conditions or deposit requirements.
* **Escrow Agreement**: Details on payment holding, if applicable.

**SAMPLE OUTPUT**

**Problem:**  
 Traditional real estate transactions are often plagued by delays, high intermediary fees, and lack of transparency.

**Background:**  
 Real estate deals involve complex documentation and third-party verification, increasing costs and time.

**Relevance:**  
 A blockchain-based booking system with smart contracts can streamline transactions, providing security, transparency, and automation.

**Objectives:**  
 Develop a secure, efficient real estate booking system using smart contracts to automate and validate transactions, reduce intermediaries, and enable transparent ownership transfers.

**Result :**

**Ex.NO. 2 : Write the software requirement specification document**

**AIM:**

To do requirement analysis and develop Software Requirement Specification (SRS) for real estate management system using smart contracts.

**ALGORITHM:**

1. **Define the Purpose**
   * Identify the core purpose of the SRS document (e.g., to provide a detailed description of the system's functional and non-functional requirements).
2. **Gather Requirements**
   * Conduct discussions with stakeholders to gather all requirements.
   * Define the requirements based on user needs, system functionality, and technical constraints.
3. **Draft the SRS Document Structure**
   * Establish the sections to include, such as Introduction, Overall Description, Functional and Non-Functional Requirements, etc.
4. **Write the Introduction Section**
   * Specify the purpose of the system.
   * Define the scope of the project, including high-level goals and objectives.
   * List key definitions, acronyms, and references used in the document.
5. **Describe the Overall System**
   * Outline the system context and operating environment (e.g., blockchain framework, smart contract functionality).
   * Identify the user classes and characteristics (e.g., buyers, sellers, agents).
   * List dependencies on other systems or technologies.
6. **Introduction**

* The aim is to build a secure, transparent, and efficient real estate platform leveraging blockchain and smart contracts.
* This document outlines the requirements for the development of the system.
* The project will automate real estate transactions, reduce intermediaries, and ensure contract enforceability.
* Target users include buyers, sellers, agents, and property managers.
* Key terms like blockchain, smart contracts, and immutability are defined.

**2. Overall Description**

* The system operates in a blockchain environment, ensuring decentralized and tamper-proof transactions.
* Users can list, buy, and sell properties via automated smart contracts.
* The platform supports multi-user roles and manages property information transparently.
* It interfaces with financial institutions for payment processing.
* Assumptions include blockchain stability and user familiarity with digital transactions.

**3. System Features**

* Property listing and management for verified users.
* Smart contract-based transactions to automate payments and ownership transfers.
* Secure storage of transaction history on the blockchain.
* User authentication and authorization for secure access.
* Real-time notifications and updates on transaction status.

**4. External Interface Requirements**

* Web-based UI for user interaction with responsive design for multiple devices.
* APIs to connect with third-party services for identity verification and payment.
* Integration with blockchain nodes to facilitate smart contract execution.
* Data exchange protocols for secure transmission of sensitive information.
* Compatibility with digital wallets for cryptocurrency transactions.

**5. System Requirements**

* Functional: Support for user registration, property listing, transaction tracking, and payment handling.
* Performance: Fast transaction processing and quick smart contract execution.
* Security: Encrypted data storage and secure blockchain transactions.
* Reliability: High uptime and fault tolerance for critical operations.
* Scalability: Ability to handle a growing number of users and transactions.

**6. Use Case Diagrams**

* Depicts interactions between users (buyers, sellers, agents) and the system.
* Shows primary use cases such as "List Property," "Buy Property," and "View Contract."
* Includes user authentication, property search, and transaction completion processes.
* Highlights system responses for each action initiated by users.
* Provides a visual outline of each user role’s permissions and interactions.

**7. System Models**

* Illustrates system architecture, showing data flow between components (UI, blockchain, database).
* Describes interactions between front-end, back-end, and smart contract layers.
* Displays data storage on the blockchain and interactions with external payment gateways.
* Shows component-level communication for real-time data updates.
* Includes sequence diagrams to visualize order of operations during transactions.

**8. Security and Privacy**

* Ensures user data encryption and privacy through blockchain immutability.
* Implements role-based access control for secure user interactions.
* Uses multi-factor authentication for account security.
* Includes periodic security audits to identify and address vulnerabilities.
* Protects sensitive information such as personal data and payment details.

**9. Maintenance and Support**

* Regular updates to incorporate user feedback and improve system performance.
* Scheduled blockchain node maintenance for system stability.
* Bug-tracking and resolution process to ensure smooth operations.
* Dedicated support team for user assistance and troubleshooting.
* Documentation and training for easy onboarding of new users and admins.

**SAMPLE OUTPUT:**

**1. Introduction**

* Purpose: To automate real estate transactions securely using blockchain and smart contracts.
* Goals: Streamline property transactions, ensure transparency, and eliminate intermediaries.
* Target Users: Buyers, sellers, agents, and property managers.
* Project Scope: Focus on property listing, contract automation, and transaction finalization.
* Key Definitions: Blockchain, Smart Contracts, Decentralization.

**2. Overall Description**

* System Environment: Operates on a blockchain with smart contract capability.
* User Roles: Buyers, sellers, agents with specific access levels.
* Dependencies: Requires blockchain infrastructure and digital payment systems.
* Operational Assumptions: Users have internet access and familiarity with online transactions.
* Main Processes: Listing properties, managing contracts, and executing payments.

**3. System Features**

* Property Listing: Enables users to list and view property details.
* Automated Contracts: Smart contracts enforce transaction terms.
* Payment Processing: Integrates with digital wallets for secure payments.
* Transaction History: Immutable storage of records on the blockchain.
* User Notifications: Real-time updates on transaction status.

**4. External Interface Requirements**

* User Interface: Web-based application with responsive design.
* APIs: Integration with third-party verification and payment services.
* Blockchain Nodes: Connects with blockchain for smart contract functionality.
* Data Transfer: Secure protocols for handling sensitive user data.
* Digital Wallets: Compatibility with cryptocurrency transactions.

**5. System Requirements**

* Functional: Property listing, contract management, and payment processing.
* Performance: Real-time response and fast contract execution.
* Security: Data encryption and blockchain-backed integrity.
* Reliability: High availability with backup and recovery mechanisms.
* Scalability: Accommodates increasing user and transaction volume.

**6. Use Case Diagrams**

* User Authentication: Ensures secure login for different user roles.
* Property Listing: Process for adding and viewing properties.
* Transaction Process: Outlines steps for completing property purchases.
* Notifications: User receives alerts during key transaction stages.
* Admin Role: Manage user accounts and system settings.

**7. System Models**

* Architecture Diagram: Shows interaction between UI, blockchain, and database.
* Data Flow Diagram: Depicts data movement through system components.
* Sequence Diagram: Steps for a buyer’s journey from listing to transaction.
* Component Diagram: Highlights modules like UI, blockchain, and APIs.
* Interaction Diagram: Visualizes role-specific interactions with the system.

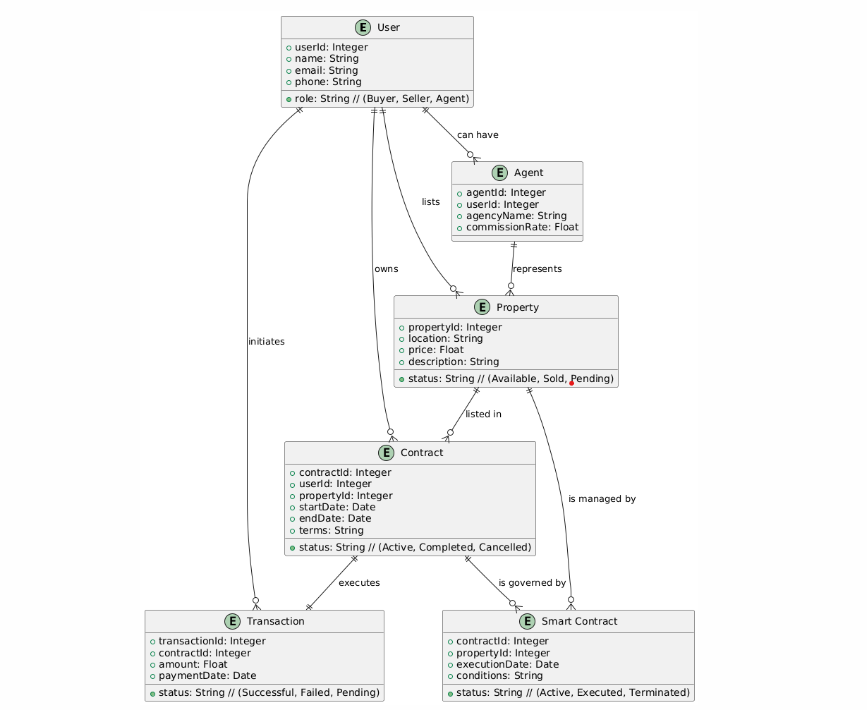
**8. Security and Privacy**

* Data Encryption: Protects user data in storage and during transmission.
* Access Control: Role-based access for different user levels.
* Multi-Factor Authentication: Enhances login security.
* Blockchain Integrity: Ensures transaction records are immutable.
* Regular Audits: Scheduled security assessments to ensure system safety.

**9. Maintenance and Support**

* Software Updates: Regular improvements and security patches.
* Technical Support: Available for user issues and troubleshooting.
* System Monitoring: Monitors performance and error logs continuously.
* User Documentation: Guides for users and administrators.
* Training Sessions: Onboarding for new users and system administrators.

**DIAGRAM**



**Result :**

**3. ENTITY RELATIONSHIP MODEL**

**AIM:**

To Draw the Entity Relationship Diagram for real estate booking system.

**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

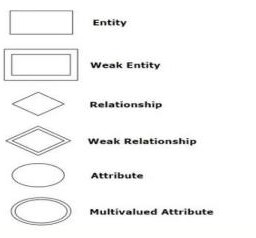
Primary Keys

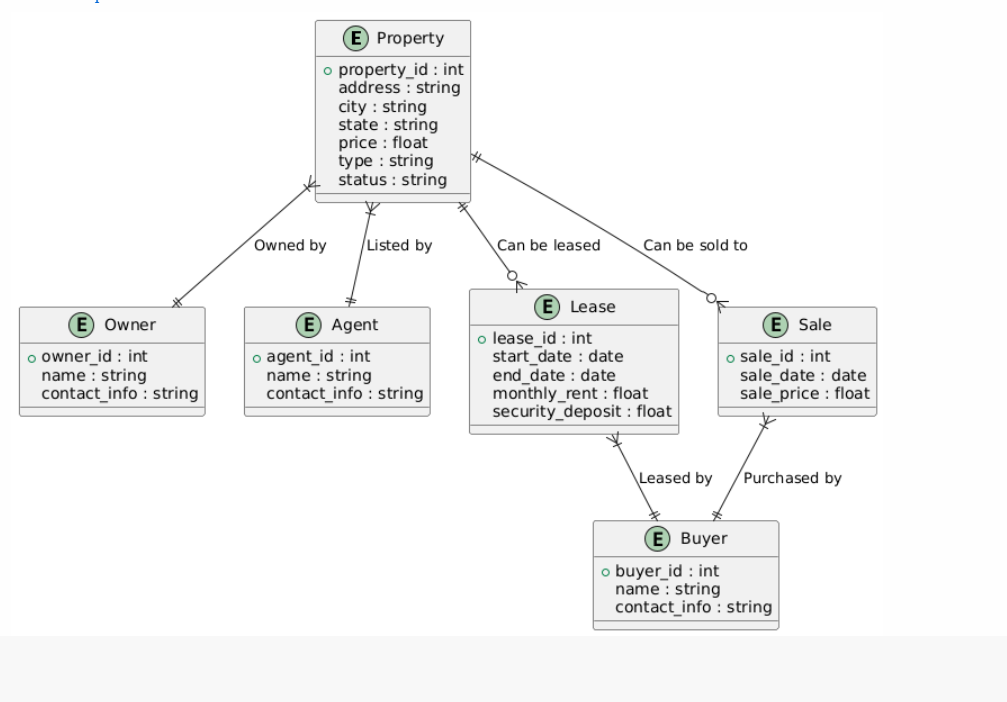
Attributes

Mapping of Attributes with Entities

### ER DIAGRAM:

SYMBOLS:





**Result:**

**4. DATA FLOW DIAGRAM**

**AIM:**

To Draw the Data Flow Diagram for real estate booking system 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

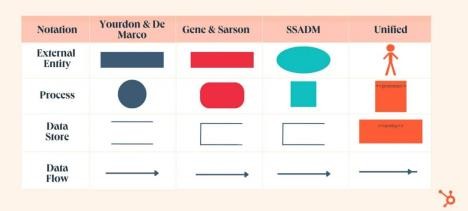
**INPUT:**

Processes

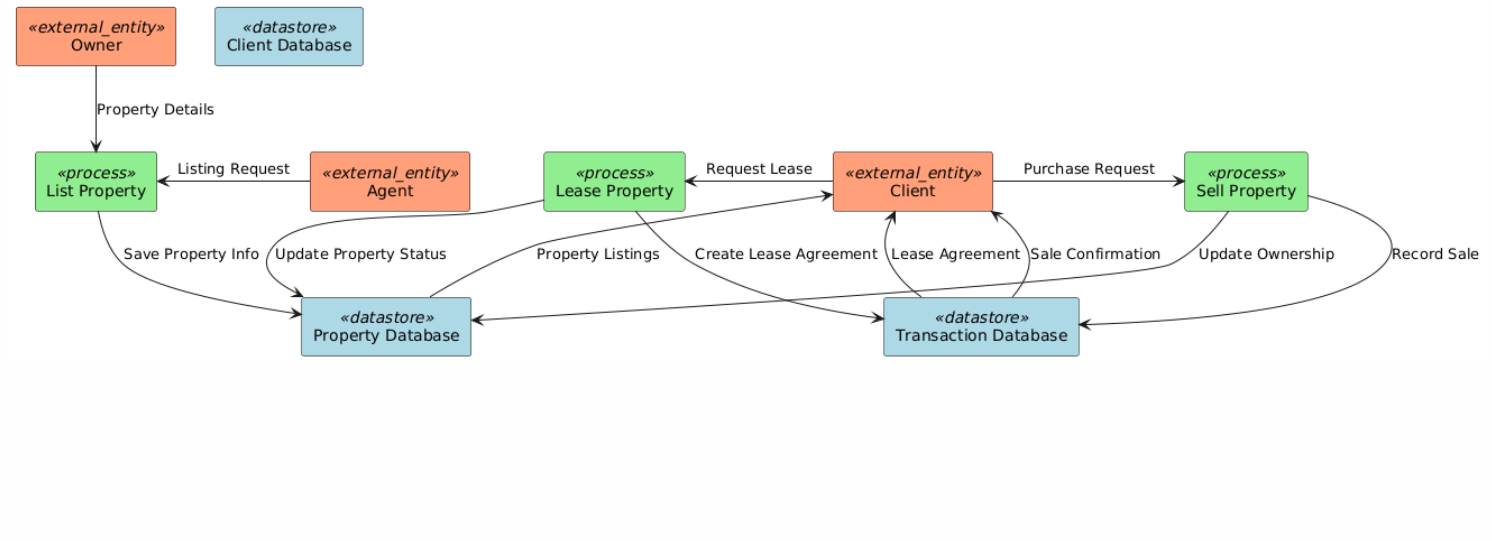
Datastores

External Entities

**Sample diagrams:**



**Diagram:**



**Result:**

**5. USE CASE DIAGRAM**

**AIM:**

To Draw the Use Case Diagram for real estate booking system.

**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

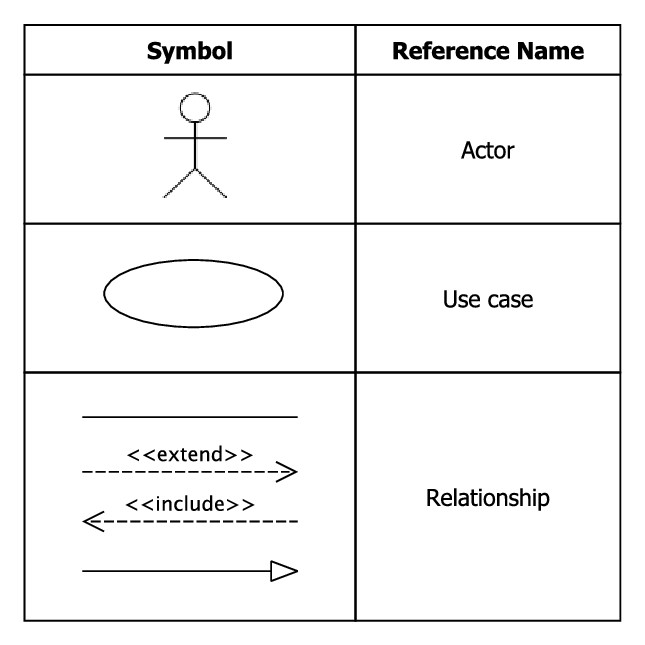
**INPUTS:**

Actors

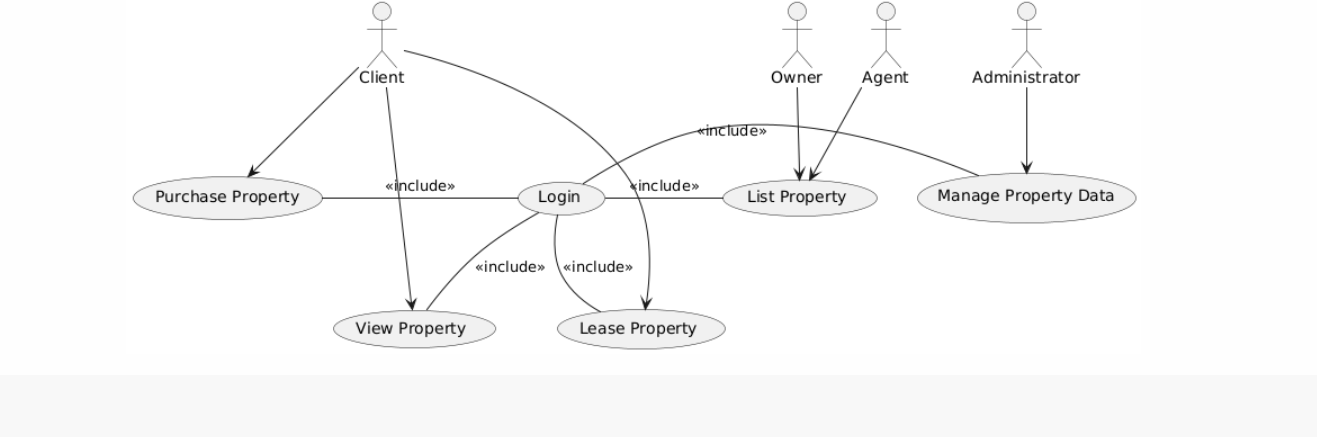
Use Cases

Relations

**Sample Symbols:**



**Diagram:**



**Result :**

**6. ACTIVITY DIAGRAM**

**AIM:**

To Draw the activity Diagram for real estate booking system.

**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

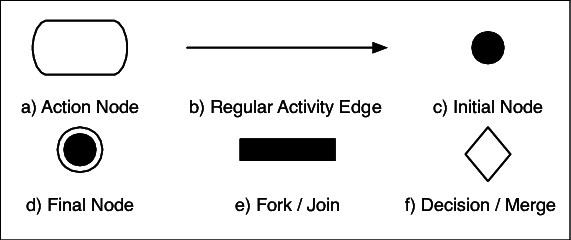
Decision Points

Guards

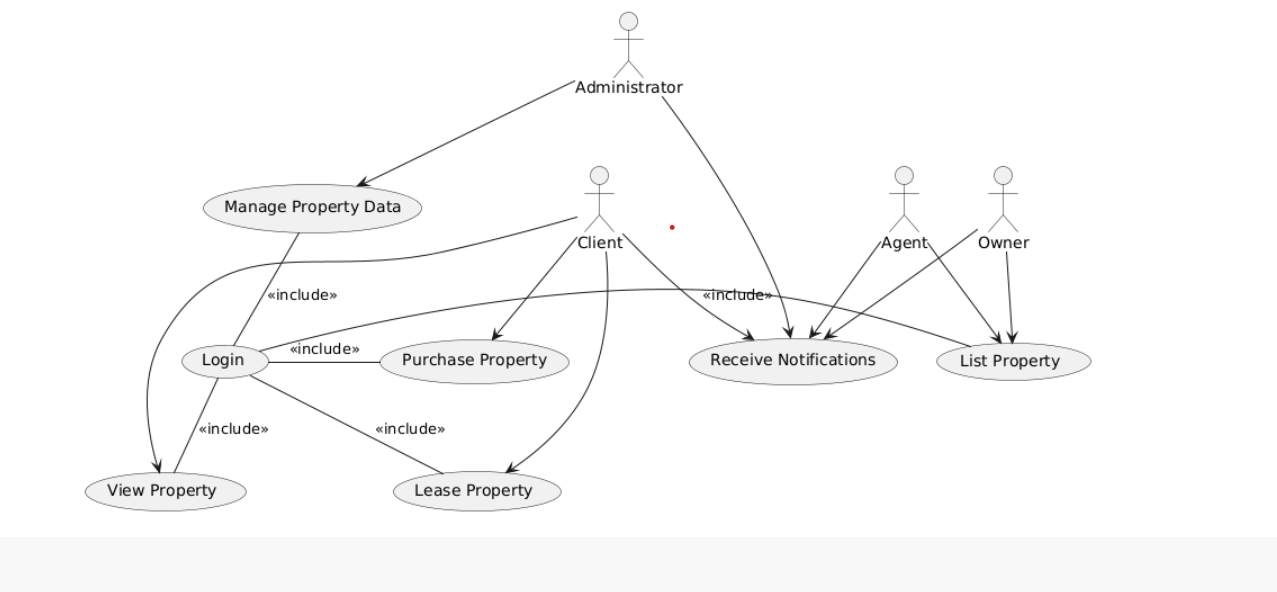
Parallel Activities

Conditions

**Sample Symbols:**



**Diagram:**

****

**Result** :

**7. STATE CHART DIAGRAM**

**AIM:**

To Draw the State Chart Diagram for real estate booking system.

**ALGORITHM:**

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

STEP-2: Identify the states.

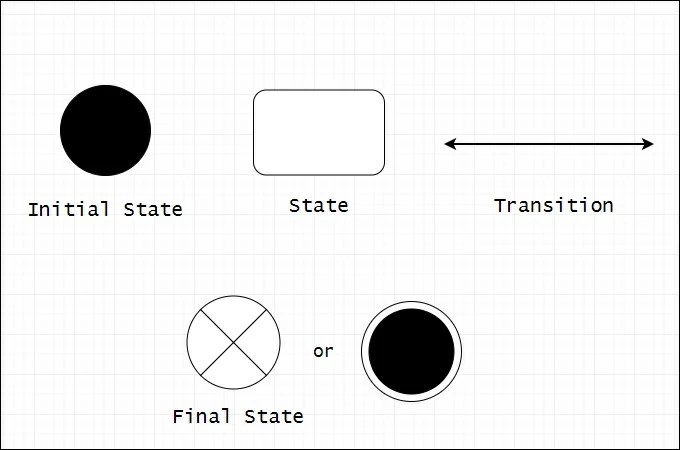
STEP-3: Identify the events.

**INPUTS:**

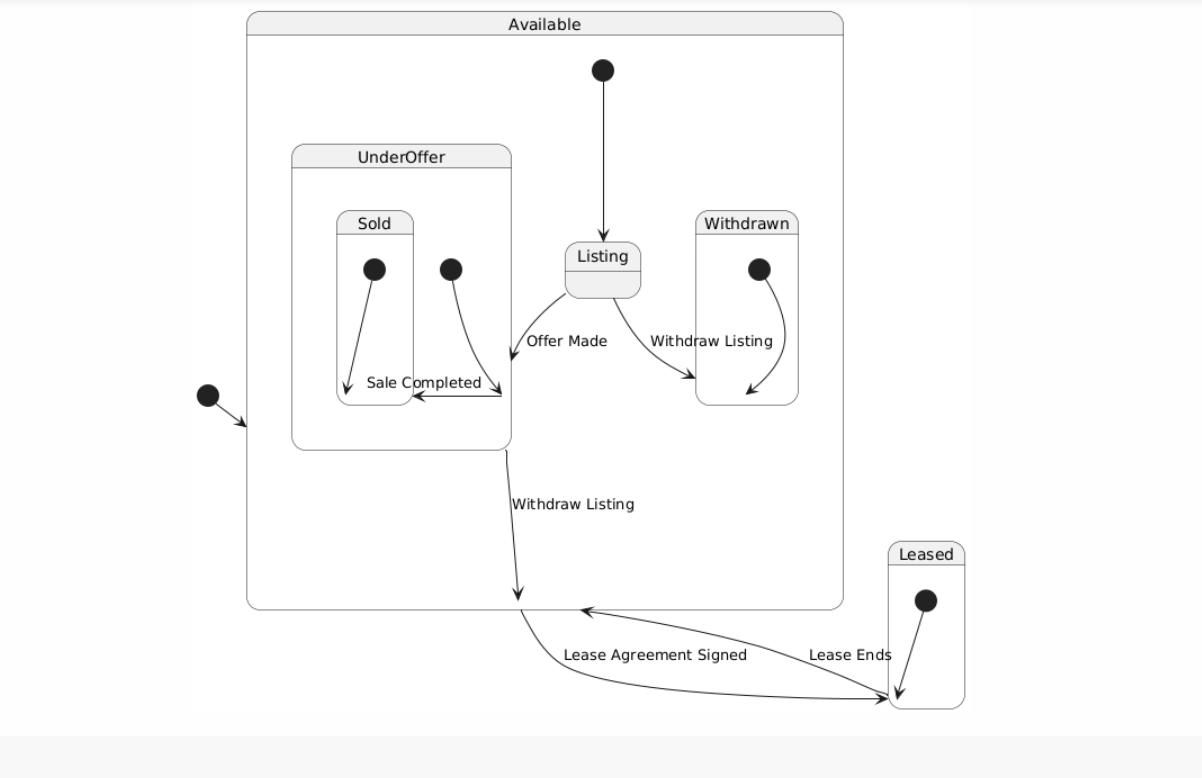
Objects

States

Events

**Sample Symbol:**

**Diagram:**



**Result** :

**8. SEQUENCE DIAGRAM**

**AIM:** To Draw the Sequence Diagram for real estate booking system.

**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:**

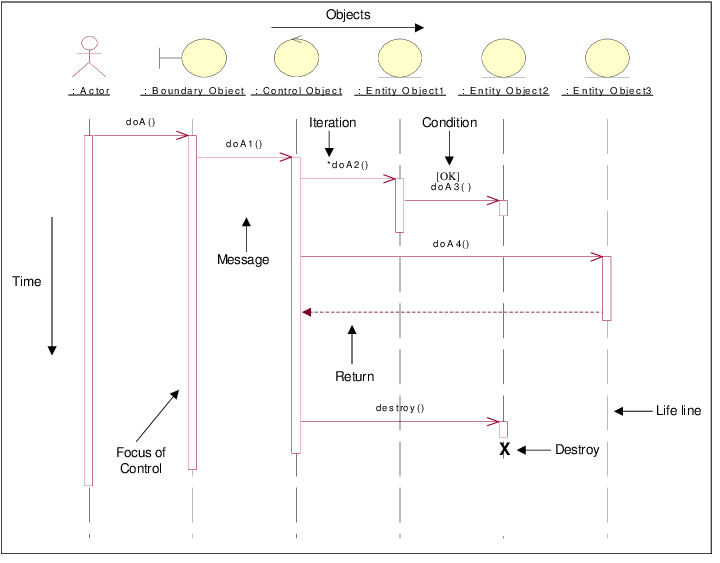
Objects taking part in the interaction.

Message flows among the objects.

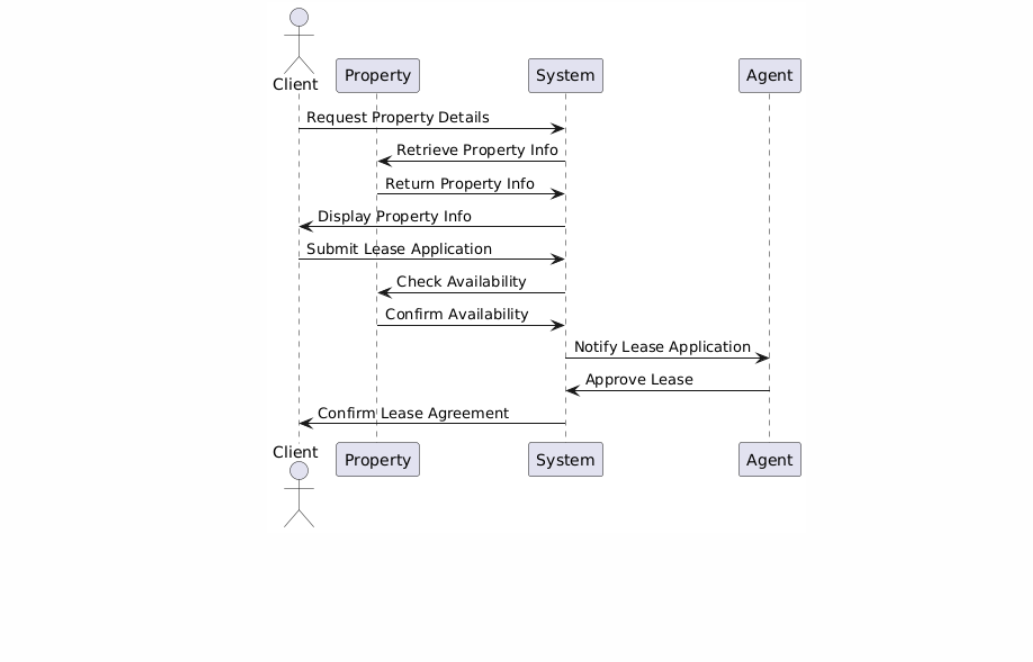
The sequence in which the messages are flowing.

Object organization.

**Sample Diagram:**



**Diagram:**

****

**Result** :

**9. COLLABORATION DIAGRAM**

**AIM:**

To Draw the Collaboration Diagram for real estate booking system

**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:**

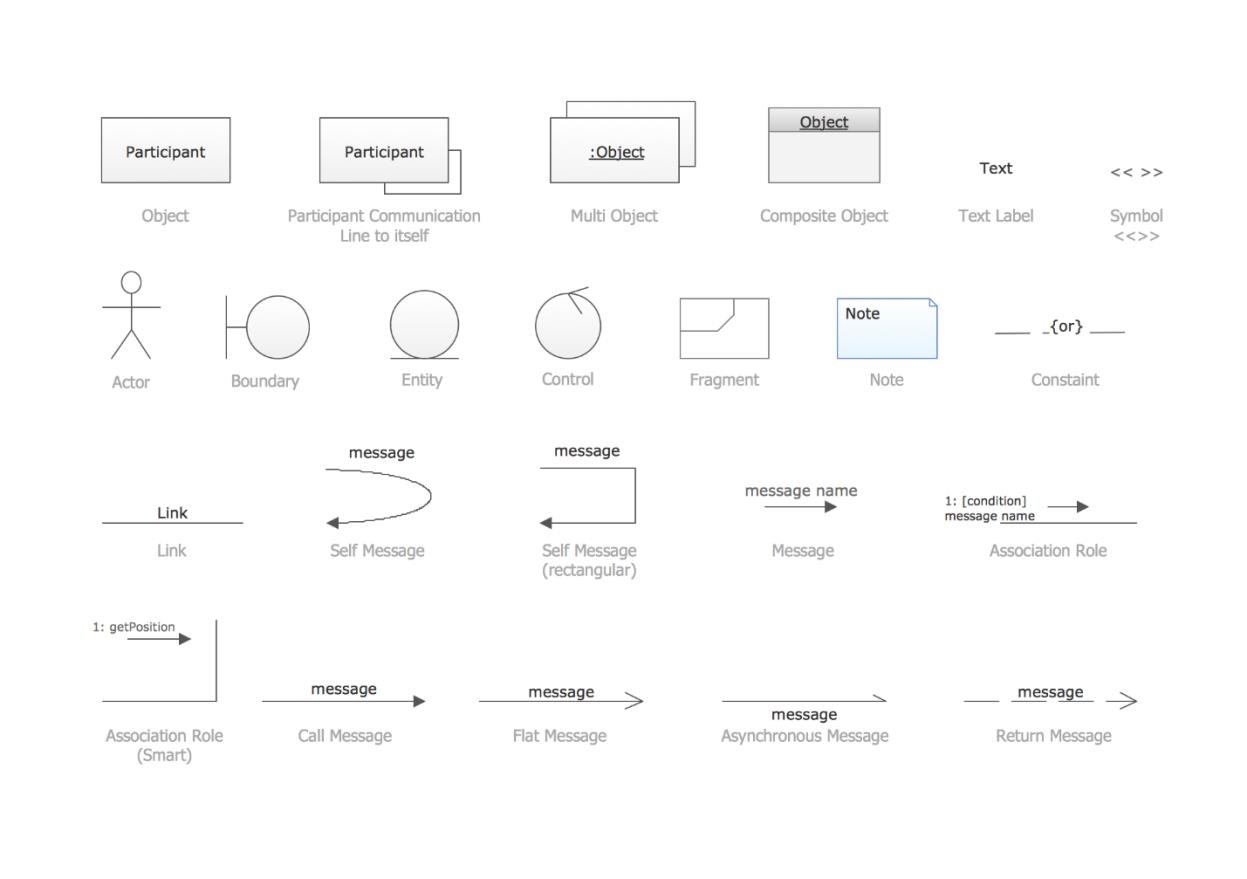
Objects taking part in the interaction.

Message flows among the objects.

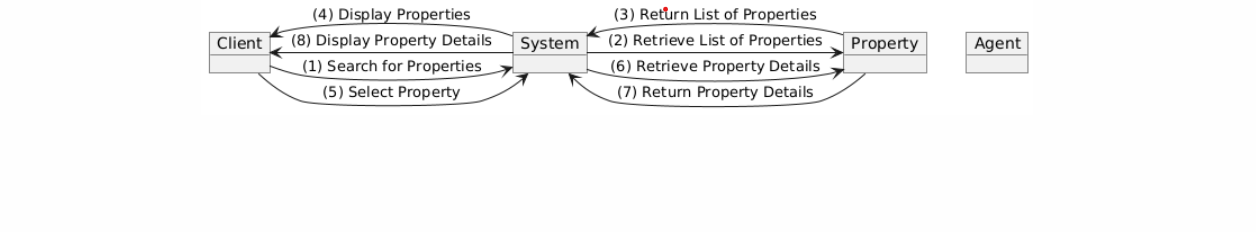
The sequence in which the messages are flowing.

Object organization.

**Sample Symbol:**



**Diagram:**



**Result** :

**10. CLASS DIAGRAM**

**AIM:**

To Draw the Class Diagram for real estate booking system.

**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:**

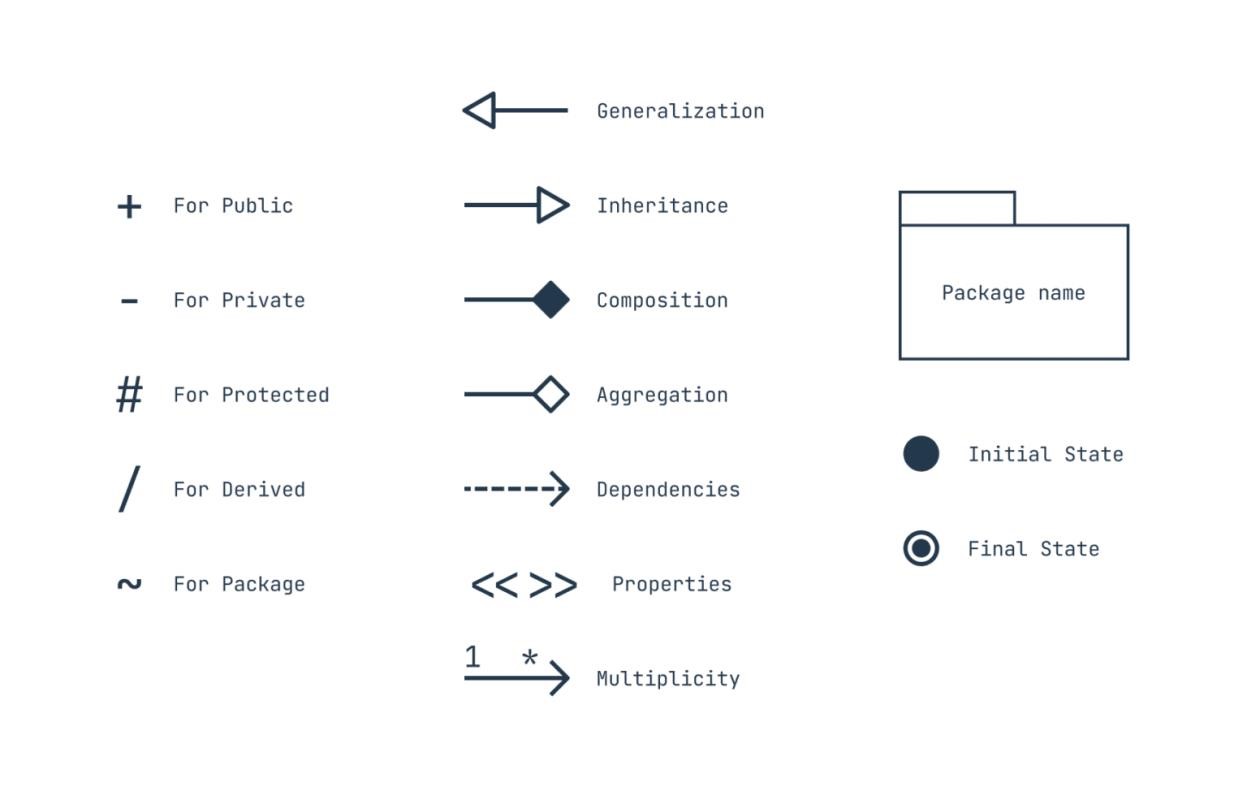
1. Class Name

2. Attributes

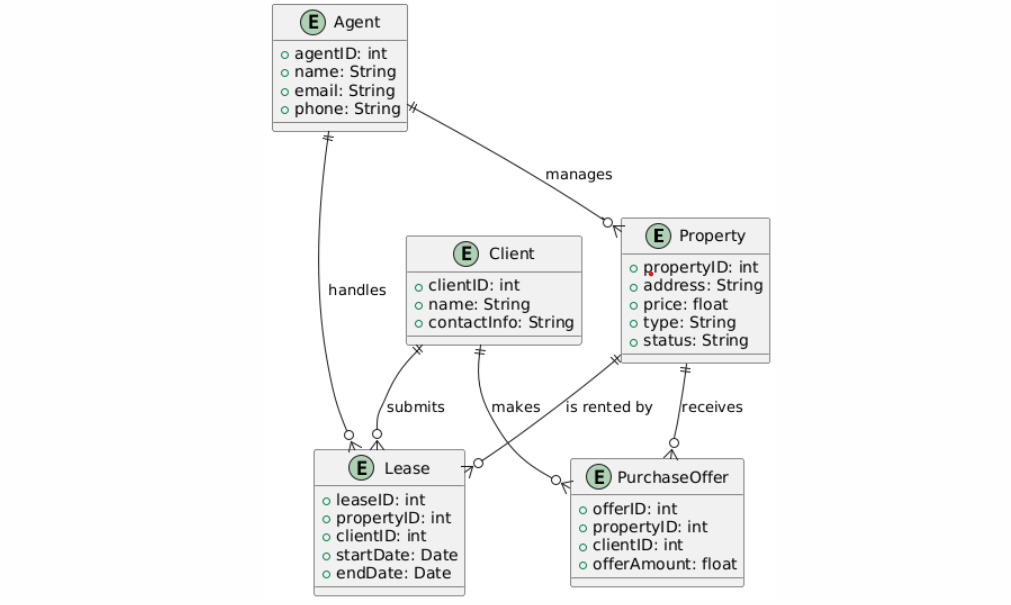
3. Methods

4. Visibility Notation

**Sample Symbol :**



**Diagram:**



# Result:

**CODE FOR MINI PROJECT**

## CODE:

## IMPLEMENTATION CODE USING PYTHON

contract RealEstateBooking {

struct Property {

uint id;

string name;

string location;

uint price;

address payable owner;

bool isBooked;

}

mapping(uint => Property) public properties;

uint public propertyCount = 0;

function addProperty(string memory \_name, string memory \_location, uint \_price) public {

propertyCount++;

properties[propertyCount] = Property(

propertyCount,

\_name,

\_location,

\_price,

payable(msg.sender),

false

);

}

function bookProperty(uint \_id) public payable {

Property storage property = properties[\_id];

require(\_id > 0 && \_id <= propertyCount, "Invalid property ID");

require(msg.value == property.price, "Incorrect amount");

require(!property.isBooked, "Property already booked");

property.isBooked = true;

property.owner.transfer(msg.value);

}

function getProperty(uint \_id) public view returns (

uint,

string memory,

string memory,

uint,

address,

bool

) {

Property memory property = properties[\_id];

return (property.id, property.name, property.location, property.price, property.owner, property.isBooked);

}

}

from web3 import Web3

# Connect to local blockchain (Ganache or similar)

web3 = Web3(Web3.HTTPProvider("http://127.0.0.1:8545"))

# Check connection

if not web3.isConnected():

print("Failed to connect to blockchain.")

exit()

# Set default account (buyer/seller)

web3.eth.defaultAccount = web3.eth.accounts[0]

# Contract ABI and Address

contract\_abi = [

# Paste the ABI of the deployed smart contract here

]

contract\_address = "0xYourDeployedContractAddress" # Replace with actual address

# Load the contract

contract = web3.eth.contract(address=contract\_address, abi=contract\_abi)

def add\_property(name, location, price):

tx = contract.functions.addProperty(name, location, price).transact()

web3.eth.wait\_for\_transaction\_receipt(tx)

print(f"Property '{name}' added successfully!")

def book\_property(property\_id, amount):

tx = contract.functions.bookProperty(property\_id).transact({"value": amount})

web3.eth.wait\_for\_transaction\_receipt(tx)

print(f"Property with ID {property\_id} booked successfully!")

def get\_property\_details(property\_id):

property\_details = contract.functions.getProperty(property\_id).call()

print("Property Details:")

print(f"ID: {property\_details[0]}")

print(f"Name: {property\_details[1]}")

print(f"Location: {property\_details[2]}")

print(f"Price: {property\_details[3]} wei")

print(f"Owner: {property\_details[4]}")

print(f"Is Booked: {property\_details[5]}")

# Example User Interaction

add\_property("Luxury Apartment", "New York", 1000000)

get\_property\_details(1)

book\_property(1, 1000000)

get\_property\_details(1)

**Example Inputs and Outputs**

**User Inputs:**

1. **Add a Property**:
   * Name: Luxury Apartment
   * Location: New York
   * Price: 1000000 wei
2. **Get Property Details** for ID 1.
3. **Book Property** ID 1 with 1000000 wei.

**Outputs:**

1. **After adding:**

Property 'Luxury Apartment' added successfully!

1. **Property details (before booking):**

Property Details:

ID: 1

Name: Luxury Apartment

Location: New York

Price: 1000000 wei

Owner: 0x... (Owner Address)

Is Booked: False

1. **After booking:**

Property with ID 1 booked successfully!

1. **Property details (after booking):**

Property Details:

ID: 1

Name: Luxury Apartment

Location: New York

Price: 1000000 wei

Owner: 0x... (Owner Address)

Is Booked: True

**CONCLUSION:**

The Real Estate Booking System leverages smart contracts to provide a secure, efficient, and transparent platform for property transactions. By integrating blockchain technology, the system eliminates intermediaries, reduces costs, and minimizes the risks of fraud and data tampering. The use of smart contracts automates and enforces the terms of agreements, ensuring trust and reliability among all parties involved.

This project highlights the potential of technology in revolutionizing the real estate industry by simplifying processes, enhancing customer experience, and fostering greater transparency. It sets the foundation for future advancements in property management and digital asset transactions, paving the way for a more accessible and streamlined real estate ecosystem.