# Apartment Leasing Management System

Phase II

System Design: 03.25.2024

Umar Abdul Aziz | Shriya Jaknalli | Aashlesha Voditelwar
CSCI 5333 Database Management System
Section-1, Spring 2024
University of Houston-Clear Lake, Houston, TX 77058

## List of Topics

## Introduction/The detailed description of the system

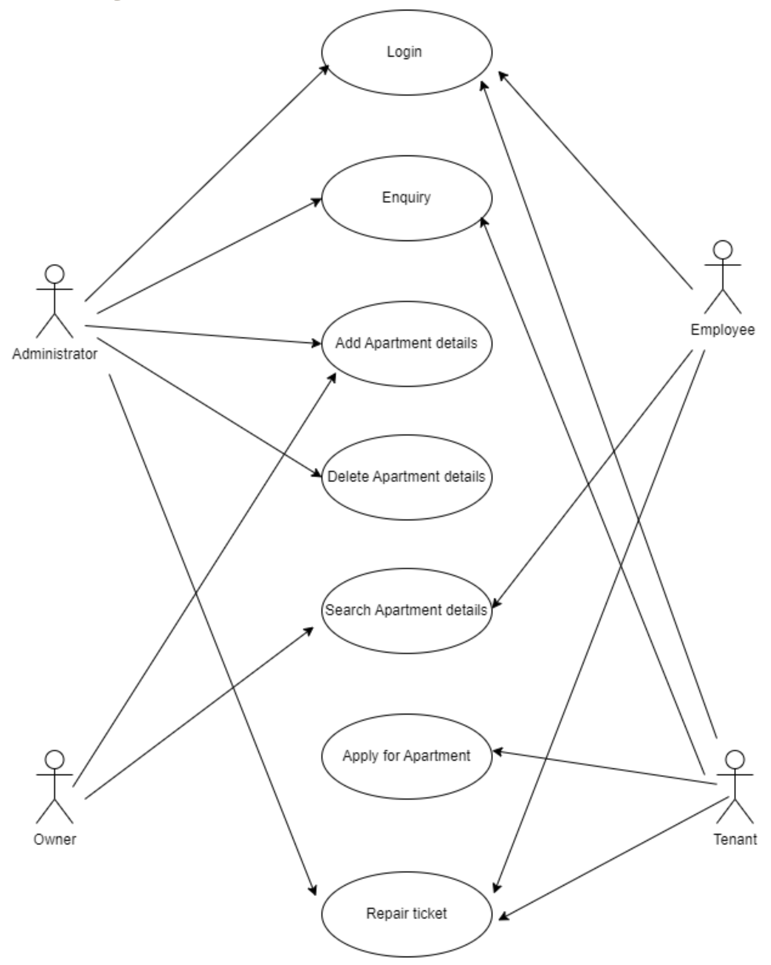
Apartment Leasing Management System (ALMS) is built to cater to the specific data requirements essential for effective apartment rental management. Below is a detailed description of the key components and functionalities of the system:

- Owners Management:
  - Identity and Contact Information: ALMS stores comprehensive details of apartment owners, including their identities and contact information.
  - Owned Apartments: The system maintains a record of apartments owned by each landlord, facilitating efficient tracking and management.
- Tenants Management:
  - Identity and Contact Details: ALMS stores personal information and contact details of tenants for communication and identification purposes.
  - Leasing Details: The system records leasing agreements, including lease durations, rental amounts, and payment schedules.
  - Age: Tenant age information is stored to ensure compliance with legal regulations.
- Apartments Management:
  - Unit Details: ALMS maintains detailed information about apartment units, including size, type (e.g., studio, one-bedroom), and location within the complex.
- Payments Management:
  - Financial Transactions: ALMS keeps a comprehensive record of all financial transactions related to rent payments, security deposits, and other fees.
- Employees Management:
  - Staff Information: The system stores details of employees involved in apartment management, including their roles, contact information, and working hours.
- Parking Management:
  - Parking Spot Details: ALMS maintains information about parking spots within the complex and their allocation to specific tenants.
- Complaints Management:
  - Tracking and Resolution: The system provides functionality for tenants to log complaints, which are then tracked and resolved by the management team.

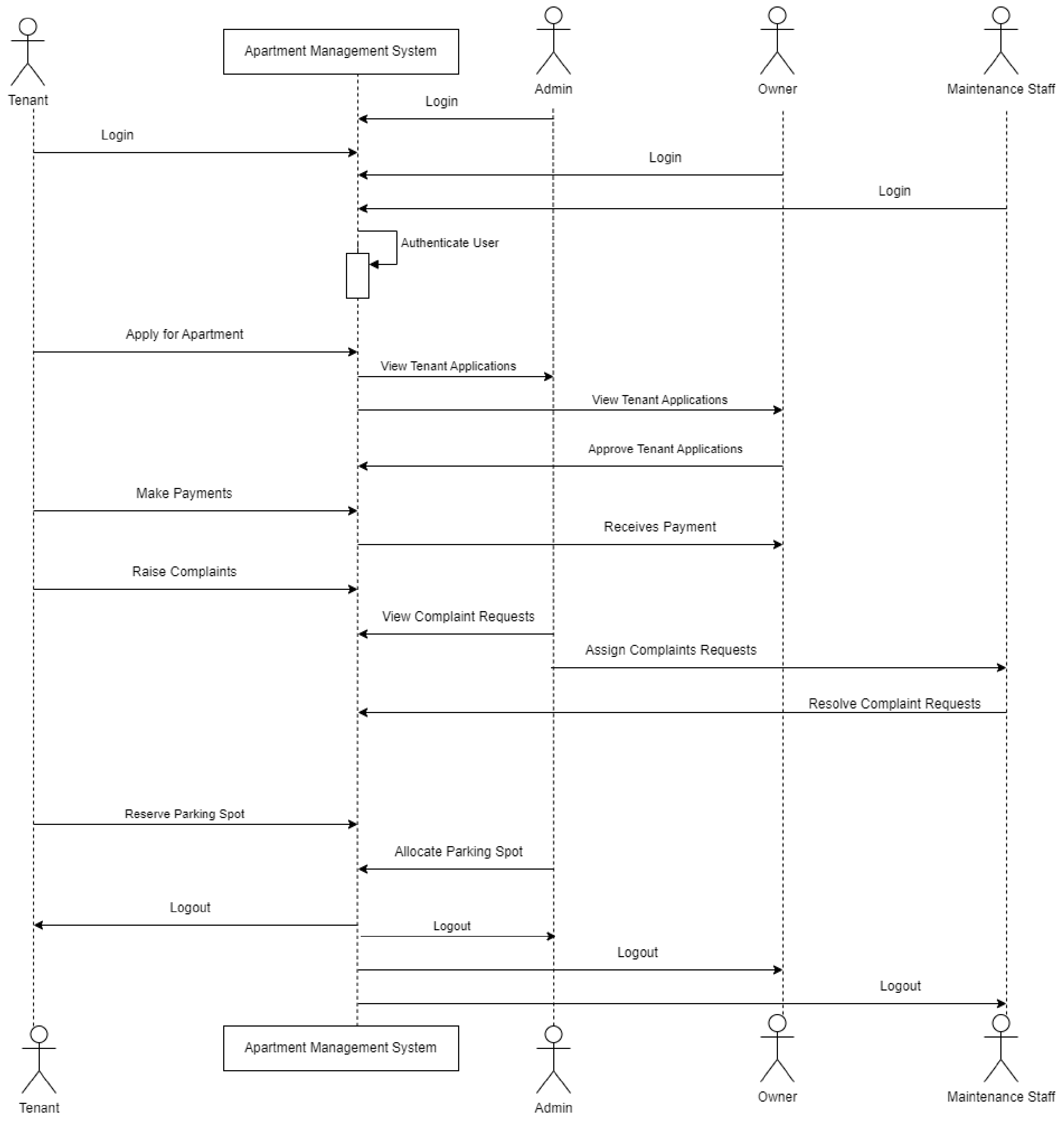
- **Apartment Applications Management:**
  - **Rental Application Handling: ALMS facilitates the management of rental applications from prospective tenants, including application status tracking and documentation storage.**
- **Blocks Management:**
  - **Organizational Structure: Information regarding different blocks within the apartment complex is stored for organizational purposes and efficient management.**
- **Access Control:**
  - **User Login Information: ALMS implements robust access control mechanisms, ensuring secure system access for both tenants and employees.**

## UML use-case/sequence/activity diagrams of the system

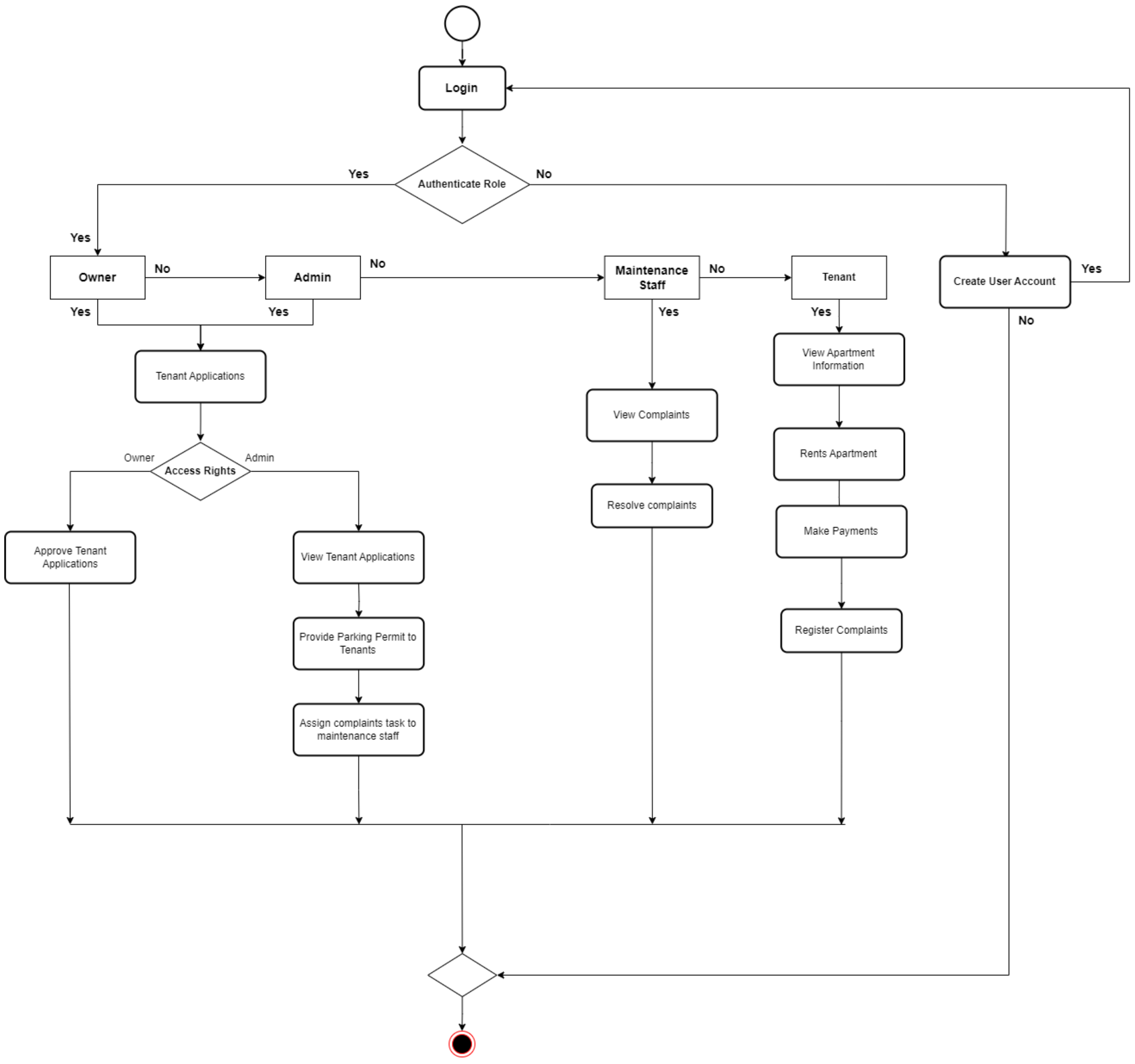
- **UML Use case diagram**



## ➢ UML Sequence diagram



## ➢ UML Activity Diagram



## Description of the overall system's data requirements

The system is designed to manage apartment rentals, which involves several key operations such as tracking apartment ownership, tenant leasing, payments, and maintenance activities. To facilitate these operations, the system requires data related to:

- Owners: Identity, contact information, and apartments owned.
- Tenants: Identity, contact details, age, and leasing details including payments.
- Apartments: Details about apartment units, including size, type, and location.
- Payments: Details of financial transactions related to rent and other fees.
- Employees: Information on staff members, including their roles and working hours.
- Parking: Details on parking spots and their allocation to tenants.
- Complaints: Tracking and resolution of tenant complaints.
- Apartment Applications: Management of rental applications from prospective tenants.
- Blocks: Information regarding the different blocks within the complex for organizational purposes.
- Access Control: User login information for secure system access by tenants and employees.

## Entity-set designation

- Owner
  - Attributes: Owner ID, Name, SSN, Address, Phone Number
  - Description: Represents the individuals or entities that own apartments.

- Apartment
  - Attributes: Apartment Number (Apt No), Bedrooms, Type, Area, Floor, Address
  - Description: Represents individual apartment units available for rent.

- **Tenant**
  - **Attributes: Tenant ID, SSN, Age, Permanent Address, Phone, Name**
  - **Description: Represents individuals who rent or are seeking to rent apartments.**

- **Apartment Employee**
  - **Attributes: Employee ID (Emp ID), Name, Phone, Shift Timings**
  - **Description: Represents employees working for the apartment complex, including administrative and maintenance staff.**

- **Apartment Application**
  - **Attributes: Name, Email, Phone Number, Apartment Number**
  - **Description: Represents applications submitted by potential tenants for renting apartments.**

- **Complaint**
  - **Attributes: Complaint ID, Complaint Description, Date**
  - **Description: Represents complaints made by tenants or other stakeholders.**

- **Block**
  - **Attributes: Block ID, Block Name, Address**
  - **Description: Represents a group or building within the apartment complex.**

- **Parking**
  - **Attributes: Spot Number, Type**
  - **Description: Represents parking spots available for tenants or owners.**

- **Login**
  - **Attributes: Email, Password**
  - **Description: Represents the credentials required for users to access the system.**

## Relationship-set designation.

- Receives (between Owner and Payment)
  - Cardinality: One-to-Many
  - Description: Indicates that an owner can receive many payments.

- Approves (between Owner and Apartment Application)
  - Cardinality: Many-to-Many
  - Description: Indicates that an owner can approve many applications and an application can be approved by many owners.

- Makes (between Tenant and Payment, and between Tenant and Complaint)
  - Cardinality: Many-to-Many
  - Description: Indicates that a tenant can make many payments and many complaints.

- Owns (between Owner and Apartment)
  - Cardinality: Many-to-Many
  - Description: Indicates that an owner can own many apartments and an apartment can have many owners.

- Rents (between Tenant and Apartment)
  - Cardinality: Many-to-Many
  - Description: Indicates that a tenant can rent many apartments and an apartment can be rented by many tenants.

- Leases (between Tenant and Parking)
  - Cardinality: One-to-Many
  - Description: Indicates that a tenant leases one or more parking spots.

- Manages (between Apartment Employee and Apartment)
  - Cardinality: Many-to-Many
  - Description: Indicates that an employee can manage many apartments and an apartment can be managed by many employees.

- ❖ Handles (between Apartment Employee and Complaint)
    - ➢ Cardinality: Many-to-Many
    - ➢ Description: Indicates that an employee can handle many complaints and a complaint can be handled by many employees.

- ❖ Consist of (between Apartment and Block)
    - ➢ Cardinality: Many-to-One
    - ➢ Description: Indicates that many apartments make up a block.

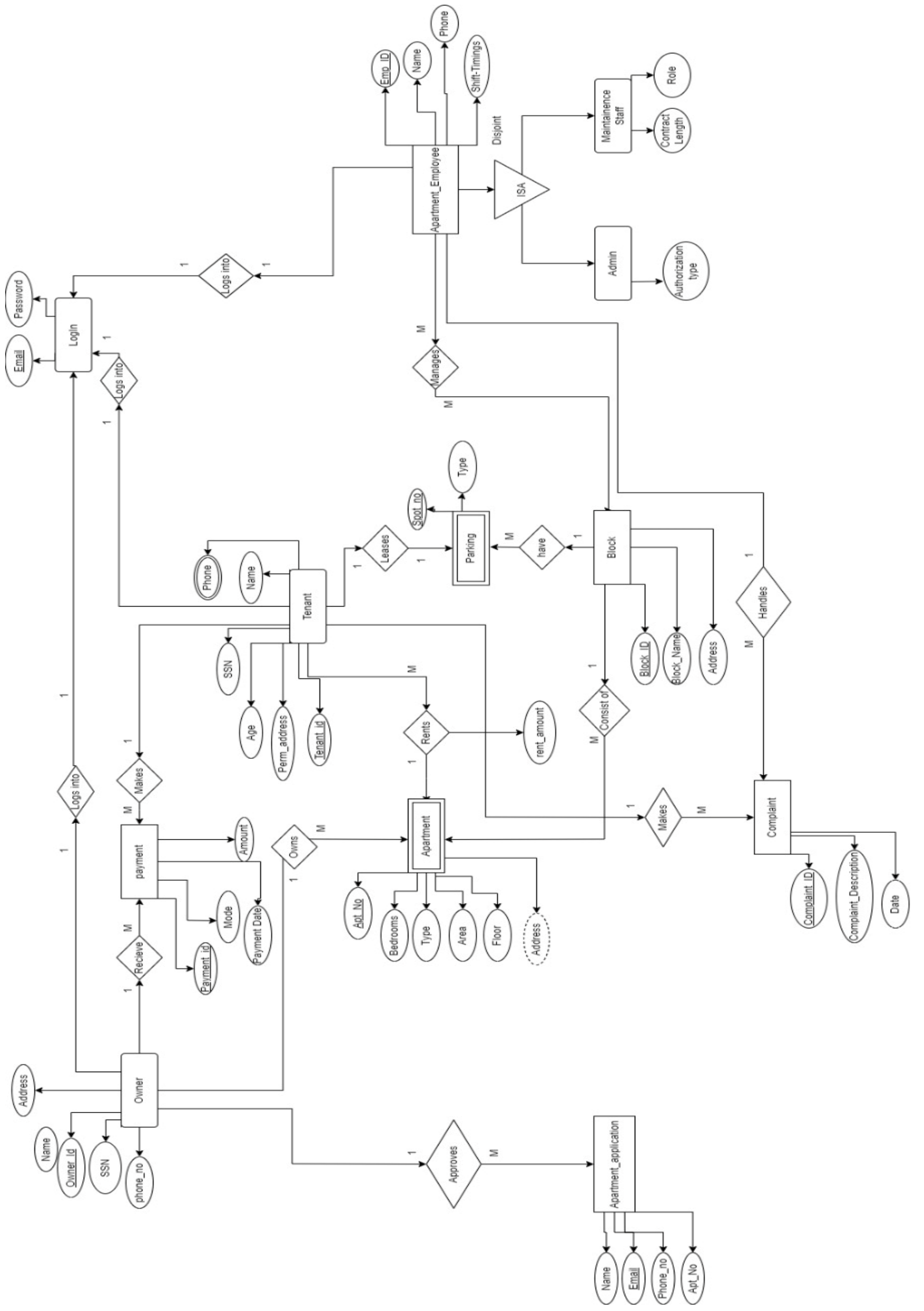
- ❖ Logs into (between Tenant, Apartment Employee and Login)
    - ➢ Cardinality: One-to-One
    - ➢ Description: Indicates that a tenant or an employee log into the system using one set of login credentials.

- ❖ Have (between Parking and Block)
    - ➢ Cardinality: Many-to-Many
    - ➢ Description: Indicates that a block can have many parking spots and a parking spot can belong to many blocks.

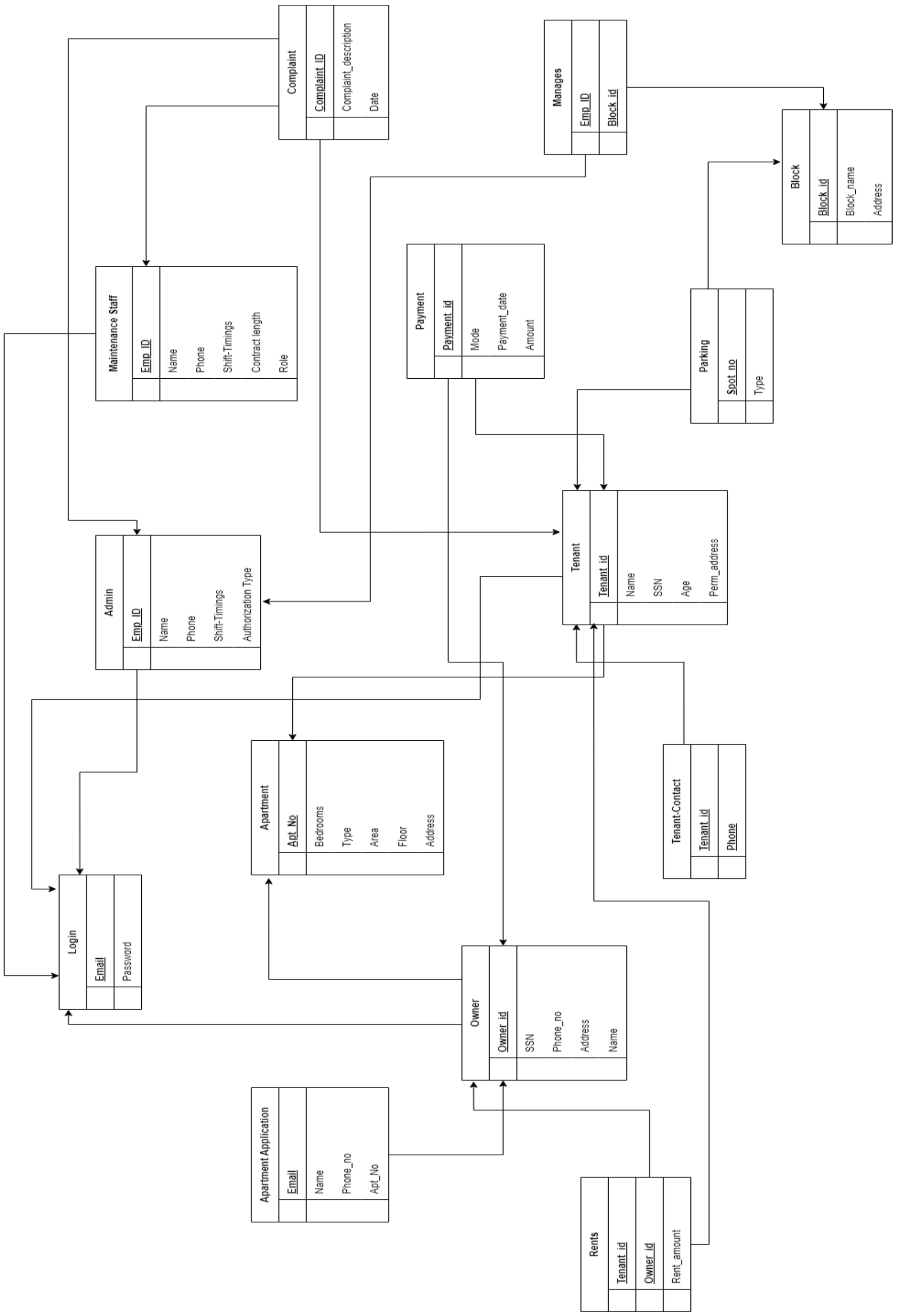
## E-R Diagram with relevant assumptions

### Assumptions

- ❖ All entities have total participation.
- ❖ Owner's phone number is a single-valued attribute since we want to avoid giving personal numbers.
- ❖ Multiple tenants can share a single apartment.
- ❖ One login per person only
- ❖ Owners can receive multiple payments for multiple apartments.
- ❖ 1 tenant can reserve 1 parking spot only.
- ❖ One apartment can accommodate multiple tenants hence we need multiple phone numbers.


Login
Email
Password
Logs into
Apartment_Employee
Emp_ID
Name
Phone
Shift-Timings
Disjoint
ISA
Maintainence Staff
Role
Contract Length
Admin
Authorization type
Manages
Parking
Spot_no
Type
have
Block
Block_ID
Block_Name
Address
Handles
Tenant
Phone
Name
SSN
Age
Perm_address
Tenant_id
Leases
Rents
rent_amount
Consist of
Makes
Complaint
Complaint_ID
Complaint_Description
Date
Apartment
Apt_No
Bedrooms
Type
Area
Floor
Address
Owns
payment
Amount
Mode
Payment Date
Payment_id
Recieve
Makes
Logs into
Owner
Address
Name
Owner_Id
SSN
phone_no
Approves
Apartment_application
Name
Email
Phone_no
Apt_No
1
M

## Reduce the ER-Diagram to tables and Eliminating Redundancy



## Relational Schema

*Note: The attributes underlined are primary keys and the ones which are bold-underlined are foreign keys.*

- ❖ Owner
  (Owner_id, SSN, Phone_no, Address, Name, **Email**)
- ❖ Payment
  (Payment_id, Mode, Payment_date, Amount, **Owner_id**, **Tenant_id**)
- ❖ Login
  (Email, Password)
- ❖ Admin
  (Emp_ID, Name, Phone, Shift-Timings, Authorization Type, **Email**)
- ❖ Maintenance Staff
  (Emp_ID, Name, Phone, Shift-Timings, Contract Length, Role, **Email**)
- ❖ Complaint
  (Complaint_ID, Complaint_description, Date, **Emp_ID**, **Tenant_id**)
- ❖ Tenant
  (Tenant_id, Age, SSN, Name, Perm_address, **Apt_No**, **Email**)
- ❖ Tenant-Contact
  (**Tenant_id**, **Phone**)
- ❖ Apartment
  (Apt_No, Bedrooms, Type, Area, Floor, Address, **Block_id**, **Owner_ID**)
- ❖ Block
  (Block_id, Block_name, Address)
- ❖ Parking
  (Spot_no, Type, **Block_id**, **Tenant_id**)
- ❖ Manages
  (**Emp_ID**, **Block_id**)

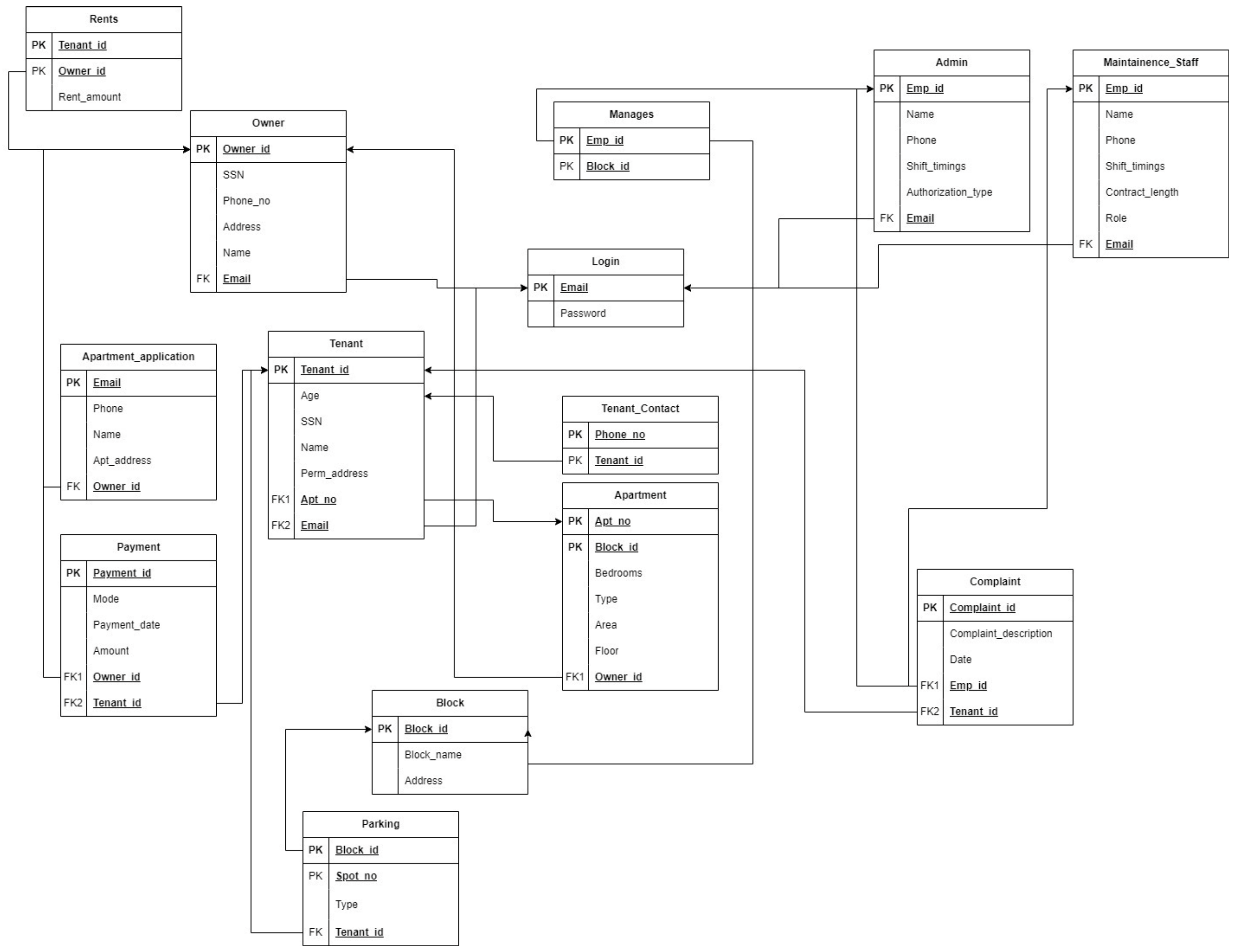
- ❖ Rents

  (Tenant_id, Owner_id, Rent_amount)

- ❖ Apartment_application

  (Email, Phone, Name, Apt_address, **Owner_id**)

## Schema Diagram



## Conclusion

This design document presents a comprehensive blueprint for the apartment rental management system. Through a series of UML diagrams, we have shown the system's workflow, capturing the dynamic interactions among users, processes, and the system itself. The ER diagram has laid out the data relationships and structures essential to the system, which were then refined into a relational schema to ensure data integrity and efficiency.

The schema diagram provides a clear visual representation of the database structure, facilitating ease of understanding and implementation. This document provides clear guidelines for the system's construction and a reference for maintaining consistency throughout the development process.

As we move forward, the focus will shift towards implementing the design, ensuring that the system meets both functional and non-functional requirements. Regular reviews and updates to the design may be necessary to adapt to evolving user needs and technological advancements.

******