Requirements:

I want to create a real estate management web application where a real estate company can take requests for development from a landlord to manage rental sale share and flat rent space for commercial and rent space for residential, an admin can accept requests from web visitors from the website by choosing a property and admin can see the request and after analyzing document can create an appointment or reject the request. after the appointment, they can make rentals, develop, or other issues. or cancel the process. if they agree they can make an agreement and store the agreement on the database. also, an admin takes requests from an engineer to requirements for developing part workers and materials and the admin can manage them from a vendor or labor market. engineer can give updates to their work and investors can watch the workflow through web applications. investor can receive their property through agreement requirements and the admin can store delivery paper on the database and update the work status. Also, admin accounts can manage vendors, labor, staff salaries, store vouchers, and many more accounts part. admin and the property owner can accept a request from the rental to leave or disclose the agreement and leave the property. and also property owners can post their for rent advertisements. rental can request any issue to solve like maintenance for stay and they can update the agreement. also set a minimum notice time to disclose the agreement from the rental or property owner.

now give me a models of this web application with full process. from tender drop to complete project and help to give rents and manage full accounts and staff

Probably solutions Models:

Here's an outline of the models and processes for the real estate management web application:

User Authentication and Authorization:

* Users can register and login as landlords, tenants, engineers, investors, or administrators.
* Authentication and authorization mechanisms are implemented to ensure secure access to different features based on user roles.

Property Management:

* Landlords can submit requests for property management, specifying details such as type (rental, sale, share), location, size, and amenities.
* Administrators review and approve/reject property management requests.
* Approved properties are listed on the website for visitors to view and make inquiries.

Appointment Management:

* Administrators schedule appointments for property viewings or meetings with landlords, tenants, or engineers.
* Appointment details, including date, time, location, and participants, are stored in the database.

Agreement Management:

* After property viewings or meetings, administrators can create rental agreements or development agreements based on discussions and negotiations.
* Agreements are digitally signed by relevant parties and stored securely in the database.

Development Management:

* Engineers submit requests for development projects, specifying requirements for materials, labor, and timelines.
* Administrators review and approve/reject development requests, manage vendors and labor, and track project progress.

Workflow Monitoring:

* Investors can monitor the progress of development projects through the web application, viewing updates and milestones provided by engineers.

Account Management:

* Administrators manage accounts, including vendor payments, staff salaries, store vouchers, and other financial transactions.
* Comprehensive reporting and analytics tools are available to track expenses, revenues, and profitability.

Rental Management:

* Tenants can request maintenance or report issues through the web application, which are logged and addressed by administrators or property managers.
* Rental agreements can be updated as needed, with provisions for minimum notice periods for termination or disclosure.

Advertisement Management:

* Landlords can post advertisements for available properties for rent or sale on the website, including descriptions, photos, and contact information.
* Visitors can search and filter properties based on criteria such as location, type, size, and amenities.

Notification and Communication:

* The system sends notifications and reminders to users for appointments, agreements, maintenance requests, and other important events.
* Built-in communication features allow users to interact with administrators, property managers, tenants, and other stakeholders.

By implementing these models and processes, the real estate management web application can effectively handle property management, development projects, rental management, accounts management, and communication among stakeholders.

Use case:

+---------------------------------------+

| Real Estate System |

+---------------------------------------+

| |

| Administrator |

| |

+---------------------------------------+

| |

| |

+--------|------------------|-----------+

| v v |

| Property Management Agreement |

| Management |

+---------------------------------------+

| |

| Development |

| Management |

+---------------------------------------+

| |

| |

+--------|------------------|-----------+

| v v |

| Rental Management Advertisement |

| |

+---------------------------------------+

Explanation:

**Administrator:** This actor represents the system administrator who has access to all functionalities of the system.

**Property Management:** This use case includes functionalities related to managing properties such as submitting requests, scheduling appointments, and approving/rejecting requests.

Agreement Management: This use case involves creating, signing, and storing rental or development agreements.

**Development Management:** This use case covers the management of development projects, including submitting requests, approving/rejecting requests, and tracking progress.

**Rental Management:** This use case includes functionalities for managing rental properties, such as handling maintenance requests, updating rental agreements, and dealing with tenant issues.

**Advertisement:** This use case allows landlords to post advertisements for available properties on the website.

These use cases represent the main functionalities of the real estate management system and illustrate the interactions between different actors and system components.

Relational Database table:

CREATE TABLE Properties (

property\_id INT AUTO\_INCREMENT PRIMARY KEY,

type VARCHAR(100) NOT NULL,

location VARCHAR(255) NOT NULL,

size DECIMAL(10, 2) NOT NULL,

amenities TEXT,

status ENUM('Available', 'Occupied', 'Under Maintenance') NOT NULL

);

CREATE TABLE RentalAgreements (

agreement\_id INT AUTO\_INCREMENT PRIMARY KEY,

property\_id INT NOT NULL,

tenant\_id INT NOT NULL,

terms TEXT,

duration INT NOT NULL,

rent DECIMAL(10, 2) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

FOREIGN KEY (property\_id) REFERENCES Properties(property\_id),

FOREIGN KEY (tenant\_id) REFERENCES Tenants(tenant\_id)

);

CREATE TABLE MaintenanceRequests (

request\_id INT AUTO\_INCREMENT PRIMARY KEY,

property\_id INT NOT NULL,

description TEXT NOT NULL,

status ENUM('Pending', 'In Progress', 'Completed') NOT NULL,

request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

completion\_date TIMESTAMP,

FOREIGN KEY (property\_id) REFERENCES Properties(property\_id)

);

CREATE TABLE Advertisements (

advertisement\_id INT AUTO\_INCREMENT PRIMARY KEY,

property\_id INT NOT NULL,

advertisement\_content TEXT NOT NULL,

publish\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expiration\_date TIMESTAMP,

FOREIGN KEY (property\_id) REFERENCES Properties(property\_id)

);

CREATE TABLE Tenants (

tenant\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL

);

CREATE TABLE Administrators (

admin\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE Engineers (

engineer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL

);

CREATE TABLE Requests (

request\_id INT AUTO\_INCREMENT PRIMARY KEY,

engineer\_id INT NOT NULL,

description TEXT NOT NULL,

status ENUM('Pending', 'Approved', 'Rejected') NOT NULL,

request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

completion\_date TIMESTAMP,

FOREIGN KEY (engineer\_id) REFERENCES Engineers(engineer\_id)

);

CREATE TABLE Agreements (

agreement\_id INT AUTO\_INCREMENT PRIMARY KEY,

request\_id INT NOT NULL,

agreement\_content TEXT NOT NULL,

creation\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (request\_id) REFERENCES Requests(request\_id)

);

CREATE TABLE Vendors (

vendor\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL

);

CREATE TABLE Laborers (

laborer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL

);

CREATE TABLE Staff (

staff\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

address TEXT NOT NULL,

position VARCHAR(100) NOT NULL,

salary DECIMAL(10, 2) NOT NULL

);

CREATE TABLE Accounts (

account\_id INT AUTO\_INCREMENT PRIMARY KEY,

account\_name VARCHAR(255) NOT NULL,

account\_type ENUM('Income', 'Expense', 'Asset', 'Liability') NOT NULL,

balance DECIMAL(10, 2) NOT NULL

);

CREATE TABLE Transactions (

transaction\_id INT AUTO\_INCREMENT PRIMARY KEY,

account\_id INT NOT NULL,

transaction\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

description TEXT,

amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (account\_id) REFERENCES Accounts(account\_id)

);

This schema includes tables for managing properties, rental agreements, maintenance requests, advertisements, tenants, administrators, engineers, requests, agreements, vendors, laborers, and staff. It establishes relationships between tables using foreign keys where necessary for maintaining data integrity.

With this updated schema, the Accounts table still manages the financial accounts, and the Transactions table tracks individual transactions associated with these accounts. You can use the account\_type field to categorize accounts (e.g., income, expense, etc.), and the balance field to maintain the current balance of each account.

When a financial transaction occurs (e.g., vendor payment, staff salary, etc.), a new record is inserted into the Transactions table with details such as the account ID, transaction date, description, and amount. This allows for detailed tracking of all financial activities within the system.