**Design a Database Schema for LivSmart Project**

**Secretary is the administrator of our system**

1. **Users Table:**

This table stores information about different types of users (residents, managers, secretaries, security guards).

| **Field** | **Type** |
| --- | --- |
| Unit Code | VARCHAR(PK) |
| Full Name | VARCHAR |
| E-mail | VARCHAR |
| Password | VARCHAR |
| Phone Number | VARCHAR |
| Role(Resident, manager, secretary, security guard) | VARCHAR |

1. **Resident Table**

**Relationship:** One to many relationship with the Resident Document Table

| **Field** | **Type** |
| --- | --- |
| Unit Code | VARCHAR(FK) |
| Resident ID | INT(PK) |
| Profession | VARCHAR |
| Family Members of Residents | INT |
| Document ID | INT(FK) |
| Image | BLOB(Binary Large Object) |
| NID Number/Birth Certificate Number | LONG INT |

1. **Management Table**

This table stores the manager and security guard information.

| **Field** | **Type** |
| --- | --- |
| Unit Code | VARCHAR(FK) |
| Management ID | INT(PK) |
| Management Role(manager, security guard) | VARCHAR |
| NID Number/Birth Certificate Number | LONG INT |
| Image | BLOB(Binary Large Object) |
| Registration Date | TIMESTAMP |

**4. ResidentDocument Table**

| **Field** | **Type** |
| --- | --- |
| Document ID | INT(PK) |
| Document Type( NID, Driving License, Birth Certificate) | VARCHAR |
| Attachment |  |

1. **Service Requests Table**

This table manages service requests submitted by residents.

| **Field** | **Type** |
| --- | --- |
| Request ID/service id | INT(PK) |
| Unit Code | VARCHAR(FK) |
| Service Type(Plumbing, electrical) | VARCHAR |
| Description | TEXT |
| Request Date | TIMESTAMP |
| Status(Complete, Pending) | ENUM |
| Service Provider ID(ID of the service provider) | INT(FK) |

1. **Service Payment Table**

This table manages service requests submitted by residents.

| **Field** | **Type** |
| --- | --- |
| Payment ID | INT(PK) |
| Service ID | VARCHAR(FK) |
| Amount | VARCHAR |
| Payment Status | ENUM |
| Payment Due Date | DATE |
| Payment Method |  |

1. **Event Registration Table**

This table tracks parking registrations and allocations for residents.

| **Field** | **Type** |
| --- | --- |
| Event Registration ID | INT(PK) |
| Unite Code | INT(FK) |
| Registration Date | TIMESTAMP |
| Status(pending, completed) | ENUM |
| Payment ID | INT (FK) |

1. **Events Table**

This table stores information about community events.

| **Field** | **Type** |
| --- | --- |
| Event ID | INT(PK) |
| Event Type | VARCHAR |
| Event Name | VARCHAR |
| Event Date | TIMESTAMP |
| Location | VARCHAR |
| Created By(ID of the secretary or manager who created the event) | INT(FK) |
| Participants Number | INT |
| Description | TEXT |
| Event Status | ENUM |

1. **Parking Registration Table**

This table tracks parking registrations and allocations for residents.

| **Field** | **Type** |
| --- | --- |
| Parking Request ID | INT(PK) |
| Unite Code | INT(FK) |
| Vehicle ID | VARCHAR |
| Types of Vehicle(Bike, Car) | VARCHAR |
| Vehicle Registration No | VARCHAR |
| Registration Date | TIMESTAMP |
| Status(pending, completed) | ENUM |

1. **Parking Slot Table**

This table tracks parking slot allocations for residents.

| **Field** | **Type** |
| --- | --- |
| Slot ID | INT(PK) |
| Status | ENUM |
| Vehicle ID | VARCHAR(FK) |

1. **Complaints Table**

This table stores complaints submitted by residents.

| **Filed** | **Type** |
| --- | --- |
| Complaint ID | INT(PK) |
| Unit Code | INT(FK) |
| Complaint Type(Maintenance, noise, security, other, parking) | VARCHAR |
| Complain description | TEXT |
| Submission Date | TIMESTAMP |
| Status(Pending, In Progress, Resolved, Closed) | ENUM |

1. **Feedback Table**

This table stores complaints submitted by residents.

| **Field** | **Type** |
| --- | --- |
| Feedback ID | INT(PK) |
| Unit Code | INT(FK) |
| Feedback Description | TEXT |

1. **Security Log Table**

This table tracks visitor logs and security events.

| **Field** | Type |
| --- | --- |
| Log ID | INT(PK) |
| Unit Code | VARCHAR(FK) |
| Visitor Type | VARCHAR |
| Visitor Name | VARCHAR |
| Entry Date | DATE |
| Exit Date | DATE |
| Entry Time | TIMESTAMP |
| Exit Time | TIMESTAMP |

1. **Amenity Booking Table**

This table stores the booking information and details.

| **Field** | Type |
| --- | --- |
| Booking ID | INT(PK) |
| Unit Code | INT(FK) |
| Booking Date | DATE |
| Booking Time | TIMESTAMP |
| Status(Pending, Completed, Canceled) | ENUM |

1. **CommunityTrade Table**

This table stores the selling and buying information

| **Field** | **Type** |
| --- | --- |
| Buyer ID(User unit code who sells the item) | INT(FK) |
| Item Name | VARCHAR |
| Item Description | TEXT |
| Item Price | VARCHAR |
| Trade Type(sale, trade, free) | ENUM |
| Posting Time | TIMESTAMP |
| Posting Date | DATE |
| Status(active, sold out) | ENUM |
| Category(electronics, furniture,clothing etc.) | VARCHAR |
| Buyer ID (User unit code who purchase the item) | INT(FK) |
| Transaction Date | TIMESTAMP |

1. **Announcement Table**

This table stores community-wide announcements such as important updates, event notifications, or system messages that are displayed to users.

| **Field** | **Type** |
| --- | --- |
| Announcement ID | INT(PK) |
| Created By(ID of the secretary or manager who created the event) | INT(FK) |
| Title | VARCHAR |
| Description | TEXT |
| Announcement Date | TIMESTAMP |

1. **Directory**

This table stores the members information

| **Field** | **Type** |
| --- | --- |
| User ID | INT(PK) |
| Member Name | VARCHAR |

1. **Notifications Table**

This table manages notifications sent to users.

| **Field** | **Type** |
| --- | --- |
| Notification ID | INT(PK) |
| Unit Code | INT(FK) |
| Notification Message | TEXT |
| Sent Date | TIMESTAMP |
| Notification Type | ENUM |
| Priority(emergency etc) | VARCHAR |
| Status(Read, Unread, Archive) | ENUM |

1. **Service Provider Table**

This table stores the service provider information. One to many relationships between Service request table.

| **Field** | **Type** |
| --- | --- |
| Service Provider ID | INT(PK) |
| Contact Number | VARCHAR |
| Provider Name | VARCHAR |
| Service Type | VARCHAR |

* **User -> Resident (1-to-1)**:

A user is a resident

1. **User -> Management (1-to-1)**:

A user holds a management role

1. **User -> SecurityLog (1-to-many)**:

A user has security logs

1. **User -> Notification (1-to-many)**:

A user receives notifications

1. **ServiceRequest -> ServiceProvider (Many-to-Many)**:

A service request is handled by many service providers, and a service provider handles many service requests

1. **ServiceRequest -> Resident (Many-to-Many)**:

Many service requests are submitted by a resident

1. **EventRegistration -> Event (Many-to-Many)**:

Many event registrations are linked to an event

1. **EventRegistration -> ServicePayment (1-to-1)**:

An event registration has a payment

1. **ServicePayment -> ServiceRequest (1-to-1)**:

A payment is made for a service request

1. **Feedback -> Resident (Many-to-1)**:

Many feedbacks are given for an event

1. **Resident -> ParkingRegistration (1-to-Many)**:

Many parking registrations are for a parking slot

1. **Resident—>Booking (1-to-many)**:

Many bookings are made by a resident

1. **CommunityTrade -> Resident (Many-to-1)**:

Many trades are initiated by a resident

1. **Complaint -> Resident (Many-to-1)**:

Many complaints are made by a resident

1. **SecurityLog -> User (Many-to-1)**:

Many security logs are for a user