User stories

- As an administrator, I want to manage apartment details, so that I can keep all apartment and block information up to date.
- As a resident, I want to make payments for administration services online, so that I can manage my expenses conveniently.
- As a resident, I want to view my payment history, so that I can track my past payments and ensure all dues are cleared.
- As an administrator, I want to track pending payments, so that I can remind residents of overdue amounts and manage the complex's finances effectively.
- As a resident, I want to update my personal information (such as phone number or email), so that I can receive relevant communications from the apartment complex.
- As an administrator, I want to manage and track reservations for common spaces, so that I can prevent double bookings and keep things organized.
- As a security guard, I want to have a list of residents and their apartment numbers, so that I can control who enters the complex.
- As a resident, I want to receive confirmation for my space reservation, so that I am assured of my booking.
- As an administrator, I want to generate reports on payments and space reservations, so that I can analyze trends and make informed decisions.
- As a resident, I want to leave feedback or complaints about services, so that my concerns are heard and addressed.
- As a resident, I want to schedule moving dates and times within the app, so that I can inform the management and avoid conflicts with other residents moving in/out.
- As a resident, I want to receive reminders for my space reservations, so that I don’t forget my booking and miss out on using common spaces.
- As a resident, I want to be able to report noisy neighbors or other disturbances, so that the administration can take appropriate action.
- As an administrator, I want to manage resident permissions for common space reservations, so that certain residents (like those with overdue payments) are restricted from booking.
- As a resident, I want access to a community bulletin board, so I can view announcements, events and classifieds within the apartment complex.

Technical and design Considerations/decisions

Elimination of the Administrator Entity

During the database design process, we initially considered including an Administrator entity. However, after evaluating the specific needs and the prepared modules, it became evident that the Administrator entity did not fulfill any essential function. The existing modules for managing payments, reservations, and resident information could be handled without a distinct Administrator entity. For instance, system roles and permissions can be defined without requiring a separate table or entity. The role-based structure already covered by the Role entity provides sufficient flexibility to assign administrative permissions where needed.
Therefore, the Administrator entity was eliminated from the final design, simplifying the database structure and avoiding unnecessary complexity. By focusing only on essential entities, we ensure that the database is optimized for the specific needs of the apartment management system.

Scalability and Future Expansion
The current design is scalable, allowing for the easy addition of future modules and functionalities. For example, additional modules for handling Maintenance Requests, Visitor Management, or Community Events can be seamlessly integrated into the existing database. The modular structure of the database and its well-defined relationships ensure that future expansion will not require significant changes to the core schema. This approach ensures the system can evolve to meet future needs without requiring a complete overhaul.
By building the system with scalability in mind, we ensure that it can accommodate future growth and more complex functionalities as the apartment management system expands.

Step 1: Define components

- Payments
- Blocks and apartments
- Services (moving, common spaces)

Step 2: Define entities

E1. Apartments
E2. Payment
E3. Resident
E4. Reservation

E5. Moving
E6. Space
E7. Notification
E8. Block
E9. Role
E10. Administrator

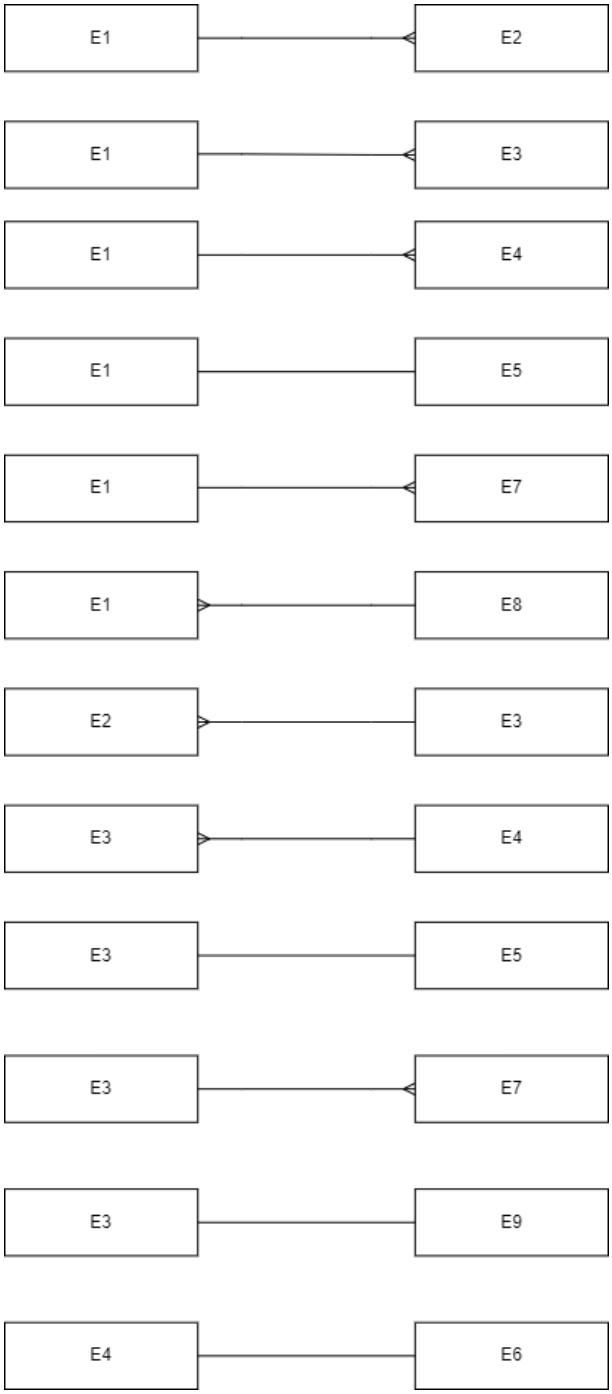
Step 3: Define Attributes per Entity

E1. Apartments: Id, block id, size, resident id.
E2. Payment: Id, Resident id, Apartment id, date, amount, status.
E3. Resident: id, name, phone, email, apartment id, role id.
E4. Reservation: id, resident id, space id, date, start time, end time, status, apartment id.
E5. Moving: id, resident id, apartment id, date, start time, end time.
E6. Space: id, name, capacity, available hours.
E7. Notification: id, resident id, type, date, apartment id.
E8. Block: id, number, number apartments.
E9. Role: id, type.
E10. Administrator: id, name, phone, email

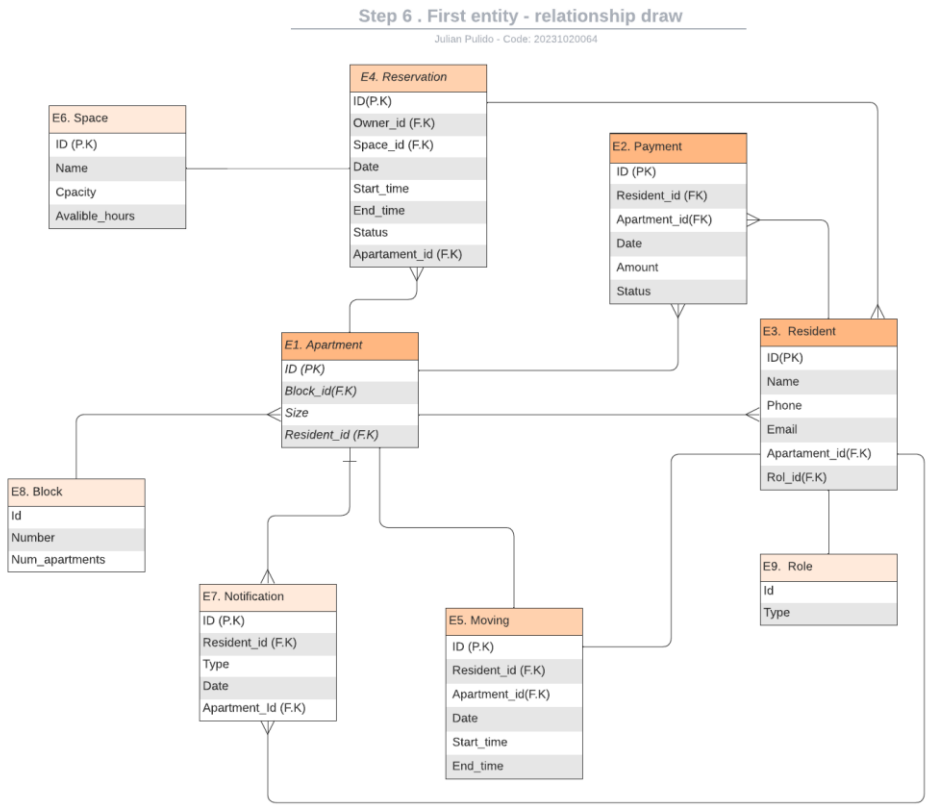
Step 4: Define relationships

xxxxxx	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
E1	XXXXX									
E2		XXXXX								
E3			XXXXX							
E4				XXXXX						
E5					XXXXX					
E6						XXXXX				
E7							XXXXX			
E8								XXXXX		
E9									XXXX	
E10										XXXXX

Step 5: Define relationship types


E1
E2
E1
E3
E1
E4
E1
E5
E1
E7
E1
E8
E2
E3
E3
E4
E3
E5
E3
E7
E3
E9
E4
E6

Step 6: First Entity-Relationship Draw



Step 7 and step 8 are not performed because in the previous steps there were no many-to-many relationships.

Step 9: Get Data-Structure E-R M

E1. Apartment

- ID: INT (PK) – Integer, primary key
- Block_id: INT (FK) – Integer, foreign key referencing the Block entity
- Size: FLOAT or DECIMAL – Represents the apartment size (e.g., square meters)
- Resident_id: INT (FK) – Integer, foreign key referencing the Resident entity

E2. Payment

- ID: Integer, primary key
- Resident_id: Integer, foreign key referencing the Resident entity
- Apartment_id: Integer, foreign key referencing the Apartment entity
- Date: Date, Represents the payment date
- Amount: Float, Represents the payment amount (monetary value)

- Status: String, Represents the payment status (e.g., "Paid", "Pending")

E3. Resident

- ID: Integer, primary key
- Name: String, Resident's name
- Phone: String, Resident's phone number
- Email: String, Resident's email
- Apartment_id: Integer, foreign key referencing the Apartment entity
- Role_id: Integer, foreign key referencing the Role entity

E4. Reservation

- ID: Integer, primary key
- Resident_id: Integer, foreign key referencing the Resident entity
- Space_id: Integer, foreign key referencing the Space entity
- Date: Reservation date
- Start_time: time, Reservation start time
- End_time: time, Reservation end time
- Status: String, Reservation status (e.g., "Confirmed", "Pending")
- Apartment_id: Integer, foreign key referencing the Apartment entity

E5. Moving

- ID: Integer, primary key
- Resident_id: Integer, foreign key referencing the Resident entity
- Apartment_id: Integer, foreign key referencing the Apartment entity
- Date: Moving date
- Start_time: time, Moving start time
- End_time: time, Moving end time

E6. Space

- ID: Integer, primary key
- Name: String, Name of the common space (e.g., "Gym", "Pool")
- Capacity: integer, Capacity of the space
- Available_hours: String, Available hours for the space

E7. Notification

- ID: Integer, primary key
- Resident_id: Integer, foreign key referencing the Resident entity
- Type: String, Notification type (e.g., "Payment Reminder", "Event")
- Date: Date the notification was sent
- Apartment_id: Integer, foreign key referencing the Apartment entity

E8. Block

- ID: Integer, primary key
- Number: Integer, Block number
- Num_apartments: Integer, Number of apartments in the block

E9. Role

ID: Integer, primary key

Type: String, Role type (e.g., "Admin", "Resident")

Step 10: Get Data-Structure E-R M