Project Name: Realty Management System
Course Name: DAMG 6210 Data Management and Database Design
Year: Spring 2023

Team Members:

Name	Contact Information
Mani Deepak Reddy Aila	aila.m@northeastern.edu
Pranitha Poosa	poosa.p@northeastern.edu
Makarand Rajaram Zende	zende.m@northeastern.edu
Siddhi Sudesh Sawant	sawant.sid@northeastern.edu
Ajin Abraham	lnu.aj@northeastern.edu

Database Purpose:

The rental property management system aims to simplify the process of managing rental properties for landlords and property owners. It offers a centralized and effective platform for tracking tenant details, rental payments, maintenance requests, and other property-related tasks. This system allows property owners to effortlessly oversee their rental properties, monitor income and expenses, and make informed decisions to enhance their property portfolio. Additionally, the system provides tenants with a convenient payment and billing experience, making rent payments and maintenance requests quick and straightforward. In summary, the rental property management system intends to provide a stress-free and efficient approach to rental property management.

Business Problems Addressed:

Through this Rental Property Management System, we plan to address the following business problems:

- Allow rental property managers to efficiently manage and maintain their apartments, units, and tenant relationships in a single centralized system.
- Provide insights and analytics to property managers to improve their property operations, such as identifying popular unit types, preferred locations from customer preferences for targeted marketing.
- Enable property managers to efficiently handle tenant information and track lease agreements, payments, and maintenance requests.
- Allow customers to easily search and compare available rental properties based on their preferences and requirements, making a seamless experience for their rental property search.
- Enable property managers to collect and analyze tenant feedback to improve their properties and services, ultimately enhancing the overall tenant experience.

Business Rules:

- Each Apartment has one or more ApartmentUnits.
- Each Apartment has an ApartmentAddress.
- Each Apartment has an ApartmentType.
- Each Apartment is owned by a Landlord.
- Each ApartmentUnit has an UnitType.
- Each ApartmentUnit has a UnitAvailability.
- Each ApartmentUnit may have zero or more Maintenance Requests.
- Each ApartmentUnit may receive zero or more Payments.
- Each ApartmentUnit may have zero or more Leases.
- Each Tenant may provide zero or many Feedbacks.
- Each Employee may handle one or more Apartments.
- Each Employee may assist zero or more customers.
- Each Customer has one or more CustomerPreferences.

Design Rules:

- Use Crow's Foot Notation.
- Specify the primary key fields in each table by specifying PK beside the fields.
- Draw a line between the fields of each table to show the relationships between each table. This
  line should be pointed directly to the fields in each table that are used to form the relationship.
- Specify which table is on the one side of the relationship by placing a one next to the field where
  the line starts.
- Specify which table is on the many side of the relationship by placing a crow's feet symbol next to the field where the line ends.

Design Decisions:

Entity Name	Why Entity Included	How Entity is related to other Entities
Apartment	The primary function of a database is to gather information about an apartment, including its name, address, and the number of units. It also store details about the apartment's type and the landlords who lease them, as well as information about the available units in the apartment.	As the core entity in the database, the Apartment entity has many-to-one relation to Employee and Landlord, one-to-many relation to ApartmentUnit and one-to-one relation with ApartmentAddress. This entity helps us to know the details of landlords i.e owners of the property, find the details of the apartment like location, zip code etc., and the information of the units present in that particular apartment.
ApartmentUnit	Another key purpose of the database is to maintain information about different units present in each of the apartments so that it is	This entity has a many-to-one relationship with Apartment because each apartment can have multiple units. It has

	easy to obtain each of the units' details. This entity will have unit specific details like unit number, rent amount and the apartment it belongs to.	one-to-many relation with Payment, UnitAvailability, Lease,Maintenance Requests. It has an identifying relationship with Lease.
ApartmentAddress	The entire location specific data related to a specific apartment such as street address, city, state and zip code are stored in ApartmentAddress.	This entity has a one-to-one relationship with the Apartment entity since each Apartment has a unique address.
UnitAvailability	All the details regarding a specific unit such as availability, and the exact dates for which an Unit is available is stored under UnitAvailability.	The ApartmentUnit has a one-to-many relationship with UnitAvailability as it keeps track of the availability of an Apartment.
UnitType	The rental management system holds data of the Unit types in an apartment.	The UnitType has a one-to-many relationship with CustomerPreferences entity. It has an identifying relationship with CustomerPreferences.
Lease	The Lease object contains the information about the lease agreement for specific unit and includes various other attributes such as duration of lease term , number of people on lease and the security deposit details.	The Tenant is related to ApartmentUnit through an associate entity, Lease, as they have many-to-many relationship between them. The Lease entity is related to the Tenant entity as it links a Tenant and ApartmentUnit.
Tenant	This entity contains details of the individual who has rented an ApartmentUnit in an Apartment.	The Tenant entity has a one-to-many relationship with the Lease entity and Feedback entity. It has an identifying relationship with the Lease entity.
Feedback	The tenant's experience about the apartment unit is	The Feedback entity relates to Tenant in

	maintained with the date and tenant details in this entity.	one-to-many relation to track a specific tenant's feedback.
Payment	The Payment entity stores all the transactional details of a tenant including payment date and amount per payment.	The Payment entity is having many-to-one relation with the ApartmentUnit, as payment is done for that particular unit.
MaintenanceRequest	The MaintenanceRequest entity keeps a track of all the maintenance requests by a specific tenant such as request description, request date and the current status.	The MaintenanceRequest entity is having many-to-one relation with apartment unit, as request id raised by a unit in the apartment.
Employee	The Employee entity stores all the basic details of the employees such as name and contact info.	The employee entity is related to the one-to-many customers, along with this employee is having one-to-many relation to keep track of the apartments.
Landlord	The Landlord entity stores all the basic details of the employees such as name and contact info.	The Landlord entity is related to the apartment entity having one-to-many relationship, as the landlord can own one or more apartments.
Customer	The Customer entity stores all the basic details of the customers such as name and contact info including the employee who's supporting a customer.	The Customer entity is related to the employee entity having many-to-one relationship, as it links a customer and its associated employee also customer has one-to-many relation with customer preference which is identifying relationship.
CustomerPreferences	The CustomerPreferences entity stores all the	The CustomerPreferences entity is an associate entity

	preferences of a particular customer such as preferred unit type, minimum rent, maximum rent and locality.	between Customer and UnitType entities as they have a many-to-many relationship. A Customer can have multiple UnitTypes as preference and a UnitType can be preferred by multiple Customers.

 **Design Decisions:**

1. **Apartment:**
    - ApartmentID (PK): Unique identifier for the apartment
    - ApartmentName: Name of the apartment
    - LandlordID (FK1): ID of the landlord associated with the apartment
    - EmployeeID (FK2): ID of the employee associated with the apartment
    - AddressID(FK3): ID of the address associated with the apartment
2. **ApartmentUnit:**
    - UnitID (PK): Unique identifier for the Apartment unit
    - ApartmentID (FK1): ID of the apartment associated with the unit
    - UnitTypeID (FK2): ID of the UnitType Entity
    - UnitNumber: Unit number
    - RentAmount: Rent amount for the unit
3. **ApartmentAddress**:
    - AddressID (PK): Unique identifier for the address
    - StreetAddress: Street address of the property
    - City: City of the property
    - State: State of the property
    - ZipCode: Zip code of the property
4. **UnitType:**
    - UnitTypeID (PK): Unique identifier for the unit type
    - UnitTypeName: Type of unit (e.g., "studio", "1-bedroom", "2-bedroom")
    - UnitTypeDescription: Description for the Unit Type
5. **UnitAvailability:**
    - UnitAvailabilityID (PK): Unique Identifier for the UnitAvailability
    - UnitID (FK): ID of Unit associated

- UnitAvailabilityStartDate: Start Date for the availability of Unit
- UnitAvailabilityEndDate: End Date for the availability of Unit
- isAvailable: Whether the Unit is available for rent

6. **Tenant:**
    - TenantID (PK): Unique identifier for the tenant
    - TenantFirstName: FirstName of the tenant
    - TenantLastName: LastName of the tenant
    - EmailAddress: Email Address of the tenant
    - PhoneNumber: Phone Number of the tenant
7. **Maintenance Request:**
    - MaintenanceRequestID (PK): Unique identifier for the maintenance request
    - UnitID (FK): ID of the unit associated with the maintenance request
    - RequestDate: Date the maintenance request was made
    - RequestDescription: Description of the maintenance issue
    - Status: Status of the maintenance request (e.g., "pending", "in progress", "completed")
8. **Lease:**
    - TenantID (PK, FK1): ID of the tenant associated with the lease
    - UnitID (PK, FK2): ID of the unit associated with the lease
    - StartDate: Start date of the lease
    - EndDate: End date of the lease
    - SecurityDeposit: Security Deposit amount for the unit
9. **Payment:**
    - PaymentID (PK): Unique identifier for the payment
    - UnitID (FK): ID of the unit associated with the payment
    - PaymentDate: Date the payment was made
    - Amount: Amount paid
10. **Landlord:**
    - LandlordID (PK): Unique identifier for the landlord
    - LandlordFirstName: FirstName of the landlord
    - LandlordLastName: LastName of the landlord
    - LandlordContactNumber: Contact of the landlord
11. **Employee**:
    - EmployeeID (PK): Unique identifier for the employee
    - EmployeeFirstName: FirstName of the employee
    - EmployeeLastName: LastName of the employee
    - EmployeeContact: Contact of the employee

12. **Customer:**
    - CustomerID (PK): Unique identifier for the customer
    - EmployeeID (FK): ID of the employee associated with the customer
    - CustomerFirstName: FirstName of the customer
    - CustomerLastName: LastName of the customer
    - CustomerContact: Contact of the customer
13. **Customer Preferences:**
    - CustomerID (PK, FK1): ID of the customer associated with the customer preference
    - UnitTypeID(PK, FK2): ID of the UnitType associated with the customer preference
    - MinimumRent: Minimum rent amount the customer is willing to pay
    - MaximumRent: Maximum rent amount the customer is willing to pay
14. **Feedback:**
    - FeedbackID (PK): Unique identifier for the feedback
    - TenantID (FK): ID of the tenant associated with the feedback
    - FeedbackDate: Date the feedback
    - FeedbackDescription: Description of the feedback