Name: Pranav Raj Sowrirajan Balaji

Date: 7 November, 2024

Home Rental Management System

This system is designed to help property owners manage rental properties and tenants efficiently. It includes features for tracking property listings, tenant applications, lease agreements, payments, maintenance requests and tenant referrals. The Tenant Referral Program adds a layer of community engagement by allowing current tenants to refer friends or colleagues. If the referral leads to a lease, the referring tenant earns rewards, such as rent discounts or cash incentives. This feature encourages tenants to spread the word, improving property occupancy rates for property owners.

Rules

- A **property owner** can own multiple properties and list them on the website.
- A **property** can only be rented by one tenant at a time, but over time, it can have multiple tenants.
- A **tenant** can apply to rent multiple properties but can have only one active lease.
- Each **application** is associated with a specific tenant and property.
- A **lease agreement** connects a tenant and a property for a specific duration.
- Tenants must make **payments** for their rent, which are tracked for each lease.
- Tenants can submit **maintenance requests** for properties they rent.
- Tenants can refer other potential tenants and earn rewards if the referral leads to a lease.

Nouns & Verbs:

Entities

1. **Property Owner**: Represents a person or company that owns rental properties.

   Attributes:

   - propertyowner_ID
   - Name
   - Email
   - PhoneNumber
   - Address
   - TaxID

   Verbs/actions: List

2. **Property**: Represents a rental property owned by a property owner.
    a. Attributes:
        - property_ID,
        - Address
        - City
        - State
        - ZipCode
        - PropertyType (e.g., Apartment, House)
        - NumberOfRooms
        - RentAmount
        - propertyowner_ID

Verbs/actions: List

3. **Tenant**: Represents a person renting a property.

Attributes:

- tenant_ID,
- Name
- Email
- PhoneNumber
- LeaseStartDate
- LeaseEndDate
- property_id

Verbs: Apply, Rent, submit

4. Application: Represents an application submitted by a tenant to rent a property. (Class serving as medium to split many to many relation from property - tenant UML diagram to the ERD diagram)

Attributes:

- application_ID,
- tenant_ID
- property_ID
- ApplicationDate,
- ApplicationStatus (e.g., Pending, Approved, Rejected).

Verbs: Apply

5. Lease Agreement: Represents the rental agreement between a property owner and a tenant.

Attributes:

- lease_ID
- property_ID
- tenant_ID
- LeaseStartDate
- LeaseEndDate
- RentAmount
- SecurityDepositAmount.

Verbs: connects, active

6. Payment: Represents a payment made by a tenant for rent.

Attributes:

- payment_ID
- tenant_ID
- lease_ID
- PaymentDate
- Amount
- PaymentStatus (e.g., Paid, Late)

7. Maintenance Request: Represents a maintenance request made by a tenant.

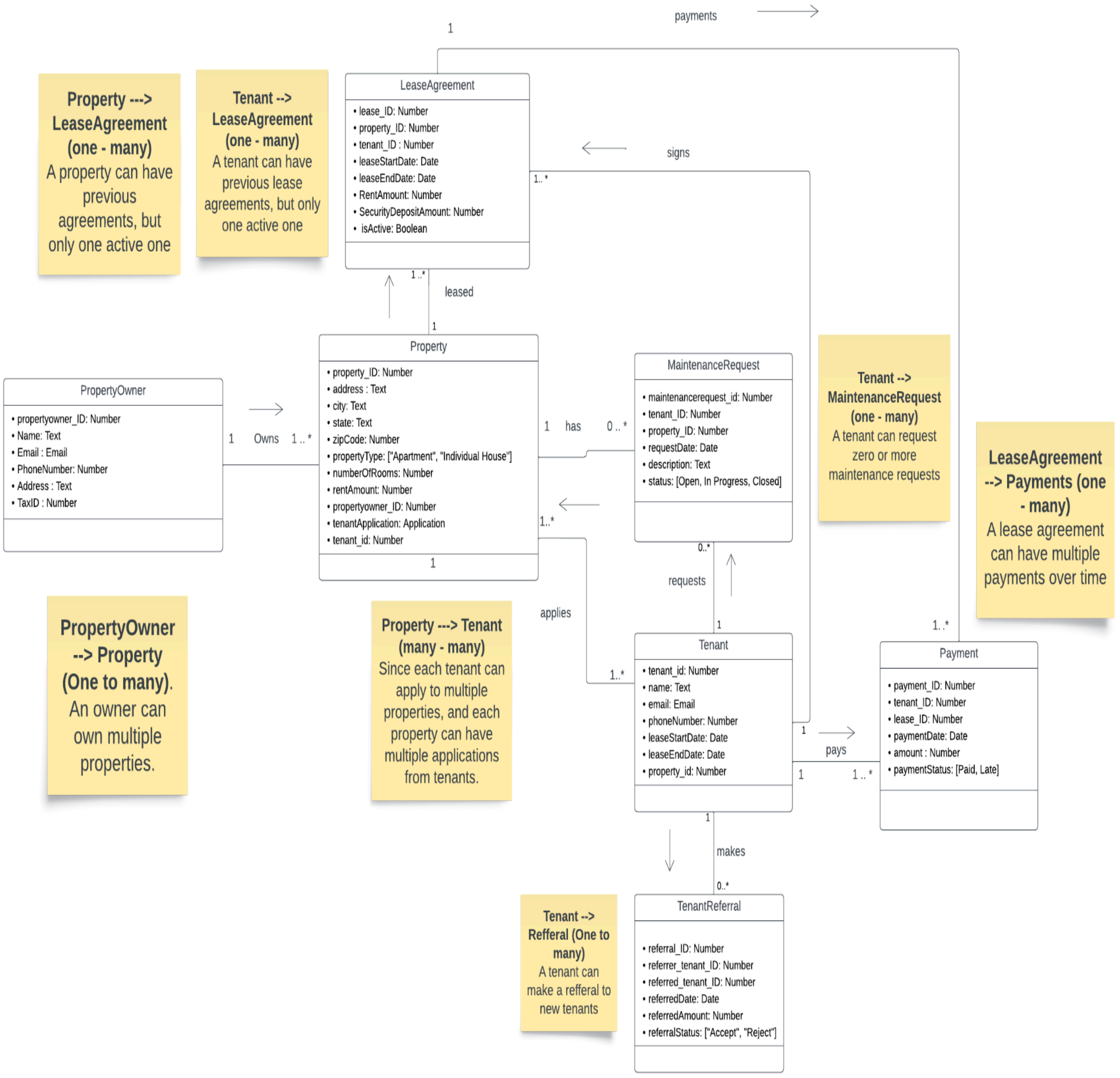
Attributes:

- request_ID
- tenant_ID
- property_ID
- RequestDate
- Description
- Status (e.g., Open, In Progress, Completed)

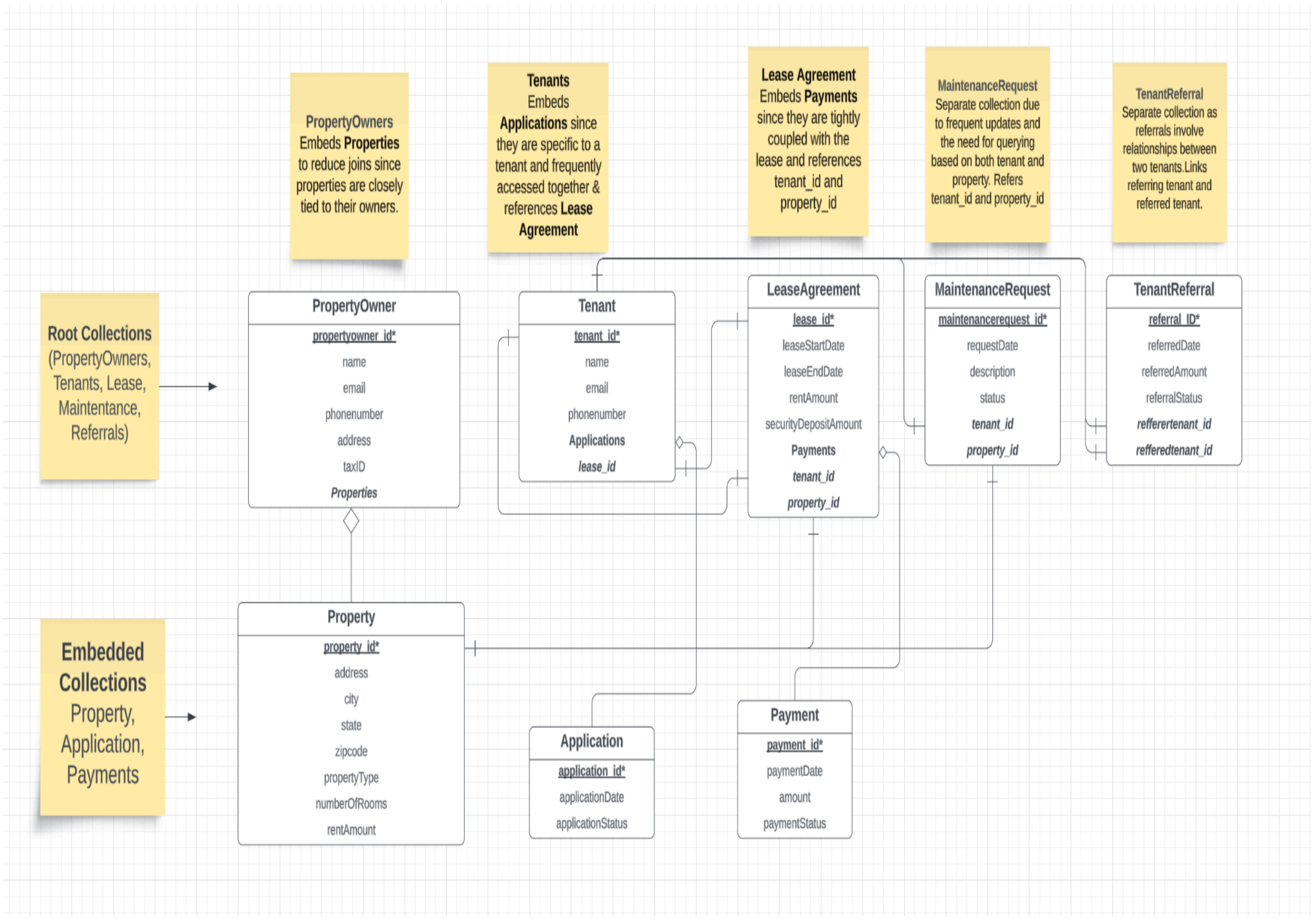
8. **TenantReferral**: Tenant can give referrals to potential clients / tenants.

- referral_ID
- referrer_tenant_ID
- referred_tenant_ID
- ReferralDate
- RewardAmount
- ReferralStatus (Accept, Reject)

Conceptual Model (UML):


payments
1
LeaseAgreement
• lease_ID: Number
• property_ID: Number
• tenant_ID : Number
• leaseStartDate: Date
• leaseEndDate: Date
• RentAmount: Number
• SecurityDepositAmount: Number
• isActive: Boolean
signs
1.. *
1 ..*
leased
1
Property ---> LeaseAgreement (one - many) A property can have previous agreements, but only one active one
Tenant --> LeaseAgreement (one - many) A tenant can have previous lease agreements, but only one active one
PropertyOwner
• propertyowner_ID: Number
• Name: Text
• Email : Email
• PhoneNumber: Number
• Address : Text
• TaxID : Number
1
Owns
1 .. *
Property
• property_ID: Number
• address : Text
• city: Text
• state: Text
• zipCode: Number
• propertyType: ["Apartment", "Individual House"]
• numberOfRooms: Number
• rentAmount: Number
• propertyowner_ID: Number
• tenantApplication: Application
• tenant_id: Number
1
1
has
0 .. *
1..*
MaintenanceRequest
• maintenancerequest_id: Number
• tenant_ID: Number
• property_ID: Number
• requestDate: Date
• description: Text
• status: [Open, In Progress, Closed]
0..*
requests
1
Tenant --> MaintenanceRequest (one - many) A tenant can request zero or more maintenance requests
LeaseAgreement --> Payments (one - many) A lease agreement can have multiple payments over time
applies
1..*
Tenant
• tenant_id: Number
• name: Text
• email: Email
• phoneNumber: Number
• leaseStartDate: Date
• leaseEndDate: Date
• property_id: Number
1
pays
1
1 .. *
1..*
Payment
• payment_ID: Number
• tenant_ID: Number
• lease_ID: Number
• paymentDate: Date
• amount : Number
• paymentStatus: [Paid, Late]
PropertyOwner --> Property (One to many). An owner can own multiple properties.
Property ---> Tenant (many - many) Since each tenant can apply to multiple properties, and each property can have multiple applications from tenants.
1
makes
0..*
Tenant --> Refferal (One to many) A tenant can make a refferal to new tenants
TenantReferral
• referral_ID: Number
• referrer_tenant_ID: Number
• referred_tenant_ID: Number
• referredDate: Date
• referredAmount: Number
• referralStatus: ["Accept", "Reject"]

Logical Model (MongoDB Collections)


PropertyOwners Embeds Properties to reduce joins since properties are closely tied to their owners.
Tenants Embeds Applications since they are specific to a tenant and frequently accessed together & references Lease Agreement
Lease Agreement Embeds Payments since they are tightly coupled with the lease and references tenant_id and property_id
MaintenanceRequest Separate collection due to frequent updates and the need for querying based on both tenant and property. Refers tenant_id and property_id
TenantReferral Separate collection as referrals involve relationships between two tenants.Links referring tenant and referred tenant.
Root Collections (PropertyOwners, Tenants, Lease, Maintentance, Referrals)
Embedded Collections Property, Application, Payments
PropertyOwner
propertyowner_id*
name
email
phonenumber
address
taxID
Properties
Tenant
tenant_id*
name
email
phonenumber
Applications
lease_id
LeaseAgreement
lease_id*
leaseStartDate
leaseEndDate
rentAmount
securityDepositAmount
Payments
tenant_id
property_id
MaintenanceRequest
maintenancerequest_id*
requestDate
description
status
tenant_id
property_id
TenantReferral
referral_ID*
referredDate
referredAmount
referralStatus
reffererTenant_id
refferedtenant_id
Property
property_id*
address
city
state
zipcode
propertyType
numberOfRooms
rentAmount
Application
application_id*
applicationDate
applicationStatus
Payment
payment_id*
paymentDate
amount
paymentStatus

Collection Structures

1. ***PropertyOwners***
   - Embeds properties to reduce joins since properties are closely tied to their owners.
   - Properties are typically not queried independently but always in the context of their owner.

JSON Structure:

```
{
 "_id": "owner123",
 "name": "John Doe",
 "email": "john@example.com",
 "phone": "123-456-7890",
 "address": "123 Elm Street",
 "taxID": "TAX12345",
 "properties": [
   {
     "_id": "property456",
     "address": "456 Oak Street",
     "city": "Metropolis",
     "state": "CA",
     "zipCode": "90210",
     "type": "Apartment",
     "rooms": 3,
     "rent": 2000
   },
   {
     "_id": "property789",
     "address": "789 Pine Street",
     "city": "Springfield",
     "state": "IL",
     "zipCode": "62704",
     "type": "House",
     "rooms": 4,
     "rent": 2500
   }
 ]
}
```

2. ***Tenants***

- Embeds applications since they are specific to a tenant and frequently accessed together.
- References active leases as these can be queried independently for financial or legal purposes.

JSON Structure:

```
{
  "_id": "tenant789",
  "name": "Jane Smith",
  "email": "jane@example.com",
  "phone": "987-654-3210",
  "applications": [
    {
      "_id": "application001",
      "propertyID": "property456",
      "applicationDate": "2024-11-01",
      "status": "Pending"
    },
    {
      "_id": "application002",
      "propertyID": "property789",
      "applicationDate": "2024-11-05",
      "status": "Approved"
    }
  ],
  "activeLeaseID": "lease001"
}
```

3. LeaseAgreements

- Embeds payments since they are tightly coupled with the lease.
- References the tenant and property for relational integrity and to avoid duplication.

JSON Structure:

```
{
  "_id": "lease001",
  "tenantID": "tenant789",
  "propertyID": "property456",
  "startDate": "2024-12-01",
  "endDate": "2025-12-01",
  "rent": 2000,
  "securityDeposit": 1000,
  "payments": [
    {
      "_id": "payment001",
      "date": "2024-12-05",
      "amount": 2000,
      "status": "Paid"
    },
    {
      "_id": "payment002",
      "date": "2025-01-05",
      "amount": 2000,
      "status": "Late"
    }
  ]
}
```

4. MaintenanceRequests

- Separate collection due to frequent updates and the need for querying based on both tenant and property.
- References tenant and property for efficient lookups.

JSON Structure

```
{
  "_id": "request001",
  "tenantID": "tenant789",
  "propertyID": "property456",
  "date": "2024-11-08",
  "description": "Leaky faucet",
  "status": "In Progress"
}
```

5. TenantReferrals

- Separate collection as referrals involve relationships between two tenants.
- Links referring tenant and referred tenant.

JSON Structure:

```
{
  "_id": "referral001",
  "referrerTenantID": "tenant789",
  "referredTenantID": "tenant123",
  "date": "2024-11-10",
  "reward": 200,
  "status": "Accepted"
}
```