LEASE MANAGEMENT SYSTEM AND DATABASE

I. Requirement Analysis

Application description

Purpose and Scope:
The application aims to depict the lease management system for a real estate company. It centers around the signing of leases by tenants and guarantors, detailing the lease involved and the methods of enforcing the lease. It also manages information regarding the employees associated with buildings and the complaints by tenants. The application provides auxiliary algorithms such as identifying vacant apartments and checking whether a tenant has paid rent for a specific month. The application does not manage specific payments and employee costs, which are assumed to be recorded elsewhere (such as in the company's spending database). Instead, it tracks customers that are late on rent payments. It also stores complaints issued from the tenant and resolved by the manager of a specific building.

Database description

Entities and their attributes

1. Lease: **The lease** is the focus of the database, with foreign keys pointing to all parties mentioned in the lease. It is associated with several attributes, including the start date, end date, and the monthly rent. It is identified by a unique leaseId, which is an artificial key.

2. Apartment: **The apartment** is a weak entity with the following attributes: **(i)** size, representing the area of the apartment in square feet **(ii)** room number, representing its location in the apartment building. It can be identified by its primary key room number and the foreign key of its building (the address of the building).

3. Apartment Building: **An apartment building** owned by the company is identified by its address. The group of apartments inside the apartment building are weak entities to it.

4. Tenant: **A tenant** is someone who rents an apartment. Tenants are identified by the artificial key: *id*. The database keeps track of the contact information of the tenant (name, phone number, email).

5. legalActions: **A legal action** may be taken against the tenant if he violates the lease -- i.e. by failing to pay on time. Each legal action is represented by a unique, artificially generated *legalId*. A legal action also has a type (eg. a fine, eviction etc.) and a date when it was issued.

6. Employee: **An employee** can either be a superintendent, a manager, or someone responsible for other tasks outside the scope of this database (for example, a secretary at the company headquarters). Employees are identified by an eid and have their name stored. This

database is aimed towards managing the leases of apartments, we do not cover employees in detail as that is the task of another database model.

7. Manager: **A manager** is a subcategory of the Employee entity. Managers have an email attribute as they are typically not accessible through phone numbers.

8. Superintendent: **A superintendent** is a subcategory of the Employee entity. Superintendents take care of apartment buildings.

9. Guarantor: **A guarantor** is identified by his/her social insurance number. Guarantors act as a backup for the tenant and hence, sign the lease with the tenant. They also have the attributes: email, name, and phone.

10. Payment: **Payment** entities keep track of the status of payment of each month of a lease. The primary key is the month and the foreign key is the leaseId from the Lease entity set. Payment entities can be used to determine if a lease has been paid fully or if part of it must still be paid. The month attribute depicts the month the payment corresponds to and the status is a binary value: 'Yes' or 'No.'

11.Complaint: **Complaint** entities are identified by an artificial key, their cid. They also contain a description attribute. Complaints represent some issue a tenant has with their apartment.

Relationships

1. Signs: A tenant signs a lease. There is a one-to-one relationship,since a tenant must sign one lease, and a lease can be signed by only one tenant. The signs relationship has an attribute indicating when the tenant signed the lease.

2. For: A lease is for an apartment. There is a one-to-many relationship, since a lease can only be for exactly one apartment, and an apartment can have many leases.

3. In: An apartment is in an apartment building. There is a one-to-many relationship, because an apartment cannot be in more than one building, and a building should contain more than one apartment.

4. Funds: Payments fund a specific lease. This is a many to one relationship as many payments can fund a lease but can only fund a single.

5.Enfored by: a lease may be enforced by legal actions. This is a many to one relationship as many legal actions may be used on a single lease, but each act (filing, case, etcra), only targets a specific lease.

6.Manages: a building may be managed by a superintendent. This is a many to many relationship as a superintendent can manage multiple buildings and multiple superintentdents can manage a single building.

7.Garentee: Guarantors may guarantee a lease. This is a many two one relationship as many guarantors can guarantee a lease, but each guarantor cannot guarantee more than one.

8.Issues: a tenant may issue complaints. This is a many to one as a tenant may issue many complaints, but tenants must submit their own independent complaints.

9.Resolves: a manager may resolve complaints. This is a many to one as a manager can resolve many complaints but a single complaint is only assigned to one manager.

Special constraints:
One apartment may only have one lease at any given time.
Vacant/occupied is a calculated attribute stored externally.

III. Relational Model

Entities:

Lease(leaseId, startDate, endDate, roomNo, monthlyRent, addr)
roomNo references Apartment.roomNo
addr references Apartment.addr
aptBuilding(addr, managerid)
managerid references Employee.eid

Employee(eid, name, buildingManaged)
note: Employee relation is combined with Manager and Superintendent, where only managers have email

Superintendent(eid)
addr references aptBuilding.addr
Manager (eid,email)

Tenant (id, name, email,phoneNumber)

LegalAction (legalid , name, date, leaseId)
leaseId references Lease.leaseId
Complaint (cid, description)

Guarantor (SIN, name, email, phoneNumber,leaseid)
Leaseid references lease.leaseid

Weak Entities:

Apartment(roomNo, addr, size)
addr references AptBuilding.addr
Payment(month,leaseId,status)
Leaseid references lease.leaseId

Relationship:

Manage (eid,addr)
Eid references superintendent.eid
Addr references aptBuilding.addr

Issues (cid,id,dateIssued)
Id references Tenant.id
Cid references Complaint.cid

Resolves (cid,eid,dateResolved)
Cid references Complaint.cid
Eid references manager.eid

Sign (leaseid,id,date)
Leaseid references lease.leaseId
Id references Tenant.id

Creative Aspect:

In terms of setting the model apart from others, our model contains 7 key constraints. It also contains one case of inheritance and two weak entities.