Explain your assumptions for each entity and relationship in your model. Discuss why you've modeled something as an entity rather than an attribute of another entity. Describe the cardinality of relationships, like why a student is linked to only one advisor. These assumptions might come from customer requirements or application constraints. Please clarify them.

**Assumptions for Entities:**

**User:**
- This entity represents the users of the application.
- Modeled as an entity because users have multiple attributes and can have various interactions with the system, like favoriting property listings, hence requiring their own entity.
- Assumption: Each user has a unique identity within the system.

**Listing:**
- This entity represents the dynamic offerings in the student housing market; they have attributes that can frequently change like availability dates. When a property opens a new cycle of apartments, they will have a corresponding Listing such as The Dean opens up a listing for Spring 2025, etc.
- Modeled as an entity because they have multiple complex attributes (like descriptions, availability dates, and associated properties). Listings have a one-to-many relationship with Property since a property can have multiple listings, but each listing is created by only one property.
- Assumption: A listing must be related to one and only one property but a property can have multiple listings over time, but there must exist at least one listing for each property to be listed on this website.

**Property:**
- This entity represents the physical characteristics of the housing being listed with a set of unchanging attributes like location and amenities. It is distinct from a Listing because while a Listing relates to the offer cycle(including available date), the Property relates to the immutable attributes of the real estate (like address, and contact information).
- Assumption: The physical attributes of a property remain consistent.
- Properties have a one-to-many relationship with listings and floor plans to accommodate the possibility of multiple offers and configurations.

**FloorPlan:**
- Floor plans are specific to properties and can vary within the same property, They contain detailed information that is relevant to the renting process.
- Modeled as a weak entity because it cannot exist without an associated Property.

- A Floor Plan is tied to a specific Property, therefore need to use a composite key that includes PropertyID.
- A property can have multiple floor plans,  therefore the relationship is one-to-many.

**Favorites:**
- Represents a junction table that represents the many-to-many relationship between Users and Listings, as users can favor multiple listings, and listings can be favorited by multiple users.
- It's an entity rather than an attribute due to the need to track additional information like when a listing was favorited.

**Rating:**
- This entity represents each specific rating given to properties. Each property can have multiple ratings from different users.
- Ratings provide detailed feedback, thus they need to be stored as separate entities.

**Relationship Descriptions and Cardinalities:**
**User to Favorites:**
- Cardinality: Zero-to-Many. Each user may have no favorite listings or several favorite listings.
- Assumption: Users have the flexibility to bookmark any number of listings they are interested in, but each favorite entry is unique to one user, capturing their specific interests.

**Favorites to Listing:**
- Cardinality: Many-to-Zero. A listing can have none, one, or many favorited records, indicating this property has been favorited be some specific users, but each favorite record relates to only one listing.
- Assumption: Listings are available to all users, who can mark them as favorites. This reflects the nature of listings and the variety of user preferences.

**Rating to Property:**
- Cardinality: Many-to-One. Any property may have zero or numerous ratings from different sources, but each rating corresponding to one property only.
- Assumption: Properties may be assessed by various external sources, providing a wide range of evaluations that enrich the information available to users. This also allows users to request detailed ratings description for greater insight(such as see why the property has so many 1 star/1 point ratings).

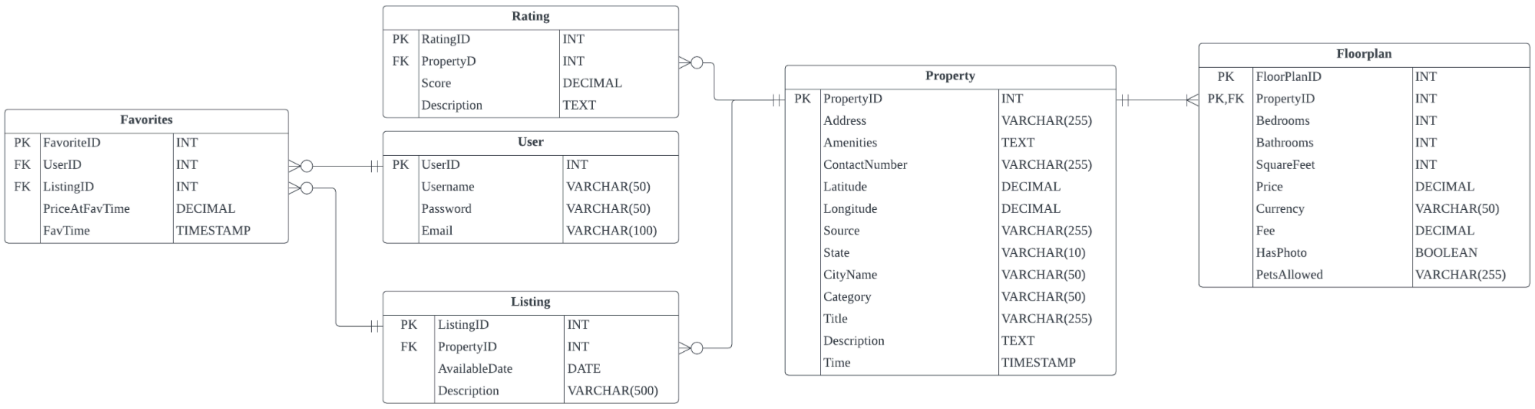
**Property to Listing:**

- Cardinality: One-to-Many. A property can be associated with 0 or multiple listings over different time periods, such as seasonal offerings(E.g.: A property may have two listings, one for spring cycle, and one for fall cycle). And each Floor Plan is uniquely identified only in the context of its Property.
- Assumption: The property is a fixed entity with changing conditions for rent, thus multiple listings can be created for different time frames. Each Floor Plan's existence is based on the Property it describes.

**Property to FloorPlan:**

- Cardinality: One-to-Many. A single property may feature one or multiple floor plans, but each floor plan is unique to that property.
- Assumption: Properties, especially larger apartments agencies, can offer different floor plans to potential tenants.

## Relational Schema:



**Rating**

| | | |
|---|---|---|
| PK | RatingID | INT |
| FK | PropertyD | INT |
| | Score | DECIMAL |
| | Description | TEXT |

**Favorites**

| | | |
|---|---|---|
| PK | FavoriteID | INT |
| FK | UserID | INT |
| FK | ListingID | INT |
| | PriceAtFavTime | DECIMAL |
| | FavTime | TIMESTAMP |

**User**

| | | |
|---|---|---|
| PK | UserID | INT |
| | Username | VARCHAR(50) |
| | Password | VARCHAR(50) |
| | Email | VARCHAR(100) |

**Listing**

| | | |
|---|---|---|
| PK | ListingID | INT |
| FK | PropertyID | INT |
| | AvailableDate | DATE |
| | Description | VARCHAR(500) |

**Property**

| | | |
|---|---|---|
| PK | PropertyID | INT |
| | Address | VARCHAR(255) |
| | Amenities | TEXT |
| | ContactNumber | VARCHAR(255) |
| | Latitude | DECIMAL |
| | Longitude | DECIMAL |
| | Source | VARCHAR(255) |
| | State | VARCHAR(10) |
| | CityName | VARCHAR(50) |
| | Category | VARCHAR(50) |
| | Title | VARCHAR(255) |
| | Description | TEXT |
| | Time | TIMESTAMP |

**Floorplan**

| | | |
|---|---|---|
| PK | FloorPlanID | INT |
| PK,FK | PropertyID | INT |
| | Bedrooms | INT |
| | Bathrooms | INT |
| | SquareFeet | INT |
| | Price | DECIMAL |
| | Currency | VARCHAR(50) |
| | Fee | DECIMAL |
| | HasPhoto | BOOLEAN |
| | PetsAllowed | VARCHAR(255) |

**User**(
    UserID: INT [PK],
    Username: VARCHAR(50),
    Password: VARCHAR(50),
    Email: VARCHAR(100)
)

**Property(**
    PropertyID: INT [PK],

```
    Address: VARCHAR(255),
    Amenities: TEXT,
    ContactNumber: VARCHAR(255),
    Latitude: DECIMAL,
    Longitude: DECIMAL,
    Source: VARCHAR(255),
    State: VARCHAR(10),
    CityName: VARCHAR(50),
    Category: VARCHAR(50) ,
    Title: VARCHAR(255),
    Description: TEXT,
    Time: TIMESTAMP
)
```

**FloorPlan(**

```
    FloorPlanID: INT [PK],
    PropertyID: INT [PK][FK to Property.PropertyID],
    Bedrooms: INT,
    Bathrooms: INT,
    SquareFeet: INT,
    Price: DECIMAL,
    Currency: VARCHAR(50),
    Fee: DECIMAL,
    HasPhoto: BOOLEAN,
    PetsAllowed: VARCHAR(255)
)
```

**Listing**(

```
    ListingID: INT [PK],
    PropertyID: INT [FK to Property.PropertyID],
    AvailableDate: DATE,
    Description: VARCHAR(500)
)
```

**Favorites**(

```
    FavoriteID: INT [PK],
    UserID: INT [FK to User.UserID],
    ListingID: INT [FK to Listing.ListingID],
    PriceAtFavTime: DECIMAL,
    FavTime: TIMESTAMP
)
```

**Rating**(
```
    RatingID: INT [PK],
    PropertyID: INT [FK to Property.PropertyID],
    Score: DECIMAL,
    Description: TEXT
)
```

## BCNF:

Our original diagram was already in BCNF. This is because all functional dependencies present in the relations are purely determined by the primary keys of those relations, meaning the left side is always a super key. Our schema is straightforward with each attribute being functionally dependent on a primary key, perfectly aligning with BCNF's requirement for determinants to be candidate keys.

**User**:
UserID → Username, Password, Email

This relation is in BCNF because the only candidate key is UserID, and all other attributes are fully functionally dependent on it.

**Property**:
PropertyID→ Address, Amenities, ContactNumber, Latitude, Longitude, Source
Latitude, Longitude -> Address

This relation is in BCNF because the only candidate key is PropertyID, and all other attributes are fully functionally dependent on it.

**FloorPlan**:
FloorPlanID + PropertyID→ Title, Description, Bedrooms, Bathrooms, SquareFeet, Price, Currency, Fee, HasPhoto, PetsAllowed, Time

This relation is in BCNF because the candidate keys are FloorPlanID and PropertyID combined, and all other attributes are fully functionally dependent on this composite key.

**Listing:**
ListingID→PropertyID, AvailableDate, Description

This relation is in BCNF because the only candidate key is ListingID, and all other attributes are fully functionally dependent on it.

**Favorites**:
FavoriteID→UserID, ListingID, PriceAtFavTime, FavTime

This relation is in BCNF because the only candidate key is FavoriteID, and all other attributes are fully functionally dependent on it.

**Rating**:
RatingID→PropertyID, Score, Description

This relation is in BCNF because the only candidate key is RatingID, and all other attributes are fully functionally dependent on it.