Digital Food Donation Matching System

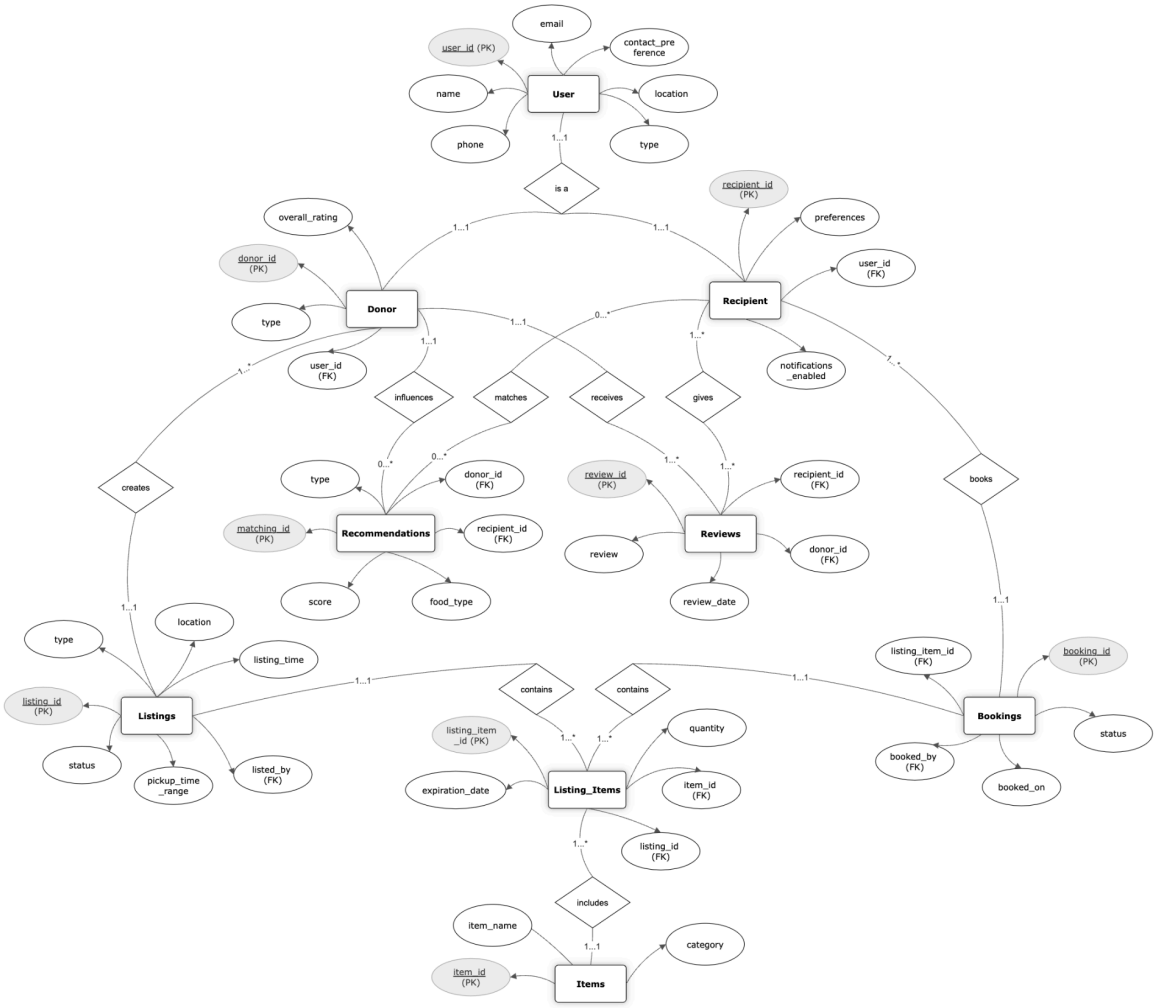
Project Track:1 Stage 2 Group Id: **070**

Project Members:

1. Sai Prakash Gorti
2. Omkar Dhekane
3. Nikunj Agarwal
4. Sai Krishna Rohith Kattamuri

ER Diagram

We have created the following ER diagram: (Link to Drive)



Entity Assumption and Description

1. **User**

Assumption: The *User* entity represents anyone interacting with the system. A *User* is modeled as an entity because it has multiple attributes and plays a key role in the system. Both *Donor* and *Recipient* have IS A relationship with the *User.*

Entity Attribute	Attribute Description	Key
user_id	Unique identifier for the user	Primary Key
email	Email address of the user	
Name	Name of the user.	
phone	Phone number of the user	
contact_preference	Preferred method of contact (e.g., email, phone)	
location	Location information for the user	
type	Specifies the type of user (Donor or Recipient)	

2. **Donor**

Assumption: Users who donate items/food. Rather than treating the *Donor* as an attribute of the *User*, it's modeled as a separate entity because it can hold additional information specific to a donor, like the rating and donor type that differentiate them from recipients.

Entity Attribute	Attribute Description	Key
donor_id	Unique identifier for the donor	Primary Key
type	Type of donor (e.g., individual, organization)	
overall_rating	The average rating given to the donor by recipients	
user_id	The foreign key refers to corresponding user	Foreign Key

3. **Recipient**

Assumption: Users who receive items. Attributes like preferences guide recommendations. Like the *Donor*, the *Recipient* is modeled as an entity because it has distinct attributes (e.g., preferences) and specific behaviors, like booking items that are different from Donors.

Entity Attribute	Attribute Description	Key
recipient_id	A unique identifier for the recipient (inherits user_id) is needed	Primary Key

preferences	Specific item preferences (e.g., food type)	
user_id	Foreign key refers to the corresponding user	Foreign Key
notifications_enabled	Boolean indicating whether notifications are enabled for the recipient	

4. **Items**

Assumption: Items represent the individual items that can be part of a listing. Items are modeled as separate entities because individual items can appear across multiple listings, and it's important to track them separately. If *Items* were just attributes of *Listing_Items*, tracking item types would become complicated. Also, we have added *Category* as an attribute here instead of a separate entity. This is because it does not have any other attribute on its own.

Entity Attribute	Attribute Description	Key
item_id	Unique identifier for the item	Primary Key
item_name	Name of the item	
category	Category of the item (e.g., protein, fruits, grocery)	

5. **Listings**

Assumption: Listings represent collections of items offered by donors. The reason for treating listings as a separate entity is that donors will have the flexibility to independently add multiple listings at any point in time. Each listing can include multiple items and have unique attributes (such as location and time). This makes managing multiple listings from the same donor easier, which wouldn't be feasible as an attribute of *Donor*.

Entity Attribute	Attribute Description	Key
listing_id	Unique identifier for the listing	Primary key
type	Type of listing (e.g., food, clothing)	
location	Location of the listing	
listing_time	Timestamp when the listing was created	
status	Current status of the listing (active, closed, etc.)	
pickup_time_range	Time window available for pickup of listed items	
listed_by	Foreign key refers to the *Donor* who created the listing	Foreign Key

6. **Listing_Items**

Assumption: This entity represents individual items in a listing. Each listing can have multiple items with their own attributes (e.g., expiration date). Managing individual items as attributes of *Listings* would lead to data redundancy, so this separation is useful.

Entity Attribute	Attribute Description	Key
listing_item_id	Unique identifier for the item within a listing	Primary Key
quantity	Quantity of the specific item listed	
expiration_date	Expiration date of the item	
listing_id	Foreign key refers to the associated listing	Foreign Key
item_id	Foreign key refers to the specific item in the listing	Foreign Key

7. **Bookings**

Assumption: *Bookings* refers to the action where a *Recipient* reserves an item. It tracks who books what and when. Booking is a core activity for recipients and involves multiple attributes that need to be tracked, like the date and status. Modeling it as an entity makes sense.

Entity Attribute	Attribute Description	Key
booking_id	Unique identifier for the booking	Primary Key
listing_item_id	Foreign key refers to the specific listing item	Foreign Key
booked_by	Foreign key refers to the recipient who booked the item.	Foreign Key
booked_on	Timestamp of when the booking was made	
status	Status of the booking (pending, confirmed, canceled)	

8. **Reviews**

Assumption: *Reviews* capture feedback from a *Recipient* about a *Donor*. It involves specific interactions between recipients and donors. We have modeled reviews as an entity because they involve multiple attributes and relationships (e.g., one recipient giving reviews to different donors). This allows flexibility in managing reviews as part of the system. And can even be useful for recommending donors for recipients having positive reviews.

Entity Attribute	Attribute Description	Key
review_id	Unique identifier for the review.	Primary Key
review	Review text provided by the recipient.	

review_date	Date the review was submitted.	
recipient_id	Foreign key refers to the recipient who wrote the review.	Foreign Key
donor_id	Foreign key referring to the donor being reviewed	Foreign Key

9. **Recommendations**

Assumption: The R*ecommendations* involve complex logic that leads to matches between donors and recipients, potentially based on preferences set by the recipient and the score of the matching algorithm. The recommendations will be generated on a daily basis based on the past entries and will be stored in the table. While searching for food from recipients' side, this table will be used to rank the results. Modeling it as an entity allows it to function as a service that connects *Donors* and *Recipients* based on various factors.

Entity Attribute	Attribute Description	Key
matching_id	Unique identifier for the recommendation.	Primary Key
type	Type of recommendation (e.g. priority, bulk, urgent, specialty)	
score	Match score between donor and recipient based on preferences. This score will help in recommending donors to the recipients.	
food_type	Type of food recommended. (perishable, non-perishable, prepared meals, groceries, produce)	
donor_id	Foreign key refers to the donor in the recommendation.	Foreign Key
recipient_id	Foreign key refers to the recipient in the recommendation.	Foreign Key

Cardinality of Relationships

1. **User is a Donor or Recipient (1..1)**
   A User can either be a Donor or a Recipient, but not both simultaneously. This is a 1:1 relationship because a single user cannot act as both a donor and a recipient at the same time.
2. **Donor creates Listings (1..*)**
   A donor may donate multiple items over time, leading to multiple listings. However, each listing is created by one specific donor. This makes the relationship between a donor and listings a one-to-many.
3. **Recipient books Bookings (1..*)**
   A Recipient can make multiple Bookings. This means a single recipient can book multiple listings, but each booking is tied to one recipient. Hence, the relationship is a one-to-many relationship between Recipient and Bookings
4. **Bookings contains Listing_items (1..1)**
   A Booking contains one Listing_item. This means each booking corresponds to one specific listing_item.The relationship is a one-to-one relationship between Bookings and Listing_items.
5. **Listings contains Listing_Items (1..*)**
   A listing may have several items (e.g., different food products), but each item must belong to one specific listing. This is a one-to-many type of relationship because a listing contains many items.
6. **Listing_Items includes Items (1..*)**
   A listing_Items may include one or more items. While an item is present in only one listing_items on items own. Thus, we modeled this relationship as one to many type.
7. **Recipient gives Reviews (1..*)**
   A Recipient can give multiple Reviews for different donors, but each Review is given by only one specific Recipient. This is a one-to-many relationship
8. **Donor receives Reviews (1..*)**
   A donor who has interacted with multiple recipients can receive multiple reviews, but each review is specific to one donor. This relationship is again a one-to-many relationship.
9. **Recipient matches Recommendations (0..*)**
   The system can generate several (or even no) recommendation(s) based on the recipient's preferences. making it a zero to many relationship
10. **Donor influences Recommendations (0..*)**
    This is a zero to many relationship as zero or more donors influences the recommendation engine to provide recommendations to recipients.

Normalization Process

All the entities conform to BCNF. In all these entities, the only determinant is the primary key, which is a super key. Thus, there are no partial or transitive dependencies.

Left:
User_id

Middle:
listed_by (FK)
booked_by (FK)

Right:
Email
Name
Phone
Contact_preference
Location
Type
Overall_rating
Listing_time
Preferences
Notifications_enabled
Item_name
Category
Status
Pickup_time_range
Expiration_date
Quantity
Booked_on
Review
Review_date
Score
food_type

Listing down the functional dependencies of each entity to prove the same:

User

- user_id → email, name, phone, contact_preference, location, type
- User entity is in BCNF
- Since user_id uniquely determines each attribute and there is only one attribute on the right-hand side of each functional dependency, this set is already on its minimal basis.

Donor

- Donor_id → overall_rating, type, user_id
- Donor entity is in BCNF
- This is already in its minimal basis because each dependency has a single attribute on the right-hand side, and all dependencies are necessary.

Recipient

- recipient_id → user_id, preferences, notifications_enabled
- Recipient entity is in BCNF
- This is already minimal because there is one attribute on the right-hand side of each dependency and no redundancy.

Listings

- listing_id → type, location, status, pickup_time_range, listed_by
- Listings entity is in BCNF
  Each functional dependency has a single right-hand side attribute and is minimal.

Items

- item_id → item_name, category
- Items entity is in BCNF
- All dependencies are minimal and non-redundant.

Listing_Items

- listing_item_id → expiration_date, quantity, item_id, listing_id
- Listing_Items is in BCNF
- All dependencies are minimal and non-redundant.

Bookings

- booking_id → status, booked_on, listing_item_id, booked_by
- Bookings entity is in BCNF
- All dependencies are minimal and non-redundant.

Reviews

- review_id → review, review_date, recipient_id, donor_id
- Reviews entity is in BCNF
- All dependencies are minimal and non-redundant.

Recommendations

- matching_id → type, score, food_type, donor_id, recipient_id
- Recommendations entity is in BCNF

- All dependencies are minimal and non-redundant.

Relational Schema

User(
user_id: INTEGER [PK],
email: VARCHAR,
name: VARCHAR,
phone: VARCHAR,
contact_preference: VARCHAR,
location: VARCHAR,
type: VARCHAR)

Donor(
donor_id: INTEGER [PK],
user_id: INTEGER [FK to User.user_id],
overall_rating: DECIMAL,
type: VARCHAR)

Recipient(
recipient_id: INTEGER [PK],
user_id: INTEGER [FK to User.user_id],
preferences: TEXT,
notifications_enabled: BOOLEAN)

Listings
listing_id: INTEGER [PK],
type: VARCHAR,
location: VARCHAR,
status: VARCHAR,
pickup_time_range: TIMESTAMP,
listed_by: INTEGER [FK to Donor.donor_id])

Items(
item_id: INTEGER [PK],
item_name: VARCHAR,
category: VARCHAR)

Listing_Items(
listing_item_id: INTEGER [PK],
expiration_date: DATE,
quantity: INTEGER,
item_id: INTEGER [FK to Items.item_id],
listing_id: INTEGER [FK to Listings.listing_id])

Bookings(
booking_id: INTEGER [PK],
status: VARCHAR,
booked_on: TIMESTAMP,
listing_item_id: INTEGER [FK to Listing_Items.listing_item_id],
booked_by: INTEGER [FK to Recipient.recipient_id])

Reviews(
review_id: INTEGER [PK],
review: TEXT,
review_date: DATE,
recipient_id: INTEGER [FK to Recipient.recipient_id],
donor_id: INTEGER [FK to Donor.donor_id])

Recommendations(
matching_id: INTEGER [PK],
type: VARCHAR,
score: DECIMAL,
food_type: VARCHAR,
donor_id: INTEGER [FK to Donor.donor_id],
recipient_id: INTEGER [FK to Recipient.recipient_id])

E.O.D