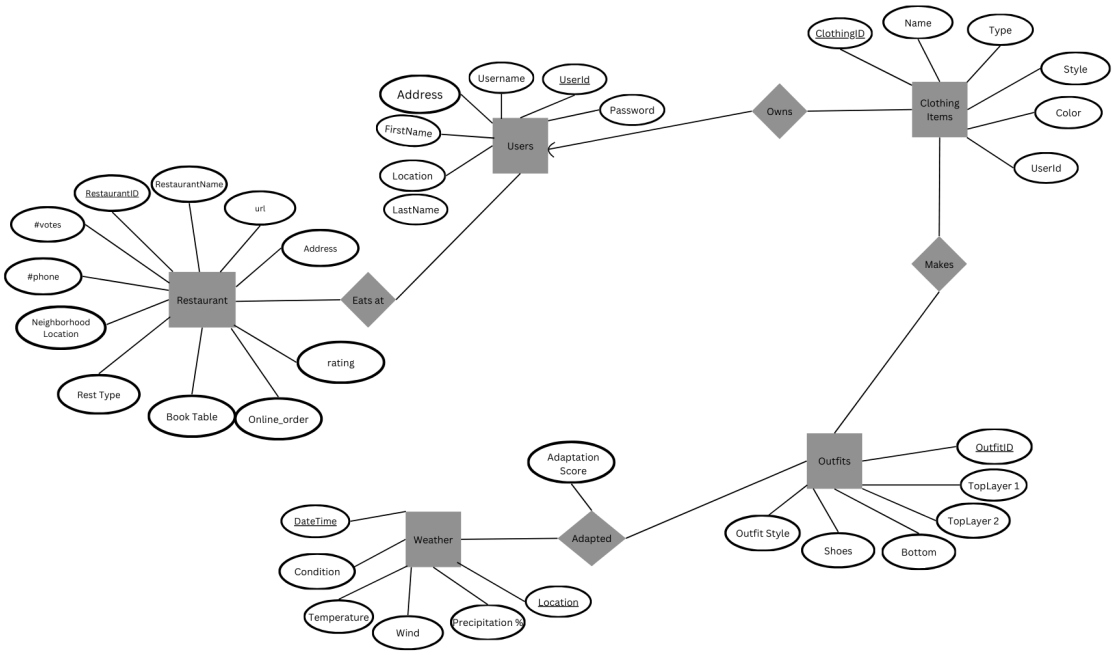
ClimaCloset - Weather Based Outfit Suggestion App
Jack Moran, Ben Goldman, Livia Bezati, and Reda Mehdaoui
Stage 2 - Team 085

Diagram



Explanation of Diagram

Entities and Their Assumptions

1. Users

- Attributes: UserId, Username, Password, FirstName, LastName, Address, Location
- Assumption:
    - Users need an account which is determined by Username and Password and they have identifying details (FirstName, LastName, Address).
    - Location is included to personalize services, such as weather-based outfit recommendations.
- Why an entity? Users interact with multiple entities. Keeping it as an entity allows dynamic tracking of these relationships.

2. Clothing Items

- Attributes: ClothingID, Name, Type, Style, Color, UserId
- Assumption:
    - Each clothing item has a unique identifier and is categorized by Type (e.g., jacket, pants), Style (e.g., casual, formal), and Color.
    - The UserId attribute links each clothing item to a specific user as a user owns many clothing items.
    - Users enter the clothing items they own, therefore each entered clothing item has exactly one owner
- Why an entity? Clothes are independently owned by users, so they need separate attributes and relationships.

3. Outfits

- Attributes: OutfitID, TopLayer 1, TopLayer 2, Bottom, Shoes, Outfit Style
- Assumption:
    - Outfits consist of multiple layers (e.g., tops, bottoms, shoes).
    - Each outfit has an associated Style (e.g., winter, summer, formal).
- Why an entity? Instead of storing outfits directly within users, outfits are dynamically created using different clothing items, making them reusable.

4. Weather

- Attributes: DateTime, Condition, Temperature, Wind, Precipitation %, Location
- Assumption:
    - Weather conditions are location-based and influence outfit recommendations.
- Why an entity? Weather varies over time and location, making it necessary to store this information separately for outfit recommendations.

5. Restaurants

- Attributes: RestaurantID, RestaurantName, url, Address, #votes, #phone, Neighborhood Location, Rest Type, Book Table, Online_order
- Assumption:
    - Users interact with restaurants, which have attributes like location, type, and booking options.
- Why an entity? Restaurants are independent with unique characteristics, allowing users to associate visits.

Relationships and Their Cardinalities

1. Owns (Users to Clothing Items)

- Cardinality: (1:M)
- Assumption:
    - A user owns multiple clothing items, but each clothing item belongs to only one user.
- Why? Users manage their wardrobe in the app, so ownership must be tracked somehow.

2. Makes (Clothing Items to Outfits)

- Cardinality: (M:M)
- Assumption:
    - Clothing items can be part of multiple outfits, and an outfit consists of multiple clothing items.
- Why? Outfits are dynamically composed of clothing items rather than predefined.

3. Adapted (Weather to Outfits)

- Cardinality: (M:M)
- Has an adaptation score for each weather
- Assumption:
    - Weather influences multiple outfits, and each outfit can be recommended for multiple weather conditions.
- Why? Outfits should be recommended based on weather conditions.

4. Eats at (Users to Restaurants)

- Cardinality: (M:M)
- Assumption:
    - A user can eat at multiple restaurants, and a restaurant serves multiple users.
- Why? Users may visit various restaurants and provide ratings.

Normalization of Database

1. Users

Minimal Basis: UserID → Username, Password, FirstName, LastName, Location

- UserID is a superkey, ensuring 3NF and BCNF compliance.

2. Clothing Items

Minimal Basis: ClothingID → Name, Type, Style, Color, UserID

- ClothingID is a superkey, ensuring 3NF and BCNF compliance.

3. Weather

Minimal Basis: (DateTime, Location) → Condition, Temperature, Wind, Precipitation%

- (DateTime, Location) forms a composite primary key, ensuring 3NF and BCNF compliance.

4. Outfits

Minimal Basis: OutfitID → OutfitStyle, TopLayer1, TopLayer2, Shoes, Bottom

- OutfitID is a superkey, ensuring 3NF and BCNF compliance.

5. Restaurants

Minimal Basis: RestaurantID → RestaurantName, Address, URL, Phone, Votes, Rating, NeighborhoodLocation, RestType, BookTable, OnlineOrder

- RestaurantID is a superkey, ensuring 3NF and BCNF compliance.

Relationships

1. Owns (Users → Clothing Items)

Minimal Basis: (UserID, ClothingID) → Ownership

- (UserID, ClothingID) is a composite key, ensuring BCNF compliance.

2. Makes (Clothing Items → Outfits)

Minimal Basis: (ClothingID, OutfitID) → OutfitComposition

- (ClothingID, OutfitID) is a composite key, ensuring BCNF compliance.

3. Eats At (Users → Restaurants)

Minimal Basis: (UserID, RestaurantID) → DiningPreference

- (UserID, RestaurantID) is a composite key, ensuring BCNF compliance.

4. Adapted (Weather → Outfits)

Minimal Basis: (DateTime, Location, OutfitID) → WeatherAdaptation

- (DateTime, Location, OutfitID) is a composite key, ensuring BCNF compliance.

Conceptual Design to Logical Design (Relational Schema)

Users(Username:VARCHAR(50),
UserId:INT [PK],
Password:VARCHAR(255),
Address:VARCHAR(255),
FirstName:VARCHAR(50),
LastName:VARCHAR(50).
Location:INT [FK to Weather.Location]);

Weather(Location:INT [PK],
Precipitation:DECIMAL,
Wind:DECIMAL,
Temperature:DECIMAL,
Condition:VARCHAR(255),
DateTime:VARCHAR(50) [PK]);

ClothingItems(UserId [FK to Users.UserId],
ClothingId:INT [PK],
Color:VARCHAR(50),
Style:VARCHAR(50),
Type:VARCHAR(50),
Name:VARCHAR(50));

Outfits(OutfitID [PK],
TopLayer1:INT [FK to ClothingItems.ClothingId],
TopLayer2:INT [FK to ClothingItems.ClothingId],
Bottom:INT [FK to ClothingItems.ClothingId],
Shoes:INT [FK to ClothingItems.ClothingId],
OutfitStyle:VARCHAR(10));

Restaurant(RestaurantId:INT [PK],
Address:VARCHAR(255),
url:VARCHAR(255),

RestaurantName:VARCHAR(100),
votes:INT,
phoneNum:INT,
NeighborhoodLocation:VARCHAR(50) [FK to Users.Location],
RestType:VARCHAR(50),
BookTable:BOOLEAN,
OnlineOrder:BOOLEAN,
rating:DECIMAL);