**Database Systems Project Part II**

**Logical Schema Optimization and Unstructured Data Collection**

**Team Members: Aruzhan Bolatova,** [**Arya Kotibhaskar**](mailto:aak10234@nyu.edu)

**Project Title:** Personalized Nutrition Recommendation System - MoodBite

The logical schema represented in the ER model organizes and defines entities, attributes, and their relationships within a database designed for personalized nutrition and health tracking. Below is a detailed breakdown of the schema's components:

### **Key Entities and Their Attributes**

1. **Users**

Purpose: Represents individuals using the application, serving as the central entity linking various user-specific data.

Attributes: user\_id, username, email, password.

Relationships: Has one UserProfile. Associated with multiple MoodTracking, DietaryPreferences, MealPlans, RecipePreferences, and HealthGoals.

1. **UserProfiles**

Purpose: Stores personal details and physical metrics of users for personalized recommendations and tracking.

Attributes: profile\_id, user\_id, age, gender, height, weight, activity\_level.

Relationships: Belongs to a single User.

1. **MoodTags**

Purpose: Defines mood categories for tracking and recipe recommendations based on mood.

Attributes: id, name, description.

Relationships: Linked to multiple Recipes. Associated with MoodTracking entries.

1. **Recipes**

Purpose: Stores information about recipes, including metadata, dietary labels, and associated mood tags.

Attributes:  
recipe\_id, name, recipe\_link, ingredients, mood\_tag\_id, image\_link, summary, cuisines, dish\_type, instruction\_steps, spoonacular\_score, servings, price\_per\_serving, caloric\_breakdown, DietLabels, vegetarian, vegan, gluten\_free.

Relationships: Linked to MoodTags. Used in MealPlans, RecipePreferences, RecipeRatings, RecipeAnalytics, and Nutrition.

1. **Ingredients**

Purpose: Represents individual ingredients for shopping list creation and recipe components.

Attributes: ingredient\_id, name.

Relationships: Used in ShoppingList.

1. **MealPlans**

Purpose: Manages user-specific meal plans by linking recipes to users and defining meal types.

Attributes: meal\_plan\_id, user\_id, meal\_type, recipe\_id, servings.

Relationships: Links Users to Recipes.

1. **DietaryPreferences**

Purpose: Stores user-specific dietary preferences and restrictions for personalized suggestions.

Attributes: preference\_id, user\_id, diet\_type, restrictions.

Relationships: Belongs to a User.

1. **Nutrition**

Purpose: Provides nutritional details for each recipe to support dietary analysis and health tracking.

Attributes: recipe\_id, calories, fat, protein, carbs, fiber, sugar, sodium.

Relationships: Linked to a Recipe.

1. **ShoppingList**

Purpose: Organizes ingredients required by users for recipes, supporting meal preparation.

Attributes: shopping\_list\_id, user\_id, ingredient\_id, quantity.

Relationships: Links Users to Ingredients.

1. **HealthGoals**

Purpose: Tracks user health objectives such as caloric intake and macronutrient targets for fitness goals.

Attributes: goal\_id, user\_id, goal\_type, target\_calories, target\_protein, target\_carbs, target\_fat.

Relationships: Belongs to a User.

1. **MoodTracking**

Purpose: Records user moods over time for analyzing trends and improving recipe recommendations.

Attributes: mood\_entry\_id, user\_id, date, mood\_tag\_id, energy\_level, stress\_level.

Relationships: Links Users to MoodTags.

1. **Analytics Tables**

Purpose:

**RecipeAnalytics**: Tracks views and favorites for recipes.

**RecommendationAnalytics**: Tracks user interactions with recipe recommendations.

**RecipePreferences**: Tracks user-specific preferences for recipes.

1. **NutritionTracking**

Purpose: Logs daily nutritional intake for users to monitor health and dietary balance.

Attributes: tracking\_id, user\_id, date, total\_calories, total\_protein, total\_carbs, total\_fat.

Relationships: Belongs to a User.

### 

### **Relationships**

1. **Users and Profiles**: One-to-One relationship.
2. **Users and Mood Tracking**: One-to-Many relationship. A user can have multiple mood entries.
3. **Users and Health Goals**: One-to-Many relationship. A user can set multiple health goals.
4. **Users and Dietary Preferences**: One-to-One relationship. A user has one dietary preference entry.
5. **Users and Recipes**: Many-to-Many relationship via Favorites and Recipe Preferences.
6. **Recipes and Ingredients**: Many-to-Many relationship via JSON data in ingredients attribute.
7. **Users and Meal Plans**: One-to-Many relationship. A user can create multiple meal plans.
8. **Users and Shopping List**: One-to-Many relationship. A user can have multiple shopping list entries.

### **Optimization**

2 database implementation files had been submitted under names ‘moodbite1520 optimizedfinal’ and ‘moodbite1520 not optimized’. The optimization that has been done are:

### **1. Indexing for Performance and Optimized Lookup**

* **Indexes added in moodbite1520:**
  + idx\_name on Ingredients(name)
  + idx\_moodtag\_name on MoodTags(name)
  + Multiple indexes on Recipes (e.g., vegetarian, vegan, gluten\_free, mood\_tag\_id, cuisines)
  + idx\_username and idx\_email on Users for unique lookups.
* **Purpose:** These indexes enhance query performance, particularly for lookups and filters on frequently accessed fields.

### **2. Use of Generated Columns**

* **Generated Columns in Recipes:** vegetarian, vegan, gluten\_free derived from the DietLabels JSON column.
* **Purpose:** This normalization extracts structured data from JSON, improving query efficiency and enabling direct indexing. It reduces the computational cost of repeatedly parsing JSON.
* By making generated columns (vegetarian, vegan, etc.) explicit, the database facilitates flexible queries without relying heavily on JSON parsing functions, which are computationally intensive.

### **3. Data Integrity and Consistency**

* Both databases enforce **foreign keys** and cascading deletions, but moodbite1520's normalization ensures better alignment of data structure with access patterns.
* Adding indexes on fields like cuisines, mood\_tag\_id, and user details helps maintain data consistency by preventing duplicates or allowing quick checks during inserts/updates.

**Normalization**

### **1. Breaking Down JSON Fields**

* In the original schema, fields like DietLabels and caloric\_breakdown in the Recipes table are stored as raw JSON.
* In the optimized schema, **functional dependencies** have been addressed by introducing **generated columns** for vegetarian, vegan, and gluten\_free derived from DietLabels. This helps avoid excessive dependence on JSON queries and allows direct indexing for efficient querying.

### **2. Functional Dependency Separation**

* **First Normal Form (1NF)**: Ensures that all data is atomic (no repeating groups or arrays).
  + In both schemas, most fields follow 1NF by using atomic fields.
  + The optimized schema adds **indexes** on attributes like name in Ingredients and MoodTags, and on generated columns from DietLabels. This ensures faster lookups and compliance with 1NF.
* **Second Normal Form (2NF)**: Eliminates partial dependencies on a composite primary key.
  + The original schema largely achieves this by avoiding composite primary keys. However, the optimized schema ensures better data integrity by normalizing JSON data and enforcing constraints like GENERATED ALWAYS for derived attributes.
* **Third Normal Form (3NF)**: Removes transitive dependencies (non-prime attributes depend only on primary keys).
  + Fields like vegetarian, vegan, and gluten\_free are directly tied to the primary key (recipe\_id) through derived values, removing potential transitive dependency issues.

### **3. Normalization of Mood Tags and MoodTag-Related Data**

* The MoodTags table remains separate in both schemas to prevent data duplication and improve reusability across different recipes and tracking entries.
* **Normalization** is further achieved by ensuring direct relationships between MoodTags and Recipes, along with indexed foreign keys for faster lookups.

### **4. Eliminating Redundancies in User-Related Tables**

* User-specific attributes like dietary preferences and profiles are separated into distinct tables (UserProfiles and DietaryPreferences), adhering to **normalization principles** by storing only relevant data in these tables.

### **5. Normalization of Analytics and Derived Data**

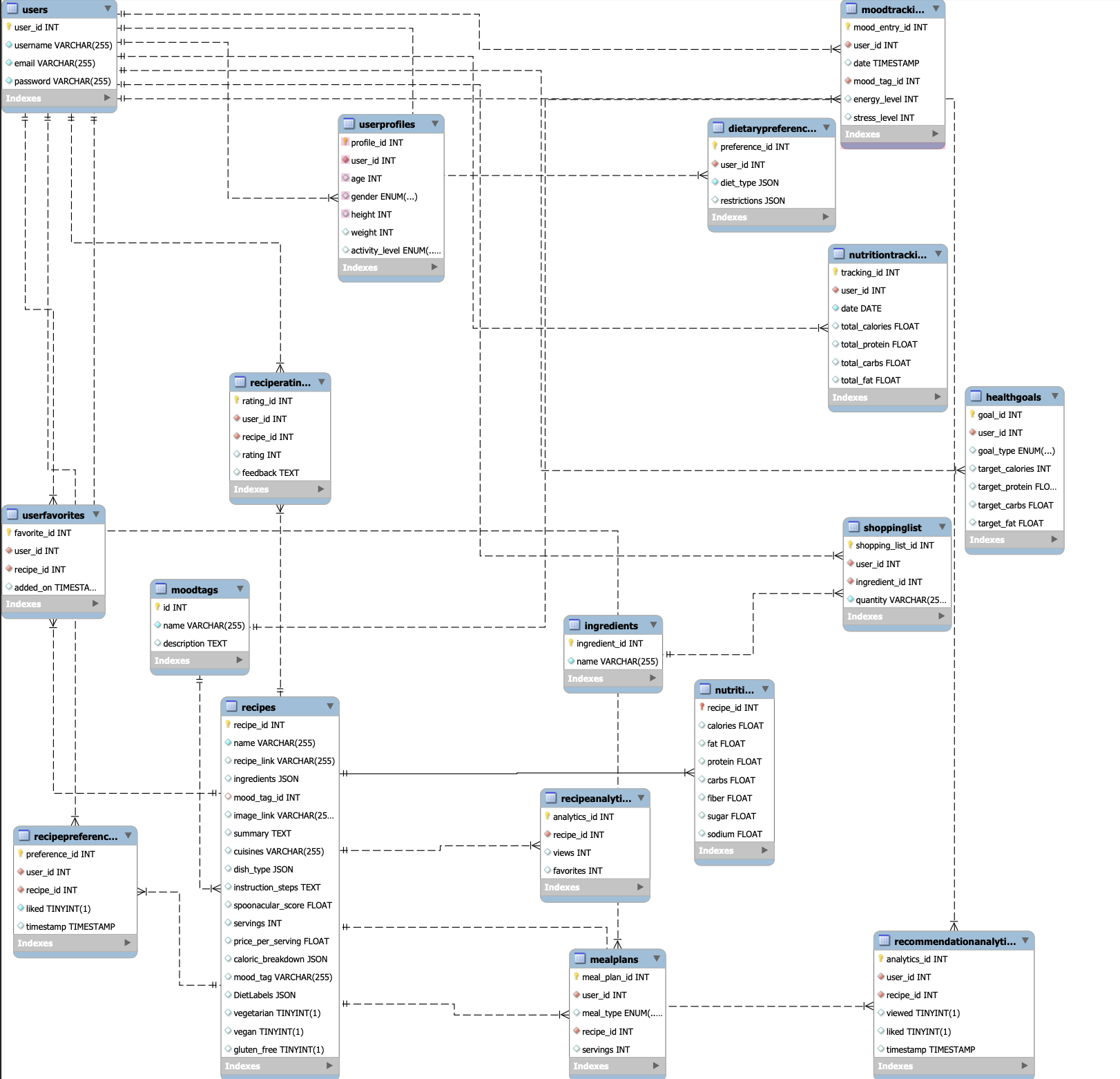
* Fields like RecipeAnalytics and RecommendationAnalytics track views, favorites, and user interactions separately rather than including these attributes in the Recipes table, ensuring data is not duplicated or updated redundantly.
* These tables are normalized to hold only derived or interaction-based data while linking back to primary entities via foreign keys.

### **Use Cases**

* **Personalized Recommendations:** The schema enables tailored recipe suggestions based on user preferences, mood, health goals, and dietary restrictions.
* **Nutritional Tracking:** Tracks user dietary intake against health goals and provides detailed nutritional insights.
* **Meal Planning:** Supports creation of custom meal plans and shopping lists.
* **User Engagement:** Tracks favorites, ratings, and analytics for recipe interaction.

### **Scalability**

The schema is designed for scalability with normalized tables and JSON-based storage for flexible data structures. Additional entities, such as exercise tracking or community interactions, can be integrated without major restructuring.



**Datasets**

Here’s a description of the datasets currently used in the MoodBite data lake:

1. Recipes (recipe\_details.csv)

Format: CSV

Description: This dataset was manually created using Spoonacular API, cleaned and formatted to CSV. It has 503 recipes that were fetched using different filters provided by Spoonacular API to ensure diversity of recipes for varying dietary preferences and emotional goals.

Use Case: Used as data lake of recipes from where we query recipes that match user input.

2. Nutritional analysis of each recipe (nutrition\_details.csv)

Format: CSV

Description: This dataset was captured along with recipes using Spoonacular API, each recipe has its own table of nutrients content such as Iron, Copper, Magnesium, Zinc, Iodine and so on.

Use Case: Used to match and fetch recipe with the highest matching score to emotional goal based on nutritional content of recipe and recommended elements intake

3. List of Ingredients for all recipes(Ingredients.csv)

Format: CSV

Description: A list of ingredients that are needed for all the recipes in the database was also captured along with recipes using Spoonacular API.

Use case: Used to match and fetch recipe with the highest matching score of ingredients required with relation to emotional goal

4. Emotional goal - Nutrients dataset (GoalNutrients.csv) and Nutrient - Ingredients dataset (NutrientIngredients.csv)

Format: CSV

Description: manually created data set based on scientific - medical records of positive effects of nutrients on mental health that correlated certain mood tag to multiple nutrients that can help achieve that goal

Use case: Matches emotional goal to nutrients that are required and further is to match emotional goal to ingredients rich in those certain nutrients. This data is further used to fetch recipe that has these nutrients in nutritional\_details.csv and contains these ingredients.

**External Datasets**

1. **Trader Joe’s Grocery’s list with prices** 
   * **Source**: [List of Trader Joe’s groceries](https://github.com/BrandonBell2025/SweetViolet/tree/main/Trader_Joes)
   * **Description**: This dataset provides prices of Trader Joe’s groceries
   * **Purpose**: Incorporating price calculation feature in application would accommodate users from different socioeconomic backgrounds and allow decision based on one more criteria, which is price