Smart Mess Management System

- Database Schema for Level-1
- Karthik K (200010024)

User

- ID - Unique identifier
- Username - User’s username
- Password - User password (hashed)
- Email - The email address of the user
- PhoneNumber - mobile number of the user
- Role - an array of roles the user will have (general, messCommitte, messAdmin, Admin)
- FirstName - First name of the user
- LastName - Last name of the user
- LastLoginDate - Date of the user's last login

Mess

- ID - Unique identifier
- messName - name of the mess (bhoopali, keeravani, etc..)
- Capacity - maximum number of users it can handle
- Location - address/location details of the mess (optional but helpful)
- Rating - average/overall rating of the mess

FoodItem

- ID - Unique identifier
- Name - Item name (rice, milk, egg, …)
- Image - Image of the food item
- Allergens - an array of allergens associated with the item
- Calories - Caloric content of the item

- Category - primary (rice, sambar ..), secondary (curry items), tertiary (cut fruits, pickles ..) (display order will be primary>secondary>tertiary)

MenuTable

- ID - Unique identifier
- Day - Day of the week (Sunday, Monday…)
- MealType - Type of the meal (breakfast, lunch, snacks, dinner, ….)
- MealItems - an array of food items (an array of foreign keys referring to the FoodItems table)
- messID - Foreign key referring to the Mess table

Feedback

- ID - Unique identifier
- User - ID of the user who given feedback (Foreign key to User table)
- Feedback - actual feedback given by the user
- Image - any images if attached (optional)
- Date - The date when the feedback form was floated
- messID - Foreign key referring to the Mess table

Notifications

- ID - Unique identifier
- Title - Title of the notification
- Message - the content of the notification
- Date - The date when the notification was floated
- messID - Foreign key referring to the Mess table

Note:

- By introducing the "Mess" table, I've successfully established a structured relationship between the different entities and their respective mess affiliations. This approach eliminates the potential complexities of incorporating various messes later, as any new mess can seamlessly be added directly to the mess table.

- In order to prevent unnecessary repetition of data (for example, having the same food item listed multiple times in a single week), I've organized the schema for storing menu data using two distinct tables. The first table is designed to store information about individual food items. The second table is responsible for constructing the actual weekly mess menu.

- Additionally, to enhance the menu's organization, a "Category" field has been included. This field helps determine the order in which food items are displayed on the menu. For instance, items like rice are shown before curry items and curry items are shown before condiments like pickles.