DATABASE MANAGEMENT SYSTEM PROJECT REPORT

DATABASE SYSTEM FOR A NUTRITIONIST

By

Hiba Hassan 21001780

Sara Mashaal 21000795

Database Systems

January 2023

Dr Fatemeh Jahedpari

Dr Hend ElMohandes

March 6th , 2023

Table of Contents

1. Brief Description 3
    - Project Overview
    - Project Objectives
2. EER Diagram 4
3. Relational Model 5
4. MySQL Database Implementation 6
    - Task 3 – Creating tables
    - Task 4 – Populating tables with data
    - Task 5 – Queries
5. User Guide 23
6. Bonus Features 24
    - User Guide for JAVA GUI
7. Summary of work 27

Brief Description

Project Overview

The project involves designing and implementing a database system for a nutritionist to keep track of their clients' information, daily meals, and ingredients. The database system consists of several tables, including a client table, a meal table, a food table, an ingredient table, and a category table.

The database system is made to make it simple and quick for the nutritionist to get data on each client's daily meals and monitor their progress over time. The technique can also be used by the nutritionist to assess each client's calorie intake and make tailored dietary improvement suggestions.

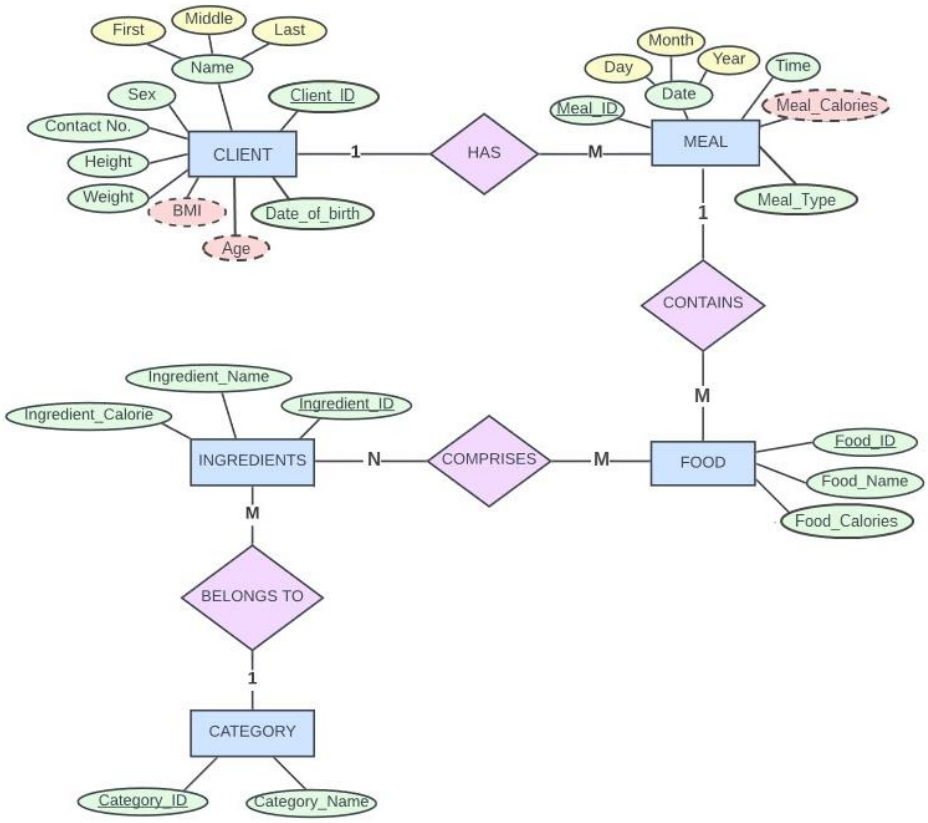
MySQL, a popular relational database management system, is utilised to implement the database system. The system is made to be scalable and versatile, making it simple to add or change the contents of the database.

As a bonus, JAVA was used to implement the GUI for the database.

Project Objectives

- To understand and develop an EER diagram which consists of relations, entities and attributes.
- To convert the EER diagram to a relational model.
- To implement the database using MySQL.

EER Diagram



Five entities were created for this EER Diagram. They were **CLIENT**, **MEAL**, **FOOD**, **INGREDIENT** and **CATEGORY**.

The CLIENT entity contains information about the client, such as their name, contact details, sex, date of birth, age, height, weight and BMI. The MEAL entity includes information about the meals consumed, such as the date, time, type of meal and the total calories of the meal. The FOOD entity contains information about the different types of foods, including their ID, name, and calorie count. The INGREDIENT entity contains information about the specific ingredients used in the foods, including their ID, name, and calorie count. Finally, the CATEGORY entity contains information about the categories of ingredients, such as fruits, vegetables, grains, protein foods, and dairy.

CLIENT has one-to-many relation with MEAL as one client can have many meals per day.

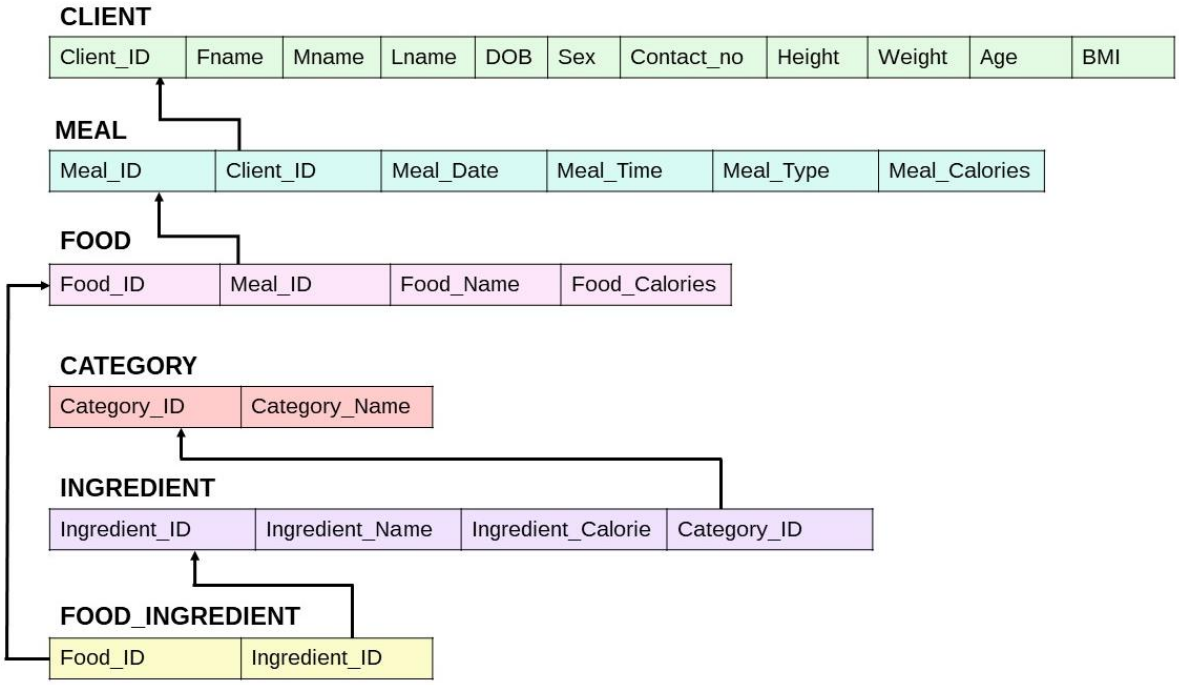
MEAL has one-to-many relation with FOOD as one meal can include many foods.

FOOD has many-to-many relation with INGREDIENT to show that food contains many ingredients and the same ingredients can belong to many foods.

INGREDIENT has a many-to-one relation with CATEGORY as many ingredients can belong to one category.

Overall, this database can be used to track clients' meals and provide insights into their nutritional intake.

Relational Model



Six tables were created when EER diagram was converted to a relational model. They are **CLIENT**, **MEAL**, **FOOD**, **INGREDIENT**, **CATEGORY**, and **FOOD_INGREDIENT**. Each has attributes from the EER diagram.

The MEAL table has a foreign key of Client_ID to know which meal belongs to which client.

The FOOD table has foreign key of Meal_ID to know which foods belong to a certain meal.

The INGREDIENT table has a foreign key to Category_ID to categorize the ingredients of the food.

The FOOD_INGREDIENT has foreign keys of Food_ID and Ingredient_ID to know which ingredient belongs to which food.

MySQL Database Implementation

A database called nutritionist was first created.

```
DROP DATABASE IF EXISTS NUTRITION_DATABASE;

CREATE DATABASE NUTRITION_DATABASE;

USE NUTRITION_DATABASE;
```

Task 3 – Creating tables

Then the tables for Client, Meal, Food, Category, Ingredient and Food_Ingredient was created with primary keys and acceptable foreign keys.

```
CREATE TABLE Client (
    Client_id INTEGER PRIMARY KEY NOT NULL,
    First_name VARCHAR(50),
    Middle_name VARCHAR(50),
    Last_name VARCHAR(50),
    Date_of_birth DATE NOT NULL,
    Sex ENUM('Male','Female'),
    Contact_no VARCHAR(15),
    Height FLOAT(5,2),
    Weight FLOAT(5,2),
    Age INT,
    BMI FLOAT(5,2) GENERATED ALWAYS AS (Weight / (Height * Height))
);

-- For updating age automatically
CREATE TRIGGER age_update
BEFORE INSERT ON Client
FOR EACH ROW
SET NEW.Age = TIMESTAMPDIFF(YEAR, NEW.Date_of_birth, CURRENT_DATE);

CREATE TABLE Meal (
    Meal_id INTEGER PRIMARY KEY NOT NULL,
    Client_id INTEGER,
    Meal_date DATE,
    Meal_time TIME,
    Meal_type ENUM('Breakfast','Morning Snack','Lunch','Evening Snack','Dinner'),
    Meal_calories INT,
    FOREIGN KEY (Client_id) REFERENCES Client(Client_id)
);

CREATE TABLE Food (
    Food_id INT NOT NULL,
    Meal_id INTEGER,
    Food_name VARCHAR(50),
    Food_calories INTEGER,
    PRIMARY KEY (Food_id),
    FOREIGN KEY (Meal_id) REFERENCES Meal(Meal_id)
```

```
);


CREATE TABLE Category (
    Category_id CHAR(3) PRIMARY KEY NOT NULL,
    Category_name VARCHAR(50)
);

CREATE TABLE Ingredient (
    Ingredient_id INTEGER PRIMARY KEY NOT NULL,
    Ingredient_name VARCHAR(50),
    Ingredient_calorie INTEGER,
    Category_id CHAR(3),
    FOREIGN KEY (category_id) REFERENCES Category(Category_id)
);

CREATE TABLE Food_ingredient (
    food_id INTEGER,
    ingredient_id INTEGER,
      PRIMARY KEY (food_id, ingredient_id),
    FOREIGN KEY (food_id) REFERENCES Food(Food_id),
    FOREIGN KEY (ingredient_id) REFERENCES Ingredient(ingredient_id)
);
```

Task 4 – Populating the tables with data

The data of 10 clients were inserted into the database. All the tables were populated for these 10 clients.

```
INSERT INTO Client (Client_id, First_name, Middle_name, Last_name, Date_of_birth,
Sex, Contact_no, Height, Weight) VALUES

(1, 'John', 'Robert', 'Doe', '1980-05-25', 'Male', '+1-555-123-4567', 1.80, 70.0),
-- healthy

(2, 'Jane', 'Marie', 'Smith', '1992-02-14', 'Female', '+1-555-987-6543', 1.65,
60.0), -- healthy

(3, 'Peter', 'David', 'Jones', '1975-11-01', 'Male', '+1-555-555-5555', 1.75,
80.0), -- overweight

(4, 'Amy', 'Elizabeth', 'Brown', '1988-07-10', 'Female', '+1-555-111-2222', 1.60,
45.0), -- underweight

(5, 'William', 'Jacob', 'Davis', '1982-03-08', 'Male', '+1-555-444-3333', 1.90,
110.0), -- obese

(6, 'Emily', 'Grace', 'Wilson', '1995-09-17', 'Female', '+1-555-222-1111', 1.55,
40.0), -- underweight

(7, 'Michael', 'Anthony', 'Taylor', '1979-12-23', 'Male', '+1-555-777-8888', 1.85,
90.0), -- overweight

(8, 'Olivia', 'Madison', 'Anderson', '1990-06-05', 'Female', '+1-555-333-4444',
1.70, 70.0), -- healthy
```

```
(9, 'David', 'Lee', 'Thomas', '1970-04-19', 'Male', '+1-555-222-3333', 1.80,
105.0), -- obese

(10, 'Sophia', 'Avery', 'White', '2000-01-02', 'Female', '+1-555-666-7777', 1.60,
55.0); -- healthy


INSERT INTO Meal (Meal_id, Client_id, Meal_date, Meal_time, Meal_type)

VALUES

-- John Doe's meals

(1, 1, '2023-02-26', '08:00:00', 'Breakfast'),

(2, 1, '2023-02-26', '12:30:00','Lunch'),

(3, 1, '2023-02-26', '18:30:00', 'Dinner'),

-- Jane Smith's meals

(4, 2, '2023-02-26', '07:30:00', 'Breakfast'),

(5, 2, '2023-02-26', '13:00:00','Lunch'),

(6, 2, '2023-02-26', '19:00:00','Dinner'),

-- Peter Jones's meals

(7, 3, '2023-02-26', '08:30:00','Breakfast'),

(8, 3, '2023-02-26', '12:00:00','Lunch'),

(9, 3, '2023-02-26', '16:00:00','Evening Snack'),

(10, 3, '2023-02-26', '19:30:00','Dinner'),

-- Amy Brown's meals

(11, 4, '2023-02-26', '13:00:00','Lunch'),

(12, 4, '2023-02-26', '18:00:00','Dinner'),

-- William Davis's meals

(13, 5, '2023-02-26', '07:30:00','Breakfast'),

(14, 5, '2023-02-26', '11:00:00','Morning Snack'),

(15, 5, '2023-02-26', '13:30:00','Lunch'),

(16, 5, '2023-02-26', '16:30:00','Evening Snack'),

(17, 5, '2023-02-26', '19:30:00','Dinner'),

-- Emily Wilson's meals

(18, 6, '2023-02-26', '07:00:00','Breakfast'),

(19, 6, '2023-02-26', '20:30:00','Dinner'),

-- Michael Anthony's meals

(20, 7, '2023-02-26', '07:30:00','Breakfast'),

(21, 7, '2023-02-26', '13:00:00','Lunch'),

(22, 7, '2023-02-26', '15:30:00','Evening Snack'),
```

```
(23, 7, '2023-02-26', '19:00:00','Dinner'),

-- Olivia Anderson's meals
(24, 8, '2023-02-26', '07:30:00','Breakfast'),
(25, 8, '2023-02-26', '13:30:00','Lunch'),
(26, 8, '2023-02-26', '19:30:00','Dinner'),
-- David Thomas's meals
(27, 9, '2023-02-26', '08:30:00','Breakfast'),
(28, 9, '2023-02-26', '10:00:00','Morning Snack'),
(29, 9, '2023-02-26', '12:00:00','Lunch'),
(30, 9, '2023-02-26', '16:00:00','Evening Snack'),
(31, 9, '2023-02-26', '19:30:00','Dinner'),
-- Sophia White's meals
(32, 10, '2023-02-26', '07:30:00','Breakfast'),
(33, 10, '2023-02-26', '13:00:00','Lunch'),
(34, 10, '2023-02-26', '19:00:00','Dinner');

INSERT INTO Food (Food_id, Meal_id, Food_name, Food_calories) VALUES
-- John Doe's breakfast
(1, 1, 'Oatmeal', 158),
(2, 1, 'Scrambled Eggs', 326),
-- John Doe's lunch
(3, 2, 'Grilled chicken breast', 284),
(4, 2, 'Caesar salad', 250),
-- John Doe's dinner
(5, 3, 'Mashed Potatoes', 250),
(6, 3, 'Grilled Salmon', 412),

-- Jane Smith's breakfast
(7, 4, 'Banana and Almond Butter Smoothie', 200),
-- Jane Smith's lunch
(8, 5, 'Roast beef and cheddar sandwich', 450),
(9, 5, 'Vegetable stir-fry', 250),
-- Jane Smith's dinner
(10, 6, 'Roasted Turkey', 350),
```

```
-- Peter Jones's breakfast
(11, 7, 'Bagel with Cream Cheese', 450),
(12, 7, 'Donut', 195),
-- Peter Jones's lunch
(13, 8, 'Cheeseburger', 750),
(14, 8, 'French Fries', 312),
-- Peter Jones's evening snack
(15, 9, 'Garlic bread with cheese', 300),
-- Peter Jones's dinner
(16, 10, 'Pizza (pepperoni)', 350),

-- Amy Brown's lunch
(17, 11, 'Ham and Swiss sandwich', 350),
-- Amy Brown's dinner
(18, 12, 'Fish and Chips', 500),

-- William Davis's breakfast
(19, 13, 'Pancakes with Butter and Syrup', 520),
(20, 13, 'Cinnamon Roll', 310),
-- William Davis's morning snack
(21, 14, 'Pizza rolls', 220),
-- William Davis's lunch
(22, 15, 'Chicken cheese sandwich', 950),
-- William Davis's evening snack
(23, 16, 'Mozzarella sticks', 350),
-- William Davis's dinner
(24, 17, 'Lasagna', 850),

-- Emily Wilson's breakfast
(25, 18, 'Yogurt and Granola', 200),
(26, 18, 'Orange Juice', 80),
-- Emily Wilson's dinner
(27, 19, 'Fried Rice', 350),
```

```
-- Micheal Anthony's breakfast
(28, 20, 'Fried Chicken and Waffles', 800),
-- Micheal Anthony's lunch
(29, 21, 'Shrimp fried rice', 850),
-- Micheal Anthony's evening snack
(30, 22, 'Candy Bar', 250),
(31, 22, 'Potato Chips', 150),
-- Micheal Anthony's dinner
(32, 23, 'Beef burger with cheese', 700),
(33, 23, 'Chocolate Ice Cream', 300),

-- Olivia Anderson's breakfast
(34, 24, 'Eggs and Toast', 300),
(35, 24, 'Orange Juice', 80),
-- Olivia Anderson's lunch
(36, 25, 'Turkey and cheese wrap', 350),
-- Olivia Anderson's dinner
(37, 26, 'Grilled Salmon', 300),

-- David Thomas's breakfast
(38, 27, 'Bagel with Cream Cheese', 450),
(39, 27, 'Apple Juice', 150),
-- David Thomas's morning snack
(40, 28, 'Cheese Puffs', 130),
-- David Thomas's lunch
(41, 29, 'Meatball sub', 800),
(42, 29, 'Baked Potato', 200),
-- David Thomas's evening snack
(43, 30, 'Cookies', 200),
-- David Thomas's dinner
(44, 31, 'Beef stroganoff', 600),
(45, 31, 'Chocolate Milkshake', 200),

-- Sophia White's breakfast
(46, 32, 'Egg and Vegetable Scramble', 300),
```

```
(47, 32, 'Apple juice', 117),
-- Sophia White's lunch
(48, 33, 'Apple and Peanut Butter Sandwich', 250),
-- Sophia White's dinner
(49, 34, 'Grilled salmon with roasted vegetables', 400),
(50, 34, 'Lemonade', 120)
;

UPDATE Meal SET Meal_calories = (
    SELECT SUM(Food_calories)
    FROM Food
    WHERE Food.Meal_ID = Meal.Meal_ID
);

INSERT INTO Category (category_id, category_name) VALUES
('FRT', 'Fruits'),
('VEG','Vegetables'),
('GRN','Grains'),
('PRT', 'Protein Foods'),
('DAP','Dairy Products');

INSERT INTO Ingredient(Ingredient_id, Ingredient_name, Ingredient_calorie,
Category_id) VALUES
(1,'Rolled Oats', 150, 'GRN'),
(2,'Egg', 70,'PRT'),
(3,'Chicken', 284, 'PRT'),
(4,'Lettuce', 8,'VEG'),
(5,'Cheese', 122, 'DAP'),
(6,'Potatoes', 130, 'VEG'),
(7,'Salmon Fillet', 412, 'PRT'),
(8, 'Banana', 105, 'FRT'),
(9, 'Almond Butter', 95, 'PRT'),
(10, 'Beef', 350, 'PRT'),
(11, 'Bread', 100, 'GRN'),
(12, 'Broccoli', 55, 'VEG'),
(13, 'Carrots', 52, 'VEG'),
```

```
(14, 'Turkey', 350, 'PRT'),
(15, 'Bagel', 250, 'GRN'),
(16, 'Bun', 120, 'GRN'),
(17, 'Garlic', 4, 'VEG'),
(18, 'Butter', 102, 'DAP'),
(19,'Pepperoni', 140, 'PRT'),
(20, 'Pizza Dough', 280, 'GRN'),
(21, 'Ham', 220, 'PRT'),
(22, 'Fish Fillet', 250, 'PRT'),
(23, 'Pancake Mix', 150, 'GRN'),
(24, 'Milk', 8, 'DAP'),
(25, 'Tomato Sauce', 50, 'VEG'),
(26, 'Lasagna Noodles', 200, 'GRN'),
(27, 'Yogurt', 60, 'DAP'),
(28, 'Rice', 200, 'GRN'),
(29, 'Orange', 70, 'FRT'),
(30, 'Flour',300, 'GRN'),
(31, 'Shrimp', 150, 'PRT'),
(32, 'Chocolate', 100, 'PRT'),
(33, 'Apple', 80, 'FRT'),
(34,'Peanut Butter', 190, 'GRN'),
(35, 'Lemon', 80, 'FRT')
;

INSERT INTO Food_Ingredient (food_id, ingredient_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(4, 5),
(4, 6),
(5, 6),
(6, 7),
(7, 8),
(7, 9),
```

```
(8, 5),
(8, 10),
(8, 11),
(9, 12),
(9, 13),
(10, 14),
(11, 5),
(11, 15),
(12, 30),
(12, 32),
(13, 5),
(13, 10),
(13, 16),
(13, 25),
(13, 4),
(14, 6),
(15, 17),
(15, 5),
(15, 11),
(16, 19),
(16, 20),
(16, 5),
(17, 21),
(17, 5),
(17, 11),
(17, 4),
(18, 22),
(18, 6),
(19, 23),
(19, 18),
(20, 30),
(20, 18),
(21, 5),
(21, 20),
(21, 25),
```

```
(22, 3),
(22, 5),
(22, 11),
(22, 25),
(23, 5),
(23, 25),
(23, 20),
(24, 25),
(24, 26),
(24, 10),
(25, 27),
(25, 28),
(25, 32),
(26, 29),
(27, 28),
(27, 13),
(27, 2),
(28, 3),
(28, 30),
(29, 31),
(29, 28),
(30, 32),
(30, 24),
(31, 6),
(32, 5),
(32, 10),
(32, 16),
(32, 25),
(32, 4),
(33, 32),
(33, 24),
(34, 2),
(34, 11),
(34, 18),
(35, 29),
```

```
(36, 14),
(36, 5),
(36, 30),
(36, 25),
(37, 7),
(38, 5),
(38, 15),
(39, 33),
(40, 5),
(40, 18),
(40, 30),
(41, 10),
(41, 5),
(41, 11),
(42, 6),
(43, 32),
(43, 30),
(44, 10),
(45, 32),
(45, 24),
(46, 2),
(46, 4),
(46, 12),
(46, 13),
(47, 33),
(48, 33),
(48, 34),
(48, 11),
(49, 7),
(49, 4),
(49, 12),
(49, 13),
(50, 35);
```

Task 5 – Queries

a. Enable the user to insert, update, delete a record.

Inserting a new client

```
INSERT INTO Client (Client_id, First_name, Middle_name, Last_name,
Date_of_birth, Sex, Contact_no, Height, Weight)

VALUES (11, 'Linda', 'Sue', 'Johnson', '1990-08-20', 'Female', '+1-555-123-
4567', 1.65, 75.0);
```

Updating an existing record in client

```
UPDATE Client

SET Client.Contact_no = '+1-555-111-1111', Client.Weight = 71.0

WHERE Client.Client_id = 1;
```

Deleting a record in client

```
DELETE FROM Client

WHERE Client.Client_id = 11;
```

b. Show information of all the clients

```
Select * from Client;
```

Client_id	First_name	Middle_name	Last_name	Date_of_birth	Sex	Contact_no	Height	Weight	Age	BMI
1	John	Robert	Doe	1980-05-25	Male	+1-555-111-1111	1.80	71.00	42	21.91
2	Jane	Marie	Smith	1992-02-14	Female	+1-555-987-6543	1.65	60.00	31	22.04
3	Peter	David	Jones	1975-11-01	Male	+1-555-555-5555	1.75	80.00	47	26.12
4	Amy	Elizabeth	Brown	1988-07-10	Female	+1-555-111-2222	1.60	45.00	34	17.58
5	William	Jacob	Davis	1982-03-08	Male	+1-555-444-3333	1.90	110.00	40	30.47
6	Emily	Grace	Wilson	1995-09-17	Female	+1-555-222-1111	1.55	40.00	27	16.65
7	Michael	Anthony	Taylor	1979-12-23	Male	+1-555-777-8888	1.85	90.00	43	26.30
8	Olivia	Madison	Anderson	1990-06-05	Female	+1-555-333-4444	1.70	70.00	32	24.22
9	David	Lee	Thomas	1970-04-19	Male	+1-555-222-3333	1.80	105.00	52	32.41
10	Sophia	Avery	White	2000-01-02	Female	+1-555-666-7777	1.60	55.00	23	21.48

c. Show meal information of a specific client for a specific date

```
SELECT Meal.Meal_id, Meal.Meal_time, Meal.Meal_type, Food.Food_id,
Food.Food_name, Food.Food_calories

FROM Meal

INNER JOIN Food ON Meal.Meal_id = Food.Meal_id

WHERE Meal.Client_id = 1 AND Meal.Meal_date='2023-02-26';
```

Meal_id	Meal_time	Meal_type	Food_id	Food_name	Food_calories
1	08:00:00	Breakfast	1	Oatmeal	158
1	08:00:00	Breakfast	2	Scrambled Eggs	326
2	12:30:00	Lunch	3	Grilled chicken breast	284
2	12:30:00	Lunch	4	Caesar salad	250
3	18:30:00	Dinner	5	Mashed Potatoes	250
3	18:30:00	Dinner	6	Grilled Salmon	412

d. Show information of the foods which their calories are less than the average of foods registered in the database

```
SELECT Food_name, Food_calories FROM Food
WHERE Food_calories < (SELECT AVG(Food_calories) FROM Food);
```

food_name	food_calories
Oatmeal	158
Scrambled Eggs	326
Grilled chicken breast	284
Caesar salad	250
Mashed Potatoes	250
Banana and Almond Butter Smoothie	200
Vegetable stir-fry	250
Roasted Turkey	350
Donut	195
French Fries	312
Garlic bread with cheese	300
Pizza (pepperoni)	350
Ham and Swiss sandwich	350
Cinnamon Roll	310
Pizza rolls	220
Mozzarella sticks	350
Yogurt and Granola	200
Orange Juice	80
Fried Rice	350
Candy Bar	250
Potato Chips	150
Chocolate Ice Cream	300
Eggs and Toast	300
Orange Juice	80
Turkey and cheese wrap	350
Grilled Salmon	300
Apple Juice	150
Cheese Puffs	130
Baked Potato	200
Cookies	200
Chocolate Milkshake	200
Egg and Vegetable Scramble	300
Apple juice	117
Apple and Peanut Butter Sandwich	250
Lemonade	120

e. Show the name of the client who consumed the highest amount of calorie in a specific date

```
SELECT c.First_name, c.Middle_name, c.Last_name
FROM Client c
INNER JOIN Meal m ON c.Client_id = m.Client_id
INNER JOIN Food f ON m.Meal_id = f.Meal_id
```

```
WHERE m.Meal_date = '2023-02-26'
GROUP BY c.Client_id
ORDER BY SUM(f.Food_calories) DESC
LIMIT 1;
```

First_name	Middle_name	Last_name
William	Jacob	Davis

f. Show information of the youngest and oldest client

```
SELECT * FROM Client
WHERE Date_of_birth = (SELECT MIN(Date_of_birth) FROM Client)
   OR Date_of_birth = (SELECT MAX(Date_of_birth) FROM Client);
```

Client_id	First_name	Middle_name	Last_name	Date_of_birth	Sex	Contact_no	Height	Weight	Age	BMI
9	David	Lee	Thomas	1970-04-19	Male	+1-555-222-3333	1.80	105.00	52	32.41
10	Sophia	Avery	White	2000-01-02	Female	+1-555-666-7777	1.60	55.00	23	21.48

g. For each food that their calorie is higher than average, show the number of clients who consumed that food.

```
SELECT f.Food_name, COUNT(DISTINCT m.Client_id) AS Num_clients
FROM Food f
JOIN Meal m ON f.Meal_id = m.Meal_id
WHERE f.Food_calories > (SELECT AVG(Food_calories) FROM Food)
GROUP BY f.Food_id;
```

Food_name	Num_clients
Grilled Salmon	1
Roast beef and cheddar sandwich	1
Bagel with Cream Cheese	1
Cheeseburger	1
Fish and Chips	1
Pancakes with Butter and Syrup	1
Chicken cheese sandwich	1
Lasagna	1
Fried Chicken and Waffles	1
Shrimp fried rice	1
Beef burger with cheese	1
Bagel with Cream Cheese	1
Meatball sub	1
Beef stroganoff	1
Grilled salmon with roasted veget...	1

Note: *The same clients who consumed multiple high-calorie foods may be included in this query. This query reveals the foods high in calories as well as who consumed them. In the database there's only one client for each high-calorie food as we tried to keep unique foods for each client.*

h. For each ingredient in the database, show the average of calories of the foods that contain that specific ingredient.

```
SELECT i.Ingredient_id, i.Ingredient_name, AVG(f.Food_calories) AS
Avg_calories

FROM Ingredient i

INNER JOIN Food_ingredient fi ON i.Ingredient_id = fi.ingredient_id

INNER JOIN Food f ON fi.food_id = f.Food_id

GROUP BY i.Ingredient_id;
```

Ingredient_id	Ingredient_name	Avg_calories
1	Rolled Oats	158.0000
2	Egg	319.0000
3	Chicken	678.0000
4	Lettuce	458.3333
5	Cheese	456.6667
6	Potatoes	277.0000
7	Salmon Fillet	370.6667
8	Banana	200.0000
9	Almond Butter	200.0000
10	Beef	691.6667
11	Bread	485.7143
12	Broccoli	316.6667
13	Carrots	325.0000
14	Turkey	350.0000
15	Bagel	450.0000
16	Bun	725.0000
17	Garlic	300.0000
18	Butter	315.0000
19	Pepperoni	350.0000
20	Pizza Dough	306.6667
21	Ham	350.0000
22	Fish Fillet	500.0000
23	Pancake Mix	520.0000
24	Milk	250.0000
25	Tomato Sauce	595.7143
26	Lasagna Noodles	850.0000
27	Yogurt	200.0000
28	Rice	466.6667
29	Orange	80.0000
30	Flour	330.8333
31	Shrimp	850.0000
32	Chocolate	224.1667
33	Apple	172.3333
34	Peanut Butter	250.0000
35	Lemon	120.0000

i. Show the average calorie consumption of females and males.

```
SELECT c.Sex, AVG(f.Food_calories) AS Avg_calorie_consumption
FROM Client c
JOIN Meal m ON c.Client_id = m.Client_id
JOIN Food f ON m.Meal_id = f.Meal_id
GROUP BY c.Sex;
```

Sex	Avg_calorie_consumption
Male	406.7813
Female	274.8333

j. Show what was the most consumed food, along with its calorie, category, number of people consumed.

```
SELECT
  f.Food_name,
  SUM(f.Food_calories) AS Total_calories,
  c.Category_name,
  COUNT(DISTINCT m.Client_id) AS Num_of_people_consumed
FROM
  Food f
  JOIN Food_ingredient fi ON f.Food_id = fi.food_id
  JOIN Ingredient i ON fi.ingredient_id = i.Ingredient_id
  JOIN Category c ON i.Category_id = c.Category_id
  JOIN Meal m ON f.Meal_id = m.Meal_id
GROUP BY
  f.Food_name
ORDER BY
  COUNT(DISTINCT m.Client_id) DESC
LIMIT 1;
```

Food_name	Total_calories	Category_name	Num_of_people_consumed
Apple Juice	267	Fruits	2

k. Calculate weight status of each client.

```
SELECT First_name, Middle_name, Last_name, Weight, Height, BMI,
CASE
    WHEN BMI < 18.5 THEN 'Underweight'
    WHEN BMI >= 18.5 AND BMI < 25 THEN 'Healthy '
    WHEN BMI >= 25 AND BMI < 30 THEN 'Overweight'
```

```
    ELSE 'Obese'
END AS Weight_status
FROM Client;
```

First_name	Middle_name	Last_name	Weight	Height	BMI	Weight_status
John	Robert	Doe	71.00	1.80	21.91	Healthy
Jane	Marie	Smith	60.00	1.65	22.04	Healthy
Peter	David	Jones	80.00	1.75	26.12	Overweight
Amy	Elizabeth	Brown	45.00	1.60	17.58	Underweight
William	Jacob	Davis	110.00	1.90	30.47	Obese
Emily	Grace	Wilson	40.00	1.55	16.65	Underweight
Michael	Anthony	Taylor	90.00	1.85	26.30	Overweight
Olivia	Madison	Anderson	70.00	1.70	24.22	Healthy
David	Lee	Thomas	105.00	1.80	32.41	Obese
Sophia	Avery	White	55.00	1.60	21.48	Healthy

l. For each client weight status (underweight, healthy, overweight, obese) show the average time of the day that have consume their food on a specific date.

```
SELECT
    CASE
        WHEN BMI < 18.5 THEN 'Underweight'
        WHEN BMI >= 18.5 AND BMI < 25 THEN 'Healthy'
        WHEN BMI >= 25 AND BMI < 30 THEN 'Overweight'
        ELSE 'Obese'
    END AS Weight_status,
    AVG(EXTRACT(HOUR FROM meal_time)) AS Avg_meal_hour
FROM
    client
    INNER JOIN meal ON client.client_id = meal.client_id
GROUP BY
    Weight_status;
```

Weight_status	Avg_meal_hour
Healthy	12.9167
Overweight	13.6250
Underweight	14.5000
Obese	13.1000

m. Show the ingredient which is used mostly in a specific week for obese clients

```
SELECT i.Ingredient_name, COUNT(fi.ingredient_id) AS Count
FROM Food f
JOIN Meal m ON f.Meal_id = m.Meal_id
```

```
JOIN Client c ON m.Client_id = c.Client_id
JOIN Food_ingredient fi ON f.Food_id = fi.food_id
JOIN Ingredient i ON fi.ingredient_id = i.Ingredient_id
WHERE m.Meal_date BETWEEN '2023-02-21' AND '2023-02-27'
AND c.BMI >= 30 -- obese clients only
GROUP BY i.Ingredient_name
ORDER BY count DESC
LIMIT 1;
```

Ingredient_name	Count
Cheese	6

User Guide

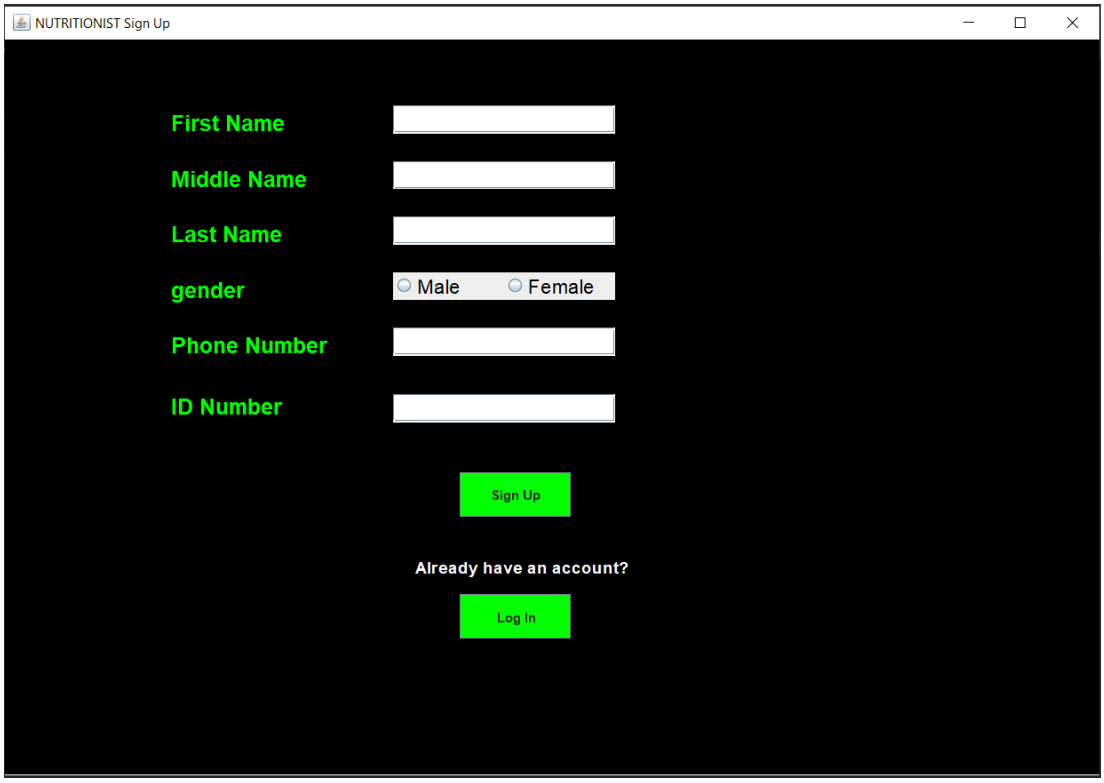
Open the **Nutrition_database.sql** file in MySQL Workbench to access the database. The user can run all the queries simultaneously or one at a time as they see fit because the file contains all the tables and queries. They can also add more queries if they require as the database is extremely flexible.

Bonus Features

A separate database was used for the GUI implementaion of the project.

User Guide for JAVA GUI:

1. Extract the project from the zip file and store in folder, then use your preferable IDE to open it.
2. Open the main.java file and run it, you will get the Sign Up page :

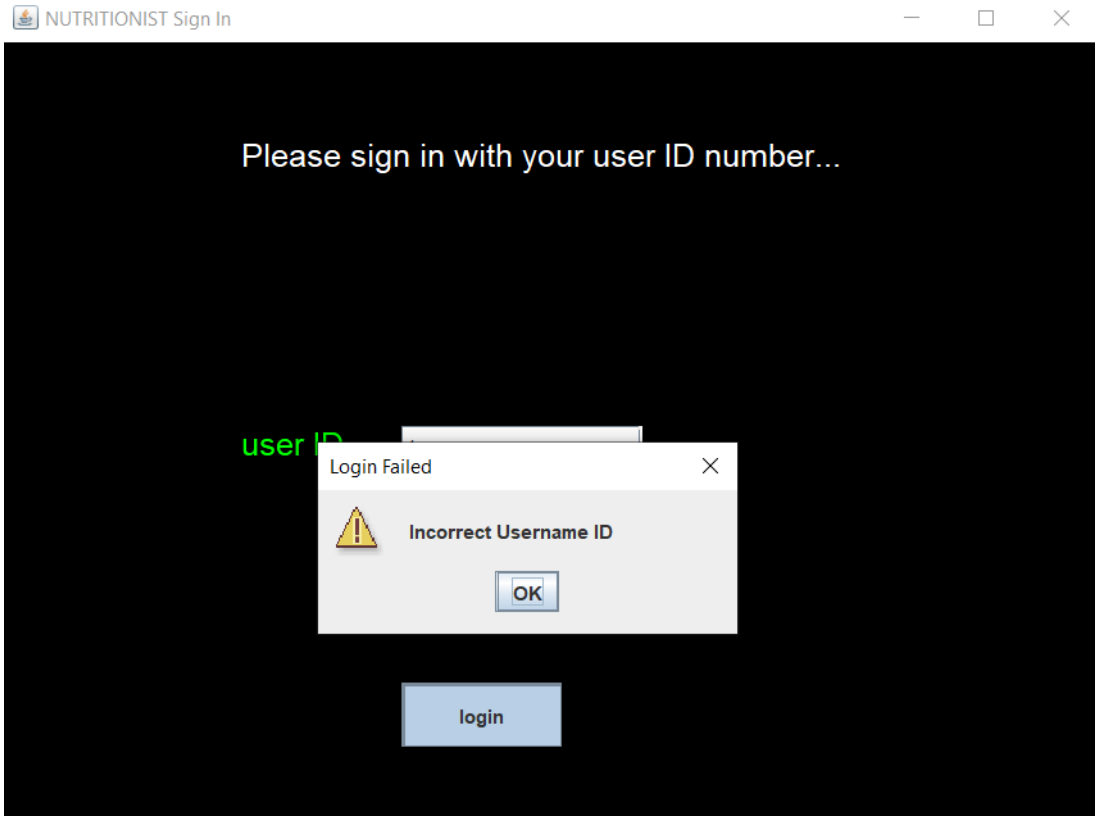


Enter your information and press Sign Up and the information will be stored in the database:

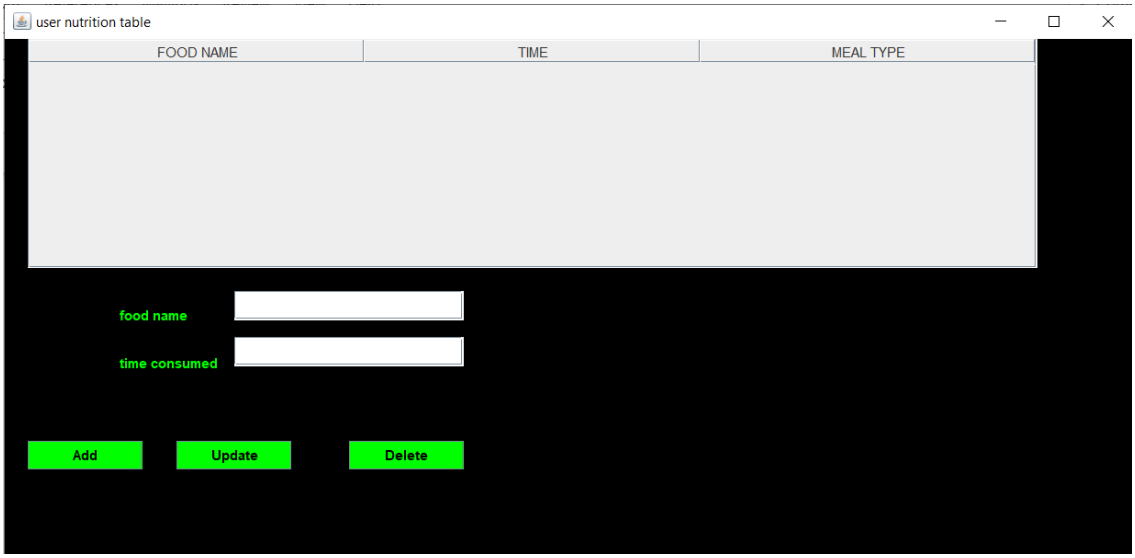
First_name	Middle_name	Last_name	Sex	phone_num	Client_id
sara	medhat	mashaal	Female	1248345	123
NULL	NULL	NULL	NULL	NULL	NULL

3. You will go next to the Sign In page:

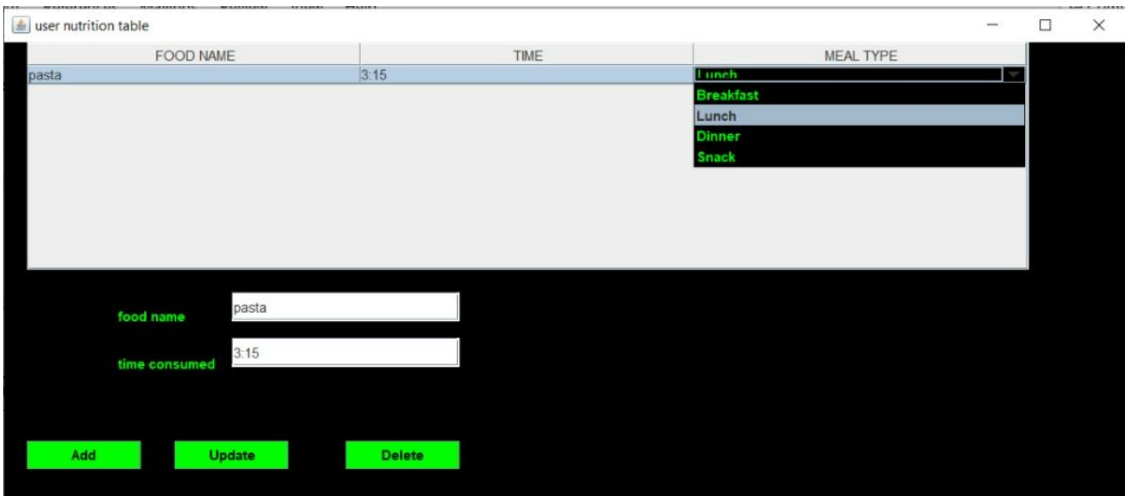
Enter the ID number that you entered earlier in the Sign Up page, if it was incorrect you wil get the following error message:



4. If the login was successful you will go to the next page where you can enter the meal intake information and it will all be updated in the database:



To add a new record enter the values in both of the text boxes and press the Add button, for the meal type select your meal from the combo box:



For updating and deleting, select the entire row with your mouse and put new values for the delete and press the update button , and for the delete press the delete button.

Summary of work

Both of the team members worked on the EER diagram and EER diagram to relational model conversion. Hiba Hassan worked on the MySQL database implementation while Sara Mashaal worked on the GUI Bonus Application using JAVA programming language.