
# CMP257: Web Application Programming

## Spring 2025

## Section #2
## Group #15

## Title:

## SmartMealz

**Submitted by:**


| Name | ID |
|---|---|
| Ahmed Mehaisi | b00094989 |
| Haider Raza | b00096026 |
| Mohammed Imtiaz | b00097219 |
| Mohamed Alawadhi | b00094286 |

# Table of Contents

## 1. Introduction

### a. Description of the application idea

SmartMealz is a dynamic, web-based meal planning platform designed to simplify healthy eating by tailoring recommendations to each user's body profile. It features an interactive Body Assessment page that calculates BMI and caloric needs, then suggests meals from a categorized menu (high-protein, balanced, plant-based, and desserts). Users browse attractive meal cards, add items to a cart, and complete a streamlined checkout. The core idea is to make personal nutrition effortless and ingenious, hence the name SmartMealz.

### b. Proposed features and functionalities

- **Body Assessment Form**: Users input height, weight, age, and gender to calculate BMI and daily caloric needs; supports tracking of food intake.
- **Meal Recommendations**: Dynamically filters meal options by BMI category (underweight, normal, overweight) and user goals (gain, maintain, lose weight).
- **Meals Display**: Grid of interactive meal cards showing name, macros (protein/carbs/fat), calories, price, and image.
- **Cart & Checkout**: Summary page with selected meals, quantities, total price, and order confirmation.
- **Responsive Design**: Layout adapts to desktop, tablet, and mobile using CSS media queries and Bootstrap grid.
- **User Authentication (Optional Extension)**: Sign-up/login screens with encrypted passwords.
- **Language Translation**: Google Translate API integration for multilingual support.

**c. Roles and responsibilities of each group member**

- **Haider Raza** – Full Stack Developer + SQL Integration report setup and drafting
- **Mohammed Imtiaz** – Back-end logic, Java Spring Boot setup, controller routing, Git version control, report specifics
- **Ahmed Mehaisi** – Back-end development, UI structure, database design, ER diagrams, SQL integration
- **Mohamed Alawadhi** – Front-end development (HTML, CSS), testing, bug fixing, report review

**2. Design Phase**

**a. Overview of the initial design of the application**

The initial concept for SmartMealz centered around personalizing meal plans based on user preferences. However, the idea evolved to focus on **automated health-based recommendations** using BMI as the driving input. This shift led to a streamlined design with three main flows:

1. **Body Assessment Input**,

2. **Meal Recommendations**,

3. **Checkout Summary**.

   The design emphasizes clarity and responsiveness, allowing users to quickly understand their health range and receive meals suited to their profile.

**b. Entity-relationship (ER) Diagrams**

(See Appendix figure 1)

The system consists of the following key entities:

- **User**: id, email, password, height, weight, bmi, created_at
- **Meal**: id, name, category, calories, protein, carbs, fat, price, image_url
- **Order**: id, user_id, order_date, status
- **OrderItem**: id, order_id, meal_id, quantity

**c. Relational database schema mapping**

(See Appendix figure 2)

SmartMealz's data consists of **five** tables that mirror our domain model. `users` hold the login credentials (email , UNIQUE, Bcrypt-hashed password) and act as the parent for everything else. Each time a user records their body metrics, a new row is written to `bmi_records`, linked by a foreign key with `ON DELETE CASCADE,` so deleting an account automatically removes its history. Meals are stored once in `meals`, so we can sort and filter efficiently. An order is split into two tables—`user_orders` (the header) and `order_items` (the line items)—giving us a

classic 1-to-many header-detail pattern and letting us run fast roll-ups like “total calories in an order.”

**d. Wireframes/mockups or screenshots of the user interface**

Include mockups or screenshots of:

- **Body Assessment Page**: Input form for height & weight with real-time BMI display.
- **Meals Page**: Responsive grid layout using Thymeleaf loops.
- **Checkout Summary**: Table listing selected meals, quantities, and totals.

**3. Development Phase**

**a. Front-end Development**

**i. Technologies and tools used for web design:**

- **HTML5** – for structuring the web pages
- **CSS3** – for custom styling (selectplan.css, meals.css, etc.)
- **Bootstrap 5** - utility classes for navigation bar.
- **JavaScript** - Button functionality and event Listeners

**ii. Description of the user interface and user experience design:**

The interface is designed for simplicity and ease of use. Key design elements include:

- A clean, fixed **navigation bar** with links to Home, Body Assessment, Meals, and Checkout.
- A **BMI calculator form** where users input their height and weight to get instant feedback.
- A **meal display grid** with attractive meal cards showing meal names, calories, macros, and prices.
- A **cart/checkout summary** that visually displays selected meals before order confirmation.

**iii. Code snippets/screenshots:**

```
Python
<!-- meals.html snippet -->
<div class="row">
  <div class="col-md-4" th:each="meal : ${meals}">
      <div class="card meal-card">
      <img th:src="@{${meal.imageUrl}}" class="card-img-top">
      <div class="card-body">
      <h5 class="card-title" th:text="${meal.name}"></h5>
      <p th:text="${meal.calories} + ' kcal'"></p>
      <button class="btn btn-primary add-cart"
data-id="[[${meal.id}]]">Add to Cart</button>
      </div>
      </div>
  </div>
</div>
```

(See Appendix figure 4)

**b. Back-end Development**

**i. Description of the object-oriented design:**

Data access is done using repository interfaces or raw SQL using JdbcTemplate. Controllers serve as liaisons between the front and back ends, processing HTTP requests and managing service-layer actions. This layered approach improves readability, testability and is consistent with typical MVC (Model-View-Controller) design concepts.

Our Java back end is cleanly organized into models, services, repositories, and controllers, and uses key OOP principles like inheritance. For example, we created a BaseEntity class (with fields like id and createdAt) that both Meal and OrderEntry (along with User and BMIRecord) extend. That way, all our entities share common properties without repeating code. Business logic like filtering meals by calories or saving BMI records lives in service classes behind interfaces, while controllers simply route web requests to those services. We wire everything together with dependency, validate inputs (e.g., no negative weights), and have a single error handler to catch and consistently display problems. With clear method names and Javadoc comments throughout, the result is a back end that is easy to follow, maintain, and extend.

**ii. Implementation of multi-threading (if applicable):**

No explicit multi-threading used. Spring Boot, which uses Jakarta Servlets, already implements multithreading..

**iii. Frameworks used:**

- **Spring Boot** – for handling HTTP requests, MVC routing, application setup, dependency management, and embedded Tomcat.
- **Thymeleaf** – for linking Java data objects into HTML pages. (Java template engine)
- **Spring MVC** – to separate front-end and back-end responsibilities, and for model-view-controller patterns.

**iv. Java Server design:**

The Java server is designed with the Spring Boot framework, which makes it easier to create robust and scalable online applications. The server listens for HTTP requests on port 8080 and forwards them to the relevant controllers depending on the URL mappings. Error handling and redirects are used to handle faulty requests, missing data, and illegal access gracefully. The design encourages clear separation of interests, modularity, and ease of maintenance.

**c. Database Implementation**

**i. Table organization:** (See Appendix figure 5)

- **User** – stores the user's email and password
- **Meal** – stores meal details like name, calories, macros, and price
- **Order** - stores meal quantities and price from the meal ID into an entry
- **User_Orders** - stores records of orders placed by a user
- **Bmi_records** - stores BMI record(s) of user

**ii. Key SQL queries/examples:**

Read:

Select * from meals;

Insert:

INSERT INTO Meals (name, description, category, calories, protein, carbs, fat, price, image_url) VALUES

('Grilled Chicken Breast', 'Chicken breast with steamed vegetables', 'high-protein', 400, 40, 10, 12, 35, '/img/grilled_chicken_breast.png')

Update:

UPDATE user_orders SET status = 'completed' WHERE user_id = ? AND status = 'draft'

Delete:

DELETE FROM bmi_records WHERE user_id = ?

**4. Integration and Testing**

**a. Overview of the relationship database setup**

**MySQL/InnoDB with UTF8-MB4:** Ensures ACID compliance and full Unicode support.

**Relationships:**

| **Entity** | **Relationship** |
|---|---|
| users → bmi_records | One-to-Many |
| users → user_orders | One-to-Many |
| user_orders → Orders | One-to-one |
| Meals → Orders | One-to-Many |

**Integrity & Performance:**

- Foreign-key cascades delete dependent records when a user is removed.

- Indexed FKs optimize joins for filtering meals by calorie thresholds and retrieving order details.

**b. Description of how the front-end and back-end were integrated**

SpringBoot + Thymeleaf templates bind Model attributes from controllers
Bootstrap + overridden custom CSS applied once data arrives

**c. End-to-end test cases**

Testing was conducted as an iterative process to ensure SmartMealz's functionality and user experience met our design goals. Each core feature such as BMI assessment, meal recommendation filtering, and cart management was repeatedly exercised: for example, we entered a variety of height/weight combinations to validate BMI calculations and calorie thresholds, then navigated through the meals page to confirm that only appropriately categorized cards appeared and proceeded to place an order, all while monitoring our tables. End-to-end flows were exercised across all layers, from the Thymeleaf-driven front end through Spring Boot controllers to the MySQL database. After each round of manual testing, any discrepancies in UI layout, business logic, or data persistence were logged, addressed in code, and re-tested. This continuous loop of testing and refinement helped us catch edge cases (e.g., zero-quantity orders, boundary-value BMIs) and ensured the application remained robust, intuitive, and responsive throughout development.

**5. Additional Features and/or Technologies**

- **T**ranslate dropdown menu - Utilization of Google's translation API to translate the page.

- Log in and sign up (sign out/ delete) using Cookies.
- Encryption for the passwords to store in the DB (instead of plain text).

## 6. Conclusion

SmartMealz is a web application that provides personalized meal planning based on BMI and user inputs, featuring responsive design, database integration, and even language translation. Developing it reinforced our understanding of JavaScript, Java, and Spring Boot while introducing new concepts. Given its complexity, this project could serve as a senior project or be split into separate front-end and back-end courses, one focusing on user interface and the other on advanced server-side functionality.

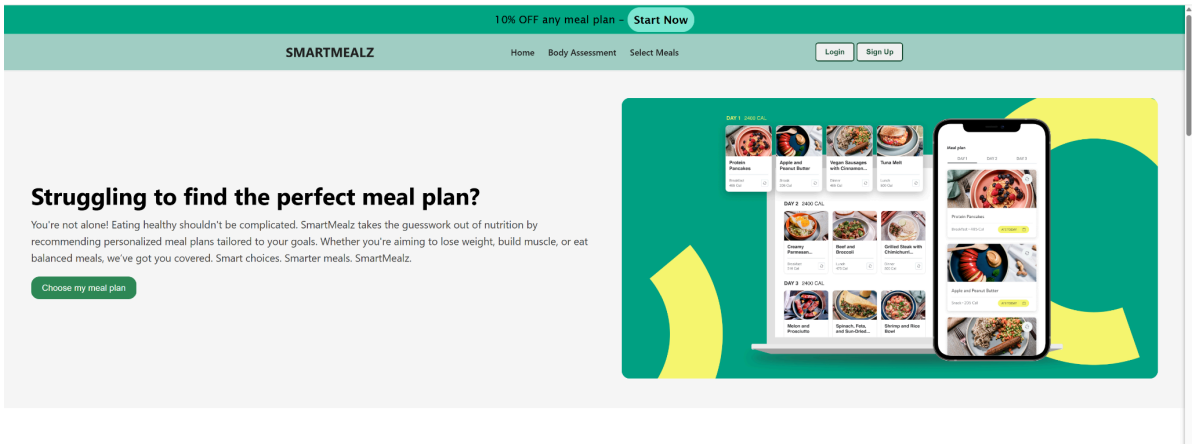
## 7. References


1. Google Translate button - https://codepen.io/barteciccio/pen/ExqwOVJ
2. Loading screen - https://codepen.io/tashfene/pen/raEqrJ
3. https://www.w3schools.com/

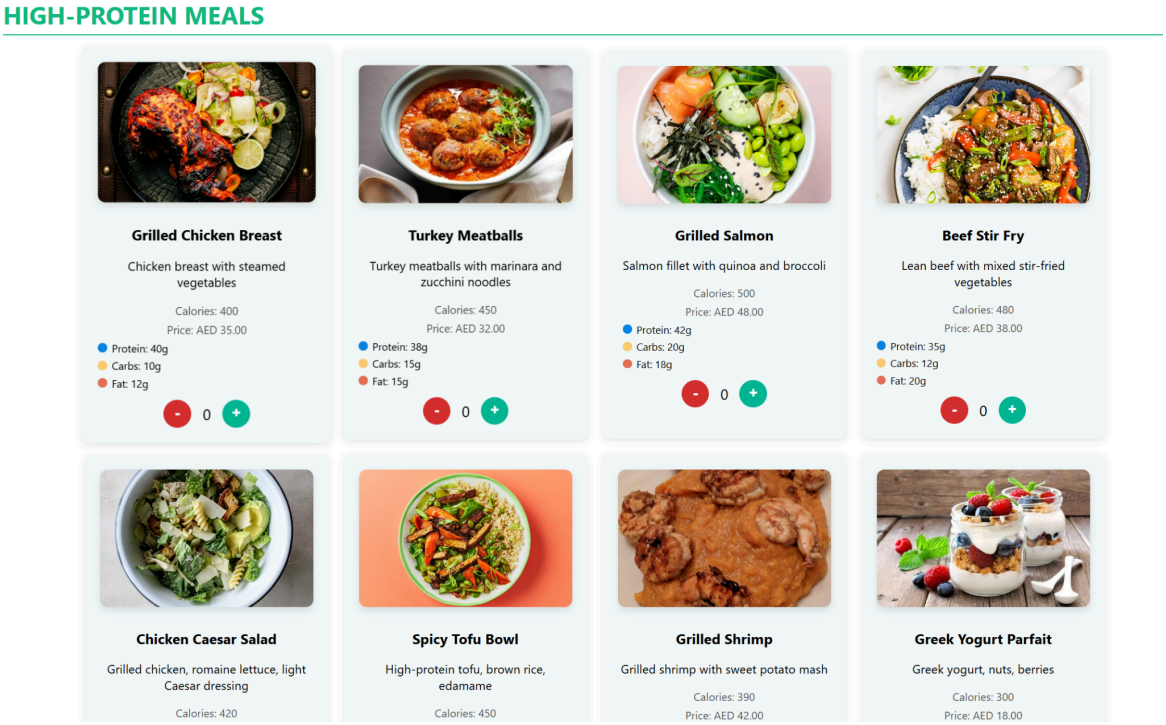

## 8. Appendix

1. **Chat GPT** - Used it for styling: fonts, generating the meals for the database, and backend setup.

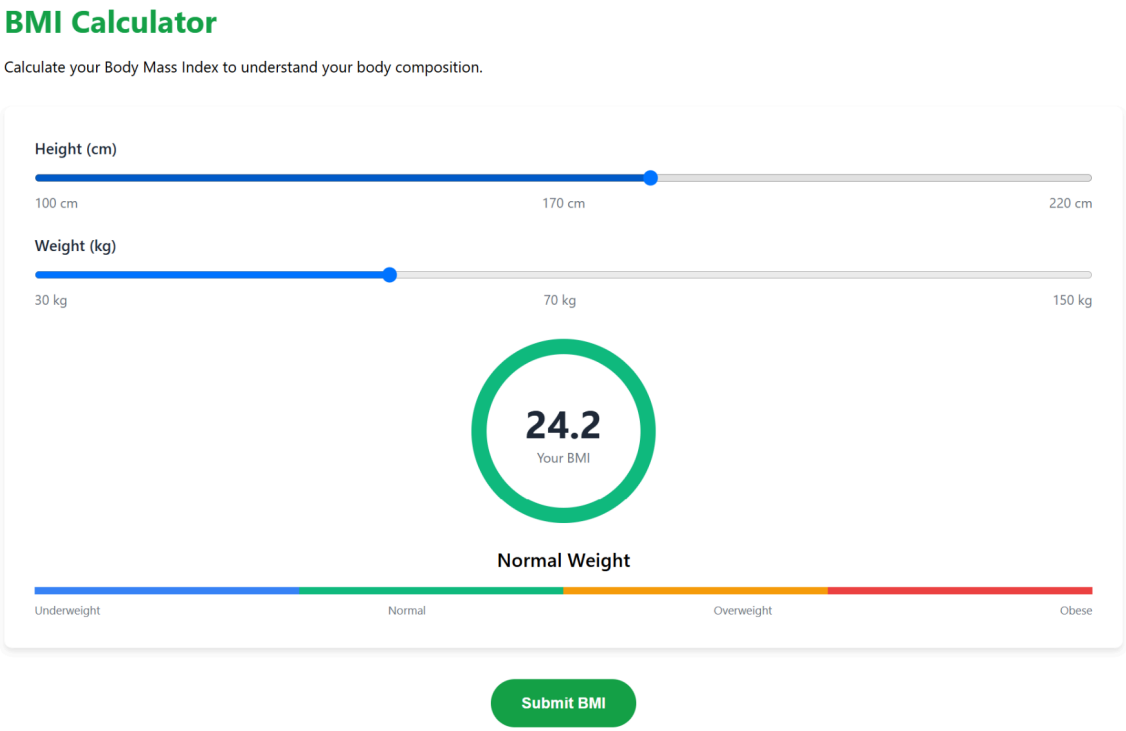
# Figures



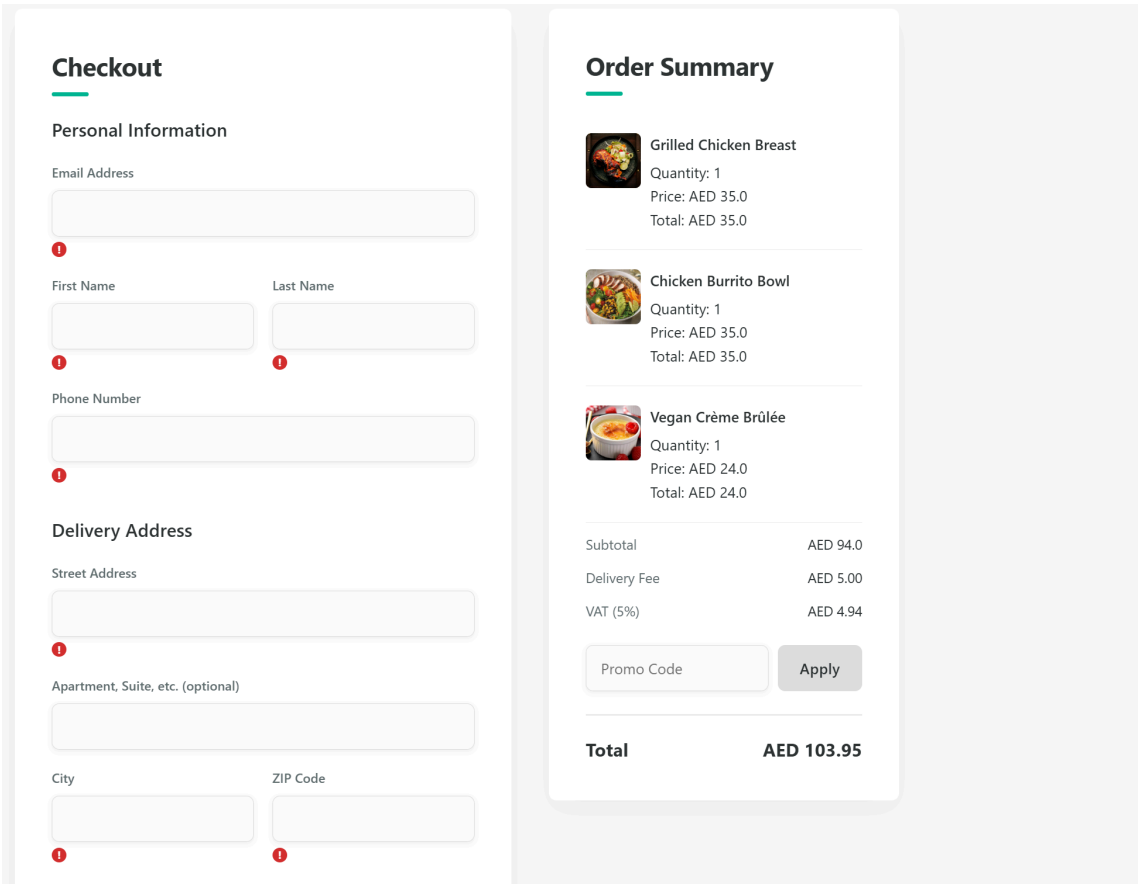

*(Figure 1.1 User Interface)*



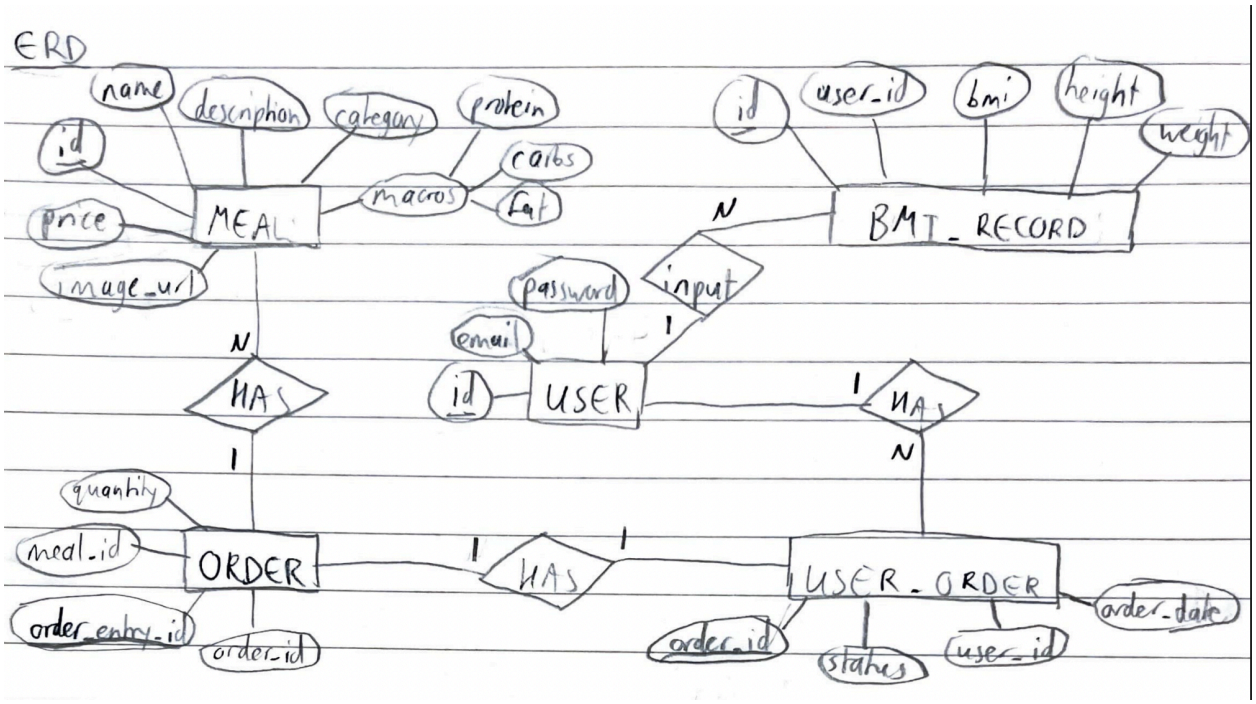

*(Figure 1.2)*


BMI Calculator
Calculate your Body Mass Index to understand your body composition.
Height (cm)
100 cm
170 cm
220 cm
Weight (kg)
30 kg
70 kg
150 kg
24.2
Your BMI
Normal Weight
Underweight
Normal
Overweight
Obese
Submit BMI

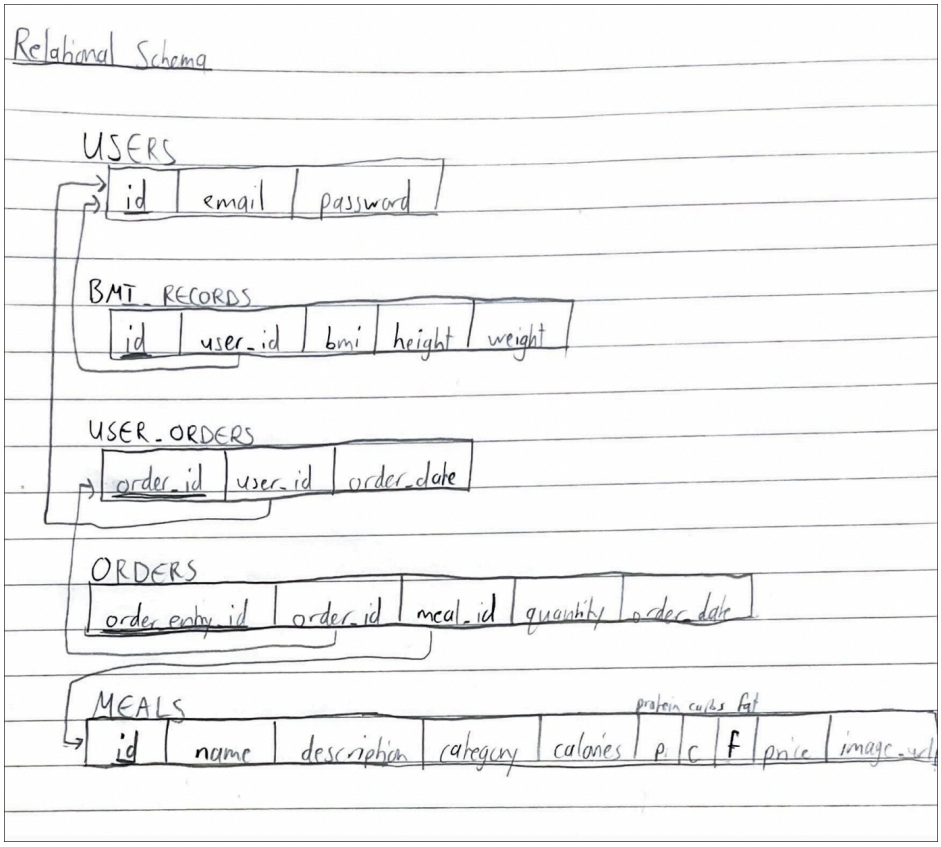
*(Figure 1.3)*


Checkout
Personal Information
Email Address
First Name
Last Name
Phone Number
Delivery Address
Street Address
Apartment, Suite, etc. (optional)
City
ZIP Code
Order Summary
Grilled Chicken Breast
Quantity: 1
Price: AED 35.0
Total: AED 35.0
Chicken Burrito Bowl
Quantity: 1
Price: AED 35.0
Total: AED 35.0
Vegan Crème Brûlée
Quantity: 1
Price: AED 24.0
Total: AED 24.0
Subtotal
AED 94.0
Delivery Fee
AED 5.00
VAT (5%)
AED 4.94
Promo Code
Apply
Total
AED 103.95

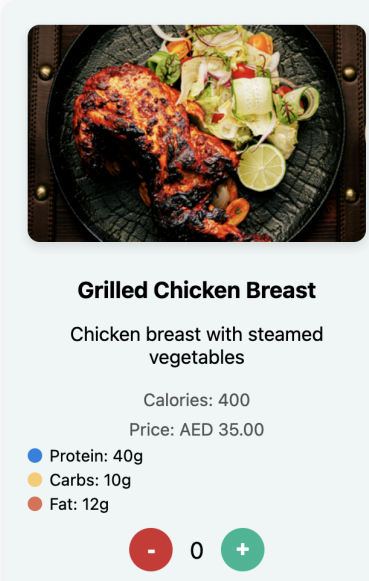
*(Figure 1.4)*


ERD
name
description
category
protein
id
carbs
macros
fat
price
MEAL
image_url
N
HAS
1
quantity
meal_id
ORDER
order_entry_id
order_id
id
user_id
bmi
height
weight
BMI_RECORD
N
input
1
password
email
id
USER
1
HAS
N
1
HAS
1
USER_ORDER
order_date
order_id
status
user_id

*(Figure 2 ERD)*




*(Figure 3 Schema)*




*(Figure 4 Card)*

```
CREATE TABLE users (
  id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  UNIQUE KEY email (email)
);
CREATE TABLE bmi_records (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  bmi VARCHAR(10),
  height VARCHAR(10),
  weight VARCHAR(10),
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
CREATE TABLE Meals (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  description TEXT,
  category VARCHAR(50),
  calories INT,
  protein INT,
  carbs INT,
  fat INT,
  price DECIMAL(6,2),
  image_url VARCHAR(255)
);
CREATE TABLE user_orders (
  order_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id BIGINT,
  order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
CREATE TABLE Orders (
  order_entry_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT,
  meal_id INT,
  quantity INT NOT NULL,
  order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (order_id) REFERENCES user_orders(order_id) ON DELETE CASCADE,
  FOREIGN KEY (meal_id) REFERENCES Meals(id) ON DELETE CASCADE
);
```

(Figure 5 *Tables creation)*

## 9. Group Project Contribution Agreement Form

**Project Title:** SmartMealz **Group Number:** 15

**Group Members – Make sure to select your registered lecture time**

1. Name: Ahmed Mehaisi ID: b00094989 Lec: 11:00
2. Name: Haider Raza ID: b00096026 Lec: 11:00
3. Name: Mohammed Imtiaz ID: b00097219 Lec: 11:00
4. Name: Mohamed Alawadhi ID: b00094286 Lec: 11:00

**Individual Contributions:**
(Describe briefly what each member contributed.)

| **Student Name** | **Contribution Summary** | **Estimated % of Total Work** |
|---|---|---|
| Ahmed Mehaisi | Back-end development, UI structure, database design, ER diagrams, SQL integration | 25% |
| Haider Raza | Front-end + dynamic backend development for select meals pages | 25% |
| Mohammed Imtiaz | Back-end logic, Java Spring Boot setup, controller routing, Git version control | 25% |
| Mohamed Alawadhi | Front-end development (HTML, CSS), testing, bug fixing | 25% |

**Total must add to 100%.**

---

**Group Agreement:**
We, Group 15, confirm that we agree with the contributions listed above and that the percentages reflect each member's work fairly.

| **Student Name** | **Signature** | **Date** |
|---|---|---|
| Ahmed Mehaisi | Ahmed Mehaisi | 2/05/2025 |
| Haider Raza | Haider Raza | 2/05/2025 |
| Mohammed Imtiaz | Mohamed Imtiaz | 2/05/2025 |
| Mohamed Alawadhi | Mohamed Alawadhi | 2/05/2025 |