

**DIETARY MANAGEMENT**

**A MINI PROJECT REPORT SUBMITTED BY**

**MOHANA PRIYA R - (230701191)**

**LAVANYA R – (230701162)**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023-2024

**BONAFIDE CERTIFICATE**

Certified that this project report **“DIETARY MANAGEMENT”** is the bonafide work of " **MOHANA PRIYA R - (230701191) & LAVANYA R – (230701162)”**

who carried out the project work under my supervision.

Submitted for practical examination held on

SIGNATURE

MS.Dharshini B S

Assistant Professor,

Computer Science and Engineering,

Rajalakshmi Engineering College (Autonomous),

Thandalam, Chennai-602 105

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who has contributed to the successful completion of this mini project.

First and foremost, I am deeply thankful to my Professor **Mrs. K. Maheshmeena** my project advisor, for their invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini project. Their expertise and encouragement have been instrumental in shaping my research and bringing this mini project to completion.

I would also like to express my appreciation to the faculty and staff of the **Computer Science and Engineering Department** at **Rajalakshmi Engineering College** for providing the necessary resources and a conducive learning environment. We express our sincere thanks to **Dr. P. Kumar, M.E., Ph.D.,** Professor and Head of the Department Computer Science and Engineering for his guidance and encouragement throughout the project work.

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support.

Thank you all for your contributions, both direct and indirect, to the success of this project.

**ABSTRACT:**

The Dietary Management System is a database-driven application designed to assist individuals in managing their dietary requirements effectively. This system aims to simplify the process of meal planning, tracking nutrition, and maintaining healthy eating habits by providing personalized recommendations based on user preferences, health goals, and nutritional requirements. It leverages a robust Database Management System (DBMS) to store, organize, and retrieve information related to various food items, recipes, dietary plans, and user profiles.

The system allows users to create and manage their profiles, input their dietary restrictions (such as allergies, preferences, or special health conditions), and receive personalized meal plans accordingly. It also tracks nutritional values such as calories, proteins, fats, carbohydrates, vitamins, and minerals, offering insights into the user's daily consumption. Additionally, the system supports the functionality for administrators to update food databases, including food types, nutrition facts, and recipe suggestions, ensuring up-to-date and accurate information for users.

The project employs SQL to efficiently manage the underlying relational database, with tables for users, food items, meal plans, and recipes. Queries are executed to fetch real-time recommendations, dietary tracking, and reports. The system aims to promote healthier eating habits by offering personalized dietary guidance and creating a structured, easy-to-navigate interface for both users and administrators.By integrating technology with nutrition science, the Dietary Management System serves as an effective tool for individuals striving to achieve and maintain a balanced, nutritious diet while enhancing their overall wellness and health.

**Table of Contents**

**1.INTRODUCTION**

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

**2. SURVEY OF TECHNOLOGIES**

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES

2.2.1 SQL

2.2.2 JAVA

**3. REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

**4. PROGRAM CODE**

**5. RESULTS AND DISCUSSION**

**6.CONCLUSION**

**1.1 Introduction**

The **Dietary Management System** is a software application designed to help users manage their daily nutrition, track meals, and achieve health goals. Using a Database Management System (DBMS), it stores and organizes data about food items, recipes, user profiles, and dietary restrictions. The system provides personalized meal plans and tracks nutritional intake, promoting healthier eating habits and supporting users in managing conditions like diabetes, obesity, and food allergies.

**1.2 Objectives**

The main objectives of the system are:

1. **Personalized Meal Planning**: Provide customized meal plans based on user preferences and health goals.
2. **Nutritional Tracking**: Track daily nutrition intake such as calories, proteins, fats, and vitamins.
3. **User Profile Management**: Allow users to create and update profiles, including health data and dietary preferences.
4. **Food Database Management**: Store and manage detailed food and recipe information, which can be updated by administrators.
5. **Health Guidance**: Offer meal suggestions and health tips to improve the user’s diet.

**1.3 Modules**

1. **User Profile Management**: Allows users to create and manage their personal information, health goals, and preferences.
2. **Meal Planning**: Generates personalized meal plans based on user profiles and dietary restrictions.
3. **Nutritional Tracking**: Tracks and reports on the user's daily food intake and nutritional values.
4. **Food Database**: Stores information about food items, including nutritional content, and allows admins to update this data.
5. **Admin Management**: Provides functionality for administrators to manage the food database and update meal plans and recipes.

Top of Form

**2.1 Software Description**

* **MySQL**:  
  MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for managing and manipulating data. It is widely used for applications that require large-scale database management, such as web applications and enterprise software. In this project, MySQL is used to store and manage data related to users, food items, meal plans, and recipes. It enables fast and efficient data retrieval and management using SQL queries.
* **JDBC (Java Database Connectivity)**:  
  JDBC is an API (Application Programming Interface) in Java that enables Java applications to interact with relational databases. It provides a standard interface for connecting and executing queries on MySQL (or other databases) directly from Java code. The JDBC allows the application to establish a connection with the MySQL database, execute SQL queries, and process the results. It serves as the bridge between the front-end Java application and the back-end MySQL database.

**2.2 Languages**

**2.2.1 SQL (Structured Query Language)**

**SQL** is the standard language used for querying and managing databases. In this project, SQL is used to interact with the **MySQL** database, which stores all the relevant information for the Dietary Management System. SQL enables users to retrieve, update, insert, and delete data from the database through structured commands.

**SQL Commands Used**:

**SELECT**: Retrieves data from one or more tables (e.g., getting user details, food items, and meal plans).

**INSERT**: Adds new records into the database (e.g., adding a new food item or user profile).

**UPDATE**: Modifies existing records in the database (e.g., updating a user's meal plan or dietary preferences).

**DELETE**: Removes records from the database (e.g., deleting a food item or user account).

**JOIN**: Combines data from multiple tables based on a related column, allowing the system to retrieve combined information (e.g., joining food item tables with user meal plans).

**2.2.2 JAVA**

* **Java** is a high-level programming language that is widely used for building scalable and secure applications. In the **Dietary Management System**, Java serves as the core language for developing the application’s user interface and logic. Java allows for the creation of dynamic, cross-platform applications that run on various operating systems.
* **Java Features Used**:

**Object-Oriented Programming**: Java supports object-oriented principles, allowing the system to model real-world entities like users, meals, food items, and recipes as objects.

**Swing/JavaFX**: These libraries can be used for creating graphical user interfaces (GUIs) for the system, enabling user interaction with the application for tasks such as entering meal preferences, tracking intake, and managing user profiles.

**JDBC**: Java Database Connectivity (JDBC) is used to connect the Java application with the MySQL database. This enables executing SQL queries from the Java application and retrieving results for processing and displaying to users.

**Exception Handling**: Java's robust exception handling ensures that errors like database connection failures or invalid user inputs are managed gracefully.

**3. Requirements and Analysis**

**3.1 Requirement Specification**

**Functional Requirements**:

* **User Registration and Profile Management**
* **Meal Planning and Recommendations**
* **Nutritional Tracking**
* **Food Database Management**
* **Reporting and Analysis**
* **Admin Features**

**Non-Functional Requirements**:

* **Usability**
* **Security**
* **Performance**

**3.2 Hardware and Software Requirements**

**Hardware Requirements**:

* **Processor**: Intel Core i3 (or higher) / AMD equivalent
* **RAM**: Minimum 4 GB
* **Hard Disk**: Minimum 500 GB of free space
* **Display**: 1280 x 800 resolution or higher
* **Network**: Internet access for database connectivity (if applicable)
* **Software Requirements**:
* **Operating System**: Windows, macOS, or Linux
* **Database**: MySQL 5.7
* **Development Environment**:
* **Java Development Kit (JDK)** version 8
* **IDE**: Eclipse, IntelliJ IDEA, or NetBeans (for Java development)
* **JDBC Driver**: MySQL JDBC Connector
* **Database Management Tool**: MySQL Workbench
* **Web Browser**: Chrome, Firefox, or Safari (if web-based interface is used)
* **Libraries**:
* Java Swing or JavaFX (for GUI development)
* JDBC API (for database connectivity)

### 3.3 Architecture Diagram

The **Dietary Management System** follows a **Client-Server** architecture, where the client (user interface) interacts with the server (database) to fetch or store data. Below is the architecture overview:

+-------------------+

| User Interface | <--- (Java Swing/JavaFX)

+-------------------+

|

v

+--------------------+

| Application Logic | <--- (Java)

+--------------------+

|

v

+------------------+ +------------------+

| JDBC (Data Layer)| <-----> | MySQL Database |

### 3.4 ER Diagram (Entity-Relationship Diagram)

The **ER Diagram** represents the key entities in the system and their relationships. Below is a simplified structure:

+-----------------+ +----------------+ +-------------------+

| User | | Meal Plan | | Food Item |

+-----------------+ +----------------+ +-------------------+

| User\_ID (PK) | | Meal\_ID (PK) | | Food\_ID (PK) |

| Name | | User\_ID (FK) | | Name |

| Age | | Food\_ID (FK) | | Calories |

| Height | | Meal\_Type | | Protein |

| Weight | +----------------+ | Fat |

| Health\_Goal | | Carbs |

| Dietary\_Prefs | | Vitamins |

+-----------------+ +-------------------+

|

v

+-------------------+

| Nutritional Info |

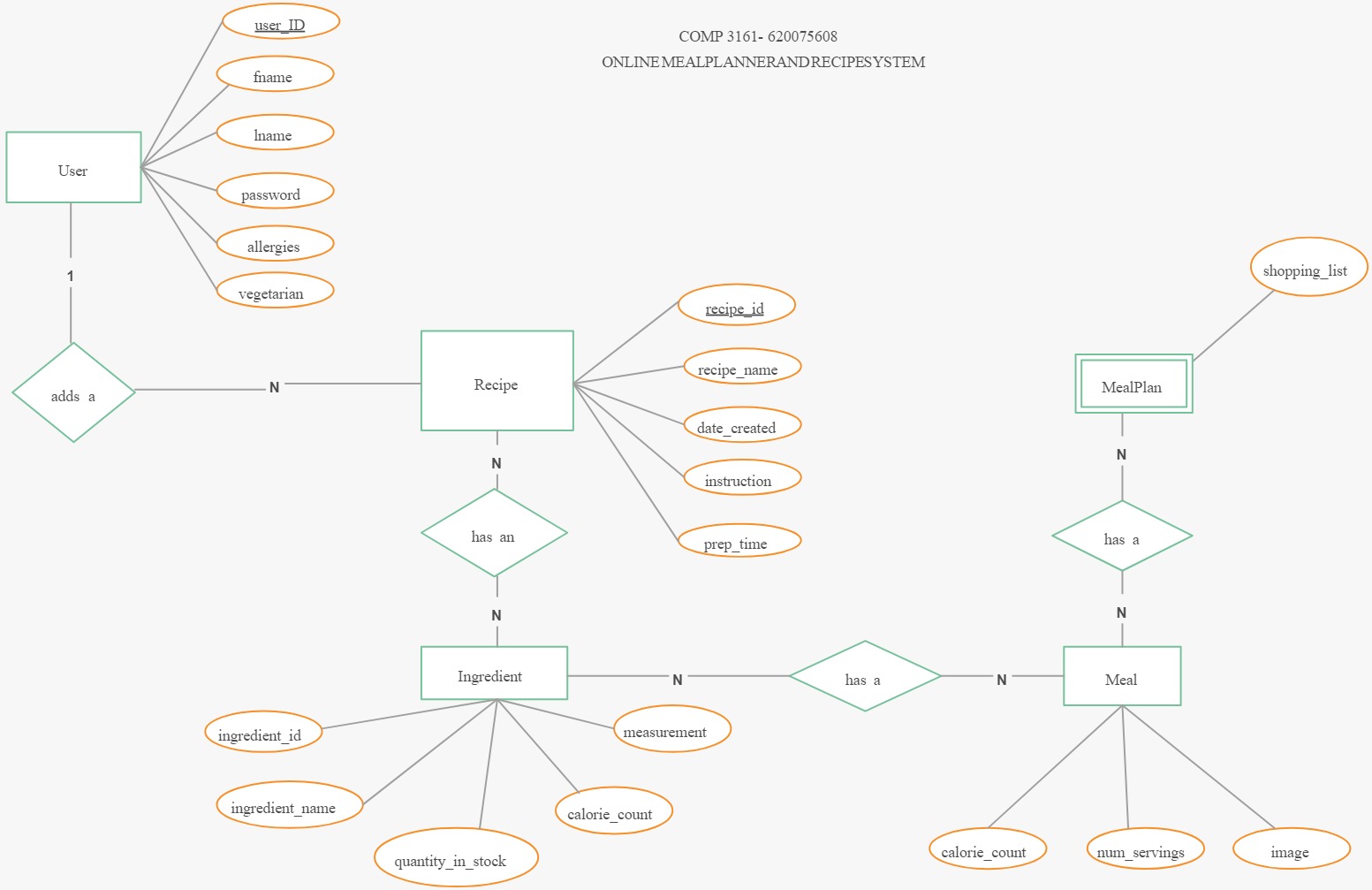
+-------------------+

| Nutrient\_ID (PK) |

| Nutrient\_Type |

| Amount |

+-------------------+

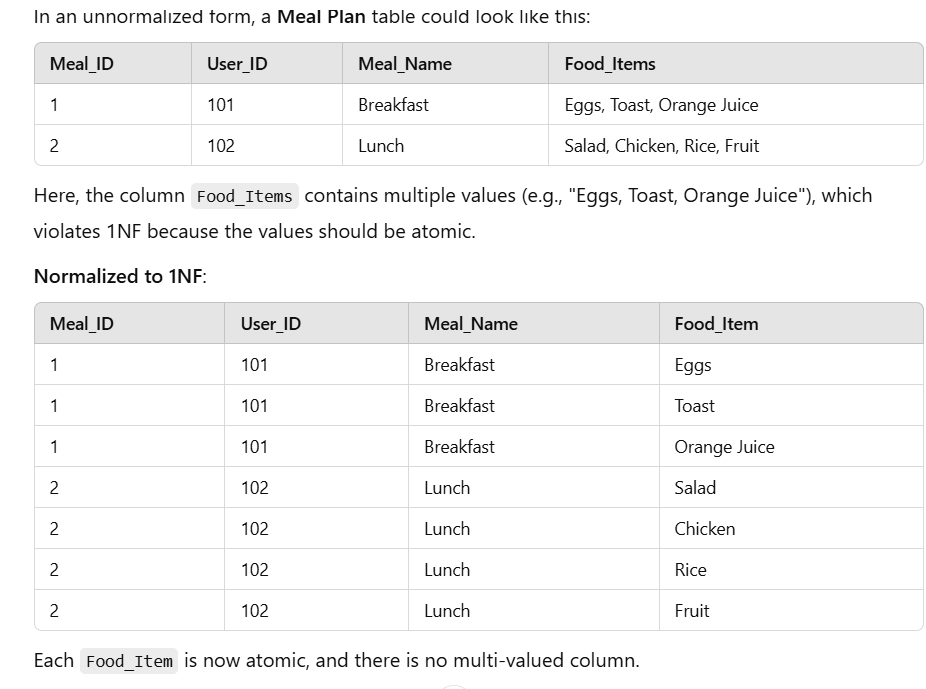


### 3.5 Normalization

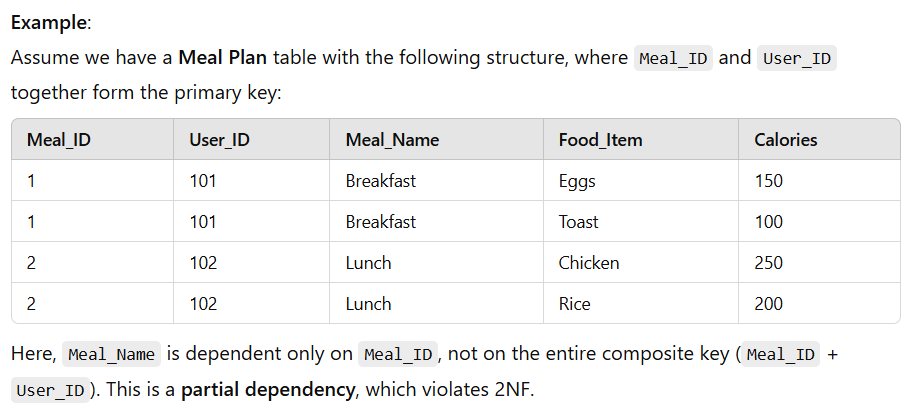
Normalization is the process of organizing data in the database to reduce redundancy and improve data integrity. The **Dietary Management System** follows **3rd Normal Form (3NF)**:

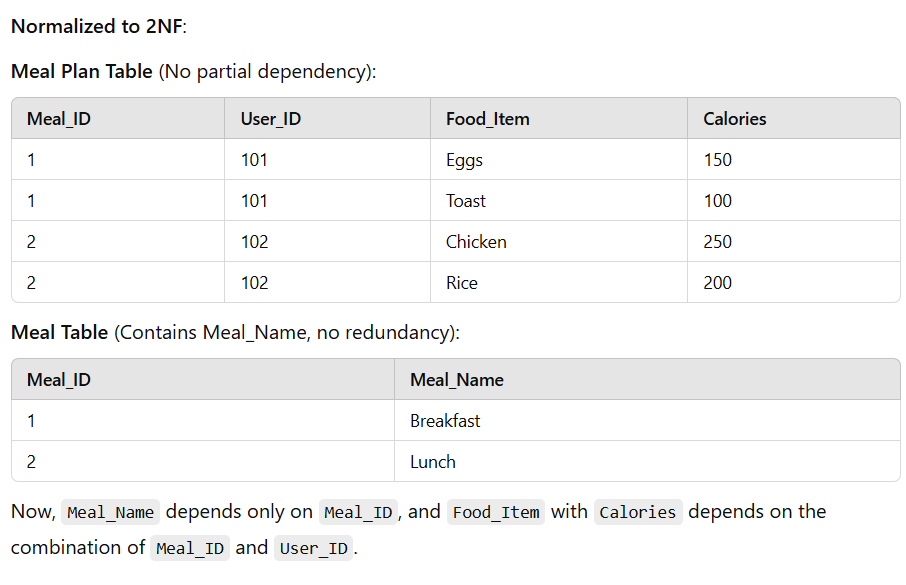
1. **1st Normal Form (1NF)**:
   * The data in the tables is atomic (no repeating groups or arrays).
   * Each column contains unique, indivisible values.

**Example:**

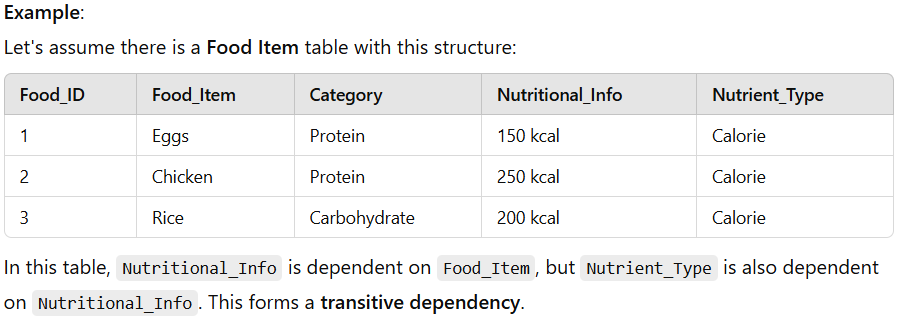


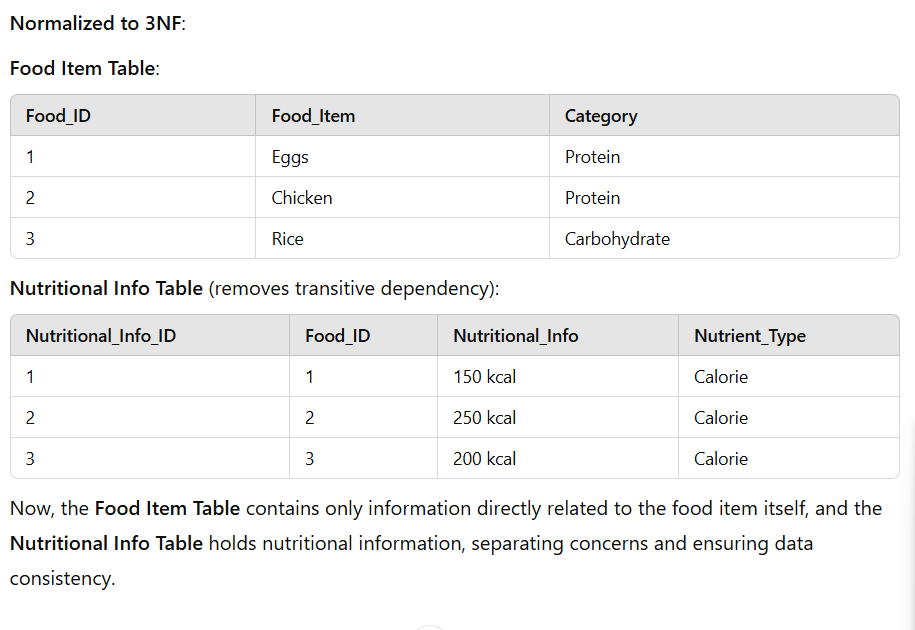
1. **2nd Normal Form (2NF)**:
   * The database is in 1NF.
   * Each non-key attribute is fully dependent on the primary key.
   * For example, a meal plan's details (user, food items) are stored separately from food nutritional information to avoid redundancy.





1. **3rd Normal Form (3NF)**:
   * The database is in 2NF.
   * There are no transitive dependencies, meaning that non-key attributes depend only on the primary key, not on other non-key attributes.
   * For instance, nutritional information is separated from the food item to avoid repeating the same information across multiple records.





This normalization ensures that the database is efficient, reduces duplication, and maintains data integrity.

**PROGRAM CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.sql.\*;

import java.time.LocalDate;

public class CNMSApp extends JFrame {

private JPanel loginPanel, mainMenuPanel, mealPanel, groceryPanel, dietaryPanel, reportPanel;

private JTextField usernameField, mealNameField, ingredientsField, caloriesField, groceryItemField, dailyCalorieField, targetCalorieField;

private JPasswordField passwordField;

private JList<String> groceryList;

private DefaultListModel<String> groceryListModel;

private JButton loginButton, saveMealButton, addItemButton, saveDietaryButton, generateReportButton, backButton;

private JButton backToHomeButton; // Declare the Back to Home button

// Database credentials

private final String url = "jdbc:mysql://localhost:3306/CNMS";

private final String user = "root";

private final String password = "root";

public CNMSApp() {

setTitle("Comprehensive Nutrition Management System");

setSize(900, 600); // Adjusted window size

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

// Initialize panels

initializeLoginPanel();

initializeMainMenuPanel();

initializeMealPanel();

initializeGroceryPanel();

initializeDietaryPanel();

initializeReportPanel();

// Set the initial panel to login

setContentPane(loginPanel);

}

// Method to initialize the Login Panel with better layout

private String currentUsername; // Declare currentUsername as an instance variable

// Method to initialize the Login Panel with better layout

private void initializeLoginPanel() {

loginPanel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

loginPanel.setBorder(BorderFactory.createTitledBorder("Login"));

loginPanel.setBackground(Color.WHITE);

// Set custom fonts and sizes

Font labelFont = new Font("Tahoma", Font.PLAIN, 16); // Bigger font for labels

Font inputFont = new Font("Arial", Font.PLAIN, 18); // Bigger font for input fields

Font buttonFont = new Font("Arial", Font.BOLD, 16); // Bigger font for buttons

usernameField = new JTextField(20); // Larger input field

passwordField = new JPasswordField(20); // Larger input field

loginButton = new JButton("Login");

// Customize the button style

loginButton.setBackground(new Color(33, 150, 243));

loginButton.setForeground(Color.WHITE);

loginButton.setFont(buttonFont);

loginButton.setFocusPainted(false);

// Set grid bag constraints for a more balanced layout

gbc.gridx = 0;

gbc.gridy = 0;

gbc.insets = new Insets(10, 20, 10, 20); // Add vertical and horizontal padding

loginPanel.add(new JLabel("Username:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

loginPanel.add(usernameField, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

loginPanel.add(new JLabel("Password:"), gbc);

gbc.gridx = 1;

gbc.gridy = 1;

loginPanel.add(passwordField, gbc);

gbc.gridx = 0;

gbc.gridy = 2;

gbc.gridwidth = 2; // Span across the columns

gbc.insets = new Insets(20, 20, 10, 20); // Vertical padding before the button

loginPanel.add(loginButton, gbc);

// Add ActionListener for loginButton

loginButton.addActionListener(e -> {

String username = usernameField.getText();

String password = new String(passwordField.getPassword());

if (authenticateUser(username, password)) {

currentUsername = username; // Store the username when login is successful

switchPanel("MainMenu"); // Proceed to main menu

} else {

JOptionPane.showMessageDialog(this, "Invalid username or password.", "Login Failed", JOptionPane.ERROR\_MESSAGE);

}

});

}

// Method to retrieve the current username

private String getCurrentUsername() {

return currentUsername; // This method will now return the logged-in username

}

// Method to initialize the Main Menu Panel

private void initializeMainMenuPanel() {

mainMenuPanel = new JPanel(new GridLayout(5, 1, 5, 5));

mainMenuPanel.setBackground(new Color(236, 239, 241));

mainMenuPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

// Create navigation buttons

JButton mealButton = new JButton("Log a Meal");

JButton groceryButton = new JButton("Manage Grocery List");

JButton dietaryButton = new JButton("Dietary Management");

JButton reportButton = new JButton("Generate Weekly Report");

backButton = new JButton("Back to Home");

// Set button styles

Font buttonFont = new Font("Arial", Font.BOLD, 14);

mealButton.setBackground(new Color(33, 150, 243));

mealButton.setForeground(Color.WHITE);

mealButton.setFont(buttonFont);

mealButton.setFocusPainted(false);

groceryButton.setBackground(new Color(33, 150, 243));

groceryButton.setForeground(Color.WHITE);

groceryButton.setFont(buttonFont);

groceryButton.setFocusPainted(false);

dietaryButton.setBackground(new Color(33, 150, 243));

dietaryButton.setForeground(Color.WHITE);

dietaryButton.setFont(buttonFont);

dietaryButton.setFocusPainted(false);

reportButton.setBackground(new Color(33, 150, 243));

reportButton.setForeground(Color.WHITE);

reportButton.setFont(buttonFont);

reportButton.setFocusPainted(false);

backButton.setBackground(new Color(255, 69, 0));

backButton.setForeground(Color.WHITE);

backButton.setFont(buttonFont);

backButton.setFocusPainted(false);

mainMenuPanel.add(mealButton);

mainMenuPanel.add(groceryButton);

mainMenuPanel.add(dietaryButton);

mainMenuPanel.add(reportButton);

mainMenuPanel.add(backButton);

mealButton.addActionListener(e -> switchPanel("MealPanel"));

groceryButton.addActionListener(e -> switchPanel("GroceryPanel"));

dietaryButton.addActionListener(e -> switchPanel("DietaryPanel"));

reportButton.addActionListener(e -> switchPanel("ReportPanel"));

backButton.addActionListener(e -> switchPanel("LoginPanel"));

}

private void initializeMealPanel() {

mealPanel = new JPanel(new GridLayout(6, 2, 10, 10)); // Adjust GridLayout to accommodate 6 rows

mealPanel.setBackground(Color.WHITE);

mealPanel.setBorder(BorderFactory.createTitledBorder("Log Meal"));

// Initialize the text fields and buttons

mealNameField = new JTextField();

ingredientsField = new JTextField();

caloriesField = new JTextField();

saveMealButton = new JButton("Save Meal");

backToHomeButton = new JButton("Back to Home"); // Add Back to Home button

// Customize Save Meal button

saveMealButton.setBackground(new Color(33, 150, 243)); // Blue color

saveMealButton.setForeground(Color.WHITE);

saveMealButton.setFont(new Font("Arial", Font.BOLD, 14));

saveMealButton.setFocusPainted(false);

// Customize Back to Home button

backToHomeButton.setBackground(new Color(76, 175, 80)); // Green color

backToHomeButton.setForeground(Color.WHITE);

backToHomeButton.setFont(new Font("Arial", Font.BOLD, 14));

backToHomeButton.setFocusPainted(false);

// Add components to the panel

mealPanel.add(new JLabel("Meal Name:"));

mealPanel.add(mealNameField);

mealPanel.add(new JLabel("Ingredients:"));

mealPanel.add(ingredientsField);

mealPanel.add(new JLabel("Calories:"));

mealPanel.add(caloriesField);

mealPanel.add(new JLabel("")); // Empty cell to balance the layout

// Add buttons

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10)); // Horizontal layout for buttons

buttonPanel.setBackground(Color.WHITE);

buttonPanel.add(backToHomeButton); // Add the Back to Home button

buttonPanel.add(saveMealButton); // Add the Save Meal button

mealPanel.add(buttonPanel); // Add the button panel to the grid

mealPanel.add(new JLabel("")); // Empty cell for balance

// Action listeners

saveMealButton.addActionListener(e -> {

String mealName = mealNameField.getText();

String ingredients = ingredientsField.getText();

int calories = Integer.parseInt(caloriesField.getText());

saveMeal(mealName, ingredients, calories);

});

backToHomeButton.addActionListener(e -> switchPanel("MainMenu")); // Switch to main menu on click

}

private void initializeGroceryPanel() {

groceryPanel = new JPanel(new BorderLayout(10, 10));

groceryPanel.setBorder(BorderFactory.createTitledBorder("Manage Grocery List"));

groceryPanel.setBackground(Color.WHITE);

// Set custom fonts and sizes for a consistent UI

Font labelFont = new Font("Tahoma", Font.PLAIN, 16); // Font for labels

Font inputFont = new Font("Arial", Font.PLAIN, 18); // Font for input fields

Font buttonFont = new Font("Arial", Font.BOLD, 16); // Font for buttons

// Initialize the grocery list model and input field

groceryListModel = new DefaultListModel<>();

groceryList = new JList<>(groceryListModel);

groceryItemField = new JTextField(20); // Larger input field for grocery item

addItemButton = new JButton("Add Item");

backToHomeButton = new JButton("Back to Home"); // Initialize Back to Home button

// Customize the Back to Home button

backToHomeButton.setBackground(new Color(76, 175, 80)); // Green color

backToHomeButton.setForeground(Color.WHITE);

backToHomeButton.setFont(buttonFont);

backToHomeButton.setFocusPainted(false);

// Customize the Add Item button

addItemButton.setBackground(new Color(33, 150, 243)); // Blue color

addItemButton.setForeground(Color.WHITE);

addItemButton.setFont(buttonFont);

addItemButton.setFocusPainted(false);

// Action listener for Add Item Button

addItemButton.addActionListener(e -> {

String item = groceryItemField.getText().trim();

if (!item.isEmpty()) {

addGroceryItem(item);

groceryListModel.addElement(item);

groceryItemField.setText("");

}

});

// Action listener for Back to Home Button

backToHomeButton.addActionListener(e -> switchPanel("MainMenu"));

// Panel for Grocery List and Buttons

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 20, 10)); // Flow layout with padding

// Add components to the button panel

buttonPanel.add(backToHomeButton);

buttonPanel.add(addItemButton);

// Panel for input field

JPanel inputPanel = new JPanel(new BorderLayout(10, 10));

inputPanel.add(groceryItemField, BorderLayout.CENTER);

inputPanel.setBackground(Color.WHITE);

inputPanel.add(buttonPanel, BorderLayout.EAST); // Place the buttons to the right of input field

// Add components to grocery panel

groceryPanel.add(new JScrollPane(groceryList), BorderLayout.CENTER); // Scrollable list

groceryPanel.add(inputPanel, BorderLayout.SOUTH); // Input area at the bottom

// Set padding and background color

groceryPanel.setBackground(Color.WHITE);

}

// Method to initialize the Dietary Panel

private void initializeDietaryPanel() {

// Using GridLayout(4, 2) as per the original layout

dietaryPanel = new JPanel(new GridLayout(4, 2, 10, 10));

dietaryPanel.setBackground(Color.WHITE);

dietaryPanel.setBorder(BorderFactory.createTitledBorder("Dietary Management"));

// Input fields for calories

dailyCalorieField = new JTextField();

targetCalorieField = new JTextField();

// Buttons

saveDietaryButton = new JButton("Save Dietary Info");

backToHomeButton = new JButton("Back to Home");

// Customize the buttons

saveDietaryButton.setBackground(new Color(33, 150, 243)); // Blue color

saveDietaryButton.setForeground(Color.WHITE);

saveDietaryButton.setFont(new Font("Arial", Font.BOLD, 14));

saveDietaryButton.setFocusPainted(false);

backToHomeButton.setBackground(new Color(76, 175, 80)); // Green color

backToHomeButton.setForeground(Color.WHITE);

backToHomeButton.setFont(new Font("Arial", Font.BOLD, 14));

backToHomeButton.setFocusPainted(false);

// Add labels and text fields to the panel

dietaryPanel.add(new JLabel("Calories Consumed Today:"));

dietaryPanel.add(dailyCalorieField);

dietaryPanel.add(new JLabel("Daily Calorie Goal:"));

dietaryPanel.add(targetCalorieField);

// Add an empty label for alignment

dietaryPanel.add(new JLabel("")); // Empty space to make room for buttons

dietaryPanel.add(new JLabel("")); // Empty space

// Add Save Dietary button and Back to Home button to the panel

dietaryPanel.add(backToHomeButton); // Add the "Back to Home" button

dietaryPanel.add(saveDietaryButton); // Add the "Save Dietary Info" button

// Action listeners for the buttons

saveDietaryButton.addActionListener(e -> {

int caloriesConsumed = Integer.parseInt(dailyCalorieField.getText());

int calorieGoal = Integer.parseInt(targetCalorieField.getText());

saveDietaryData(caloriesConsumed, calorieGoal); // Method to save dietary data

});

backToHomeButton.addActionListener(e -> {

switchPanel("MainMenu"); // Switch to Main Menu

});

}

// Method to initialize the Report Panel

private void initializeReportPanel() {

// Initialize panel with BorderLayout

reportPanel = new JPanel(new BorderLayout(10, 10));

reportPanel.setBackground(Color.WHITE);

reportPanel.setBorder(BorderFactory.createTitledBorder("Weekly Report"));

// Initialize buttons and text area

generateReportButton = new JButton("Generate Weekly Report");

backToHomeButton = new JButton("Back to Home");

JTextArea reportArea = new JTextArea();

reportArea.setEditable(false);

// Customize the Generate Report button

generateReportButton.setBackground(new Color(33, 150, 243)); // Blue

generateReportButton.setForeground(Color.WHITE);

generateReportButton.setFont(new Font("Arial", Font.BOLD, 14));

generateReportButton.setFocusPainted(false);

// Customize the Back to Home button

backToHomeButton.setBackground(new Color(76, 175, 80)); // Green

backToHomeButton.setForeground(Color.WHITE);

backToHomeButton.setFont(new Font("Arial", Font.BOLD, 14));

backToHomeButton.setFocusPainted(false);

// Add the buttons and text area to the panel

reportPanel.add(generateReportButton, BorderLayout.NORTH);

reportPanel.add(new JScrollPane(reportArea), BorderLayout.CENTER);

// Use a JPanel for the bottom section to hold both buttons

JPanel bottomPanel = new JPanel(new GridLayout(1, 2, 10, 0)); // GridLayout for buttons

bottomPanel.add(backToHomeButton);

bottomPanel.add(generateReportButton); // You can keep this or adjust as needed

reportPanel.add(bottomPanel, BorderLayout.SOUTH); // Add the button panel to the south

// Action listener for Generate Report button

generateReportButton.addActionListener(e -> {

String report = generateWeeklyReport();

reportArea.setText(report);

});

// Action listener for Back to Home button

backToHomeButton.addActionListener(e -> {

switchPanel("MainMenu"); // Switch back to the main menu

});

}

// Method to switch between panels

private void switchPanel(String panelName) {

switch (panelName) {

case "MainMenu":

setContentPane(mainMenuPanel);

break;

case "MealPanel":

setContentPane(mealPanel);

break;

case "GroceryPanel":

setContentPane(groceryPanel);

break;

case "DietaryPanel":

setContentPane(dietaryPanel);

break;

case "ReportPanel":

setContentPane(reportPanel);

break;

case "LoginPanel":

setContentPane(loginPanel);

break;

}

revalidate();

repaint();

}

// Database Connection Method

private Connection connectToDatabase() {

try {

return DriverManager.getConnection(url, user, password);

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Database connection failed!", "Error", JOptionPane.ERROR\_MESSAGE);

return null;

}

}

// User Authentication Method

private boolean authenticateUser(String username, String password) {

try (Connection conn = connectToDatabase()) {

String query = "SELECT \* FROM Users WHERE username = ? AND password = ?";

PreparedStatement ps = conn.prepareStatement(query);

ps.setString(1, username);

ps.setString(2, password);

ResultSet rs = ps.executeQuery();

return rs.next();

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

//Method to save meal

private void saveMeal(String mealName, String ingredients, int calories) {

try (Connection conn = connectToDatabase()) {

// Get the logged-in username

String username = getCurrentUsername(); // This method should return the username of the logged-in user

// Check if the username exists in Users table (optional but recommended)

String checkUserQuery = "SELECT username FROM Users WHERE username = ?";

PreparedStatement psCheckUser = conn.prepareStatement(checkUserQuery);

psCheckUser.setString(1, username);

ResultSet rs = psCheckUser.executeQuery();

if (!rs.next()) {

// Handle case where username does not exist

JOptionPane.showMessageDialog(this, "Username does not exist!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

// If the username exists, proceed with meal insertion

String query = "INSERT INTO CNMSMeals (username, meal\_name, ingredients, calories) VALUES (?, ?, ?, ?)";

PreparedStatement ps = conn.prepareStatement(query);

ps.setString(1, username); // Set username dynamically based on the logged-in user

ps.setString(2, mealName);

ps.setString(3, ingredients);

ps.setInt(4, calories);

ps.executeUpdate();

JOptionPane.showMessageDialog(this, "Meal saved successfully!");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to save meal.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// Method to Save Dietary Data

private void saveDietaryData(int caloriesConsumed, int calorieGoal) {

try (Connection conn = connectToDatabase()) {

String query = "INSERT INTO DietaryLogs (date, calories\_consumed, calorie\_goal, user\_id) VALUES (?, ?, ?, ?)";

PreparedStatement ps = conn.prepareStatement(query);

ps.setDate(1, Date.valueOf(LocalDate.now()));

ps.setInt(2, caloriesConsumed);

ps.setInt(3, calorieGoal);

ps.setInt(4, getCurrentUserId());

ps.executeUpdate();

JOptionPane.showMessageDialog(this, "Dietary information saved!");

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to save dietary information.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// Method to Generate Weekly Report

private String generateWeeklyReport() {

int totalCaloriesConsumed = 0;

int totalCaloriesGoal = 0;

try (Connection conn = connectToDatabase()) {

String query = "SELECT SUM(calories\_consumed) AS total\_consumed, SUM(calorie\_goal) AS total\_goal FROM DietaryLogs WHERE date >= DATE\_SUB(CURDATE(), INTERVAL 7 DAY) AND user\_id = ?";

PreparedStatement ps = conn.prepareStatement(query);

ps.setInt(1, getCurrentUserId());

ResultSet rs = ps.executeQuery();

if (rs.next()) {

totalCaloriesConsumed = rs.getInt("total\_consumed");

totalCaloriesGoal = rs.getInt("total\_goal");

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Failed to generate report.", "Error", JOptionPane.ERROR\_MESSAGE);

}

int caloriesLeftToBurn = totalCaloriesGoal - totalCaloriesConsumed;

String feedback = caloriesLeftToBurn > 0 ?

"Keep going! You have " + caloriesLeftToBurn + " calories left to burn this week!" :

"Great job! You've reached your weekly calorie goal!";

return "Weekly Calories Consumed: " + totalCaloriesConsumed + "\n" +

"Weekly Calorie Goal: " + totalCaloriesGoal + "\n" +

"Calories Left to Burn: " + caloriesLeftToBurn + "\n\n" + feedback;

}

// Method to Add Grocery Items

private void addGroceryItem(String item) {

try (Connection conn = connectToDatabase()) {

String query = "INSERT INTO GroceryList (item, user\_id) VALUES (?, ?)";

PreparedStatement ps = conn.prepareStatement(query);

ps.setString(1, item);

ps.setInt(2, getCurrentUserId());

ps.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to Load Grocery List from Database

private void loadGroceryList() {

try (Connection conn = connectToDatabase()) {

String query = "SELECT item FROM GroceryList WHERE user\_id = ?";

PreparedStatement ps = conn.prepareStatement(query);

ps.setInt(1, getCurrentUserId());

ResultSet rs = ps.executeQuery();

while (rs.next()) {

groceryListModel.addElement(rs.getString("item"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// Method to Get the Current User ID

private int getCurrentUserId() {

// Placeholder for current user ID; assuming a logged-in user with ID = 1

return 1;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

CNMSApp app = new CNMSApp();

app.setVisible(true);

});

}

}

### Conclusion

In conclusion, the **Dietary Management System** project effectively demonstrates the application of a Database Management System (DBMS) to manage users' dietary preferences and nutritional data. The system provides users with personalized meal plans, nutritional tracking, and health insights, aiming to support individuals in achieving their health and fitness goals. By implementing technologies such as **Java**, **MySQL**, and **JDBC**, the project ensures smooth interaction between the user interface and database, allowing for efficient data retrieval and storage.

The system’s design focuses on optimizing data storage through normalization, ensuring data integrity and minimizing redundancy. Furthermore, the architecture, including the use of **3rd Normal Form (3NF)**, ensures that the database remains scalable, efficient, and maintainable. The project also highlights the importance of security, performance, and usability in building an interactive and user-friendly application.

Overall, the **Dietary Management System** successfully integrates database management, user-friendly interfaces, and personalized meal tracking to offer a comprehensive solution for managing dietary health. This system can be further extended with additional features such as real-time updates, advanced analytics, and integration with fitness trackers for more personalized user experiences>

The over view the Dietary Management: