Comprehensive Nutrition Management System

A MINI-PROJECT BY:

S.SANJAY KRISHNA 230701288

SAUNAK RAMYA SEBESAN 230701298

in partial fulfilment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"Comprehensive Nutrition Management System"** is the bona fide work of **"S.SANJAY KRISHNA,SAUNAK RAMYA SEBESAN"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Mr. G SARAVANA GOKUL
Assistant Professor (SS),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai-602105

SIGNATURE

Ms. V. JANANEE
Assistant Professor (SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Comprehensive Nutrition Management System:

The Comprehensive Nutrition Management System (CNMS) is a Java-based application designed to assist individuals in effectively tracking their nutrition and maintaining a healthy lifestyle. The system provides the following features:

1. Manual Meal Logging:
    - Log meals and ingredients to receive nutritional breakdowns for calories, macronutrients, and micronutrients.
    - Supports QR code scanning for easy retrieval of nutritional data from food packaging.
2. Dietary Restriction and Allergy Management:
    - Create customizable profiles with dietary preferences (e.g., vegetarian, gluten-free) and allergies (e.g., nuts, dairy).
    - Receive filtered meal suggestions and alternative ingredient recommendations.
3. Weekly Progress Reports:
    - View nutritional trends and daily averages for calories and nutrients.
    - Track progress toward personalized nutrition goals.
4. Meal Prep and Grocery Planning:
    - Plan weekly meals and access a database of recipes with nutritional details.
    - Generate automated grocery lists based on selected recipes.

Java Technologies and Tools Used

- Java Swing: Used for creating the graphical user interface (GUI), including navigation panels, buttons, forms, and input fields.
- JDBC (Java Database Connectivity): Enables interaction with the backend database (e.g., MySQL) to store and retrieve user data, meal logs, and nutritional information.
- CardLayout: Facilitates smooth transitions between different UI panels for features such as meal logging, dietary management, and progress reports.
- Event Handling: Implemented using Java's ActionListener to handle user interactions like button clicks and form submissions.
- Data Validation: Ensures accurate input for meal logs, nutritional goals, and other fields.

TABLE OF CONTENTS

INTRODUCTION

The **Comprehensive Nutrition Management System (CNMS)** is a user-friendly application designed to help individuals track their nutritional intake, manage dietary preferences, and plan meals effectively. Proper nutrition is a cornerstone of a healthy lifestyle, and this system simplifies the process by offering personalized features to suit diverse user needs.

In today's fast-paced world, people often struggle to maintain balanced diets due to a lack of knowledge about nutrition, limited time for meal planning, or challenges in managing dietary restrictions and allergies. CNMS addresses these challenges by providing an intuitive interface and practical tools that empower users to take control of their nutrition.

The application leverages Java as its primary development language to ensure a reliable and scalable system. By combining Java's robust capabilities with features like meal logging, dietary management, and progress tracking, CNMS bridges the gap between technology and nutrition management, making healthy living accessible for all.

With its core functionality implemented using Java Swing for the UI and JDBC for database connectivity, CNMS offers a seamless experience for users to log meals, set dietary goals, and plan their weekly nutrition with minimal effort. Whether users aim to monitor calorie intake, accommodate dietary restrictions, or prepare balanced meals, CNMS serves as a comprehensive solution tailored to their needs.

1.2 IMPLEMENTATION

The project **Comprehensive Nutrition Management System** is implemented using Java Swing for the user interface and MySQL for database management.

1.3 SCOPE OF THE PROJECT

The Comprehensive Nutrition Management System (CNMS) is designed to

provide users with an efficient, intuitive platform to track their nutrition and maintain a healthy lifestyle. The project's scope encompasses features that address the challenges of meal tracking, dietary restriction management, and nutritional goal monitoring while leveraging modern technologies for seamless implementation.

Functional Scope

1. Manual Meal Logging
    - Feature Description: Allows users to log meals and ingredients manually. Provides a detailed nutritional breakdown, including calories, macronutrients, and micronutrients.
    - Purpose: Ensures accurate tracking of nutritional intake and promotes awareness of dietary habits.
2. Dietary Restriction and Allergy Management
    - Feature Description:
        - Users can customize dietary profiles (e.g., vegetarian, gluten-free) and specify allergies (e.g., nuts, dairy).
        - Provides meal suggestions filtered according to user preferences.
        - Recommends ingredient substitutions tailored to dietary restrictions.
    - Purpose: Facilitates safe and personalized nutrition management for users with specific dietary needs.
3. Weekly Progress Reports
    - Feature Description:
        - Generates weekly summaries of caloric intake, macronutrient ratios, and vitamin or mineral trends.
        - Includes graphical representations to visualize progress.
        - Tracks and evaluates user-defined nutritional goals.
    - Purpose: Helps users monitor their nutrition trends and adjust their habits to meet their health objectives.
4. Meal Prep & Grocery Planner
    - Feature Description:
        - Offers weekly meal planning to ensure balanced daily nutrition.
        - Includes a recipe database with nutritional data and user ratings.
        - Automatically generates categorized grocery lists based on

selected recipes.
        - Tracks pantry inventory with reminders for replenishment.
    - Purpose: Streamlines meal preparation and shopping, reducing waste and saving time.

Technical Scope

1. Backend Implementation
    - Language: Java
        - Used to develop core application logic, handle requests, and process user inputs.
    - Database Connectivity: JDBC (Java Database Connectivity)
        - Facilitates seamless interaction with the MySQL database for storing and retrieving user data, meal logs, and recipe information.
2. User Interface
    - Framework: Java Swing
        - Provides an intuitive and responsive graphical user interface (GUI) for desktop-based application users.
        - Includes features like navigation menus, form inputs, and report visualization.
3. Database Management
    - Technology: MySQL
        - Stores structured data such as user profiles, meal logs, dietary preferences, and nutritional information.
        - Ensures data integrity and quick retrieval for real-time application needs.
4. Report Generation
    - Tools Used: Java libraries for data visualization (e.g., graphs and charts).
        - Provides a visual representation of nutritional trends, ensuring easy interpretation by users.
5. Scalability and Extensibility
    - The modular design allows easy addition of new features, such as mobile app support or integration with third-party APIs for advanced nutrition tracking.
    - Prepared to support future enhancements like QR code scanning for automatic meal logging or AI-based meal recommendations.

SYSTEM SPECIFICATIONS

HARDWARE SPECIFICATIONS:

PROCESSOR: Intel i5

MEMORY SIZE: 16GB

HARD DISK: 500 GB of free space

SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE: Java, SQL

FRONT-END: Java Swing

BACK-END: MySQL

OPERATING SYSTEM: Windows 11

ENTITY RELATION MODEL

The Entity-Relationship (ER) diagram for the Comprehensive Nutrition Management System (CNMS) illustrates how the various entities interact and are related within the system. This database design ensures efficient data management and retrieval for features such as meal logging, dietary tracking, and weekly reports.

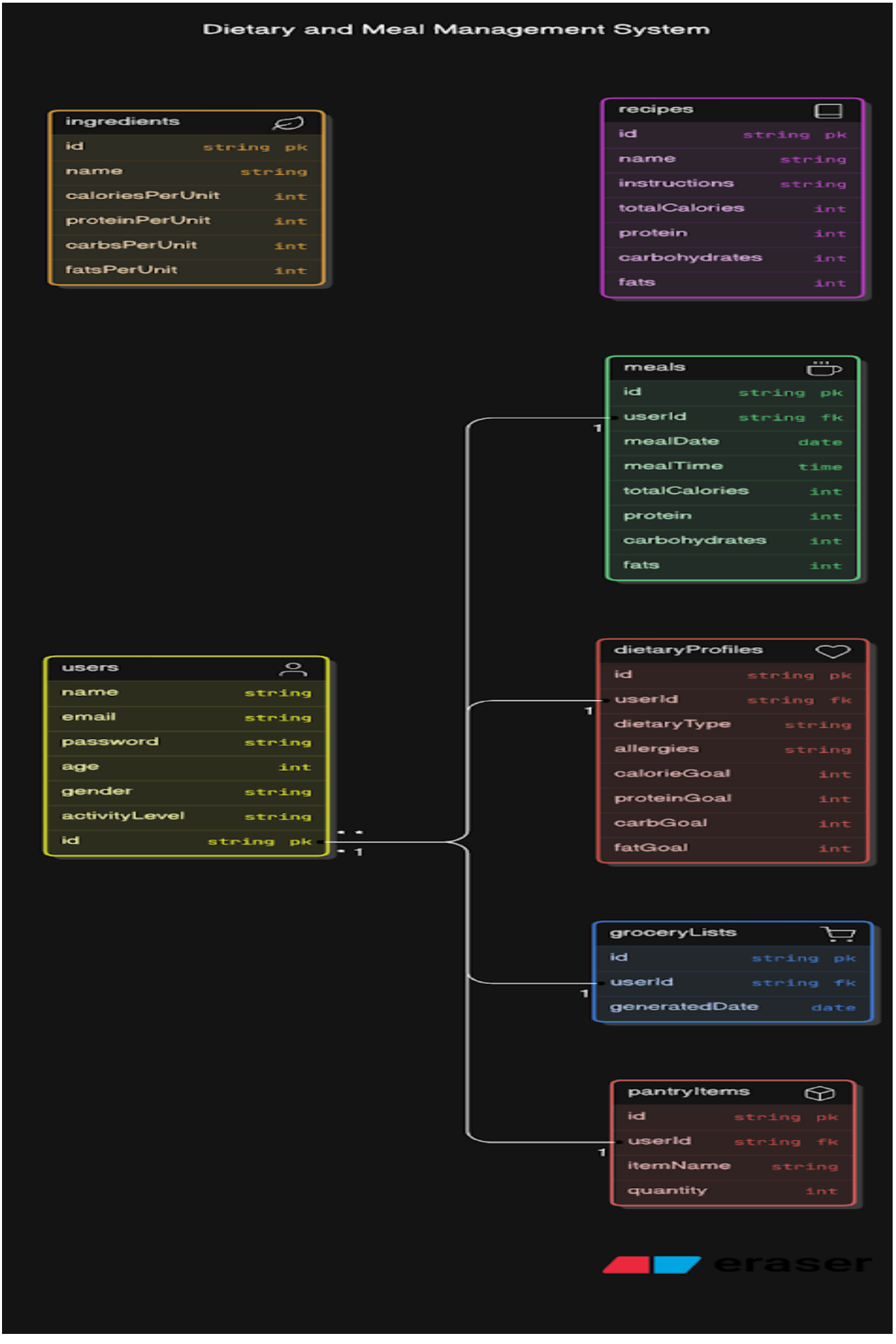
Data Integrity: Ensures consistent storage of user information, meal logs, and dietary profiles.

Scalability: Supports future expansion, such as adding new features like automated QR code scanning or AI-driven meal recommendations.

Efficient Querying: Relationships allow for easy retrieval of relevant data, such as filtering meals based on a user's dietary restrictions.

Personalization: Enables tailored suggestions and progress tracking for individual users

2.1 ER DIAGRAM


Dietary and Meal Management System
ingredients
id string pk
name string
caloriesPerUnit int
proteinPerUnit int
carbsPerUnit int
fatsPerUnit int
recipes
id string pk
name string
instructions string
totalCalories int
protein int
carbohydrates int
fats int
meals
id string pk
userId string fk
mealDate date
mealTime time
totalCalories int
protein int
carbohydrates int
fats int
users
name string
email string
password string
age int
gender string
activityLevel string
id string pk
dietaryProfiles
id string pk
userId string fk
dietaryType string
allergies string
calorieGoal int
proteinGoal int
carbGoal int
fatGoal int
groceryLists
id string pk
userId string fk
generatedDate date
pantryItems
id string pk
userId string fk
itemName string
quantity int
1
1
1
1
1
eraser

SOURCE 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, deleteItemButton, viewGroceryListButton, saveDietaryButton, generateReportButton, backButton;
    private JButton backToHomeButton, backToLoginButton;  // 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");
        backToLoginButton = new JButton("Back to Login");

        // 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);

backToLoginButton.setBackground(new Color(255, 69, 0));
backToLoginButton.setForeground(Color.WHITE);
backToLoginButton.setFont(buttonFont);
backToLoginButton.setFocusPainted(false);

mainMenuPanel.add(mealButton);
mainMenuPanel.add(groceryButton);
mainMenuPanel.add(dietaryButton);
mainMenuPanel.add(reportButton);
//mainMenuPanel.add(backButton);
mainMenuPanel.add(backToLoginButton);

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

```
        groceryButton.addActionListener(e -> switchPanel("GroceryPanel"));
        dietaryButton.addActionListener(e -> switchPanel("DietaryPanel"));
        reportButton.addActionListener(e -> switchPanel("ReportPanel"));
        backToLoginButton.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");
        deleteItemButton = new JButton("Delete Item");  // New Delete Item button
        viewGroceryListButton = new JButton("View Grocery List");  // New View Grocery List button
        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);

        // Customize the Delete Item button
        deleteItemButton.setBackground(new Color(255, 69, 0));  // Red color
        deleteItemButton.setForeground(Color.WHITE);
        deleteItemButton.setFont(buttonFont);
        deleteItemButton.setFocusPainted(false);

        // Customize the View Grocery List button
        viewGroceryListButton.setBackground(new Color(255, 165, 0)); // Orange color
```

```
        viewGroceryListButton.setForeground(Color.WHITE);
        viewGroceryListButton.setFont(buttonFont);
        viewGroceryListButton.setFocusPainted(false);

        // Action listener for Add Item Button
        addItemButton.addActionListener(e -> {
            String item = groceryItemField.getText().trim();
            if (!item.isEmpty()) {
                addGroceryItem(item);  // Add the item to the database
                groceryListModel.addElement(item);  // Update the list model with the new
item
                groceryItemField.setText("");  // Clear the input field after adding
            }
        });

        // Action listener for Delete Item Button
        deleteItemButton.addActionListener(e -> {
            // Get the text from the groceryItemField
            String itemToDelete = groceryItemField.getText().trim();

            if (!itemToDelete.isEmpty()) {
                // Check if the item exists in the list
                if (groceryListModel.contains(itemToDelete)) {
                    // Call the method to delete the item from the database and list
                    deleteGroceryItem(itemToDelete);
                    groceryListModel.removeElement(itemToDelete);  // Remove the item
from the list model
                    groceryItemField.setText("");  // Clear the input field after deletion
                } else {
                    // If the item is not found in the list, show a message
                    JOptionPane.showMessageDialog(this, "Item not found in the list.",
"Error", JOptionPane.WARNING_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(this, "Please enter an item to delete.",
```

```
"Error", JOptionPane.WARNING_MESSAGE);
            }
        });

        // Action listener for View Grocery List Button
        viewGroceryListButton.addActionListener(e -> {
            viewGroceryList();  // Call the method to load and display the grocery list
        });

        // 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);
        buttonPanel.add(deleteItemButton);  // Add Delete Item button to the panel
        buttonPanel.add(viewGroceryListButton);  // Add View Grocery List button to the panel

        // 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 view grocery list for the current user
    private void viewGroceryList() {
        try (Connection conn = connectToDatabase()) {
            int userId = getCurrentUserId();  // Get the logged-in user ID
            if (userId == -1) {
                JOptionPane.showMessageDialog(this, "User not found. Please log in first.", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Query to get all items in the grocery list for the logged-in user
            String query = "SELECT item FROM GroceryList WHERE user_id = ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setInt(1, userId);  // Set the user ID for the query
            ResultSet rs = ps.executeQuery();

            // Clear the current list model to refresh the display
            groceryListModel.clear();  // Assuming you're using a DefaultListModel for the grocery list UI

            // Populate the list with items fetched from the database
            while (rs.next()) {
                String item = rs.getString("item");
                groceryListModel.addElement(item);  // Add each item to the list model
            }

            // Display a message if the list is empty
            if (groceryListModel.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Your grocery list is empty.", "No Items", JOptionPane.INFORMATION_MESSAGE);
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to load grocery list.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    // 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 {
            Connection conn = DriverManager.getConnection(url, user, password);
            if (conn != null) {
                return conn;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to connect to the database.",
"Connection 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;
        }
    }

    // After a successful login, load the grocery list for the user
private void onLoginSuccess() {
    loadGroceryList();  // Load the grocery list for the logged-in user
```

```
}


    // Inside the method where user authentication occurs
private void authenticateUserGrocery(String username, String password) {
    boolean authenticated = authenticateUser(username, password);

    if (authenticated) {
        // Show the grocery list panel (or another panel)
        switchPanel("GroceryPanel");
        onLoginSuccess();  // Load the user's grocery list
    } else {
        JOptionPane.showMessageDialog(this, "Invalid username or password",
"Error", JOptionPane.ERROR_MESSAGE);
    }
}



    //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" +
```

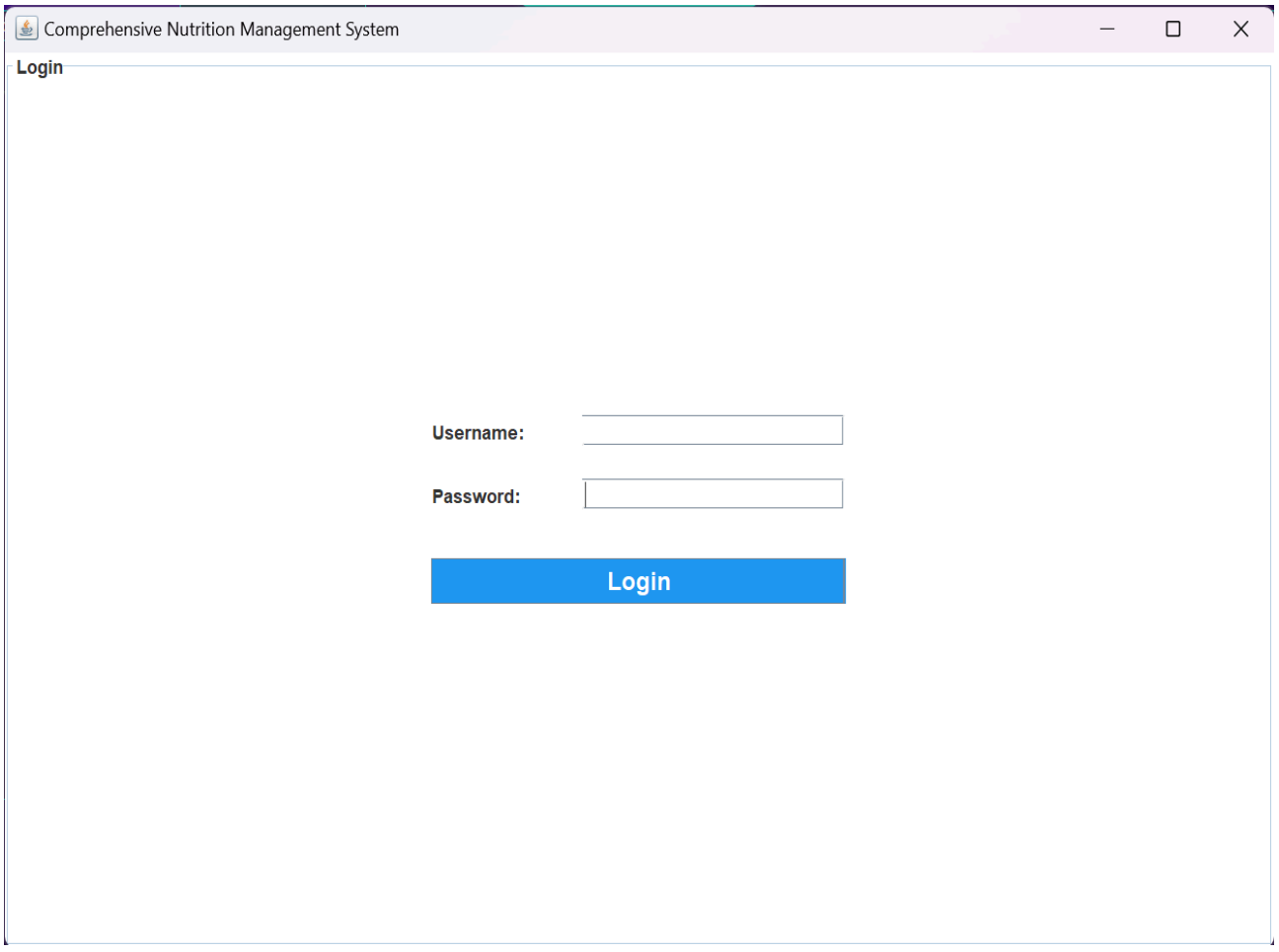
```
            "Weekly Calorie Goal: " + totalCaloriesGoal + "\n" +
            "Calories Left to Burn: " + caloriesLeftToBurn + "\n\n" + feedback;
    }

private int getCurrentUserId() {
    int userId = -1;  // Default to an invalid ID
    try (Connection conn = connectToDatabase()) {
        String query = "SELECT id FROM Users WHERE username = ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, currentUsername);  // Assuming currentUsername holds the
logged-in user's username
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            userId = rs.getInt("id");  // Set the user ID from the result
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return userId;
}

// 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();
    }
}
```

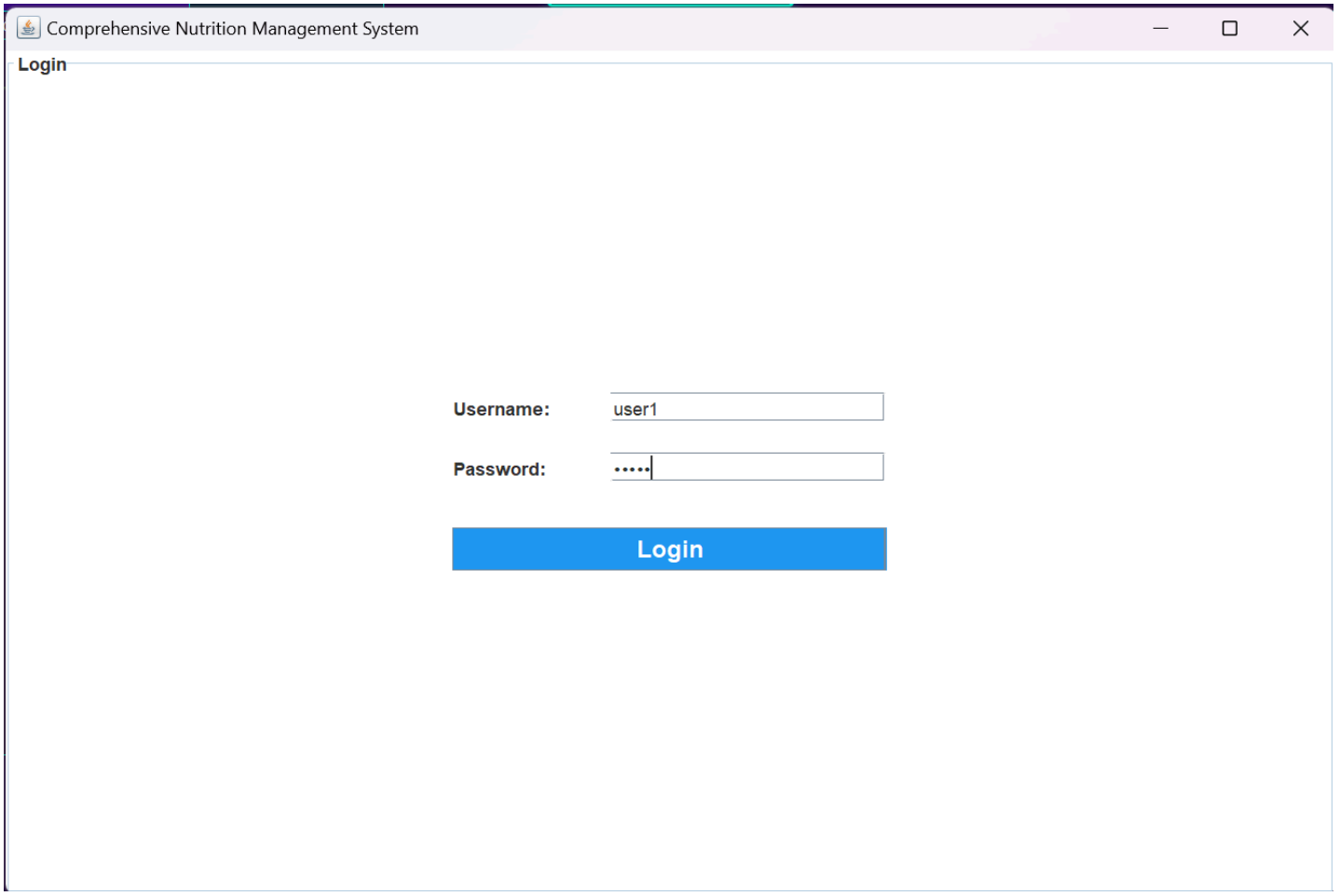
```
// Method to load grocery list from the database and display it in the UI
private void loadGroceryList() {
    groceryListModel.clear();  // Clear any existing items in the UI list model

    try (Connection conn = connectToDatabase()) {
        String query = "SELECT item FROM GroceryList WHERE user_id = ?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, getCurrentUserId());  // Use the logged-in user's ID
        ResultSet rs = ps.executeQuery();

        // Add items to the list model
        while (rs.next()) {
            String item = rs.getString("item");

            // Avoid adding the same item to the UI list multiple times
            if (!groceryListModel.contains(item)) {
                groceryListModel.addElement(item);  // Add the item to the list model
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}



    private void deleteGroceryItem(String item) {
        try (Connection conn = connectToDatabase()) {
            // Delete the item from the database for the logged-in user
            String query = "DELETE FROM GroceryList WHERE item = ? AND user_id
= ?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, item);
            ps.setInt(2, getCurrentUserId());  // Use the current logged-in user's ID
            int rowsAffected = ps.executeUpdate();
```

```
            if (rowsAffected > 0) {
                // If the deletion is successful, remove it from the list model
                groceryListModel.removeElement(item);
                JOptionPane.showMessageDialog(this, "Item deleted successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Item not found or failed to
delete.", "Error", JOptionPane.WARNING_MESSAGE);
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Database error occurred while
deleting item.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            CNMSApp app = new CNMSApp();
            app.setVisible(true);
        });
    }
}
```

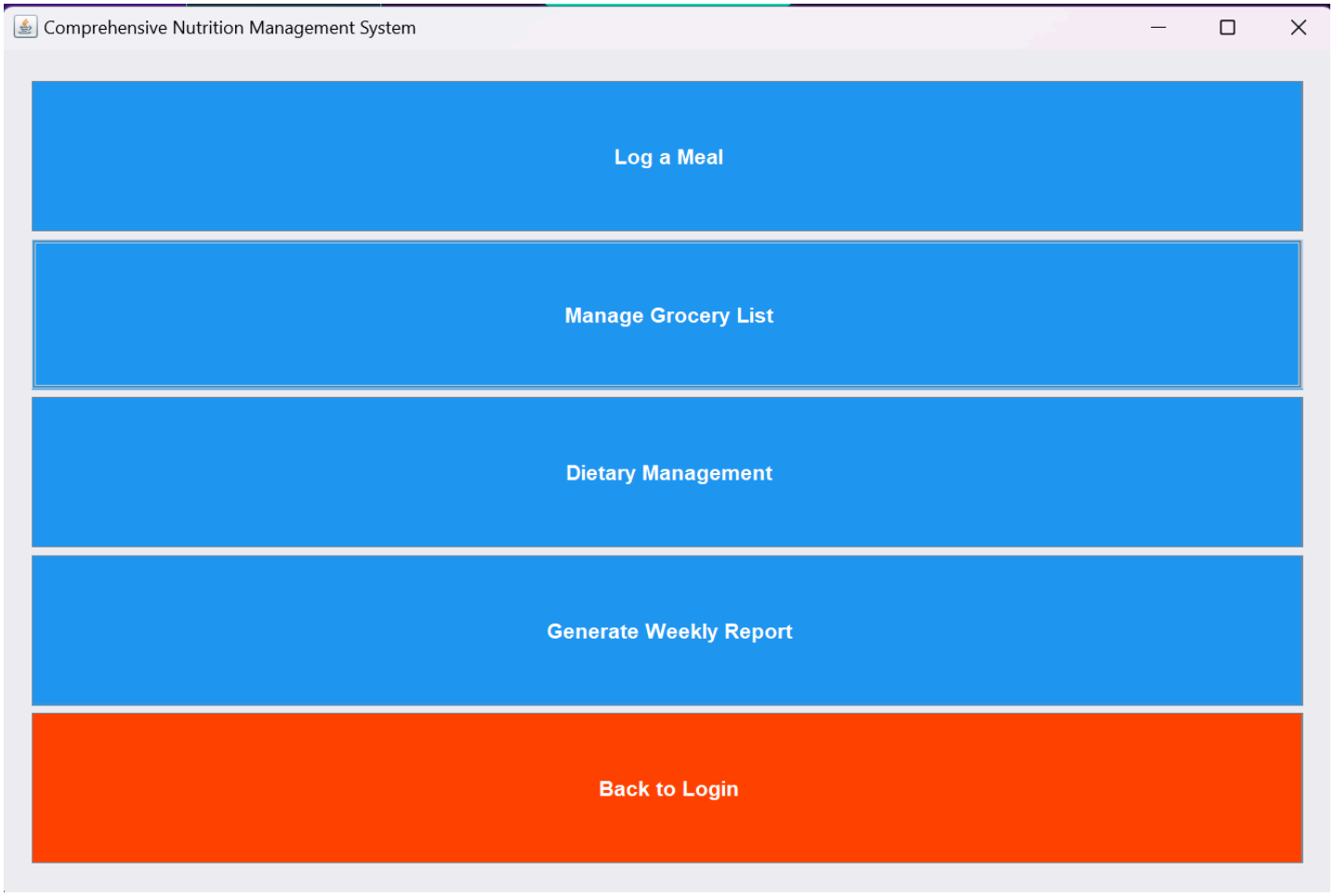
SNAPSHOTS

4.1 LOGIN PAGE:

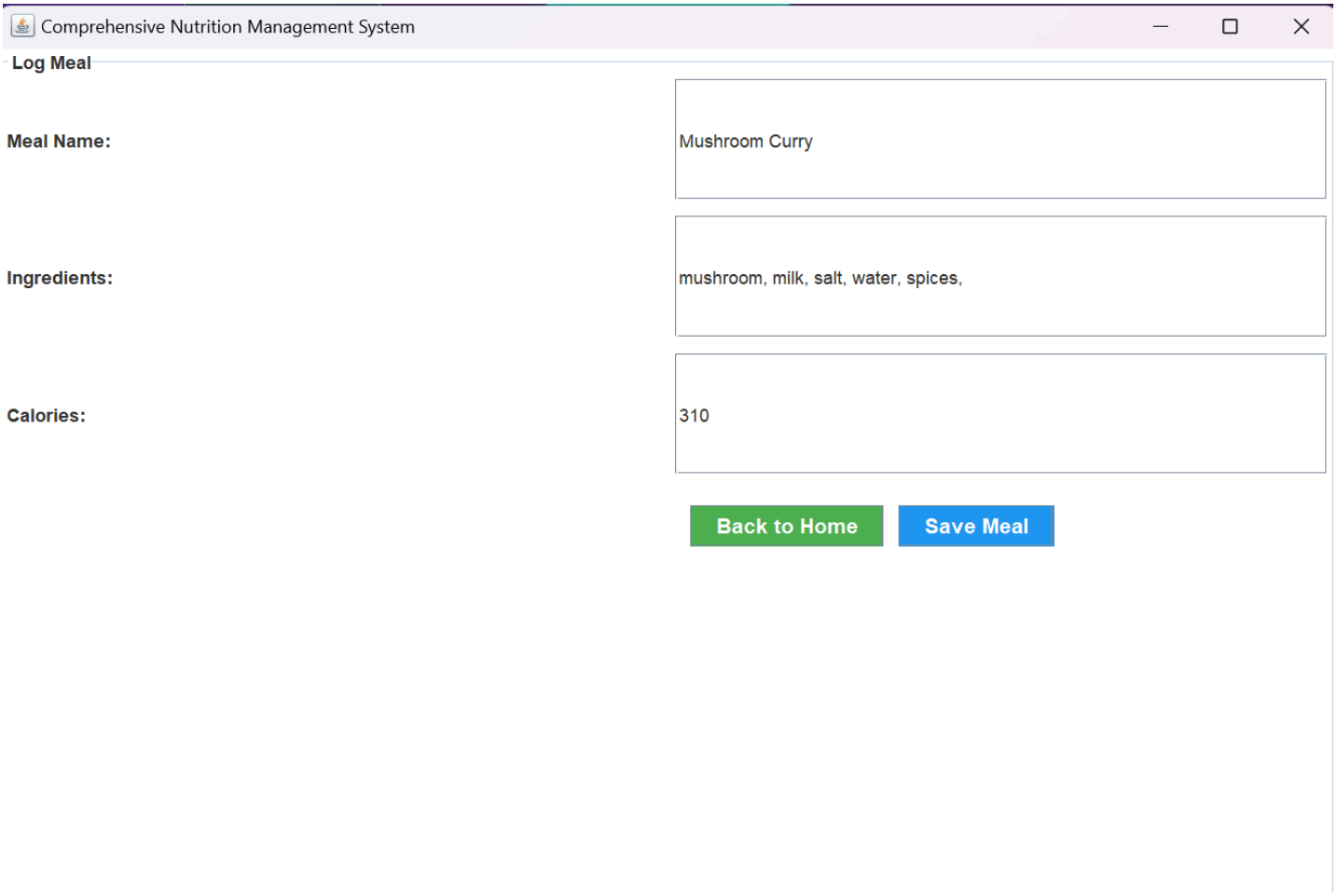

Comprehensive Nutrition Management System
Login
Username:
Password:
Login

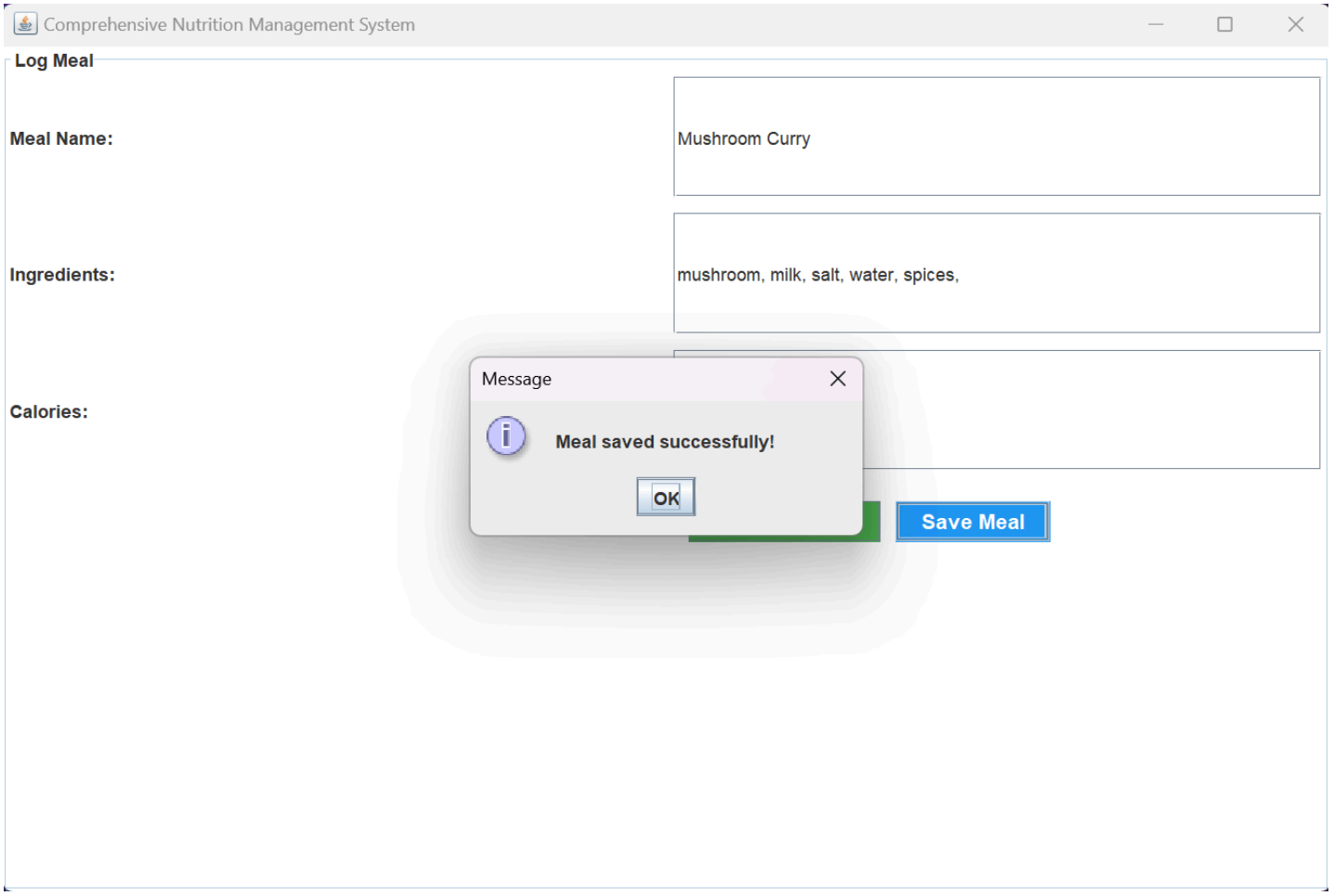


4.2 DASHBOARD

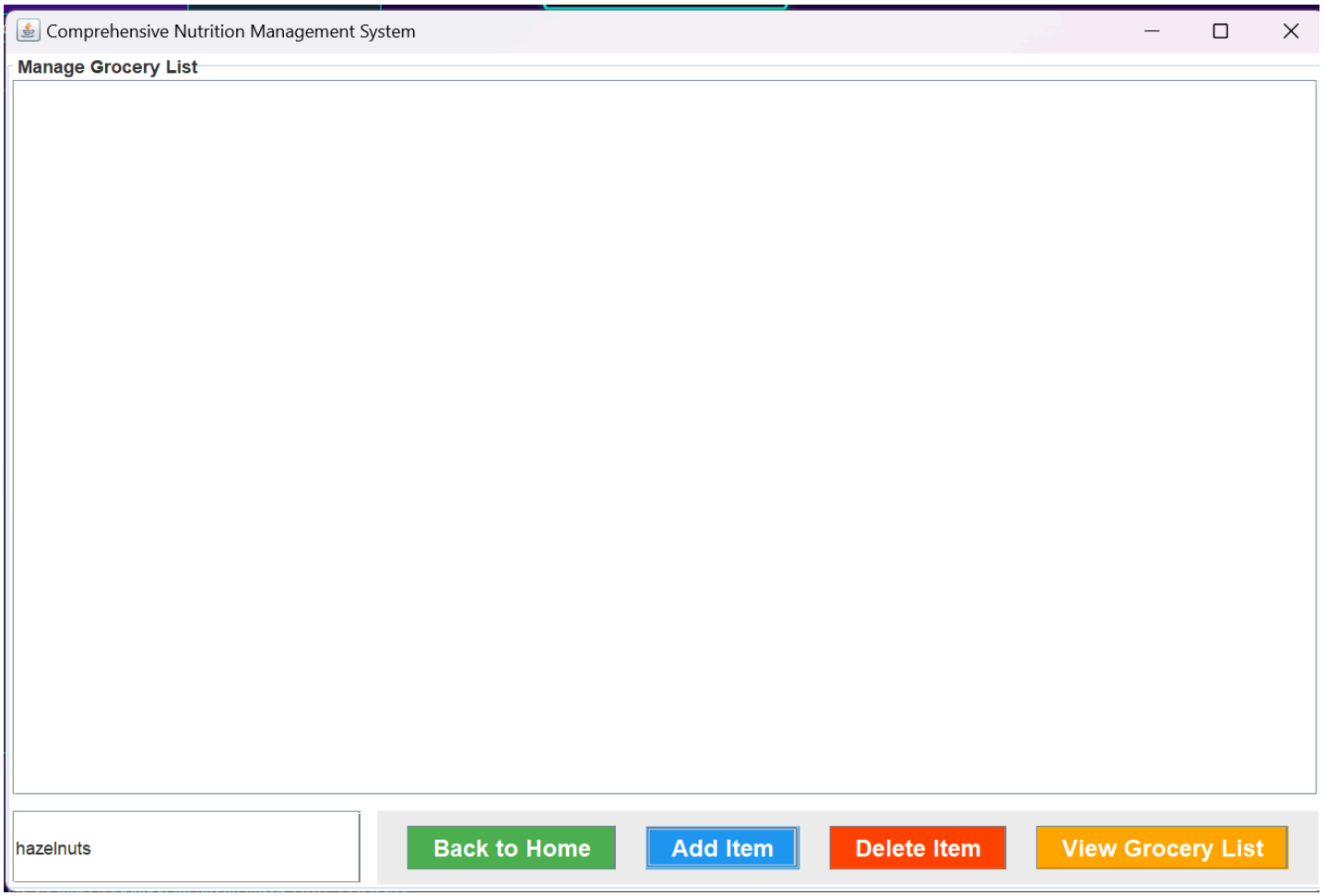

Comprehensive Nutrition Management System
Log a Meal
Manage Grocery List
Dietary Management
Generate Weekly Report
Back to Login

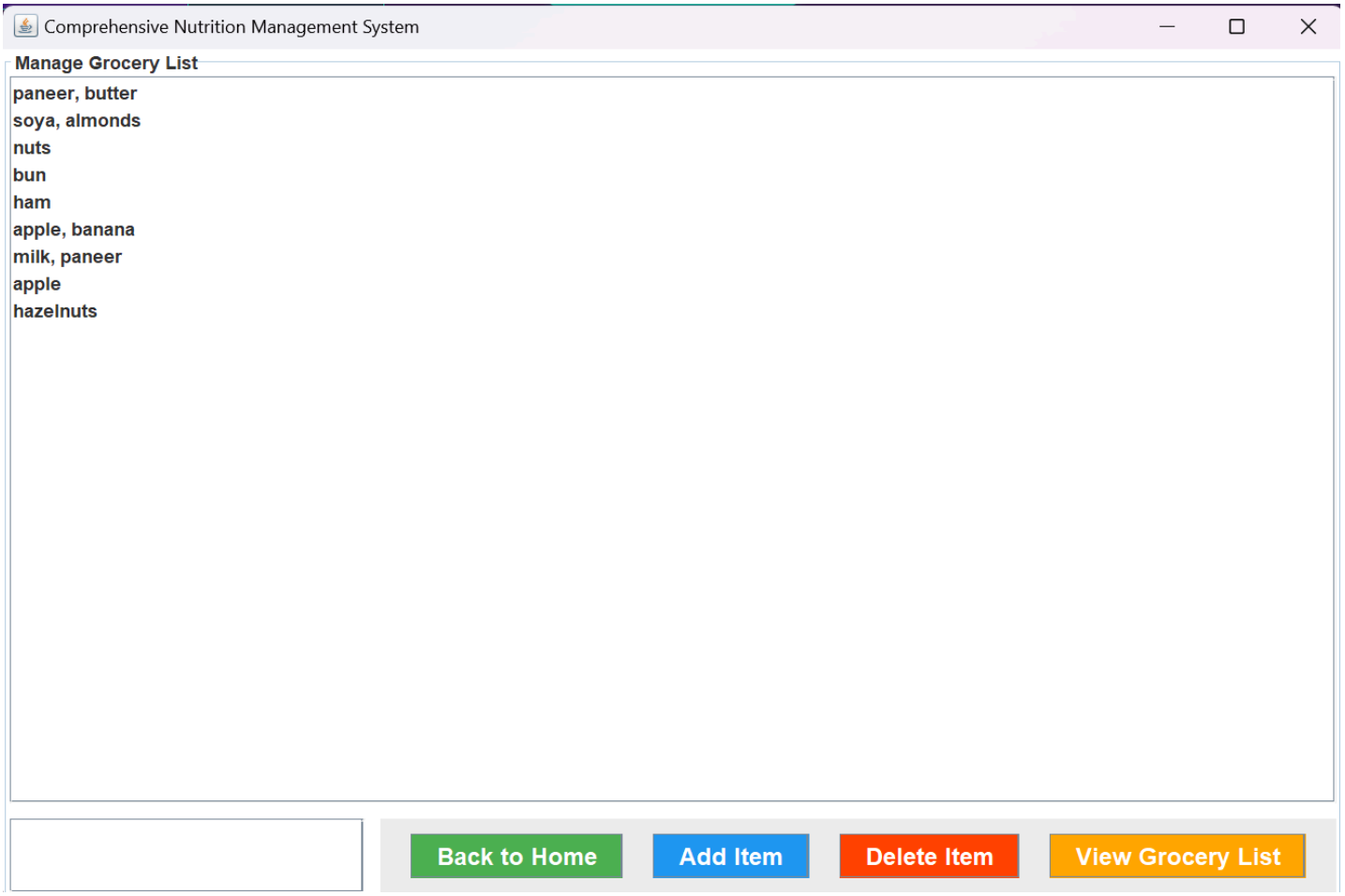
4.3 LOG MEAL PAGE:



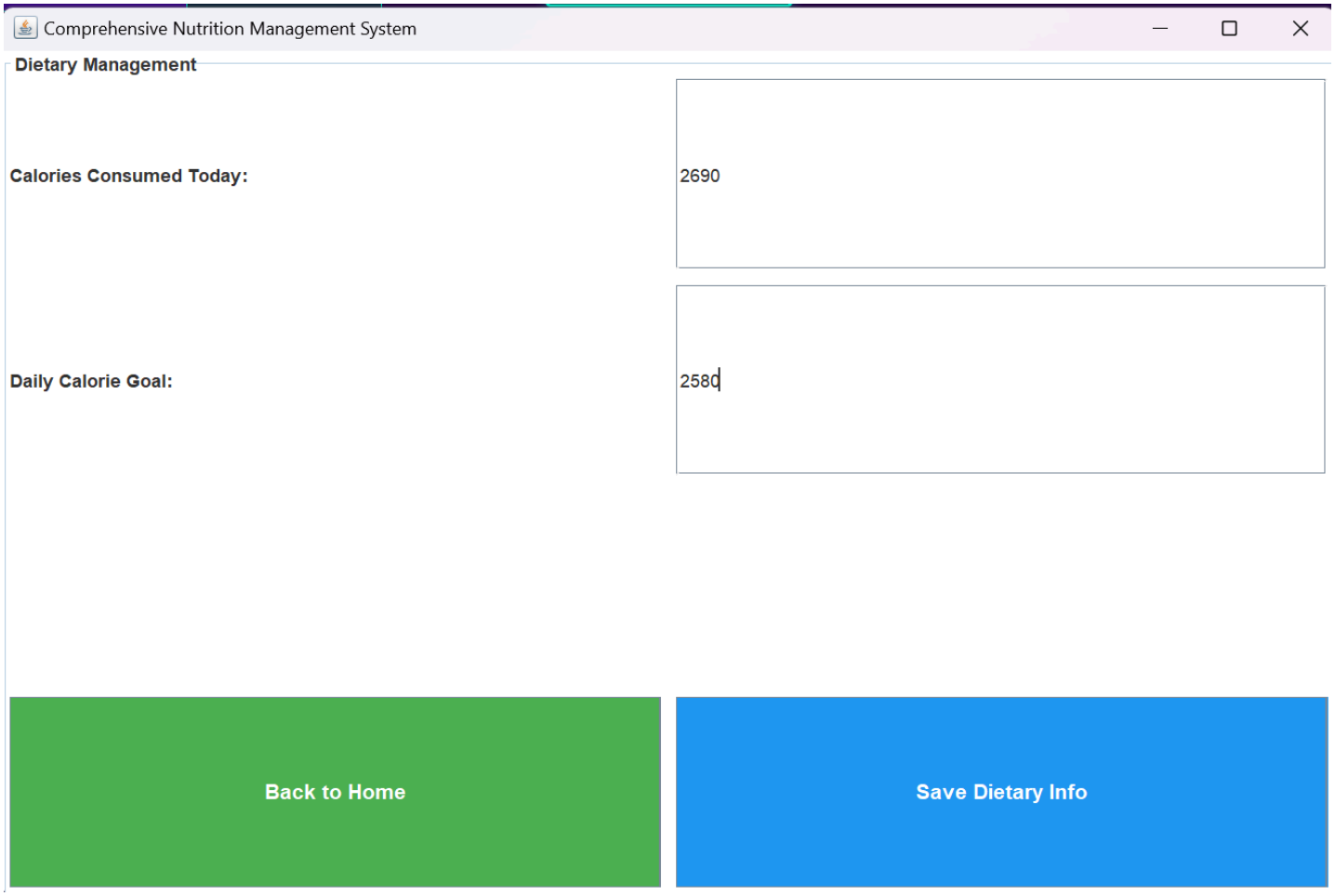

Comprehensive Nutrition Management System
Log Meal
Meal Name:
Mushroom Curry
Ingredients:
mushroom, milk, salt, water, spices,
Calories:
Message
Meal saved successfully!
OK
Save Meal

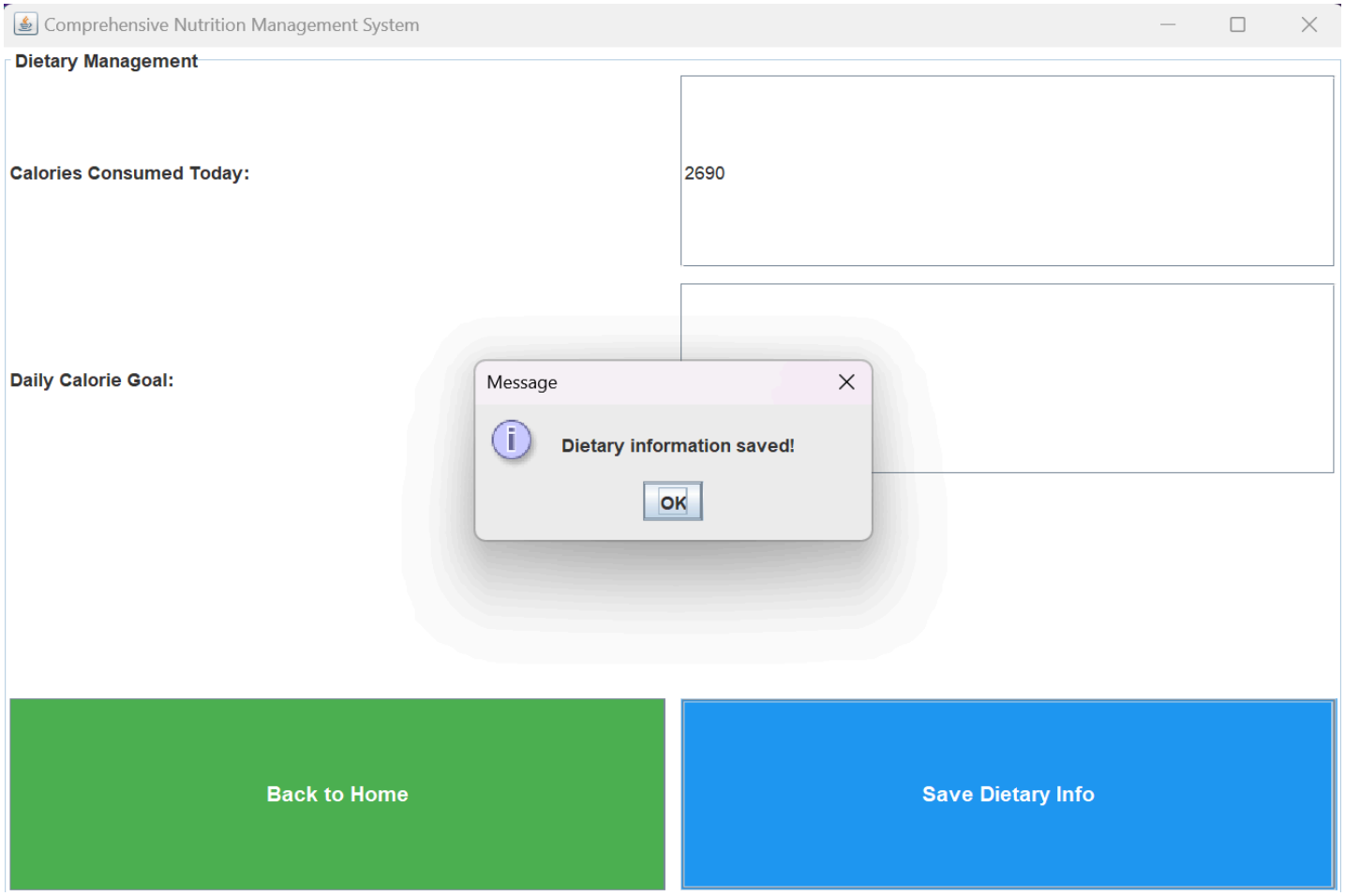
4.4 GROCERY MANAGEMENT PAGE



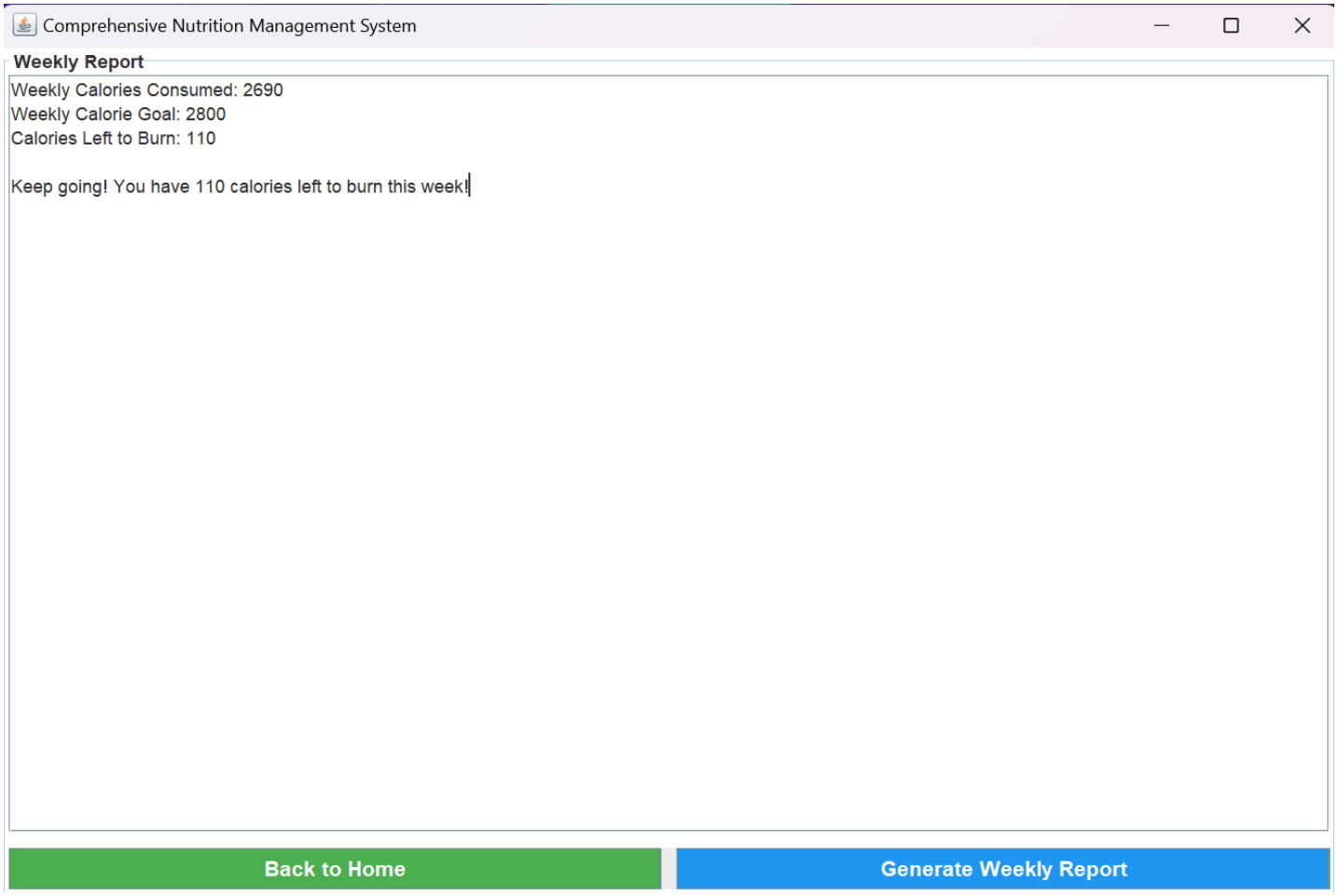

Comprehensive Nutrition Management System
Manage Grocery List
paneer, butter
soya, almonds
nuts
bun
ham
apple, banana
milk, paneer
apple
hazelnuts
Back to Home
Add Item
Delete Item
View Grocery List

4.5 CALORIE GOAL PAGE




Comprehensive Nutrition Management System
Dietary Management
Calories Consumed Today:
2690
Daily Calorie Goal:
Message
Dietary information saved!
OK
Back to Home
Save Dietary Info

4.6 WEEKLY REPORT PAGE:



CONCLUSION

The project offers a comprehensive solution for efficient nutrition management. By automating various tasks, this system enhances the accuracy and efficiency of managing nutritional data and meal planning. It provides a centralized platform for storing, retrieving, and analyzing nutrition-related information, enabling informed decision-making and streamlined dietary management. The system features a user-friendly interface for easy data input and retrieval, ensuring a smooth experience for users. With real-time tracking of nutritional intake, personalized meal recommendations, and automated report generation, the system helps optimize nutrient levels, reduce errors, and improve overall dietary health management.

REFERENCES

1. *https://www.javatpoint.com/java-tutorial*

2. *https://www.wikipedia.org/*

3. *https://www.w3schools.com/sql/*