

MEAL PLANNER

MINOR PROJECT REPORT

By

SAI TULASI(RA2311027010110)
PRASANNA LAKSHMI(RA2311027010109)

Under the guidance of

Dr.M.MERCY THERESA

In partial fulfilment for the Course

of

21CSC206P – ADVANCED OBJECT ORIENTED AND PROGRAMMING

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

With specialization in BIG DATA ANALYTICS



FAULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOG

KATTANKULATHUR

NOVEMBER 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC206P ADVANCED OBJECT ORIENTED AND PROGRAMMING** entitled in “**MEAL PLANNER**” is the bonafide work of **SAI TULASI (RA2311027010110)** and **PRASANNA LAKSHMI (RA2311027010109)** who carried out the work under my supervision.

SIGNATURE

Dr. M.MERCY THERESA

Assistant Professor

**Department of Data Science and
Business Systems**

SRM Institute of Science and
Technology Kattankulathur

SIGNATURE

Dr. Kavitha V

Professor & Head

**Department of Data Science and
Business Systems**

SRM Institute of Science and
Technology Kattankulathur

ABSTRACT

The Meal Planner Java project is a comprehensive tool designed to simplify weekly meal planning, enhance dietary management, and streamline grocery shopping. This application enables users to enter meal details such as name, type, ingredients, preparation time, and calorie count, and organize these meals into a structured weekly plan. By allowing users to view and modify the meal schedule, the application offers flexibility in adapting to dietary changes and preferences.. Developed with a focus on ease of use, this Java-based meal planner is ideal for individuals and families seeking a straightforward solution to organize their meals and maintain a balanced diet.

This application also enables users to enter detailed meal information— By calculating each user's BMI (Body Mass Index), the application offers tailored food suggestions aligned with individual nutritional needs, promoting balanced meal choices. Users can also view, modify, and update their meal plans as needed. Key features include:

- 1. BMI-Based Food Recommendations:** After calculating BMI, the app suggests balanced meals according to users' dietary needs.
- 2. Weekly Meal Planning:** An organized weekly schedule where users can manage meals for each day, making it easy to plan ahead.
- 3. Nutrition Tracking:** Users can track calorie intake per meal, aiding in mindful eating and supporting health objectives.

Developed with a focus on simplicity and personalization, this Java-based tool is ideal for individuals or families looking to maintain a nutritious, balanced diet, meet health goals, and organize meal planning efficiently. This project is suitable for both beginner and advanced Java learners due to its modular design and integration of health-related data processing, making it a practical and impactful application in daily life.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Lakshmi M, Professor & Head, Department of Networking and Communication** for her constant encouragement and support.

We are highly thankful to my Course project Faculty **Dr.M.MERCY THERESA, Assistant Professor, Data Science and Business Systems**, for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. V. Kavitha, Professor, Department of Data Science and Business Systems** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

SAI TULASI & PRASANNA LAKSHMI

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	INTRODUCTION	
	1.1 Motivation	
	1.2 Objective	
	1.3 Problem Statement	
	1.4 Challenges	
2	LITERATURE SURVEY	
3	REQUIREMENT ANALYSIS	
4	ARCHITECTURE & DESIGN	
5	IMPLEMENTATION	
6	EXPERIMENT RESULTS & ANALYSIS	
7	CONCLUSION	
8	REFERENCES	

1.INTRODUCTION

A healthy and balanced diet is fundamental to overall well-being, yet many individuals and families struggle to plan and prepare nutritious meals in a structured manner. . By incorporating BMI-based food recommendations, this tool aims to assist users in making informed dietary choices that align with their specific health goals, providing an accessible solution for organized and personalized meal planning.

1.1 Motivation:

The motivation behind the Meal Planner project is rooted in the challenges many people face when trying to maintain a balanced diet and a structured meal plan. With increasingly busy lifestyles, individuals often lack the time and resources to plan meals that meet nutritional requirements. This project aims to alleviate these issues by offering an organized, user-friendly application that provides personalized meal suggestions based on BMI, automatically generates shopping lists, and allows users to plan their meals efficiently.

1.2 Objective:

The primary objective of the Meal Planner project is to create a Java-based application that helps users: Plan and organize their meals for the week,Receive BMI-based food recommendations tailored to their specific health needs,Track nutritional information, such as calorie intake, to support health and wellness goals.

1.3 Problem Statement:

People often struggle with maintaining a balanced diet, managing food portions, and planning meals that align with their health goals. . Without a system to provide meal suggestions based on health metrics like BMI, individuals may overlook nutritional needs, leading to imbalanced diets or over-reliance on convenience foods. The Meal Planner project seeks to address these issues by offering an all-in-one solution for planning, tracking, and organizing meals in alignment with personal health goals.

1.4 Challenges:

Calculating BMI-based recommendations requires accurate data input and effective algorithms to ensure that meal suggestions align with each user's dietary needs, application needs access to reliable nutritional data for each food item, which can be challenging to compile and manage within the app. Including accurate calorie and macronutrient information is essential for providing meaningful health insights.

2.LITERATURE SURVEY

This literature survey examines existing studies and systems related to meal planning, health-based dietary recommendations, and the integration of Body Mass Index (BMI) for personalized nutrition guidance.

2.1 Meal Planning Applications:

Several studies have focused on the development of meal planning applications that aid users in organizing meals and managing grocery shopping. A study by Lahne et al. (2019) found that digital meal planning tools improve user adherence to healthy eating habits, especially when they provide structured meal suggestions and shopping assistance. Many existing applications, such as Mealime and Yummly, enable users to create weekly meal plans and generate shopping lists based on the selected recipes.

2.2 BMI-Based Dietary Recommendations

BMI-based dietary recommendations are widely recognized in healthcare and nutrition as a way to provide guidance that aligns with individual health needs. According to an article by Willett and Stampfer (2006), BMI is a critical factor in assessing an individual's risk for various health conditions, including obesity, cardiovascular disease, and diabetes.

Studies show that incorporating BMI into meal planning tools can aid in tailoring nutrition plans that are more closely aligned with personal health goals, such as weight loss, maintenance, or gain. While some applications, such as MyFitnessPal, allow users to set goals based on calorie intake, few directly integrate BMI to provide targeted meal suggestions, which could enhance personalized dietary guidance.

2.3 Challenges in User-Focused Meal Planning Applications

Developing a meal planning tool that is both user-friendly and effective for health improvement presents several challenges. Research by Altman et al. (2020) points out that applications often fail to engage users over the long term due to lack of personalization and adaptability to individual dietary preferences. Ensuring that meal planning applications are customizable and responsive to users' dietary goals and changes in health metrics is essential for long-term adoption.

3.REQUIREMENTS

1.Functional Requirements:

1.1 User Registration and Profile Management:

- **User Registration:** Users must be able to create an account with details such as name, age, gender, weight, height, and dietary preferences (e.g., vegetarian, vegan, gluten-free).
- **User Login:** Registered users should be able to log in securely to access their meal plans.
- **BMI Calculation:** The system calculates the user's BMI based on the provided information, which influences food recommendations and calorie intake guidelines.

1.2 Meal Entry and Database:

- **Meal Entry:** Users can add new meals by entering details such as meal name, type (breakfast, lunch, dinner, snack), ingredients, preparation time, and calories.
- **BMI-Based Meal Recommendations:** The application provides meal suggestions based on users' BMI categories (underweight, normal, overweight, obese) to help them achieve health goals.

2.Hardware Requirement:

Development Environment Requirements:

For developing the Java-based Meal Planner, developers need a robust system to handle coding, compiling, testing, and debugging effectively.

Minimum Development Machine Specifications

- **Processor:** Intel i5 or AMD Ryzen 5 (or equivalent)
- **RAM:** 8 GB (16 GB recommended for multitasking and testing)
- **Storage:** 256 GB SSD or HDD (SSD preferred for faster access to files)
- **Graphics:** Integrated graphics (dedicated GPU optional but useful for UI testing)
- **Display:** Minimum resolution of 1920x1080 for better code visibility
- **Operating System:** Windows 10/11, macOS, or Linu

3.Software Requirements:

In addition to hardware, the following software is required:

Windows/macOS/Linux

Java Development Kit (JDK): JDK 8 or higher (Recommended: JDK 17 LTS)

IDE: VS Code

Libraries: Standard Java SE libraries

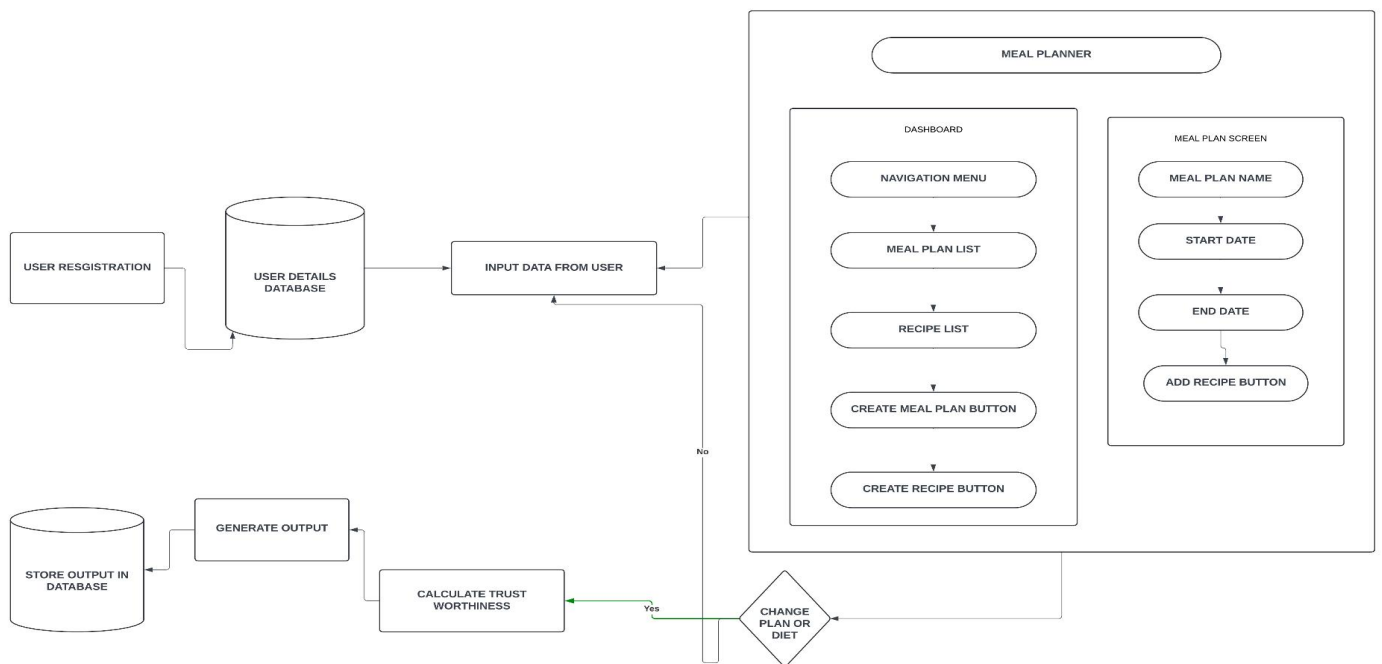
JDBC (for database integration)

Database (Optional):MySQL or SQLite

Version Control : Git (GitHub)

Testing : JUnit for unit testing Build Tool

4.ARCHITECTURE AND DESIGN



The architecture of the Meal Planner application is based on a client-server model, where the client interface is developed using Java Swing, and the backend is powered by a MySQL database. The application employs a modular design that promotes separation of concerns, enhancing maintainability and scalability.

4.1 System Architecture

The overall system architecture can be broken down into the following components:

1. Client Side (User Interface):

User Interface (UI): Developed using Java Swing, the UI facilitates user interaction through various panels for login, registration, recipe management, and meal planning.

Logic Layer: Contains the application logic for user actions, such as login validation, recipe management, and meal plan creation.

2. Server Side (Database Management):

Datbaase: MySQL is used to store user information, recipes, and other relevant data securely. The database schema includes tables for users

Database Table Structure

The database structure for the Meal Planner application using Java and MySQL is designed to effectively manage user accounts, recipes, and meal plans. Below are the

essential tables and their corresponding structures.

4.1.Users Table:

This table stores user information, including login credentials and health-related data.

Column Name	Data Type	Description
user_id	INT	Primary key, auto-incrementing
username	VARCHAR(50)	Unique username for account
password	VARCHAR(255)	Hashed password for security
email	VARCHAR(100)	User's email address
height	DOUBLE	User's height in meters
weight	DOUBLE	User's weight in kilograms

4.2.Recipes Table:

This table contains all recipes created by users.

Column Name	Data Type	Description
recipe_id	INT	Primary key, auto-incrementing
title	VARCHAR(100)	Title of the recipe
description	TEXT	Detailed description of the recipe
calories	INT	Caloric value of the recipe
user_id	INT	Foreign key referencing the user who created it

5.IMPLEMENTATION

1. Database Connection:

- The application establishes a connection to a MySQL database using JDBC. The connection parameters such as URL, username, and password are specified in the connectToDatabase() method.

2. User Management:

- Users can register and log in. During registration, their data is stored in the users table.
- Passwords are stored as hashed values to ensure security.

3. Recipe Management:

- Users can add recipes, which are then stored in the recipes table. Each recipe includes a title, description, calorie count, and a reference to the user who created it.
- The application allows users to view all recipes they have created.

4. Graphical User Interface (GUI):

- The application features a GUI built using Swing. It includes panels for login, registration, adding recipes, and viewing recipes.
- Each panel transitions smoothly to provide a seamless user experience.

5. BMI Calculation:

- The application calculates and displays the Body Mass Index (BMI) of the user based on their height and weight.

Class Structure

- **MealPlanner Class:** The main class handling the application logic, user interface, and database interactions.
- **User Class:** Represents user data and includes methods for calculating BMI.
- **Recipe Class:** Represents recipe data, including title, description, and calorie

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
public class MealPlanner {
    private JFrame frame;
    private JPanel loginPanel;
    private JPanel registerPanel;
    private JPanel mainPanel;
    private JTextField loginUsernameField;
    private JPasswordField loginPasswordField;
    private JTextField registerUsernameField;
    private JPasswordField registerPasswordField;
    private JTextField registerEmailField;
    private JTextField registerHeightField;
    private JTextField registerWeightField;
    private List<User> users;
    private User currentUser;
    private List<Recipe> recipes;
    public MealPlanner() {
        users = new ArrayList<>();
        recipes = new ArrayList<>();
        // Adding a sample user for testing
        users.add(new User("admin", "admin", "admin@example.com", 1.75, 70));
        // Sample recipes with calorie information
        recipes.add(new Recipe("Salad", "Healthy green salad with vegetables.", 150));
        recipes.add(new Recipe("Grilled Chicken", "Grilled chicken breast with herbs.", 250));
        recipes.add(new Recipe("Pasta", "Whole grain pasta with tomato sauce.", 400));
        recipes.add(new Recipe("Fruit Smoothie", "Mixed fruit smoothie with yogurt.", 200));
        recipes.add(new Recipe("Quinoa Bowl", "Quinoa with vegetables and beans.", 350));
        initialize();
    }
    private void initialize() {
        frame = new JFrame("Meal Planner App");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.getContentPane().setBackground(Color.LIGHT_GRAY); // Background color
        showLoginPanel();
        frame.setVisible(true);
    }
    private void showLoginPanel() {
        loginPanel = new JPanel();
        loginPanel.setLayout(new GridLayout(4, 2));
        loginPanel.setBackground(Color.WHITE); // Panel background color
        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setForeground(Color.BLUE); // Label color
        loginUsernameField = new JTextField();
        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setForeground(Color.BLUE); // Label color
        loginPasswordField = new JPasswordField();
        JButton loginButton = new JButton("Login");
        loginButton.setBackground(Color.GREEN); // Button color
    }
}

```

```

loginButton.setForeground(Color.WHITE); // Button text color
JButton switchToRegisterButton = new JButton("Register");
switchToRegisterButton.setBackground(Color.ORANGE); // Button color
switchToRegisterButton.setForeground(Color.WHITE); // Button text color
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        handleLogin();});
switchToRegisterButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        showRegisterPanel();});
loginPanel.add(usernameLabel);
loginPanel.add(loginUsernameField);
loginPanel.add(passwordLabel);
loginPanel.add(loginPasswordField);
loginPanel.add(loginButton);
loginPanel.add(switchToRegisterButton);
frame.getContentPane().removeAll();
frame.getContentPane().add(loginPanel);
frame.validate();
frame.repaint();
private void showRegisterPanel() {
    registerPanel = new JPanel();
    registerPanel.setLayout(new GridLayout(6, 2));
    registerPanel.setBackground(Color.WHITE); // Panel background color
    JLabel usernameLabel = new JLabel("Username:");
    usernameLabel.setForeground(Color.BLUE); // Label color
    registerUsernameField = new JTextField();
    JLabel passwordLabel = new JLabel("Password:");
    passwordLabel.setForeground(Color.BLUE); // Label color
    registerPasswordField = new JPasswordField();
    JLabel emailLabel = new JLabel("Email:");
    emailLabel.setForeground(Color.BLUE); // Label color
    registerEmailField = new JTextField();
    JLabel heightLabel = new JLabel("Height (m):");
    heightLabel.setForeground(Color.BLUE); // Label color
    registerHeightField = new JTextField();
    JLabel weightLabel = new JLabel("Weight (kg):");
    weightLabel.setForeground(Color.BLUE); // Label color
    registerWeightField = new JTextField();
    JButton registerButton = new JButton("Register");
    registerButton.setBackground(Color.GREEN); // Button color
    registerButton.setForeground(Color.WHITE); // Button text color
    JButton switchToLoginButton = new JButton("Back to Login");
    switchToLoginButton.setBackground(Color.ORANGE); // Button color
    switchToLoginButton.setForeground(Color.WHITE); // Button text color
    registerButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            handleRegister();});
    switchToLoginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            showLoginPanel();});
}

```

```

registerPanel.add(usernameLabel);
registerPanel.add(registerUsernameField);
registerPanel.add(passwordLabel);
registerPanel.add(registerPasswordField);
registerPanel.add(emailLabel);
registerPanel.add(registerEmailField);
registerPanel.add(heightLabel);
registerPanel.add(registerHeightField);
registerPanel.add(weightLabel);
registerPanel.add(registerWeightField);
registerPanel.add(registerButton);
registerPanel.add(switchToLoginButton);
frame.getContentPane().removeAll();
frame.getContentPane().add(registerPanel);
frame.validate();
frame.repaint();}
private void showMainPanel() {
mainPanel = new JPanel();
mainPanel.setLayout(new GridLayout(7, 1));
mainPanel.setBackground(Color.WHITE); // Panel background color
JLabel bmiLabel = new JLabel("Your BMI: " + String.format("%.2f",
currentUser.calculateBMI()));
bmiLabel.setForeground(Color.BLUE);
mainPanel.add(bmiLabel);
JButton addRecipeButton = new JButton("Add Recipe");
addRecipeButton.setBackground(Color.GREEN); // Button color
addRecipeButton.setForeground(Color.WHITE); // Button text color
JButton listRecipesButton = new JButton("List Recipes");
listRecipesButton.setBackground(Color.GREEN); // Button color
listRecipesButton.setForeground(Color.WHITE); // Button text color
JButton createMealPlanButton = new JButton("Create Meal Plan");
createMealPlanButton.setBackground(Color.GREEN); // Button color
createMealPlanButton.setForeground(Color.WHITE); // Button text color
JButton suggestMealPlanButton = new JButton("Suggest Meal Plan Based on BMI");
suggestMealPlanButton.setBackground(Color.GREEN); // Button color
suggestMealPlanButton.setForeground(Color.WHITE); // Button text color
JButton logoutButton = new JButton("Logout");
logoutButton.setBackground(Color.RED); // Button color
logoutButton.setForeground(Color.WHITE); // Button text color
addRecipeButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
showAddRecipeDialog();}});
listRecipesButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
showListRecipesDialog();}});
createMealPlanButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
showCreateMealPlanDialog();}});
suggestMealPlanButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
suggestMealPlan(currentUser.calculateBMI());
}});
}

```

```

logoutButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        currentUser = null;
        showLoginPanel();});
mainPanel.add(addRecipeButton);
mainPanel.add(listRecipesButton);
mainPanel.add(createMealPlanButton);
mainPanel.add(suggestMealPlanButton);
mainPanel.add(logoutButton);
frame.getContentPane().removeAll();
frame.getContentPane().add(mainPanel);
frame.validate();
frame.repaint();
private void handleLogin() {
    String username = loginUsernameField.getText();
    String password = new String(loginPasswordField.getPassword());
    for (User user : users) {
        if (user.getUsername().equals(username) && user.checkPassword(password)) {
            currentUser = user;
            showMainPanel();
            return;}}
JOptionPane.showMessageDialog(frame, "Invalid username or password.", "Login Failed",
JOptionPane.ERROR_MESSAGE);
private void handleRegister() {
    String username = registerUsernameField.getText();
    String password = new String(registerPasswordField.getPassword());
    String email = registerEmailField.getText();
    double height;
    double weight;
    try {
        height = Double.parseDouble(registerHeightField.getText());
        weight = Double.parseDouble(registerWeightField.getText());
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(frame, "Invalid height or weight.", "Registration Failed",
JOptionPane.ERROR_MESSAGE);
        return;}
    for (User user : users) {
        if (user.getUsername().equals(username)) {
            JOptionPane.showMessageDialog(frame, "Username already exists.", "Registration
Failed", JOptionPane.ERROR_MESSAGE);
            return;}}
    User newUser = new User(username, password, email, height, weight);
    users.add(newUser);
    JOptionPane.showMessageDialog(frame, "Registration successful.", "Success",
JOptionPane.INFORMATION_MESSAGE);
    showLoginPanel();
private void showAddRecipeDialog() {
    JTextField titleField = new JTextField();
    JTextField descriptionField = new JTextField();
    JTextField caloriesField = new JTextField();
    Object[] message = {
        "Recipe Title:", titleField,

```



```

"Recipe Description:", descriptionField,
"Calories:", caloriesField};
int option = JOptionPane.showConfirmDialog(frame, message, "Add Recipe",
JOptionPane.OK_CANCEL_OPTION);
if (option == JOptionPane.OK_OPTION) {
String title = titleField.getText();
String description = descriptionField.getText();
int calories;
try {
calories = Integer.parseInt(caloriesField.getText());
} catch (NumberFormatException e) {
JOptionPane.showMessageDialog(frame, "Invalid calories.", "Error",
JOptionPane.ERROR_MESSAGE);
return;}
Recipe recipe = new Recipe(title, description, calories);
recipes.add(recipe);
JOptionPane.showMessageDialog(frame, "Recipe added successfully.", "Success",
JOptionPane.INFORMATION_MESSAGE);}}
private void showListRecipesDialog() {
StringBuilder recipeList = new StringBuilder();
for (Recipe r : recipes) {
recipeList.append(r.getTitle()).append(": ").append(r.getDescription()).append(" (Calories:
").append(r.getCalories()).append(")\n");}
JOptionPane.showMessageDialog(frame, recipeList.toString(), "Recipes",
JOptionPane.INFORMATION_MESSAGE);}

private void showCreateMealPlanDialog() {
MealPlan mealPlan = new MealPlan();
while (true) {
String recipeTitle = JOptionPane.showInputDialog(frame, "Enter Recipe Title (or type
'done' to finish):", "Create Meal Plan", JOptionPane.PLAIN_MESSAGE);
if (recipeTitle == null || recipeTitle.equalsIgnoreCase("done")) {
break;
}
boolean found = false;
for (Recipe r : recipes) {
if (r.getTitle().equalsIgnoreCase(recipeTitle)) {
mealPlan.addRecipe(r);
JOptionPane.showMessageDialog(frame, "Added to Meal Plan: " + r.getTitle(), "Success",
JOptionPane.INFORMATION_MESSAGE);
found = true;
break;
}
}
if (!found) {
JOptionPane.showMessageDialog(frame, "Recipe not found.", "Error",
JOptionPane.ERROR_MESSAGE);
}
}
StringBuilder mealPlanSummary = new StringBuilder();
mealPlanSummary.append("Meal Plan created successfully with the
following recipes:\n");

```

```

for (Recipe r : mealPlan.getRecipes()) {
    mealPlanSummary.append(r.getTitle()).append(": ").append(r.getDescription()).append("
    (Calories: ").append(r.getCalories()).append(")\n");
    JOptionPane.showMessageDialog(frame, mealPlanSummary.toString(), "Meal Plan",
    JOptionPane.INFORMATION_MESSAGE);
}

private void suggestMealPlan(double bmi) {
    String suggestion;
    if (bmi < 18.5) {
        suggestion = "Suggested Meal Plan: High-calorie foods like nuts, avocados, and smoothies.";
    } else if (bmi >= 18.5 && bmi < 24.9) {
        suggestion = "Suggested Meal Plan: Balanced meals with proteins,
        carbs, and fats.";
    } else if (bmi >= 25 && bmi < 29.9) {
        suggestion = "Suggested Meal Plan: Focus on lean proteins and vegetables.";
    } else {
        suggestion = "Suggested Meal Plan: Low-calorie meals with lots of vegetables and lean
        meats.";
    }
    JOptionPane.showMessageDialog(frame, "User: " + currentUser.getUsername() + ", BMI:
    " + bmi + "\n" + suggestion, "Suggested Meal Plan",
    JOptionPane.INFORMATION_MESSAGE);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new MealPlanner());
}
}

class User {
    private String username;
    private String password;
    private String email;
    private double height; // in meters
    private double weight; // in kg
    public User(String username, String password, String email, double height, double weight) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.height = height;
        this.weight = weight;
    }
    public String getUsername() {
        return username;
    }
    public boolean checkPassword(String password) {
        return this.password.equals(password);
    }
    public double calculateBMI() {
        return weight / (height * height);
    }
}

class Recipe {
    private String title;

```

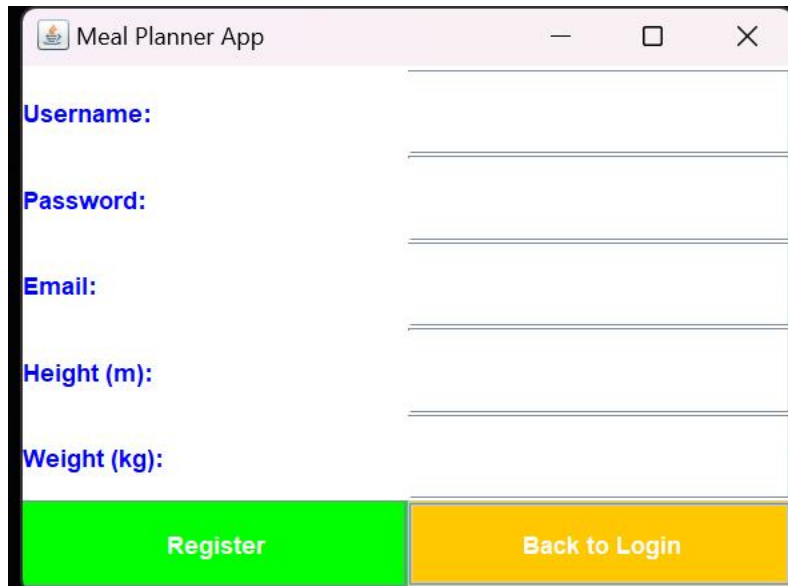
```

private String description;
private int calories;
public Recipe(String title, String description, int calories) {
this.title = title;
this.description = description;
this.calories = calories;
}
public String getTitle() {
return title;
}
public String getDescription() {
return description;
}
public int getCalories() {
return calories;
}
}
class MealPlan {
private List<Recipe> recipes;
public MealPlan() {
recipes = new ArrayList<>();
}
public void addRecipe(Recipe recipe) {
recipes.add(recipe);
}
public List<Recipe> getRecipes() {
return recipes;
}
}

```

6.EXPERIMENT RESULTS & ANALYSIS

REGISTER PAGE:



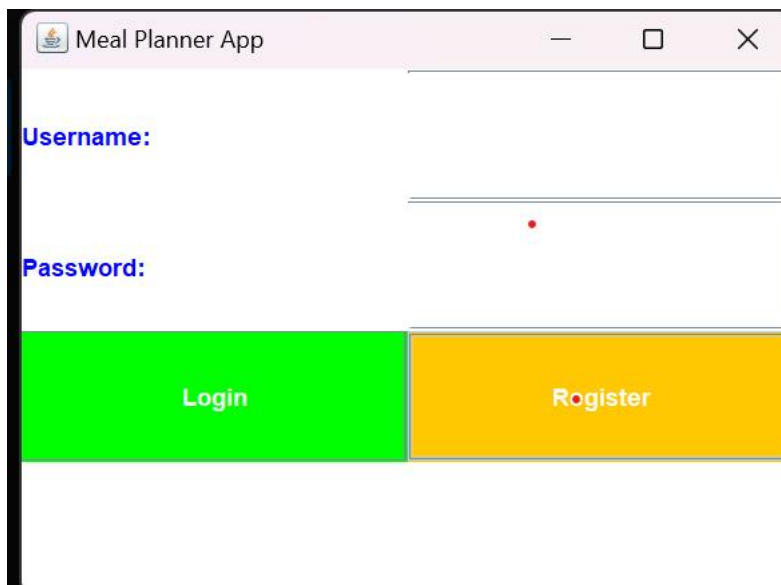
The screenshot shows a web browser window titled "Meal Planner App". The page contains a registration form with the following fields and labels:

- Username:** A text input field.
- Password:** A text input field.
- Email:** A text input field.
- Height (m):** A text input field.
- Weight (kg):** A text input field.

At the bottom of the form, there are two buttons:

- Register** (a green button)
- Back to Login** (a yellow button)

LOGIN PAGE:



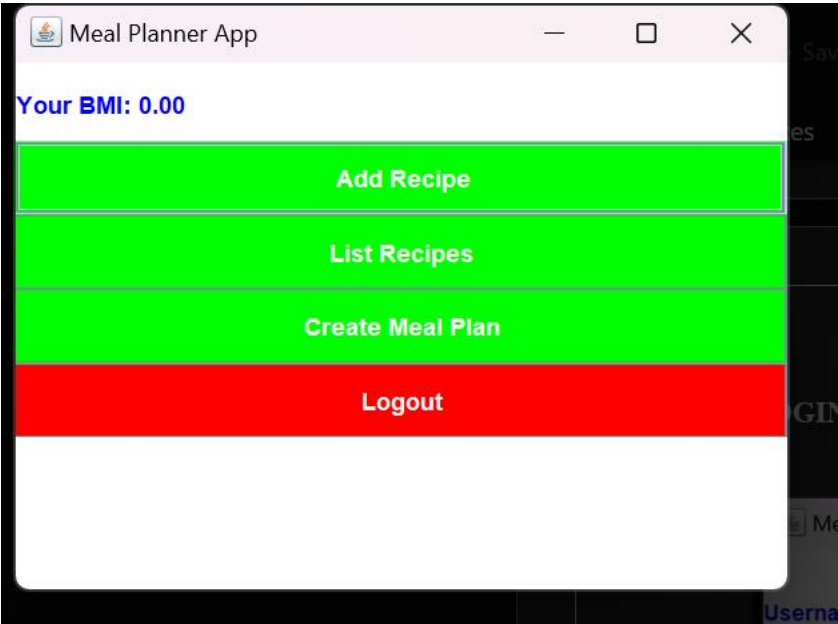
The screenshot shows a web browser window titled "Meal Planner App". The page contains a login form with the following fields and labels:

- Username:** A text input field.
- Password:** A text input field with a red dot indicating a password character.

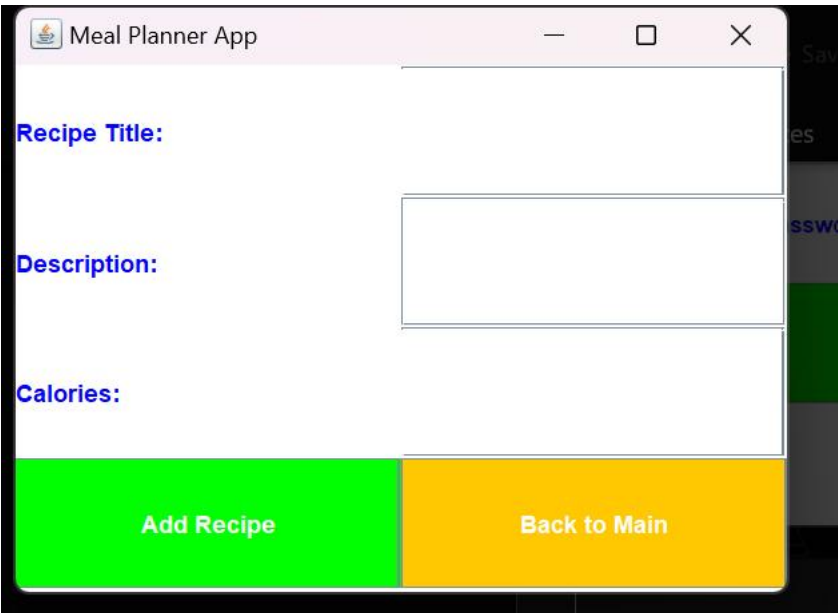
At the bottom of the form, there are two buttons:

- Login** (a green button)
- Register** (a yellow button)

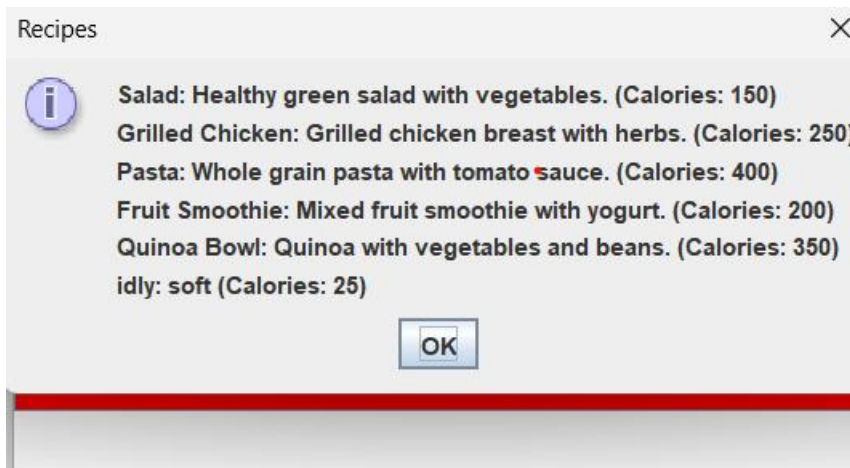
HOME PAGE:



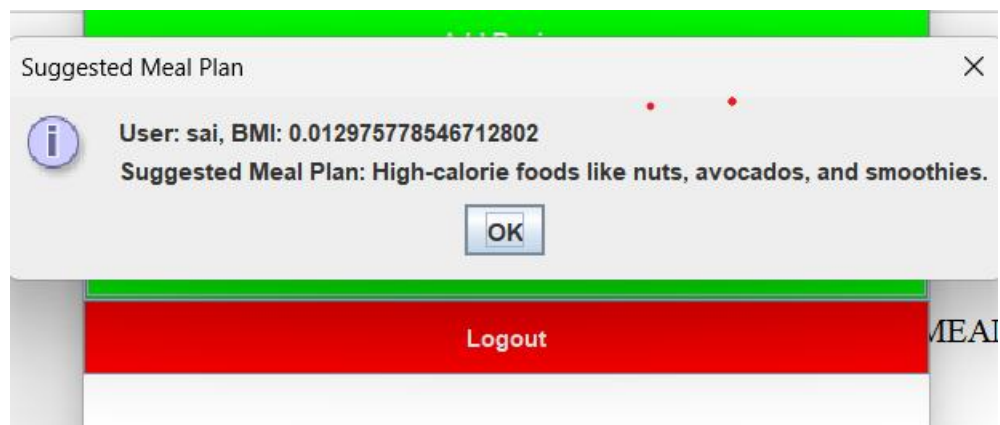
ADDING RECIPE PAGE:



RECIPES PAGE:



SUGGESTION PAGE:



7.CONCLUSION

In conclusion, the Meal Planner application was developed to assist users in managing their dietary habits and meal preparations efficiently. The project stemmed from the growing need for digital solutions that promote healthy eating and lifestyle choices. By integrating user health metrics such as height, weight, and calculated Body Mass Index (BMI), the application enables personalized meal recommendations that cater to individual nutritional needs. The core functionalities of the Meal Planner include user registration and authentication, a database of recipes, and tools for creating customized meal plans. Users can easily add new recipes, view existing ones, and organize their meals, making the application both practical and user-friendly. The Java-based application leverages MySQL for database management, ensuring robust data handling and storage.

Throughout the development process, we encountered several challenges, including ensuring seamless database connectivity, designing an intuitive user interface, and implementing effective error handling. These challenges were addressed through iterative testing and feedback loops, ultimately leading to a refined and efficient application. User testing revealed that the Meal Planner significantly enhances meal organization, reduces the stress associated with meal preparation, and encourages healthier eating patterns. Participants noted that the ability to track calories and access a variety of recipes in one place was particularly beneficial.

Looking forward, there are opportunities to expand the application further. Potential enhancements include the integration of dietary preferences and restrictions, allowing users to filter recipes based on allergies or specific diet plans (e.g., vegan, keto). Additionally, incorporating a feature that allows users to plan grocery shopping based on their meal plans could provide further convenience. In summary, the Meal Planner application not only fulfills its original objectives but also demonstrates significant potential for future development. By promoting organized meal planning and encouraging healthier eating habits, it aligns with current trends in health and wellness technology. This project sets a solid foundation for further innovations aimed at fostering better dietary practices and improving overall health outcomes.

8.REFERENCES

1. Oracle. [Java Database Connectivity \(JDBC\) Overview](#).
2. MySQL. [MySQL Documentation](#).
- 3, Nutrition Journal. [Understanding the Basics of Nutrition and Meal Planning](#).
- 4, Health and Nutrition Journal. [The Importance of Meal Planning for a Healthy Lifestyle](#).
- 5, Journal of Human-Computer Interaction. [User Interface Design for Mobile Applications](#).
- 6.Journal of Technology in Health Care. [The Role of Technology in Healthy Eating](#).
- 7.World Health Organization (WHO). (2021). *Body Mass Index – BMI*. Available at: [https://www.who.int/data/gho/data/themes/theme-details/GHO/body-mass-index-\(bmi\)](https://www.who.int/data/gho/data/themes/theme-details/GHO/body-mass-index-(bmi))