Prince Sultan University

Department of Computer & Information Sciences

Calorie Calculator

IS499 Senior Project

First **Semester, Term 232**

Instructor: Dr. Suliman Mohamed Fati

Team Members

218210865	MOHAMMED SHAFI MOHAMMED ALANAZI
220110496	ABDALRAHMAN ABDALAZIZ MOHAMAD FALAH
220110252	KHALID HASAN MUSTFA ALSHEHRI
219110687	NAWAF ALI SAEED BIN JAMAN
221111057	ABDULRAHMAN ABDULLAH SAEED ALSABER

Table of Contents

Summary

The calorie calculator system is designed to empower users by providing personalized meal plans and a user-friendly platform for managing their dietary goals. This innovative solution aims to address the diverse needs of health-conscious individuals by combining the latest technologies with a robust and scalable design. The system not only supports users in making informed dietary decisions but also simplifies the process of tracking caloric intake, calculating BMI, and planning meals tailored to individual preferences and health objectives.

This document comprehensively outlined all key aspects of the project, starting with the objectives and moving through the requirements, design, implementation strategies, testing plans, and constraints. These elements were meticulously planned and integrated to ensure that the system delivers both functionality and user satisfaction. By defining a clear roadmap and aligning it with project goals, the calorie calculator system has been developed to be efficient, reliable, and scalable.

The project leverages modern web development tools and frameworks, including React for the front-end, Node.js and Express.js for the back-end, and MongoDB as the database solution. These technologies provide a solid foundation for the system, offering several key benefits:

- Scalability: The system can accommodate a growing user base without performance degradation, thanks to its cloud-based deployment on Google Cloud Platform (GCP).
- Flexibility: The modular design allows for future enhancements, such as AI-based meal recommendations or integration with wearable devices.
- Responsiveness: A user-friendly interface ensures that the system is accessible across a variety of devices, including desktops, tablets, and smartphones.

Despite its many strengths, the project faced certain constraints and limitations, such as a strict six-month timeline, the need for Saudi Personal Data Protection Law. compliance, and resource limitations. However, these challenges were addressed through careful planning, prioritization, and risk mitigation strategies. For example, critical features like user registration, BMI calculation, and meal plan generation were prioritized to ensure the core functionality of the system was delivered on time.

The implementation of a robust admin panel further enhances the system by enabling administrators to manage food data, user accounts, and calorie values effectively. This ensures that the backend operations remain efficient and the system continues to deliver high-quality outputs to users.

In conclusion, the calorie calculator system represents a significant step forward in dietary management solutions. By integrating cutting-edge technology with user-centered design principles, it provides a practical and effective tool for improving health and wellness. The comprehensive planning and execution of this project have laid a strong foundation for its future success, ensuring that it meets both current needs and potential growth opportunities.

Acknowledgments

We extend our gratitude to the following individuals and organizations for their support and guidance throughout the project:

- Project Supervisors: For their invaluable feedback and assistance in refining the project scope and objectives.
- Development Team Members: For their dedication and collaboration during all phases of the project.

- University Resources: For providing access to tools, platforms, and infrastructure necessary for development and testing.

Future Enhancements for the Calorie Calculator System

The calorie calculator system, while robust and comprehensive in its current form, can be expanded and improved with several future enhancements to meet evolving user needs and technological advancements. These enhancements aim to increase the system's functionality, user engagement, and overall effectiveness.

1. Introduction

1.1 Purpose

The purpose of this section is to provide a comprehensive framework for the architecture and design of the calorie calculator system. By employing structured models and diagrams, this section defines the organization, components, and interactions within the system, ensuring that it aligns seamlessly with both functional and non-functional requirements. A well-documented design serves as a blueprint for development, testing, and maintenance, reducing ambiguities and streamlining the implementation process.

Key Objectives of the Software Design

1. System Architecture Definition:
   This section aims to establish a clear architectural structure, detailing how the different components of the system interact and function together. It outlines the roles and responsibilities of each component, including the front-end interface, back-end server logic, and database.
2. Component Interaction and Integration:
   The design focuses on how the front-end, back-end, and database layers communicate, ensuring seamless data flow and real-time responsiveness. It specifies the integration points, such as APIs for user authentication, BMI calculation, and personalized meal plan generation.
3. Alignment with Requirements:
   By mapping functional and non-functional requirements to specific components and interactions, the design ensures that every aspect of the system supports user needs and project goals. For example, the need for secure login and data encryption directly influences the back-end authentication mechanisms.
4. Support for Scalability and Flexibility:
   The design is structured to accommodate future enhancements and an increasing user base.

Modular components allow for easy updates, such as adding new meal planning algorithms or integrating third-party APIs.

5. Facilitate Development and Maintenance:
   A clear and detailed design reduces development risks by minimizing misunderstandings and guiding developers through the implementation process. It also simplifies debugging and future maintenance efforts by providing a documented structure for reference.
6. Visualization Through Diagrams:
   The section incorporates diagrams such as use case diagrams, component diagrams, and sequence diagrams to provide a visual representation of the system. These diagrams enhance understanding by illustrating the flow of data, interactions between components, and user scenarios.

Importance of a Strong Design Foundation

A robust software design is crucial to the success of the calorie calculator system. It ensures that the system is built on a foundation that is:

- Reliable: The architecture must support consistent functionality and high availability.
- Efficient: The design must optimize resource usage, including processing power and database queries.
- Scalable: As the user base grows, the system should handle increased demands without degradation in performance.
- Secure: The design must incorporate mechanisms to protect user data and ensure compliance with Saudi Personal Data Protection Law. and other regulatory requirements.
- User-Friendly: A well-structured design supports the creation of intuitive interfaces, ensuring a positive user experience.

1.2 Document Conventions

This documentation adheres to the APA (American Psychological Association) style, ensuring consistency and professionalism throughout. The following conventions are applied:

- Formatting: A 12-point Times New Roman font is used with 1-inch margins on all sides, and the text is double-spaced.
- Structure: Headings and subheadings are organized hierarchically, formatted in bold, and labeled (e.g., Heading 1, Heading 2).

- Title Page: Includes the project's title and the institutional affiliation.
- Figures and Tables: Included where necessary to clarify concepts and enhance readability.
- References: All sources are cited according to APA guidelines to ensure proper attribution and academic integrity.

These conventions facilitate a clear and systematic presentation of the project, making it accessible to technical and non-technical audiences alike.

1.3 Intended Audience and Reading Suggestions

The documentation is intended for the following audiences:

- Project Supervisors: To monitor the project's progress and ensure alignment with objectives.
- Development Team: To provide guidance and a shared understanding of tasks and deliverables.
- Future Developers: To enable further development and maintenance of the system.
- General Readers: To gain insights into the project's purpose, execution, and impact.

Reading Suggestions:

- Section 1: Overview of the project's goals and structure.
- Section 2: Technical details for readers with a development or engineering background.
- Section 3: Evaluation and results for assessing the project's outcomes and effectiveness.

1.4 Problem Statement

Health problems related to poor dietary habits, such as obesity and malnutrition, are increasingly common. Current tools for calorie tracking often lack personalization, failing to accommodate user-specific needs like dietary preferences or allergies. These shortcomings hinder users from achieving their health goals effectively. A solution that combines personalization, user engagement, and comprehensive nutritional tracking is needed to address this gap.

1.5 Proposed Solution

The proposed solution is a web-based calorie calculator that offers:

A user-friendly interface for entering personal details and dietary preferences.

Personalized BMI calculation and tailored 30-day meal plans that exclude allergenic foods.

Dynamic updates using AJAX for seamless interaction.

An admin control panel for managing food data, meal plans, and calorie information. This approach ensures a tailored and inclusive platform to support users in their health journeys.

1.6 Related Software Products

While several tools exist for calorie tracking, they lack key features provided by this project: MyFitnessPal: Popular but does not offer personalized meal plans.

Lose It!: Basic calorie tracking without allergy-specific customization.

Cronometer: Focuses on nutritional details but lacks dynamic updates or admin management. This project addresses these gaps, delivering a comprehensive and user-centric solution.

1.7 Project Contribution

The project's contributions include:

Personalized Nutrition: Offering meal plans tailored to user preferences and allergies.

Enhanced Management: Introducing an admin panel for efficient food and calorie management.

Improved User Experience: Leveraging AJAX for real-time updates and a seamless interface.

Health Awareness: Providing BMI feedback and calorie tracking to encourage healthier choices.

1.8 Project Work Plan

1.8.1 Project Tasks and Task Assignments

Task ID	Task Description	Assigned To	Estimated Completion Date
1	Requirement Analysis	All Team Members	Week 1
2	Database Design	All Team Members	Week 2
3	Front-End Development	All Team Members	Week 4
4	Back-End Development	All Team Members	Week 6
5	Testing and Debugging	All Team Members	Week 8
6	Final Report and Documentation	All Team Members	Week 9

1.8.2 Expected Deliverables

- Project Proposal: An initial document detailing the project's objectives and scope.
- Midway Report: A progress report summarizing completed tasks and outlining future steps.
- 1st Implementation and Coding Iteration (Second Semester): Initial development of core features.
- 2nd Implementation and Coding Iteration (Second Semester): Integration of dynamic updates and advanced functionality.
- 3rd Implementation and Coding Iteration (Second Semester): Finalized features and thorough testing.
- Final Report: A comprehensive document detailing the entire project lifecycle.
- Final Presentation: A summary of the project's objectives, implementation, and outcomes.
- Final Product: The complete, operational calorie calculator website.

1.9 Gantt Chart

Figure 1. Gantt Chart (Documentation)

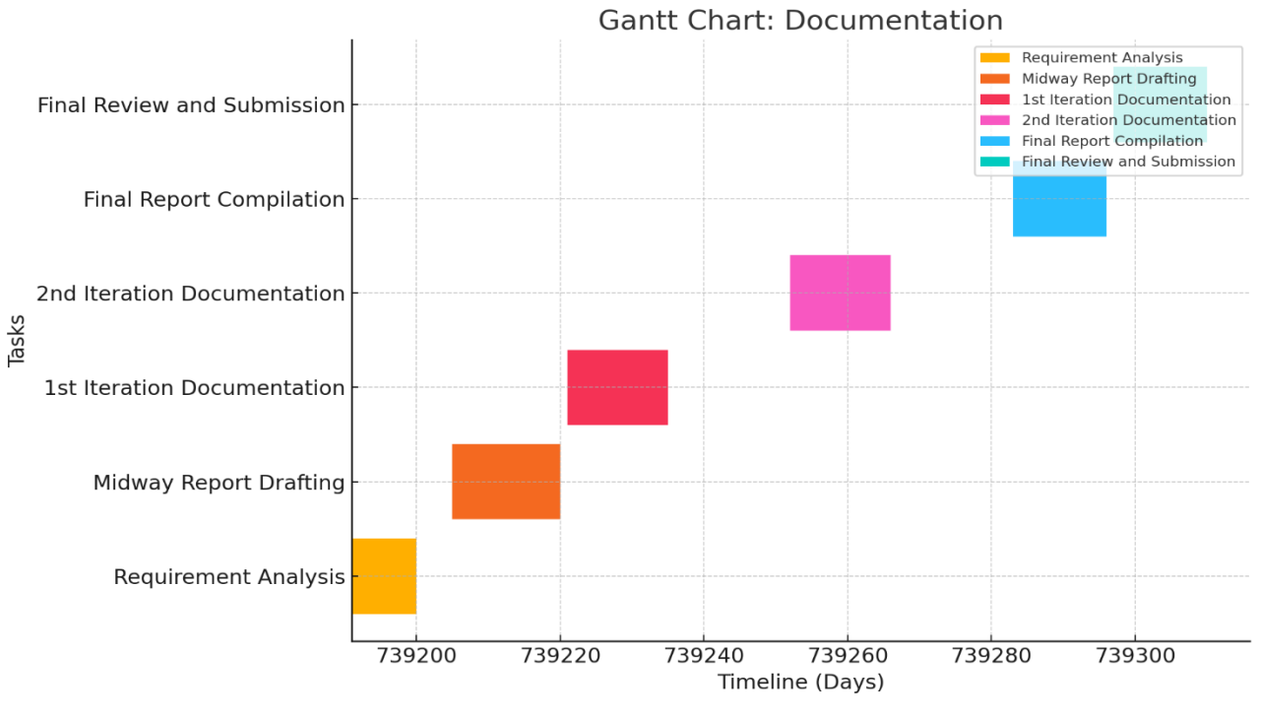


Figure 1 gantt chart Documentation

Figure 2. Gantt Chart (Implementation)

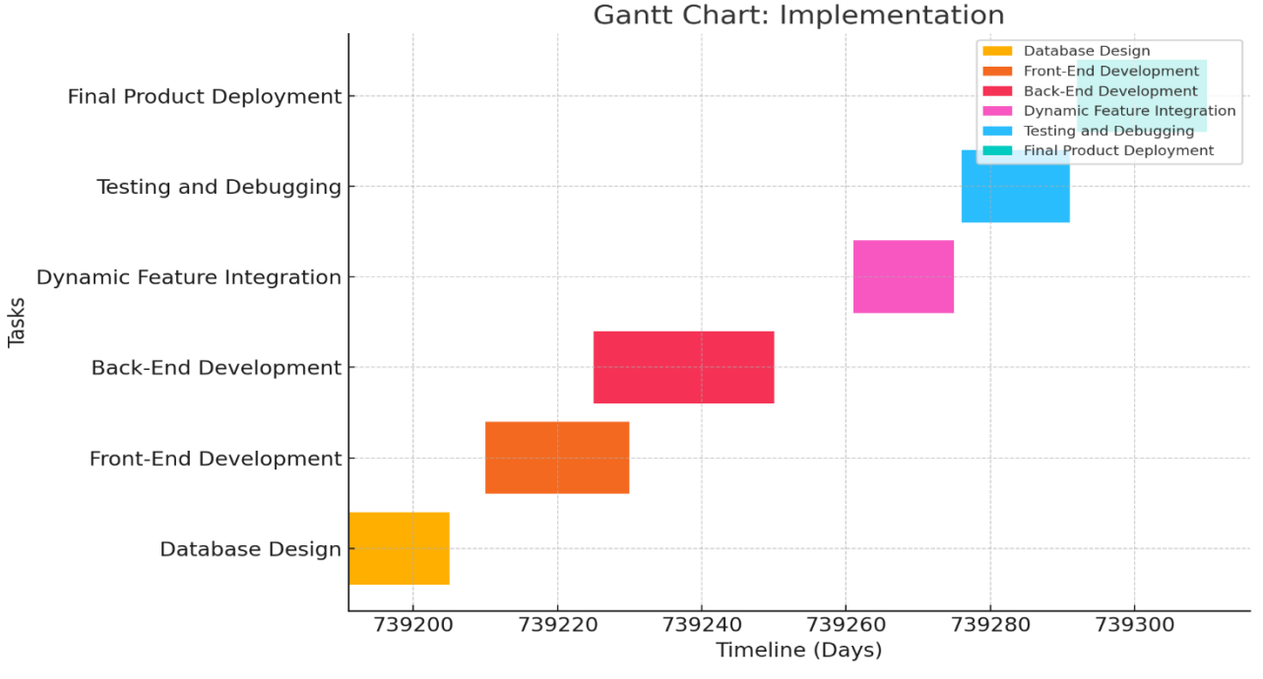


Figure 2 gantt chart implementation

1.10 Risk Management

Table 2. Risk Management Table

ID	Category	Risk Description	Likelihood	Impact	Severity	Countermeasures
1	Time	Delays in database setup	High	High	High	Allocate additional resources to mitigate delays.
2	Requirements	Scope creep due to unclear project goals	High	Medium	Medium	Establish clear and well-defined scope, monitor changes, and document reasons for them.
3	Requirements	Errors in calorie calculation logic	Medium	Medium	Medium	Perform rigorous testing and debugging to ensure calculations are accurate.
4	Compatibility	Compatibility issues with browsers	Low	Medium	Medium	Conduct thorough cross-browser testing to ensure functionality across all platforms.

5	Functionality	File upload feature malfunction	Medium	High	High	Use robust third-party libraries or frameworks to enhance reliability.
6	People	Lack of adoption due to usability issues	High	Medium	High	Understand user needs, improve UI/UX, and conduct targeted communication campaigns.
7	Coding	Errors in coding leading to broken features	Medium	Medium	Medium	Assign regular peer code reviews and establish a quality assurance process.
8	Scheduling	Dependencies and delays	High	Medium	High	Identify critical paths and prioritize tasks to minimize bottlenecks.

Survey Question	Answer Option 1	Answer Option 2	Answer Option 3	Answer Option 4	Comments/Notes
1. How would you rate the ease of registration on the system?	Excellent	Good	Average	Poor	
2. Do you find the BMI calculation accurate and helpful?	Yes	No	Neutral		
3. How satisfied are you with the personalized meal plans generated by the system?	Very Satisfied	Satisfied	Neutral	Dissatisfied	
4. Do you think the admin panel features (e.g., food management) are effective for system maintenance?	Yes	No	Neutral		
5. How important is multilingual support for your user experience?	Very Important	Important	Neutral	Not Important	
6. Do you find the system's design responsive and user-friendly across devices?	Yes	No	Neutral		
7. Would you like additional features like AI-based recommendations or integration with wearable devices?	Yes	No	Maybe		

Survey Question	Answer Option 1	Answer Option 2	Answer Option 3	Answer Option 4	Comments/Notes
8. How would you rate the system's compliance with data privacy standards (e.g., Saudi Personal Data Law)?	Excellent	Good	Average	Poor	
9. What improvements would you like to see in the meal planning feature?	[Open-ended]				
10. How likely are you to recommend this system to others?	Very Likely	Likely	Neutral	Unlikely	

2. Software Requirements Engineering Process

2.1 Purpose

The purpose of the software requirements engineering process is to establish a systematic approach for identifying, analyzing, documenting, and validating the requirements of the calorie calculator project. This process is critical to ensuring that the final product aligns with the expectations of stakeholders, meets functional and non-functional goals, and adheres to constraints such as budget, timeline, and technology. By capturing and clarifying requirements early in the project, the process mitigates risks associated with unclear, incomplete, or misunderstood requirements.

The software requirements engineering process bridges the gap between stakeholders (e.g., end-users, administrators) and developers by creating a shared understanding of the system's intended functionality. This collaboration ensures that the project delivers maximum value, minimizing the likelihood of rework or failure during later stages of development. Through iterative validation and

refinement, the process ensures that requirements remain relevant, feasible, and consistent with project objectives.

2.2 Requirements Engineering Process

The requirements engineering process is a systematic approach that defines, refines, and documents the system requirements to ensure clarity, feasibility, and alignment with the project's goals. This process is essential for bridging the gap between stakeholders and the development team, transforming abstract needs into actionable and well-structured development plans. The process consists of four interconnected and iterative phases: elicitation, analysis, specification, and validation. Each phase plays a critical role in ensuring that the system meets user expectations while adhering to technical, budgetary, and timeline constraints.

1. Elicitation

The first phase focuses on gathering and identifying the requirements from stakeholders, including end-users, administrators, and project sponsors. This step leverages various techniques such as interviews, surveys, and competitive analysis to capture a comprehensive set of needs and expectations. During elicitation:

- Stakeholders' pain points, preferences, and goals are discussed to align the system's purpose with their priorities.
- Existing solutions are analyzed to identify gaps and opportunities for improvement, ensuring the system stands out by addressing unmet needs.

2. Analysis

Once the requirements are gathered, they are rigorously examined to ensure feasibility and consistency. This phase involves:

- Prioritization: Ranking requirements by importance to determine which features must be implemented first.
- Conflict Resolution: Addressing and reconciling contradictory requirements among stakeholders.

- Feasibility Analysis: Evaluating whether the requirements can be realistically achieved within the project's technical, financial, and time constraints. This phase ensures that the requirements are practical, unambiguous, and aligned with the project’s overarching goals.

3. Specification

The analyzed requirements are then documented in a structured and detailed format, serving as a formal reference for the development team. The outputs of this phase include:

- A Functional Requirements Document (FRD) that outlines system features like user login, BMI calculation, and meal plan generation.
- A Non-Functional Requirements Document (NFRD) that specifies performance, security, and scalability expectations. Clear documentation minimizes misunderstandings and provides a solid foundation for development, testing, and maintenance.

4. Validation

In the final phase, the documented requirements are reviewed and validated to ensure they accurately reflect stakeholder needs and are free of errors. This involves:

- Stakeholder Review: Presenting the documented requirements to stakeholders for feedback and approval.
- Prototyping and Mockups: Creating early prototypes to validate user expectations and refine unclear requirements.
- Testing Alignment: Ensuring that the requirements align with testing criteria to verify that the system, once developed, can meet its defined goals.

Significance of the Requirements Engineering Process

The interconnected nature of these phases ensures that requirements are not only well-defined but also actionable and realistic. This structured framework reduces the risks of project delays, cost overruns, and user dissatisfaction by addressing potential issues early in the development cycle. By focusing on collaboration, clarity, and consistency, the requirements engineering process lays the groundwork for a successful and impactful calorie calculator system.

2.2.1 Requirements Elicitation

Requirements elicitation involves gathering information from stakeholders to identify their needs and expectations for the calorie calculator system. This phase uses various techniques to ensure a comprehensive understanding of user and system requirements. For the calorie calculator project, the following methods were employed:

Interviews: Direct discussions were held with potential users, such as individuals seeking calorie tracking solutions and meal planning tools, to understand their expectations. Topics included BMI calculation, personalized meal plans, and the flexibility of the system in accommodating dietary restrictions.

Questionnaires: Surveys were distributed to collect feedback on specific preferences, such as food categorization (vegetarian, non-vegetarian), allergy considerations, and the importance of a mobile-friendly interface. These questionnaires provided quantifiable insights into user expectations.

Prototyping: Early prototypes were developed to demonstrate key features such as the admin control panel and meal plan customization. These prototypes facilitated user engagement, allowing stakeholders to visualize the system and provide detailed feedback. This iterative feedback process refined the system's design and functionality.

Competitive Analysis: A thorough review of existing products, such as MyFitnessPal and Cronometer, was conducted to identify their strengths, limitations, and areas for improvement. Insights from this analysis informed the project's feature set and helped differentiate the calorie calculator system from competitors.

2.2.2 Requirements Analysis

After collecting the requirements, the next step was to analyze them for feasibility, prioritization, and conflict resolution. This phase involved the following activities:

- Categorization: Divided requirements into:
  - Functional Requirements: Features such as login functionality, BMI calculation, and meal plan generation.
  - Non-Functional Requirements**:** Performance metrics, system scalability, and user interface design.

- Conflict Resolution**:** Identified and resolved inconsistencies between requirements, ensuring that user preferences (e.g., allergy settings) did not conflict with system capabilities.
- Prioritization: Ranked requirements based on their importance and feasibility to ensure critical features were developed first.
- Feasibility Analysis: Evaluated each requirement against constraints like technology, timeline, and budget to ensure realistic implementation.

2.2.3 Requirements Specification

The specification phase documents the analyzed requirements in a detailed, structured format to serve as a reference for the development team. Outputs include:

- Functional Requirements Document (FRD): Defines the core functionalities such as:
  - User login and registration.
  - Personalized meal plans based on dietary preferences and allergies.
  - Admin functionality to manage calorie and food data.
- Non-Functional Requirements: Describes the performance, reliability, and security standards, including:
  - A responsive design for seamless user experience across devices.
  - Secure authentication mechanisms to protect user data.
- Use Cases: Examples include:
  - Use Case 1: A user logs in, enters their weight and dietary preferences, and receives a personalized 30-day meal plan.
  - Use Case 2: An admin adds or removes food items in the calorie database.
- Data Models: Detailed database schemas representing users, food items, and meal plans.

This specification acts as a blueprint for system design and development.

2.2.4 Requirements Validation

Validation ensures that the specified requirements are complete, accurate, and feasible. The following methods were employed for validation:

- Peer Reviews: Team members reviewed the requirements to identify ambiguities, missing details, or contradictions.

- Prototyping: Developed an initial prototype of key features, such as BMI calculation and meal plan generation, to gather feedback from stakeholders.
- Stakeholder Meetings: Organized sessions with stakeholders to confirm that requirements aligned with their expectations.
- Traceability Matrix**:** Mapped requirements to development deliverables to ensure all needs were addressed during the implementation phase.

Table 3. Requirements Validation

Validation Criteria	Yes	No	Comments
Is it technically feasible to meet these requirements?	☑		Most requirements were kept basic and straightforward, with some more complex but feasible ones.
Are there any inconsistencies or conflicts in the requirements?	☑0		Lots of focus and concentration were applied to create a consistent system without conflicts.
Are dependencies among requirements identified?	☑		Some dependencies were initially missed but resolved during implementation through hard work.
Are requirements documented in a formalized and uniform format?	☑		All requirements are documented in a structured, standardized format for easy understanding.

Validation Criteria	Yes	No	Comments
Do the requirements adequately reflect stakeholders' needs?	☑		Stakeholders' feedback was effectively incorporated, ensuring alignment with their expectations.
Is the language and terminology used understandable to all?	☑		Simple, clear language was used to ensure understanding by stakeholders and developers alike.
Are all requirements appropriate to the system's objectives?	☑		All requirements align with the project's objectives, ensuring a focused approach to development.

3. Software Requirements Specification

3.1 Purpose

The purpose of this document is to comprehensively define and detail the functional and non-functional requirements of the calorie calculator system. It serves as an essential guide for all stakeholders involved in the project, including developers, testers, and project managers, ensuring a shared understanding of what the system aims to accomplish. By clearly outlining the system's objectives and constraints, this document provides a foundation for a structured and efficient development process.

This specification has several key aims that contribute to the success of the project:

1. Clarify Objectives:
   This document offers a precise and unambiguous description of the system's purpose, primary functionalities, and scope. It sets clear expectations for what the system will deliver, ensuring alignment between stakeholder goals and the project deliverables. For example, users will benefit from tailored meal plans and a user-friendly interface, while administrators gain robust tools for managing data.
2. Facilitate Development:
   By serving as a reference point for the development team, this document ensures that the team adheres to user needs and project goals throughout the implementation phases. It minimizes the risk of misunderstandings and provides actionable requirements that translate directly into design and coding tasks.
3. Ensure Usability:
   The document outlines the requirements for the user interface and system behavior to deliver an intuitive and seamless experience for all users, including general users and administrators. Features such as responsive design, clear navigation, and personalized content aim to make the system accessible and effective for a diverse user base.
4. Support Validation and Testing:
   The specification includes criteria for assessing the system's performance, reliability, security, and usability during and after development. These criteria enable testers to verify that the system meets its requirements and identify areas for improvement before deployment.
5. Address Constraints:
   It identifies critical design, operational, and resource constraints, such as technology limitations, timeline restrictions, and budget considerations. These constraints ensure that the implementation remains realistic and achievable within the project's defined parameters.

The calorie calculator system's primary goal is to empower users with a personalized health management tool that supports informed dietary choices and facilitates calorie tracking. Users will benefit from features such as:

- Tailored Meal Plans: Customized 30-day meal plans based on dietary preferences, allergies, and caloric needs.

- Caloric Intake Tracking: Tools to monitor daily calorie consumption and provide actionable feedback.
- Health Insights: Calculations such as BMI to offer users a better understanding of their health status.

Additionally, the system includes robust administrative features for managing food data and calorie information, ensuring that the backend operations are efficient and reliable. These tools help administrators maintain the accuracy and relevance of meal plans, ultimately enhancing user satisfaction.

This document acts as a foundational blueprint for the system's design, development, and evaluation phases. It ensures that the delivered product aligns with its intended purpose and meets the expectations of all stakeholders. By providing clarity, direction, and structure, the software requirements specification significantly contributes to the success of the calorie calculator system.

3.1.1 Other Websites Comparison

Feature	User Registration	BMI Calculation	Admin Panel	Meal Suggestion Updates	Multilingual Support	Integration with Wearables	AI Recommendations	Data Privacy Compliance
Calorie Calculator	✅	✅	✅	✅	❌	❌	❌	✅
MyFitnessPal	✅	✅	❌	❌	✅	✅	❌	✅
Cronometer	✅	✅	❌	❌	✅	✅	❌	✅
Lose It!	✅	✅	❌	❌	❌	❌	❌	✅
Proposed Future Enhancements	✅	✅	✅	✅	✅	✅	✅	✅

:

3.2 Overall Description

3.2.1 Product Perspective

The calorie calculator system is a modern web-based application developed to empower users to monitor their dietary habits, manage calorie intake, and achieve their health and fitness goals. The system is designed with a user-centric approach, ensuring it provides an intuitive and seamless

experience for diverse user groups, ranging from casual users interested in maintaining a healthy lifestyle to administrators responsible for backend data management.

The application incorporates a variety of features that address key aspects of dietary management, including BMI calculation, 30-day personalized meal plans, and an administrative control panel. These features collectively provide users with actionable insights and tailored solutions for improving their nutritional habits. The system is designed to be both comprehensive and easy to use, ensuring that users of all technical backgrounds can navigate the interface and derive value from its functionality.

Key features of the calorie calculator include:

1. BMI Calculation:
    - This feature allows users to calculate their Body Mass Index based on their height and weight. It provides a health status indicator (e.g., underweight, healthy, overweight), enabling users to understand their current fitness level and tailor their dietary goals accordingly.
2. 30-Day Meal Plan Generation:
    - Users receive customized meal plans based on their input, such as dietary preferences (e.g., vegetarian, non-vegetarian), allergies, and calorie goals. This feature ensures that meal plans are both nutritious and aligned with user preferences, offering a practical approach to maintaining or improving health.
3. Admin Panel:
    - Administrators have access to a dedicated interface that allows them to manage the food database, update calorie values, and ensure the accuracy of meal plan recommendations. This feature ensures the backend data is reliable and up-to-date, directly contributing to the quality of user experiences.

The system is designed to integrate seamlessly with users' daily routines while maintaining simplicity and scalability. Its architecture enables it to handle increasing user demand and adapt to future enhancements, such as the integration of additional meal planning algorithms, user engagement tools, or AI-based health insights.

The calorie calculator is not a standalone tool but a system designed to fit into the broader ecosystem of health and fitness applications. It bridges gaps left by existing solutions by offering a higher level of personalization and dietary inclusivity. Unlike many applications that provide

generalized meal plans, this system adapts to user-specific needs such as allergies, cultural food preferences, and targeted calorie goals.

- Existing Systems: Solutions like MyFitnessPal and Cronometer focus on calorie tracking but lack robust meal planning functionalities tailored to individual preferences.
- Unique Value Proposition: The calorie calculator combines calorie tracking, meal customization, and health insights in a single platform, offering both users and administrators tools to manage dietary goals effectively.

3.2.2 Product Functions

The calorie calculator system is built to provide a wide range of functionalities that cater to both general users and administrators, ensuring a personalized and efficient user experience. These functions are designed to address the key requirements of dietary management, health tracking, and data administration. Below is a detailed explanation of the primary functions of the product:

1. User Registration

The user registration function enables individuals to create personalized accounts by providing essential details such as name, email address, password, age, gender, height, weight, dietary preferences, and any allergies. This step ensures that the system collects sufficient data to tailor the experience to individual needs.

- Core Features:
  - Input validation to ensure accurate data entry (e.g., correct email format, password strength).
  - Support for capturing dietary preferences, such as vegetarian, non-vegetarian, or vegan diets.
  - An intuitive and user-friendly registration form designed to minimize user effort.

2. Log-In and Authentication

The log-in functionality provides secure access to user profiles and system features. It ensures that only authorized users can access sensitive information, such as personal data and meal plans.

- Core Features:
  - User authentication through email and password verification.
  - Implementation of secure password hashing to protect user credentials.
  - Error handling for incorrect credentials, including user-friendly error messages.
  - Additional security measures, such as account lockout after multiple failed attempts and CAPTCHA integration to prevent automated attacks.

3. BMI Calculation

The BMI (Body Mass Index) calculation feature allows users to input their height and weight to determine their BMI, which provides a quick assessment of their health status (e.g., underweight, normal, overweight).

- Core Features:
  - Automated BMI calculation using the formula: BMI=Weight (kg)Height (m)2BMI = \frac{\text{Weight (kg)}}{\text{Height (m)}^2}BMI=Height (m)2Weight (kg)
  - Categorization of BMI results into predefined ranges (e.g., underweight, normal, overweight, obese) for easier interpretation.
  - Health tips and recommendations based on the calculated BMI to guide users toward their health goals.

This feature serves as an initial step in helping users understand their current health status and tailoring their dietary plans accordingly.

4. Personalized Meal Planning

One of the standout features of the calorie calculator system is its ability to generate personalized 30-day meal plans based on user preferences, allergies, and calorie requirements.

- Core Features:
  - Dynamic meal plan generation using user data, such as dietary preferences (vegetarian, vegan, non-vegetarian), calorie goals, and allergy restrictions.
  - Daily calorie breakdown to ensure users meet their nutritional goals.

- Exclusion of allergenic foods from meal plans, enhancing safety and user satisfaction.
- Flexibility to update meal plans based on changes in user preferences or goals.

5. Admin Panel

The admin panel is a robust interface designed for system administrators to manage the food database, calorie data, and user accounts. It ensures the accuracy and relevance of the data used for meal planning and system operations.

- Core Features:
  - Food Management: Add, edit, or delete food items and their associated calorie values.
  - User Management: Manage user accounts, including resetting passwords, handling inactive accounts, and resolving user-reported issues.
  - System Monitoring: View logs of user activity and track system performance for troubleshooting and optimization.

Comprehensive Impact

These core functions work together to provide users with an engaging and practical tool for managing their dietary habits. While users benefit from features like personalized meal plans and health insights, administrators can efficiently maintain and enhance the system's data and functionality. This dual focus ensures the calorie calculator system serves its purpose effectively for all stakeholders.

3.2.3 User Classes and Characteristics

The system targets the following user groups table 4:

User Class	**User Class**	**User Class**

General Users	Individuals who use the system to track calories, calculate BMI, and receive personalized meal plans based on their dietary preferences and health goals.	- Non-technical users with varying levels of health awareness.
Administrators	Users responsible for managing system content, including the database of food items, calorie values, and user accounts. They ensure that the system operates effectively and is updated regularly.	- Familiar with technical and administrative tasks.

3.2.4 Operating Environment

The calorie calculator system is designed to function seamlessly within a modern web-based environment, ensuring accessibility and compatibility for a wide range of users and devices. The operating environment encompasses both the client-side and server-side components, each playing a vital role in delivering a smooth and efficient user experience.

Client-Side Environment

The client-side represents the user-facing portion of the system, which includes the interface that users interact with directly. The system is optimized to work across multiple devices, including desktops, laptops, tablets, and smartphones. Key requirements and features for the client-side environment include:

1. Browser Compatibility**:**

- The system is compatible with modern web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. This ensures that users across various platforms can access the application without issues.
- Support for HTML5 and CSS3 ensures rich content delivery and advanced styling.

2. Device Responsiveness:
    - The front-end design employs responsive frameworks like Bootstrap, allowing the application to adjust its layout and functionality based on screen size and resolution. Users on smaller devices, such as smartphones, will experience a streamlined interface optimized for touch-based interactions.
3. Performance Requirements:
    - Lightweight client-side scripts are used to minimize page load times and enhance user experience. JavaScript (via React) ensures dynamic content updates without requiring full page reloads.
    - Efficient rendering of components, even on lower-end devices, guarantees accessibility to users with varying hardware capabilities.

4. Security Features**:**
    - Client-side validation prevents users from submitting invalid or malicious data, enhancing system reliability and security.
    - HTTPS ensures secure communication between the client and server, safeguarding sensitive data such as user credentials and dietary preferences.

Server-Side Environment

The server-side handles the back-end processes, including business logic, database management, and API interactions. It is designed to deliver consistent performance and reliability while meeting scalability requirements for growing user bases. Key requirements and features for the server-side environment include:

1. Web Server Requirements:

- The system is hosted on a robust web server capable of running Node.js for handling back-end operations. Node.js ensures fast, asynchronous handling of user requests, making it ideal for scalable web applications.
- Google Cloud Platform (GCP) is used to deploy the application, offering high availability, scalability, and global reach.

2. Database Requirements:
   - The application uses MongoDB, a NoSQL database, for managing user profiles, food items, and meal plans. MongoDB's flexibility in handling unstructured data makes it suitable for storing diverse user inputs, such as dietary preferences and allergy information.
   - The database supports efficient querying and indexing, ensuring quick retrieval of data during meal plan generation and other operations.
3. API Integration:
   - APIs developed using Express.js facilitate seamless communication between the client and server. These APIs handle tasks such as user authentication, meal plan generation, and BMI calculations.
   - Secure authentication mechanisms, such as JSON Web Tokens (JWT), ensure that only authorized users can access sensitive endpoints.
4. Scalability and Reliability:
   - The server-side infrastructure is designed to handle multiple simultaneous requests, ensuring minimal latency even during peak usage.
   - Load balancing and horizontal scaling capabilities on GCP provide flexibility to accommodate an increasing number of users without compromising performance.

3.2.5 Design and Implementation Constraints

The design and implementation of the calorie calculator system are subject to several constraints that shape its development, deployment, and performance. These constraints include technical requirements, time limitations, and compliance with regulatory standards, all of which ensure that the system operates effectively while adhering to project and legal requirements.

1. Scalability and Performance

One of the key design constraints is the system's ability to handle a large number of simultaneous users without any noticeable performance degradation. The calorie calculator system is expected to serve a diverse and growing user base, which may include individuals and administrators accessing the system concurrently. This requires careful consideration of the following:

- Load Management**:**
  - The system must efficiently manage concurrent requests to avoid bottlenecks or delays. For example, when multiple users are generating meal plans or accessing their profiles simultaneously, the back-end must process these requests without overloading the server.
- Infrastructure**:**
  - Deployment on a scalable platform, such as Google Cloud Platform (GCP), ensures that the system can dynamically allocate resources based on user demand. Features like load balancing and horizontal scaling enable the system to maintain performance even during peak usage.
- Optimized Codebase**:**
  - The use of asynchronous programming in Node.js ensures non-blocking operations, allowing the system to handle multiple tasks concurrently. This is critical for functions like database queries, API calls, and dynamic content updates.

2. Development Timeline

The project is constrained by a development timeline of six months, requiring efficient planning and execution of tasks. This constraint influences various aspects of the system's design and implementation:

- Incremental Development**:**
  - The project follows an iterative approach, with core functionalities such as user registration, BMI calculation, and basic meal plan generation prioritized in the early phases. Advanced features, such as admin panel enhancements, are scheduled for later iterations.

- Task Prioritization**:**
  - The development team must prioritize essential features to ensure a functional and deployable product within the timeframe. Lower-priority features may be deferred for future updates or enhancements.
- Team Coordination:
  - Effective communication and collaboration among team members are crucial to meet deadlines. Tools like GitHub for version control and task management systems ensure that progress is tracked and potential delays are addressed promptly.

3. Compliance with Saudi Personal Data Protection Law. Guidelines

Adherence to the Saudi Personal Data Protection Law.is a critical constraint for the system, especially as it involves collecting and managing sensitive user data, such as personal details and dietary preferences. This constraint affects both the design and implementation of the system in the following ways:

- Data Privacy:
  - The system must implement robust encryption mechanisms for data storage and transmission. User passwords, for example, are hashed using secure algorithms to prevent unauthorized access.
- Consent Management:
  - Users must provide explicit consent for data collection and processing during registration. The system must include a clear privacy policy outlining how data will be used and stored.
- Right to Data Access and Deletion:
  - The system must provide users with options to access, modify, or delete their data, ensuring compliance with Saudi Personal Data Protection Law.'s data rights provisions.
- Audit Trails:

- Logs of user interactions and administrative actions must be maintained securely for accountability and auditing purposes, without violating user privacy.

3.2.6 User Documentation

Comprehensive user documentation is an integral component of the calorie calculator system, ensuring that users and administrators can navigate and utilize the platform effectively. The documentation is designed to provide clear, step-by-step guidance for all functionalities, catering to both general users and system administrators. By offering accessible resources, the system reduces the learning curve and empowers users to make the most of its features.

1. User Manuals

The system includes detailed user manuals tailored for general users, covering every aspect of the platform's functionality. These manuals are structured to provide intuitive guidance, enabling users to interact with the system seamlessly. Key topics covered in the user manuals include:

- Registration:
  - Instructions on how to create an account, including filling in personal details such as name, email, password, height, weight, dietary preferences, and allergies.
  - Explanations of input validation rules, such as password strength requirements and valid email formats.
- Meal Planning:
  - Step-by-step instructions for entering dietary preferences and generating personalized 30-day meal plans.
  - Guidance on how to update preferences or adjust calorie goals to reflect changing health needs.
- BMI Calculation:
  - Directions for inputting height and weight to calculate BMI.
  - Explanations of the BMI categories (e.g., underweight, healthy, overweight) and tips for interpreting the results.
- Profile Management:
  - Tutorials on editing personal information, changing passwords, and managing account settings.

Each section of the user manual includes annotated screenshots, flow diagrams, and clear instructions to ensure users can follow along easily.

2. Admin Guides

For administrators, the system provides dedicated guides focused on managing backend data and ensuring the platform operates smoothly. These guides address the technical and operational aspects of the admin panel, equipping administrators with the tools to manage the system effectively. Key topics in the admin guides include:

- Data Management:
  - Instructions on adding, editing, and deleting food items and their associated calorie values in the database.
  - Guidelines for categorizing food items based on user preferences and allergies.
- User Account Management:
  - Steps for resetting user passwords, resolving account-related issues, and handling inactive accounts.
  - Directions for viewing user activity logs and ensuring compliance with system policies.
- System Monitoring and Maintenance:
  - Tutorials on monitoring server performance, troubleshooting common issues, and ensuring data accuracy.
  - Guidance on using logs to track administrative actions and maintain accountability.

The admin guides emphasize efficiency and accountability, helping administrators maintain the platform's reliability and security.

3. FAQs and Troubleshooting Tips

To address common user questions and technical issues, the system includes an extensive FAQ section and troubleshooting guide. These resources are designed to empower users and administrators to resolve minor issues independently, reducing the need for external support.

- FAQs:
    - Answers to frequently asked questions, such as:
        - "How do I recover my password if I forget it?"
        - "Can I update my dietary preferences after registration?"
        - "What should I do if I encounter an error while generating a meal plan?"
    - Categorized FAQs for easy navigation, covering registration, meal planning, BMI calculation, and admin tasks.
- Troubleshooting Tips:
    - Solutions to common issues, such as:
        - Steps to resolve connectivity problems or server downtime.
        - Instructions for clearing browser cache if the system interface does not display correctly.
        - Guidance on checking input errors if meal plans fail to generate.

3.2.7 Assumptions and Dependencies

The successful operation and effectiveness of the calorie calculator system are based on certain assumptions and dependencies that influence its design, implementation, and usability. These factors must be considered to ensure the system performs as expected and delivers value to its users. Below is a detailed explanation of the assumptions and dependencies that underpin the system.

1. Assumptions

The system's functionality and performance are based on several key assumptions that must hold true for it to operate as intended:

- Users Must Have Internet Access:
  The calorie calculator system is a web-based application that requires a stable internet connection for users to access its features. This assumption means:
    - Users need to be connected to the internet to register, log in, and interact with the system.
    - Data, such as meal plans and BMI results, is retrieved and processed on the server, necessitating real-time communication between the client and server.

- Accurate Data Inputs from Users:
  The system relies on users providing accurate and truthful information when entering their data. For example:
  - Correct height and weight values are critical for calculating an accurate BMI.
  - Accurate dietary preferences and allergy information are essential for generating safe and relevant meal plans.
  - Errors or inconsistencies in user-provided data may lead to incorrect outputs, such as miscalculated calorie recommendations or unsuitable meal plans.
- Users Follow Provided Guidelines:
  It is assumed that users will adhere to the instructions and guidelines provided in the user documentation. This includes correctly entering information during registration and updating preferences as needed.

2. Dependencies

The system is dependent on several external and internal factors that enable its smooth operation. These dependencies include:

- Server Reliability:
  The system relies on the stability and performance of the hosting server to handle requests, process data, and deliver outputs efficiently. Key considerations include:
  - The server must handle concurrent user requests without crashes or delays.
  - Hosting on a scalable platform, such as Google Cloud Platform (GCP), ensures that the system can adapt to fluctuations in user demand.
- Third-Party Tools for Database Management:
  The application uses MongoDB as its database management system, which plays a critical role in storing and retrieving user data, food items, and meal plans. Dependencies include:
  - The reliability of MongoDB for managing large volumes of data and maintaining data integrity.
  - The efficiency of Mongoose, an object modeling library for MongoDB, to facilitate database interactions within the application.

- Front-End and Back-End Integration:
  The system depends on seamless communication between the front-end (built with React) and back-end (built with Node.js and Express.js). This integration ensures:
  - Accurate data transfer between the user interface and the server.
  - Consistent performance and functionality across different devices and browsers.
- Security Mechanisms:
  Dependencies include third-party libraries and frameworks for implementing encryption, authentication, and authorization. For example:
  - Secure handling of user credentials through password hashing algorithms.
  - Protection of sensitive data during transmission using HTTPS protocols.

3. Risk Mitigation for Assumptions and Dependencies

To address potential risks associated with these assumptions and dependencies, the following measures are in place:

- Internet Access:
  While internet access is a requirement, the system includes user-friendly error messages and retry mechanisms to guide users in case of connectivity issues.
- Data Validation:
  Input validation mechanisms ensure that users provide data in the correct format. For example:
  - Height and weight fields accept only numerical inputs within a realistic range.
  - Dietary preferences and allergy information are collected through dropdowns or checkboxes to minimize errors.
- Server and Database Reliability:
  The system leverages cloud hosting (GCP) with load-balancing capabilities and backup systems to minimize downtime. Regular monitoring ensures the server remains operational and responsive.
- Security Measures:
  Robust security practices, such as regular updates to encryption libraries and adherence to Saudi Personal Data Protection Law. guidelines, mitigate risks associated with data breaches or unauthorized access.

3.3 External Interface Requirements

3.3.1 User Interfaces

The calorie calculator system is designed with a user-centric approach, featuring intuitive and visually appealing user interfaces that cater to both general users and administrators. These interfaces ensure seamless navigation and provide easy access to the system's functionalities. Below is a detailed description of the key user interfaces:

1. Registration Page

The Registration Page serves as the entry point for new users, enabling them to create accounts and customize their profiles. This page is designed to be user-friendly, guiding users through the process of entering their information while ensuring data accuracy.

- Features:
  - A straightforward form layout for entering personal details such as name, email, and password.
  - Fields for inputting dietary preferences (e.g., vegetarian, vegan, non-vegetarian) and allergy information.
  - Password strength validation to ensure secure account creation.
  - A checkbox for users to agree to terms and conditions, ensuring compliance with legal and privacy requirements.
- Design Elements:
  - Clear labels and placeholders for form fields to enhance usability.
  - Validation messages that provide real-time feedback on errors, such as invalid email formats or weak passwords.
  - A visually prominent "Sign Up" button that initiates the registration process.
- Purpose:

  The registration page collects essential user data to personalize their experience, laying the foundation for tailored meal plans and health tracking.

2. Dashboard

The Dashboard is the primary interface for general users, providing a centralized hub for accessing their personalized information and system features. It is designed to be both informative and interactive, ensuring users can easily navigate and understand their health insights.

- Features:
  - BMI Results: Displays the user's BMI based on their height and weight, along with a health category (e.g., underweight, healthy, overweight).
  - Meal Plans: Provides a 30-day personalized meal plan, broken down by daily calorie recommendations and meal options.
  - User Profile Section: Allows users to view and update their personal details, such as weight, dietary preferences, and calorie goals.
  - Health Tips: Offers actionable advice based on the user's BMI and dietary data to encourage healthier habits.
- Design Elements:
  - A visually engaging layout with charts and graphs to present data like calorie breakdowns and BMI trends.
  - Tabs or cards for easy access to different sections, such as meal plans, health insights, and profile settings.
  - Dynamic elements like collapsible menus or dropdowns for compact navigation on smaller devices.
- Purpose:
  The dashboard provides users with a personalized and interactive experience, helping them monitor their health and achieve their dietary goals.

3. Admin Panel

The Admin Panel is an advanced interface tailored for system administrators. It provides tools for managing the backend data and ensuring the system's smooth operation. The panel is designed with functionality and efficiency in mind, enabling administrators to oversee and update system components effectively.

- Features:
  - User Management: Tools for managing user accounts, such as resetting passwords, deactivating inactive accounts, and resolving user queries.

  - Food Database Management: Options to add, edit, or delete food items, along with their associated calorie values and dietary classifications.
  - System Logs: A log view that tracks administrative actions and user activities, providing transparency and accountability.
  - Reports and Analytics: Insights into user trends, such as popular dietary preferences or frequently used features.
- Design Elements:
  - A dashboard layout with categorized sections for quick navigation (e.g., User Management, Food Database, Logs).
  - Filters and search functionality for efficiently locating specific user accounts or food items.
  - Alerts and notifications for system updates or flagged user activities that require attention.
- Purpose:
  The admin panel empowers administrators to maintain the accuracy and relevance of the system's data, ensuring users receive reliable and up-to-date recommendations.

3.3.2 Hardware Interfaces

The calorie calculator system requires specific hardware interfaces to ensure smooth operation and compatibility between the client-side and server-side environments. These hardware requirements are designed to accommodate the performance and scalability needs of the system while ensuring accessibility for a wide range of users.

Client-Side Hardware Requirements

The client-side hardware refers to the devices used by general users and administrators to access the system. The minimum hardware specifications are as follows:

- Device Compatibility:
  - The system can run on desktops, laptops, tablets, and smartphones.
  - Devices must be capable of running a modern web browser, such as Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.
- Performance Specifications:
  - At least 2 GB of RAM to support smooth browsing and interactive web elements.

  - A processor equivalent to Intel i3 or higher to handle dynamic content rendering.
  - A stable internet connection (at least 5 Mbps) to support real-time interactions with the server.
- Purpose:

  The client-side hardware requirements ensure that users experience a responsive and functional interface across various devices without performance issues.

Server-Side Hardware Requirements

The server-side hardware forms the backbone of the calorie calculator system, hosting its database, API, and application logic. Key server-side specifications include:

- Processing Power:
  - A multi-core processor (e.g., Intel Xeon or equivalent) to handle simultaneous user requests, complex calculations (e.g., BMI, meal plans), and API responses.
- Memory and Storage:
  - At least 8 GB of RAM to manage data processing and caching efficiently.
  - SSD storage (minimum 50 GB) to store user data, food databases, and logs while ensuring high-speed read/write operations.
- Network Connectivity:
  - High-speed network connections to maintain low latency during data transmission between the client and server.
  - Support for scalable infrastructure to accommodate future growth in user base and data volume.
- Purpose:

  The server-side hardware ensures the system remains reliable, scalable, and capable of handling concurrent user activities without degradation in performance.

3.3.3 Software Interfaces

The calorie calculator system relies on various software interfaces to manage data, enable real-time interactions, and ensure secure communication between the client-side and server-side components. These interfaces form an essential part of the system's architecture.

Integration with Database Management

The system uses a robust database for managing user profiles, food items, and meal plans. The software interfaces ensure seamless interaction with the database:

- **Database**:
  - Initially, MySQL was identified as the database for structured data management. However, the system has transitioned to MongoDB to handle diverse and dynamic data sets more effectively.
  - The interface allows CRUD (Create, Read, Update, Delete) operations for user and food data, ensuring up-to-date and accurate records.
- **Object Modeling**:
  - Mongoose is used as the primary interface for managing MongoDB collections, providing a schema-based solution for application data.
- **Purpose**:
  The database interface ensures efficient data storage and retrieval while maintaining data integrity and consistency.

APIs for Real-Time Data Retrieval

Application Programming Interfaces (APIs) form the bridge between the front-end and back-end, enabling dynamic and interactive functionalities:

- **Custom APIs**:
  - APIs developed using Node.js and Express.js handle user authentication, BMI calculations, and meal plan generation.
  - Real-time updates, such as calorie values or food suggestions, are fetched using these APIs, ensuring users receive accurate and timely recommendations.
- **Third-Party APIs**:
  - The system can integrate third-party APIs for extended functionalities, such as accessing external food databases or nutritional data for enhanced meal planning.
- **Security Features**:
  - APIs are secured using JSON Web Tokens (JWT) for authentication and HTTPS for encrypted communication.
- **Purpose**:
  APIs ensure seamless communication between different system components, enabling real-time interactions and scalable integrations.

3.4 System Features

3.4.1 User Registration

- Description and Priority: High priority feature for creating user accounts.
- Stimulus/Response Sequences:
  - User inputs personal data and submits the form.
  - System validates data and creates an account.
- Functional Requirements:
  - User must provide mandatory fields like email, password, and dietary preferences.
  - System validates password strength and email format.

3.4.2 Log-In

- Description and Priority: Critical feature to ensure secure access to profiles.
- Stimulus/Response Sequences:
  - User enters email and password.
  - System verifies credentials and grants access.
- Functional Requirements:
  - System must lock accounts after multiple failed login attempts.

3.4.3 User Profiles

- Description and Priority: Medium priority feature for managing personal data and viewing meal plans.
- Functional Requirements:
  - Users can update personal details and preferences.
  - System displays calculated BMI and meal plans.

3.4.5 Discover

- Description and Priority: Optional feature for exploring additional resources or recipes.
- Stimulus/Response Sequences:
  - User clicks "Discover".
  - System fetches and displays resources.
- Functional Requirements:

- Connects to external APIs for retrieving recipes or health articles.

3.5 Nonfunctional Requirements

3.5.1 Performance Requirements

- The system should handle at least 1,000 concurrent users with minimal latency.

3.5.2 Reliability Requirements

- System uptime must be at least 99.9%.

3.5.3 Security Requirements

- All user data must be encrypted during transmission and storage.

3.5.4 Availability Requirements

- The system must be accessible 24/7, except during scheduled maintenance.

3.5.5 Usability Requirements

- The interface should be intuitive and user-friendly for non-technical users.

3.5.6 Maintainability Requirements

- The system should allow for easy updates and scalability.

3.6 Business Rules

1. User accounts must be unique and linked to a valid email address.
2. Admins are the only users allowed to modify food and calorie data.
3. Meal plans must account for user allergies and dietary preferences.
4. Sensitive user data (e.g., weight, dietary preferences) must remain confidential.

4. Software Requirements Prioritization

4.1 Purpose

The purpose of this section is to establish a clear prioritization of the functional and non-functional requirements for the calorie calculator system. Given the finite resources, time constraints, and scope of the project, prioritization is critical to ensuring the most important and impactful features are developed and implemented first. By aligning with the project's goals and stakeholders' expectations, this approach ensures that the system delivers maximum value within the defined constraints.

Prioritizing requirements helps the development team address key aspects of the system early, such as ensuring core functionalities like user registration, BMI calculations, and meal plan generation are operational before secondary features like advanced analytics or extended admin tools are developed. This structured approach allows the team to balance stakeholder expectations, technical feasibility, and available resources effectively.

Key benefits of prioritization include:

1. Focus on Critical Features:
   By identifying and addressing essential requirements first, the team ensures that the core functionalities of the system, such as user registration, login, and meal plan generation, are operational early in the development cycle. This reduces the risk of project failure and ensures that the system provides value to users even in its initial stages.
2. Alignment with Project Goals:
   Prioritization ensures that the development process aligns with the project's primary objectives, such as delivering a personalized health management tool. It prevents resource allocation to lower-priority features that may not contribute directly to the system's success.
3. Effective Resource Utilization:
   By focusing on high-priority requirements, the team can allocate time, manpower, and financial resources efficiently. This minimizes the risk of overspending or delays caused by diverting attention to less critical tasks.
4. Improved Stakeholder Satisfaction:
   Addressing stakeholder-defined priorities ensures that the delivered system meets their

expectations. For example, users are more likely to value accurate BMI calculations and tailored meal plans than additional features that do not directly enhance their experience.

5. Flexibility for Future Enhancements:
   A well-prioritized development roadmap allows the team to focus on core functionalities first, leaving room for the addition of lower-priority features, such as AI-driven recommendations or social sharing options, in future iterations.

This systematic prioritization process also ensures that non-functional requirements, such as performance, scalability, and security, are addressed in parallel with functional requirements to deliver a reliable and user-friendly system.

Prioritization Methodology

The project employs the MoSCoW prioritization technique, which categorizes requirements into four groups:

- Must-Have: Critical features without which the system cannot function.
- Should-Have: Important features that significantly enhance the system but are not mandatory for initial functionality.
- Could-Have: Desirable features that improve user experience but can be deferred if necessary.
- Won't-Have: Features that are not feasible within the current scope but may be considered for future updates.

4.2 Prioritization Model Definition

The calorie calculator system employs the MoSCoW Model for requirements prioritization, a widely recognized and effective technique for classifying project requirements based on their importance and impact. This model enables the development team to focus on delivering essential functionalities first, ensuring the system achieves its primary objectives within the constraints of time, budget, and resources. The structured approach provided by the MoSCoW Model enhances decision-making and resource allocation, ensuring that the project stays aligned with its goals and stakeholder expectations.

MoSCoW Categories

The MoSCoW Model classifies requirements into four distinct categories, each reflecting a different level of priority:

1. Must-Have

These are critical requirements that are essential for the system to function effectively. Without these features, the system would fail to meet its core objectives and would not be viable for deployment. Must-Have requirements are non-negotiable and take precedence in the development process.

- Examples:
  - User Registration and Login: Enabling users to create accounts and securely access the system.
  - BMI Calculation: Providing users with a health assessment based on their height and weight.
  - Personalized Meal Plan Generation: Delivering tailored 30-day meal plans based on user preferences, calorie goals, and allergies.
- Significance:
  These requirements form the foundation of the system and ensure its basic functionality, making them indispensable for the project's success.

2. Should-Have

These requirements significantly enhance the system and add value to the user experience, but they are not critical for the initial operation. While their absence would reduce the system's effectiveness, the core functionalities would still remain intact. Should-Have features are typically implemented after Must-Have requirements.

- Examples:
  - Admin Panel: Allowing administrators to manage food data, user accounts, and system logs.
  - Responsive Design: Ensuring the system adapts seamlessly to various devices and screen sizes.

- Dynamic Meal Updates: Allowing users to adjust their preferences and receive updated meal plans in real time.

- Significance:
  These features elevate the system's usability and efficiency, contributing to a more comprehensive and satisfying user experience.

3. Could-Have

These are desirable requirements that improve the overall user experience but can be deferred if necessary. While Could-Have features may enhance the system, their absence would not significantly impact the system's functionality or user satisfaction in the short term.

- Examples:
  - Health Progress Tracking: Visualizing changes in BMI and caloric intake over time.
  - Integration with Wearable Devices: Syncing with fitness trackers to enhance calorie tracking accuracy.
  - Social Sharing Features: Allowing users to share meal plans or progress with friends on social media.
- Significance:
  These features are often included in later iterations or updates, ensuring that development resources are focused on more critical tasks initially.

4. Won't-Have

These requirements are identified as low priority and are excluded from the current project scope. While they may be considered for future updates, they are not feasible within the existing constraints of time, budget, and resources.

- Examples:
  - Multilingual Support: Offering the application in multiple languages.
  - Advanced AI Meal Recommendations: Using machine learning to predict user preferences and recommend meals dynamically.
  - Offline Functionality: Allowing users to access limited features without an internet connection.
- Significance:
  Excluding these features allows the development team to concentrate on higher-priority requirements and deliver a functional product within the project's timeline.

Purpose and Benefits of the MoSCoW Model

The MoSCoW Model helps streamline the development process by establishing a clear hierarchy of requirements. This prioritization ensures that essential features (Must-Have) are addressed first, while secondary and tertiary features (Should-Have and Could-Have) are tackled in subsequent phases, and less critical features (Won't-Have) are deferred entirely.

Key Advantages of Using the MoSCoW Model:

1. Efficient Resource Allocation:
   Development resources, including time, manpower, and budget, are focused on the most impactful requirements, ensuring maximum return on investment.
2. Risk Mitigation:
   By addressing critical requirements first, the project minimizes the risk of delays or scope creep affecting essential functionalities.
3. Flexibility:
   The model accommodates changes in project priorities or stakeholder needs, allowing lower-priority features to be added or deferred as needed.
4. Stakeholder Satisfaction:
   Prioritizing features based on stakeholder input ensures that the system meets their expectations and delivers the most value.
5. Scalable Development:
   The model facilitates iterative development, enabling the team to release a functional product early and enhance it with additional features over time.

4.3 Rationale Behind the Model

The MoSCoW model was chosen because it provides:

Clarity: Offers a clear framework for distinguishing between essential and non-essential requirements.

Flexibility: Allows for iterative development, accommodating changes in requirements during the project lifecycle.

Stakeholder Focus: Ensures that stakeholder priorities are addressed by focusing on must-have and should-have features.

Table 5. Functional Requirements

Requirement ID	Description	Priority	Rationale
FR-1	User Registration	Must-Have	Enables user account creation, essential for personalized functionality.
FR-2	Log-In	Must-Have	Provides secure access to user profiles and system features.
FR-3	BMI Calculation	Must-Have	Core functionality for health management and meal plan customization.
FR-4	Personalized Meal Plans	Must-Have	Key feature that provides user-specific dietary recommendations.
FR-5	Admin Food Management	Should-Have	Allows administrators to manage food data for accurate calorie tracking.

Requirement ID	Description	Priority	Rationale
FR-6	Dynamic Content Updates	Should-Have	Improves user experience by ensuring seamless navigation without page reloads.

Table 6. Non-Functional Requirements

Requirement ID	Description	Priority	Rationale
NFR-1	System Performance	Must-Have	The system must handle at least 1,000 concurrent users without performance issues.
NFR-2	Security	Must-Have	Protects sensitive user data with encryption during storage and transmission.
NFR-3	Usability	Should-Have	Ensures the system is intuitive for non-technical users.
NFR-4	Availability	Should-Have	System must remain operational 99.9% of the time except during maintenance.

Requirement ID	Description	Priority	Rationale
NFR-5	Maintainability	Could-Have	Simplifies future updates and enhancements.

4.5 Prioritization Results

Requirement Type: Specifies whether the requirement is functional (directly related to system features) or non-functional (related to system quality attributes like performance, security, etc.).

Requirement ID: A unique identifier for the requirement.

Priority Level: Categorized as Must-Have, Should-Have, or Could-Have, based on its importance.

Implementation Phase: Indicates when the requirement will be implemented:

- Phase 1: Initial development phase.
- Phase 2: Secondary implementation phase.
- Future Updates: Enhancements planned after the core system is deployed.

Table 7. Requirements Prioritization

Requirement Type	Requirement ID	Priority Level	Implementation Phase
Functional	FR-1	Must-Have	Phase 1
Functional	FR-2	Must-Have	Phase 1

Requirement Type	Requirement ID	Priority Level	Implementation Phase
Functional	FR-3	Must-Have	Phase 1
Functional	FR-4	Must-Have	Phase 2
Functional	FR-5	Should-Have	Phase 2
Functional	FR-6	Should-Have	Phase 2
Non-Functional	NFR-1	Must-Have	Phase 1
Non-Functional	NFR-2	Must-Have	Phase 1
Non-Functional	NFR-3	Should-Have	Phase 2
Non-Functional	NFR-4	Should-Have	Phase 2
Non-Functional	NFR-5	Could-Have	Future Updates

5. Software Design

5.1 Purpose

The purpose of this section is to provide a comprehensive framework for the architecture and design of the calorie calculator system. By employing structured models and diagrams, this section defines the organization, components, and interactions within the system, ensuring that it aligns seamlessly with both functional and non-functional requirements. A well-documented design serves as a blueprint for development, testing, and maintenance, reducing ambiguities and streamlining the implementation process.

Key Objectives of the Software Design

1. System Architecture Definition:
   This section aims to establish a clear architectural structure, detailing how the different components of the system interact and function together. It outlines the roles and responsibilities of each component, including the front-end interface, back-end server logic, and database.
2. Component Interaction and Integration:
   The design focuses on how the front-end, back-end, and database layers communicate, ensuring seamless data flow and real-time responsiveness. It specifies the integration points, such as APIs for user authentication, BMI calculation, and personalized meal plan generation.
3. Alignment with Requirements:
   By mapping functional and non-functional requirements to specific components and interactions, the design ensures that every aspect of the system supports user needs and project goals. For example, the need for secure login and data encryption directly influences the back-end authentication mechanisms.
4. Support for Scalability and Flexibility:
   The design is structured to accommodate future enhancements and an increasing user base. Modular components allow for easy updates, such as adding new meal planning algorithms or integrating third-party APIs.

5. Facilitate Development and Maintenance:
   A clear and detailed design reduces development risks by minimizing misunderstandings and guiding developers through the implementation process. It also simplifies debugging and future maintenance efforts by providing a documented structure for reference.
6. Visualization Through Diagrams:
   The section incorporates diagrams such as use case diagrams, component diagrams, and sequence diagrams to provide a visual representation of the system. These diagrams enhance understanding by illustrating the flow of data, interactions between components, and user scenarios.

Importance of a Strong Design Foundation

A robust software design is crucial to the success of the calorie calculator system. It ensures that the system is built on a foundation that is:

- Reliable: The architecture must support consistent functionality and high availability.
- Efficient: The design must optimize resource usage, including processing power and database queries.
- Scalable: As the user base grows, the system should handle increased demands without degradation in performance.
- Secure: The design must incorporate mechanisms to protect user data and ensure compliance with Saudi Personal Data Protection Law. and other regulatory requirements.
- User-Friendly: A well-structured design supports the creation of intuitive interfaces, ensuring a positive user experience.

In summary, the software design section establishes a detailed plan for developing a calorie calculator system that meets user expectations, adheres to technical requirements, and provides a scalable and maintainable solution. By focusing on architecture, component interactions, and alignment with requirements, this section ensures the project's success through clarity, structure, and foresight.

5.2 Software Architecture

5.2.1 Architectural Pattern

5.2.1.1 Pattern Identification

The system adopts the Model-View-Controller (MVC) architectural pattern. This pattern divides the application into three interconnected components, ensuring a clear separation of responsibilities and facilitating maintainability, scalability, and reusability.

1. Model:
    - Responsibilities: The model represents the application's core data and business logic. It handles tasks such as data storage, data validation, and manipulation of the application's state.
    - Examples in the Calorie Calculator System:
        - Storing user profiles, dietary preferences, and food databases.
        - Performing calculations such as BMI or calorie counts for meal plans.
        - Ensuring data integrity and consistency.
2. View:
    - Responsibilities: The view defines the user interface (UI) and is responsible for presenting data to the user in an accessible and visually appealing format.
    - Examples in the Calorie Calculator System:
        - Displaying BMI results and 30-day meal plans to the user.
        - Providing input forms for user registration and meal preference updates.
        - Creating an admin dashboard for food database management.
3. Controller:
    - Responsibilities: The controller acts as an intermediary between the model and view. It processes user input, updates the model, and determines which view should be displayed.
    - Examples in the Calorie Calculator System:
        - Handling user inputs from the registration form and passing data to the model for storage.
        - Validating login credentials and retrieving user-specific data from the model.
        - Managing the flow between the user's interaction and the displayed content.

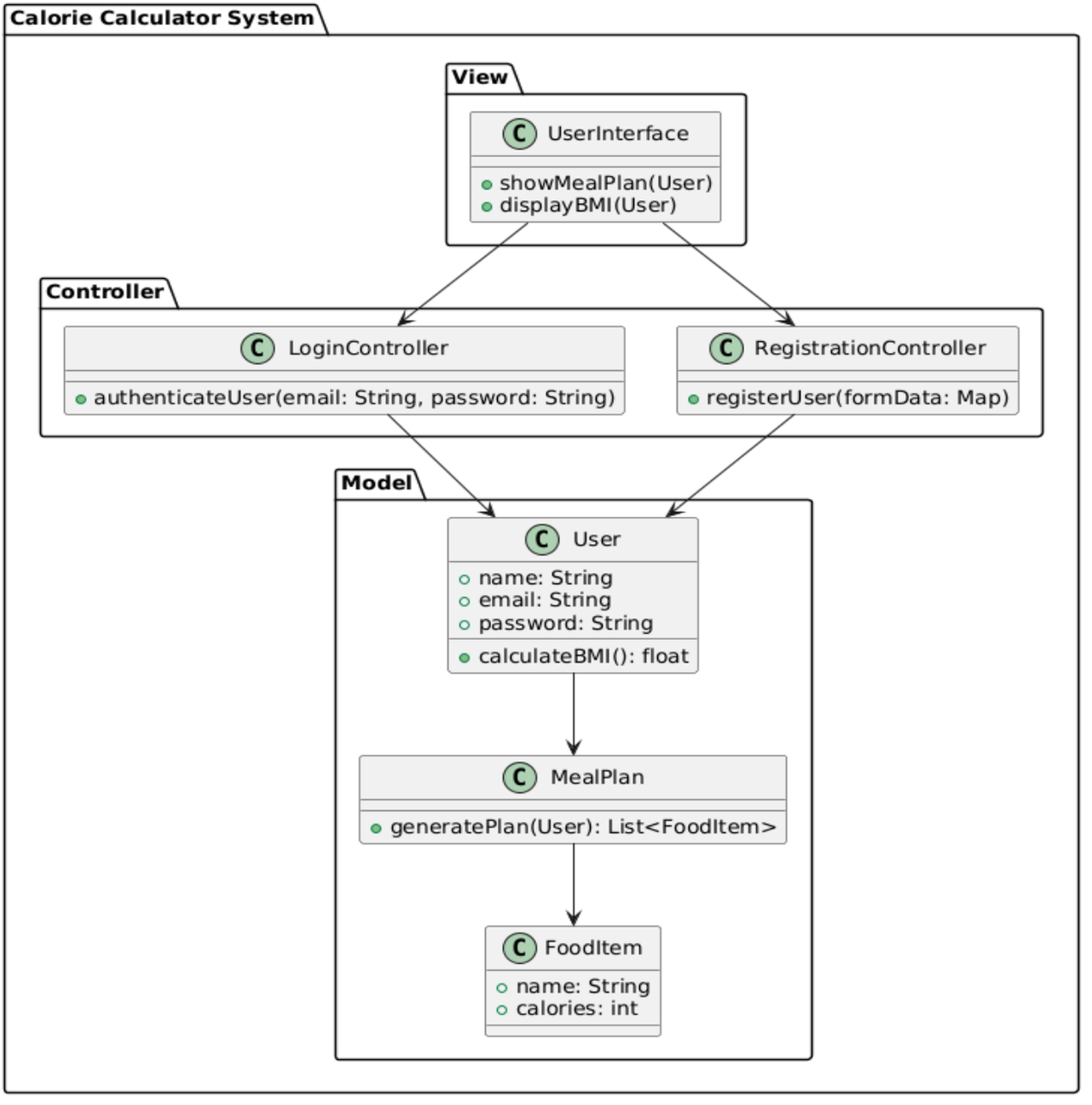


Figure 3 Architectural pattern

5.2.1.2 Pattern Justification

The MVC pattern is chosen for the calorie calculator system due to the following benefits:

1. Separation of Concerns:
    - By dividing the system into three distinct layers (Model, View, and Controller), each component can be developed and maintained independently. For example:

- Changes in the user interface (View) do not affect the core business logic (Model).
- Updates to business rules (Model) do not require modifications in the UI.

2. Scalability:
    - The MVC pattern supports future expansion, enabling developers to add new features or modules without disrupting existing functionality. For instance:
        - Adding a new feature like a "Discover Recipes" page would only involve updating the View and Controller components, while the Model remains unaffected.
3. User Experience:
    - The clear separation allows developers to focus on creating intuitive and responsive user interfaces without compromising application performance. For example:
        - Using AJAX for dynamic content updates ensures seamless interaction without page reloads.
4. Testability:
    - MVC enhances testability by isolating the application's components, making it easier to perform unit and integration testing on individual modules.

5.2.2 Architectural Design: The 4+1 View Model

Logical View: Defines the system's high-level structure, focusing on functionality.

Development View: Focuses on the organization of software modules.

Process View: Illustrates dynamic interactions between components.

Physical View: Represents deployment across hardware nodes.

User View: Captures user interaction scenarios (use cases).

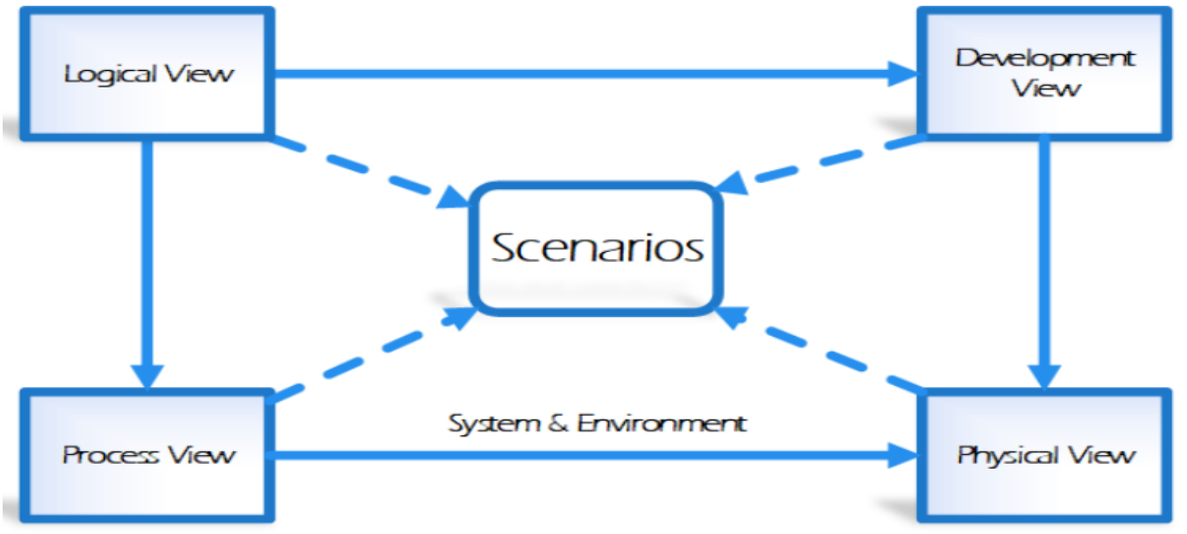


Figure 4 4+1 view model

5.3 Design Levels

5.3.1 Design Level 1 - Software System (Use Case Diagram)

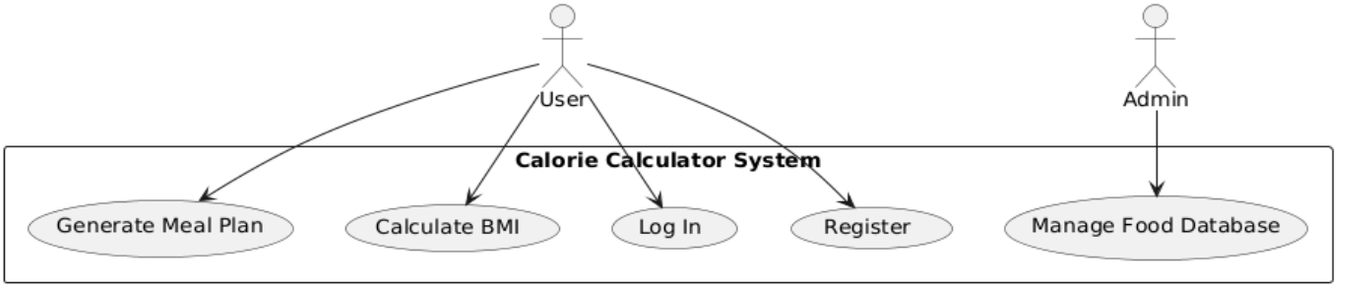


Figure 5 Use case diagram

5.3.2 Design Level 2 - Subsystems (Component Diagram)

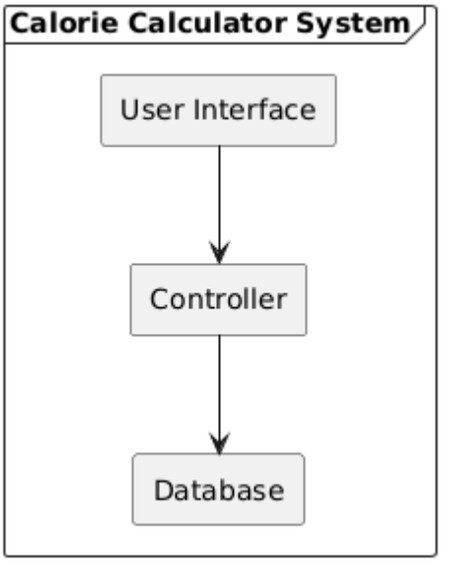


Figure 6 Component Diagram

5.3.3 Design Level 3 - Classes (Class Diagram)

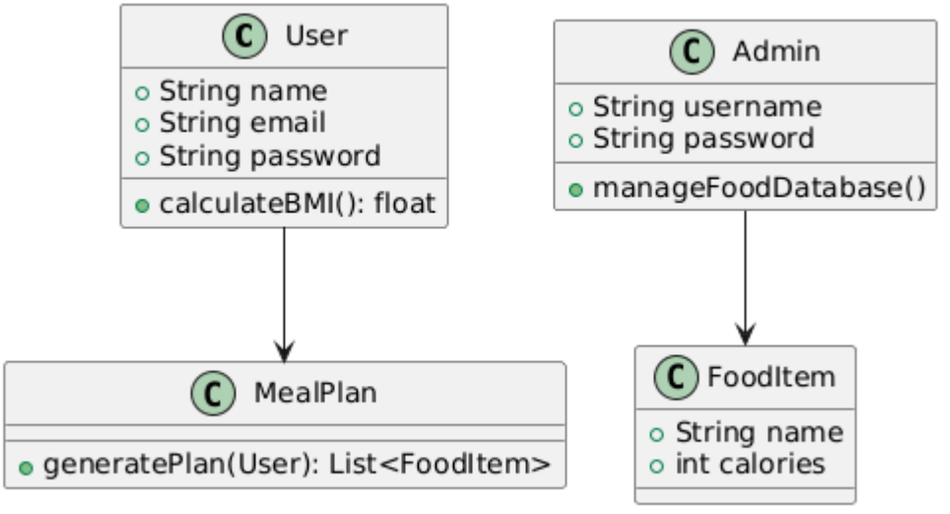


Figure 7 Class Diagram

5.3.4 Design Level 4 - Internal Route Design (Use Case Scenario)

Table 8. Use Case Scenario 1

Use Case ID	UC-1
Use Case Name	User Registration
Actors	User
Preconditions	User has access to the registration page.
Main Flow	1. User inputs details (name, email, etc.). 2. System validates inputs. 3. User account is created.
Postconditions	A new user account is stored in the database.

3.4 Activity Diagram

Elements in the Flowchart:

1. Start Node (Black Circle):
    - This represents the beginning of the process.
2. Action Nodes (Rounded Rectangles):
    - Each action or task in the registration process is represented as a step in the flowchart.
3. Decision Node (Diamond):
    - A decision point where the system evaluates the validity of user details. The outcome (Yes/No) determines the subsequent actions.
4. End Node (Black Circle with Border):
    - This marks the end of the registration process.

Steps in the Registration Flow:

1. Open Registration Page:

- The process begins when the user accesses the registration page.

2. Enter User Details:
    - The user inputs their personal details, such as name, email, and password.
3. Validation of Details:
    - The system checks if the provided details are valid:
        - Yes (Valid): If the details are valid, the system proceeds to create an account.
        - No (Invalid): If the details are invalid, the system displays an error message and returns the user to the input step.
4. Create Account:
    - If the details are valid, the system creates a new account for the user.
5. Display Success Message:
    - After successfully creating the account, the system displays a success message to the user.
6. End Process:
    - The process concludes after either successful registration or an error message.

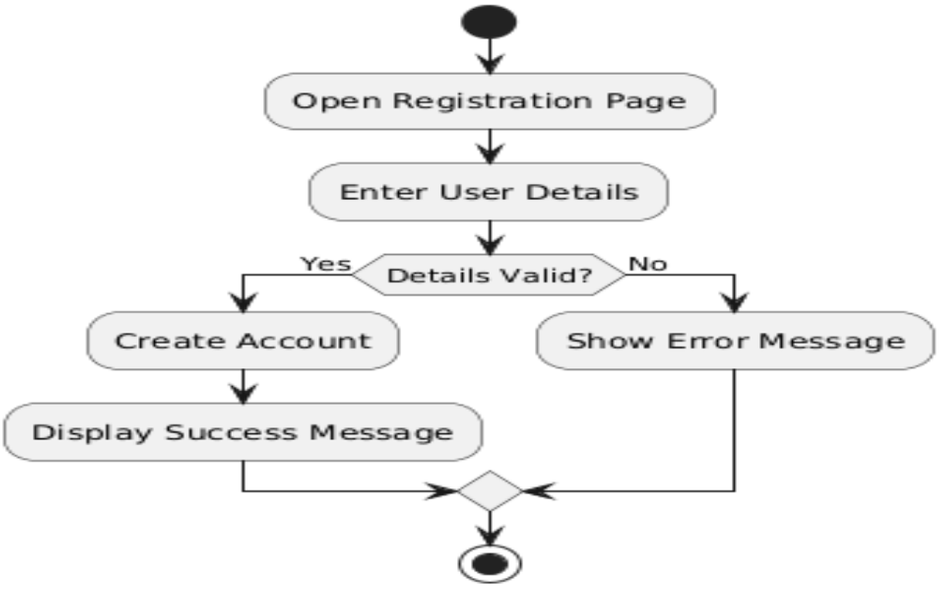


Figure 8 Activity Diagram

3.5 Sequence Diagram

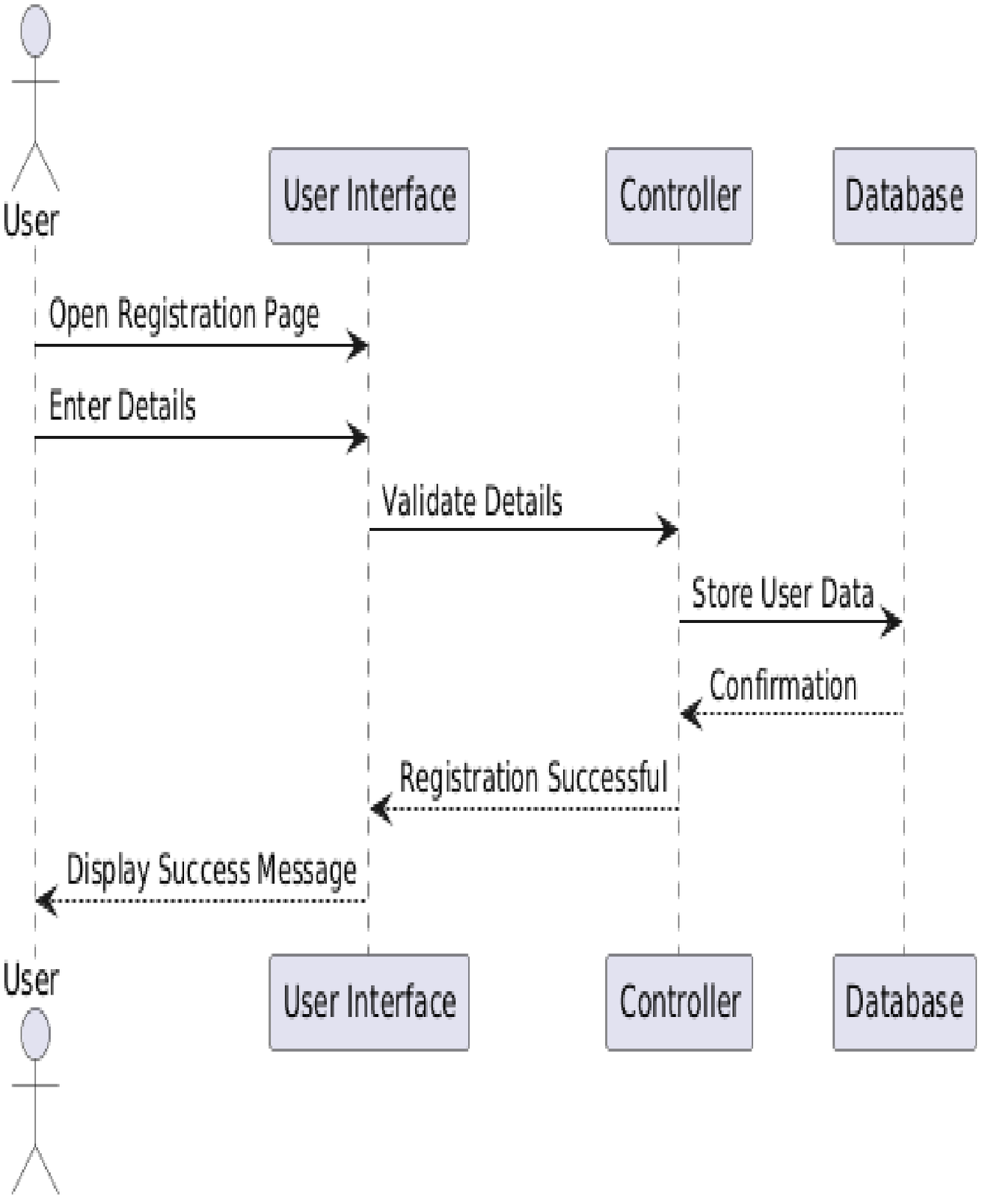


Figure 9 Sequence Dia

This is a Sequence Diagram that illustrates the process of user registration in a system. It represents the interaction between the User, User Interface, Controller, and Database to successfully complete the registration process. Here's an explanation of each step:

Actors and Components:

1. User: Represents the individual interacting with the system to register.
2. User Interface: The front-end interface that the user interacts with (e.g., web page or app screen).
3. Controller: The back-end logic that processes user requests and communicates with the database.
4. Database: The storage system that holds user data.

Steps in the Sequence:

1. User Opens Registration Page:
    - The User initiates the process by accessing the registration page through the User Interface.
2. User Enters Details:
    - The User inputs their registration details (e.g., name, email, password) via the User Interface.
3. Validation of Details:
    - The User Interface sends the input data to the Controller to validate the details. This step includes checking for required fields, email format, password strength, etc.
4. Storing User Data in Database:
    - Upon successful validation, the Controller sends the user details to the Database, which stores the information securely.
5. Database Sends Confirmation:
    - The Database confirms to the Controller that the user data has been successfully stored.
6. Registration Success Message:
    - The Controller sends a success status back to the User Interface.
    - The User Interface displays a "Registration Successful" message to the User.

Purpose:

This diagram visually represents how different components of the system collaborate to complete the user registration process. It is useful for:

- Understanding system workflows.
- Identifying areas for optimization or potential errors.
- Aligning developers and stakeholders on how the registration feature is implemented.

3.5 Business Process Management System (BPMS)

1. Introduction

Business Process Management System (BPMS) is a comprehensive platform designed to help organizations optimize, automate, and manage their business processes effectively. The primary goal of BPMS is to improve operational efficiency, ensure compliance with regulatory standards, and enhance the overall agility of the organization. By providing a centralized platform to manage workflows, BPMS bridges the gap between organizational goals and execution strategies, enabling seamless collaboration, transparency, and accountability.

2. Purpose

The purpose of this BPMS is to streamline organizational processes by providing tools for designing, automating, executing, monitoring, and optimizing workflows. The system enables organizations to:

- Improve process efficiency by identifying bottlenecks and redundancies.
- Automate repetitive tasks to save time and reduce errors.
- Enhance collaboration and communication across departments.
- Monitor process performance in real-time to ensure continuous improvement.
- Ensure compliance with regulatory and organizational standards.

3. Key Objectives

1. Process Optimization

   Identify inefficiencies in workflows and implement optimized processes to increase productivity and reduce costs.

2. Automation

   Automate repetitive and manual tasks, such as approvals, notifications, and document generation, to improve accuracy and reduce delays.

3. Collaboration and Transparency

   Facilitate communication and data sharing across teams, ensuring visibility into workflows and responsibilities.

4. Performance Monitoring

   Use analytics and reporting tools to measure process performance, enabling data-driven decisions for continuous improvement.

5. Scalability and Flexibility

   Design workflows that are scalable to meet growing organizational needs and flexible enough to adapt to changing business requirements.

6. Regulatory Compliance

   Ensure processes comply with industry-specific regulations and standards, reducing the risk of non-compliance.

4. Key Features

1. Process Modeling
   - Visual drag-and-drop interface for designing workflows.
   - Integration of business rules for decision-making processes.
   - Customizable process templates for common workflows (e.g., employee onboarding, procurement, invoicing).
2. Workflow Automation
   - Trigger-based actions to automate processes like approvals, notifications, and document routing.
   - Integration with third-party applications (e.g., ERP, CRM, email).
   - Automated escalation for overdue tasks.
3. Task Management
   - Centralized task dashboard for users to manage assigned tasks.

- Prioritization and tracking of task progress.
- Real-time updates and notifications.

4. Monitoring and Analytics
    - Real-time process monitoring with dashboards.
    - Detailed reports and KPIs to measure process efficiency.
    - Root cause analysis to identify and resolve bottlenecks.
5. Collaboration Tools
    - Shared workspaces for team collaboration.
    - Role-based access control to ensure data security.
    - Integrated communication tools (e.g., chat, email).
6. Compliance and Security
    - Audit trails to track all changes and actions in the system.
    - Role-based permissions to secure sensitive data.
    - Compliance templates for industry standards (e.g., Saudi Personal Data Protection Law., ISO).
7. Integration
    - API support for seamless integration with other systems such as HRMS, ERP, and CRM.
    - Import/export functionalities for data sharing.

5. BPMS Lifecycle

1. Process Design

   Define and model workflows using a visual interface, incorporating business rules, triggers, and user roles.
2. Process Execution

   Deploy the designed workflows, enabling users to interact with the system to complete tasks, approvals, or data submissions.
3. Process Monitoring

   Track real-time process execution to identify bottlenecks, delays, or inefficiencies.

4. Process Optimization

   Analyze performance data and feedback to refine workflows for improved efficiency and effectiveness.

6. Benefits of BPMS

1. Enhanced Efficiency

   Automating repetitive tasks and eliminating bottlenecks leads to faster process completion and cost savings.

2. Improved Collaboration

   Teams can work more cohesively with shared workspaces, centralized task dashboards, and communication tools.

3. Increased Agility

   Organizations can quickly adapt to changes in market conditions, regulations, or internal policies by modifying workflows.

4. Better Decision-Making

   Real-time analytics and reporting provide insights into process performance, enabling informed decisions.

5. Regulatory Compliance

   Built-in compliance features reduce the risk of penalties or reputational damage due to non-compliance.

7. Use Cases

1. Employee Onboarding

   Automates the entire onboarding process, from document collection to orientation scheduling, reducing time and effort.

2. Procurement Management

   Streamlines procurement workflows by automating purchase requisitions, approvals, and vendor communication.

3. Customer Support

   Ensures timely resolution of customer issues by automating ticket assignment, escalation, and resolution tracking.

4. Invoice Processing

   Speeds up invoice approvals and payments through automated workflows, ensuring accuracy and timely processing.

5. Regulatory Reporting

   Automates the generation and submission of compliance reports to regulatory bodies.

8. Challenges and Solutions

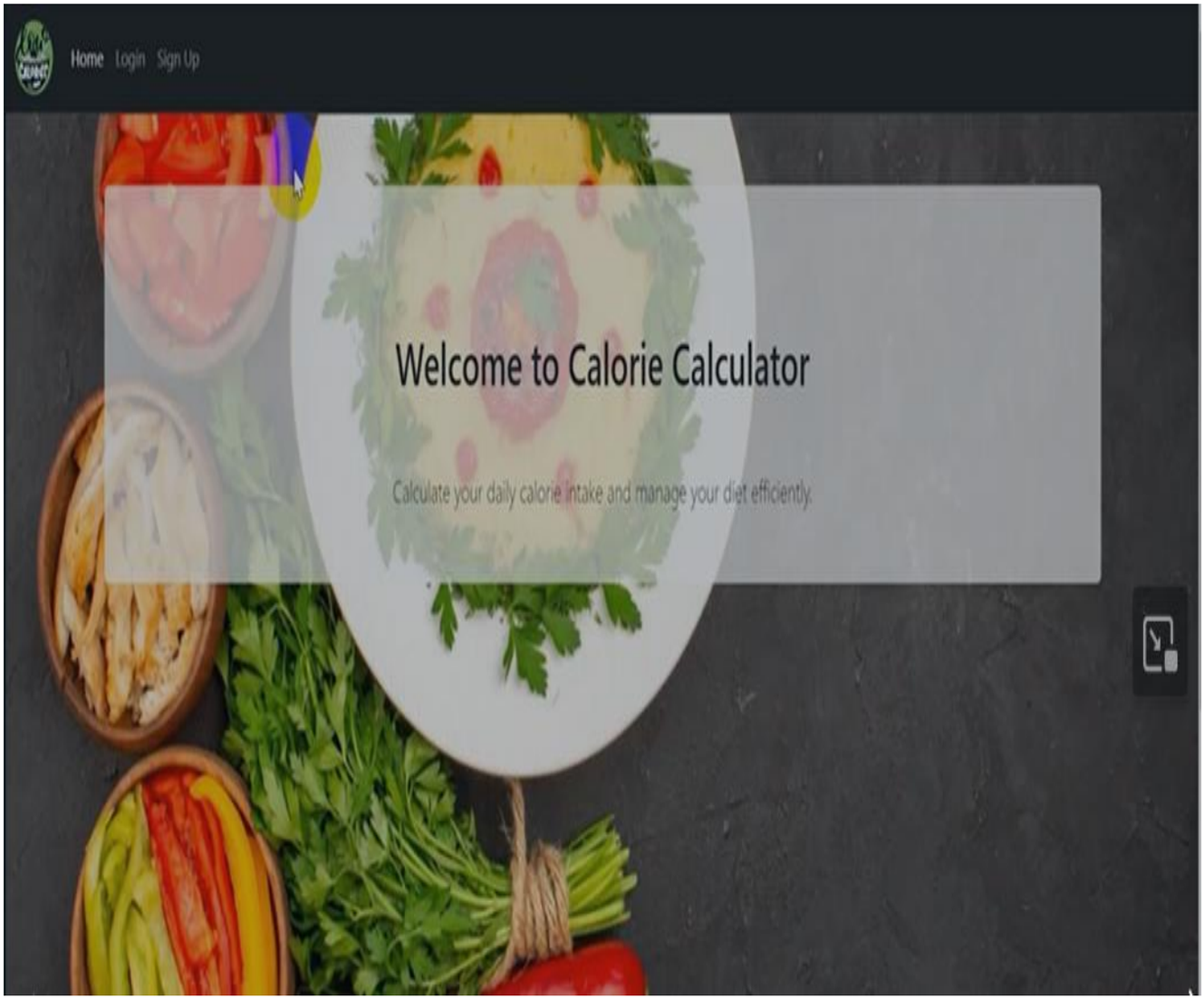
Challenge	Solution
Resistance to change	Provide training and change management support.
Integration with legacy systems	Use APIs and middleware to ensure seamless connectivity.
Complexity of process modeling	Offer user-friendly tools and pre-built templates.
Ensuring data security	Implement robust role-based access and encryption.

9. Conclusion

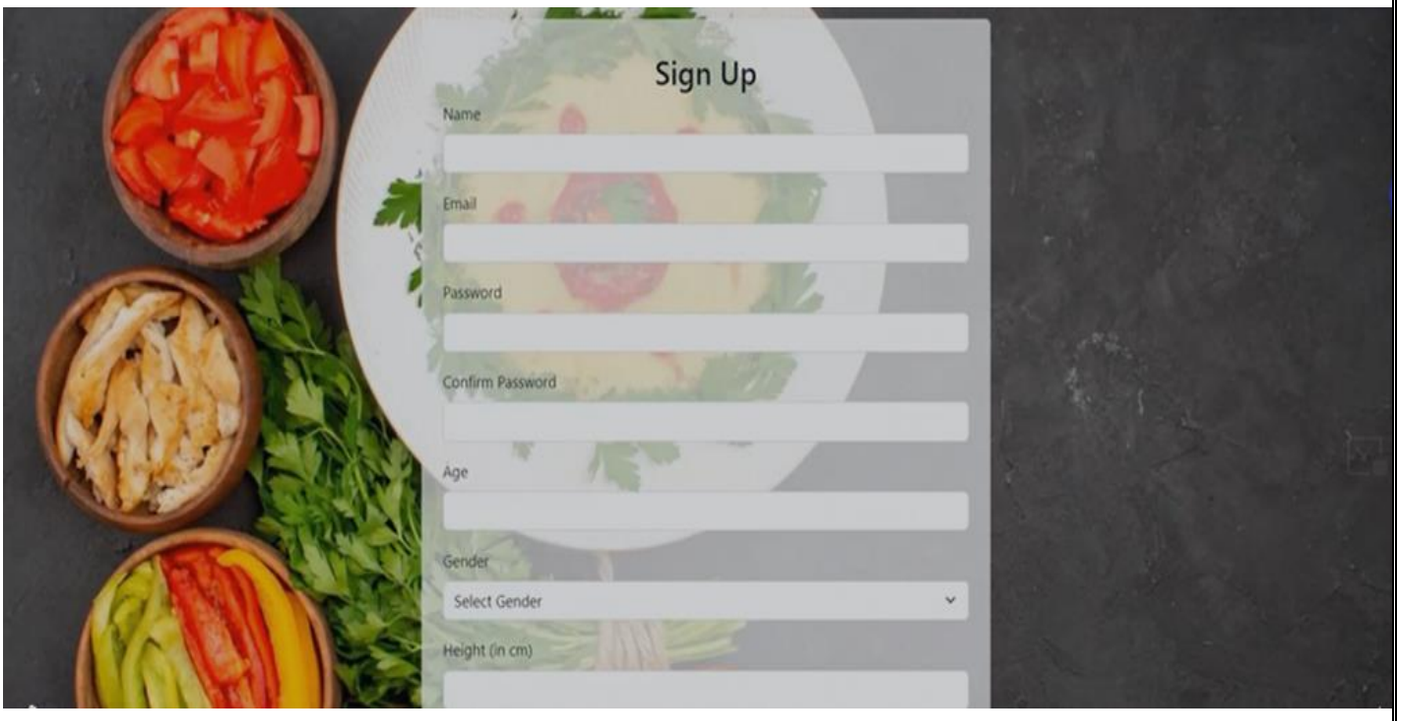
A well-implemented Business Process Management System transforms the way organizations operate by streamlining processes, improving efficiency, and ensuring adaptability. BPMS empowers businesses to stay competitive in dynamic markets while maintaining compliance and fostering collaboration. It is a strategic investment for organizations aiming to optimize performance and drive innovation.

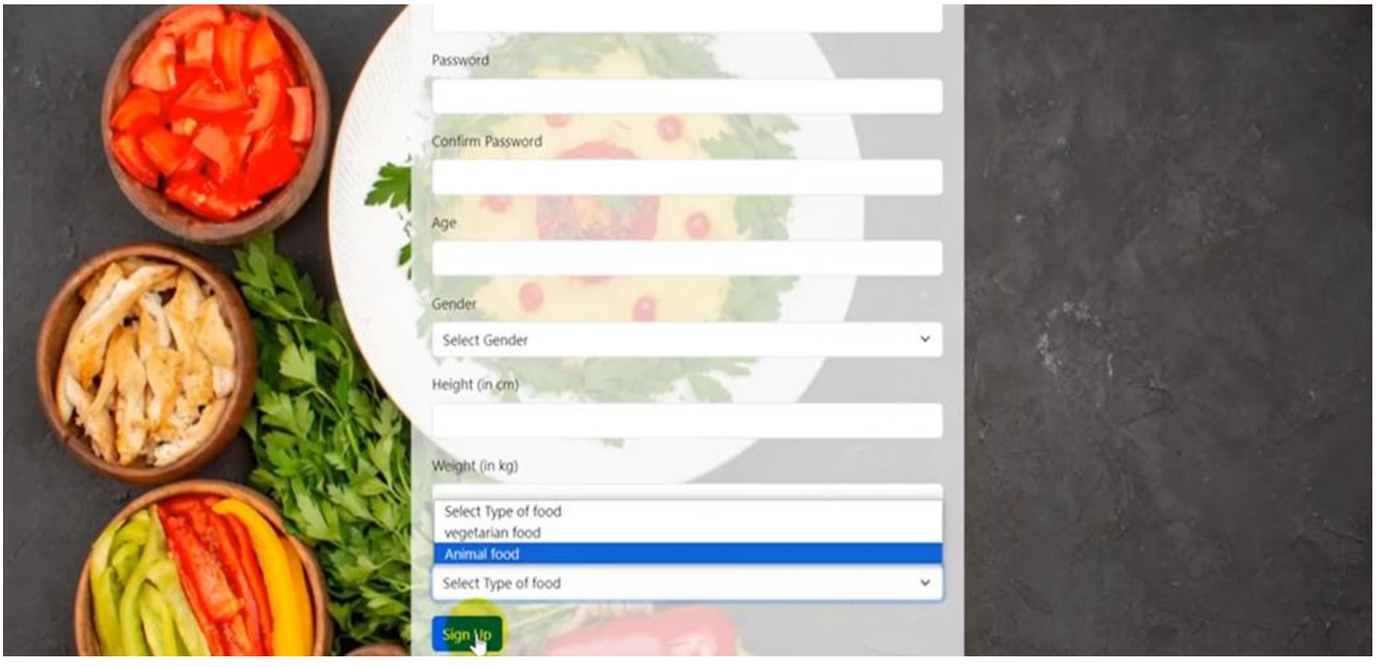
5.4 Human Interface

5.4.1 Home page

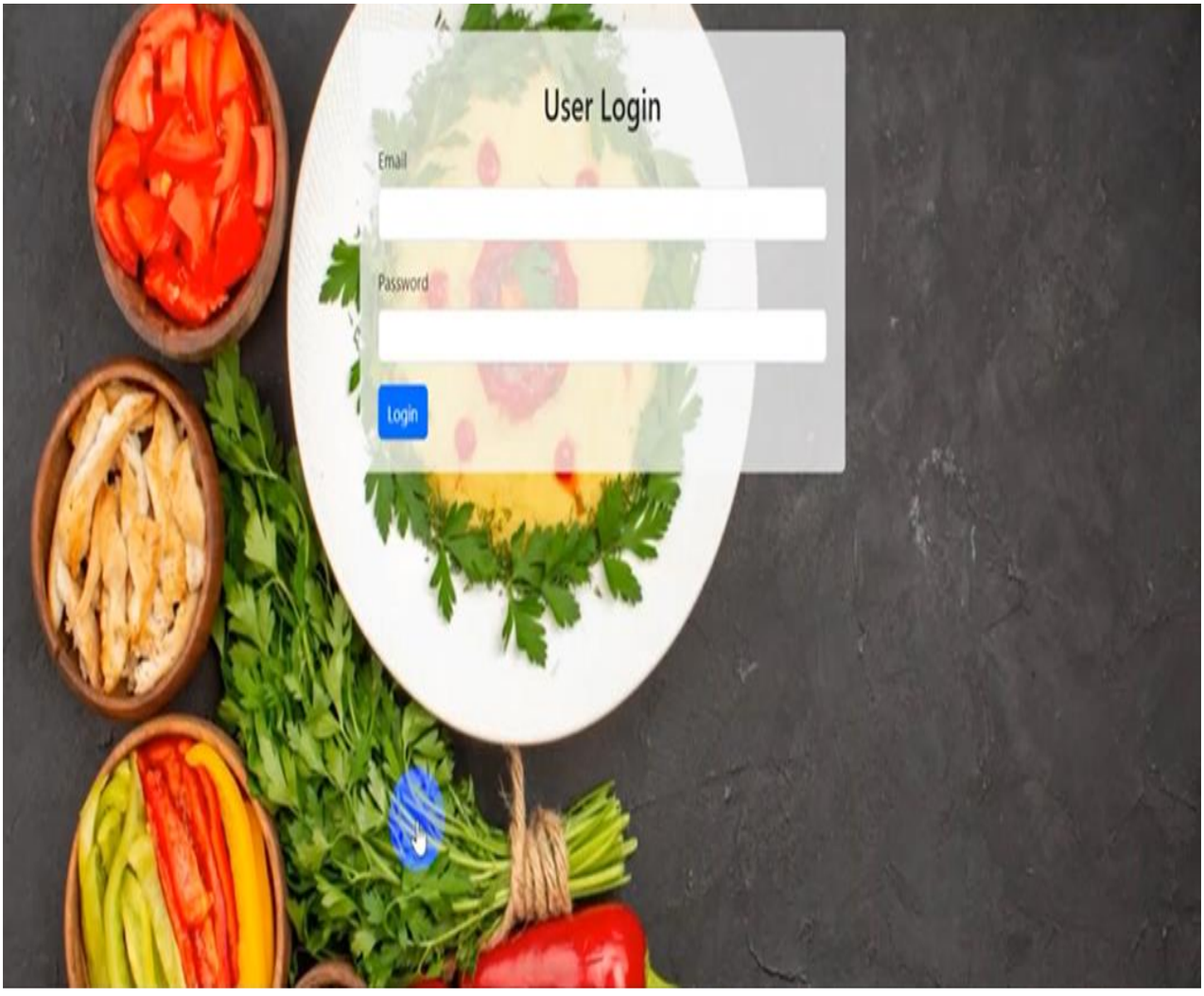


5.4.2 signup page

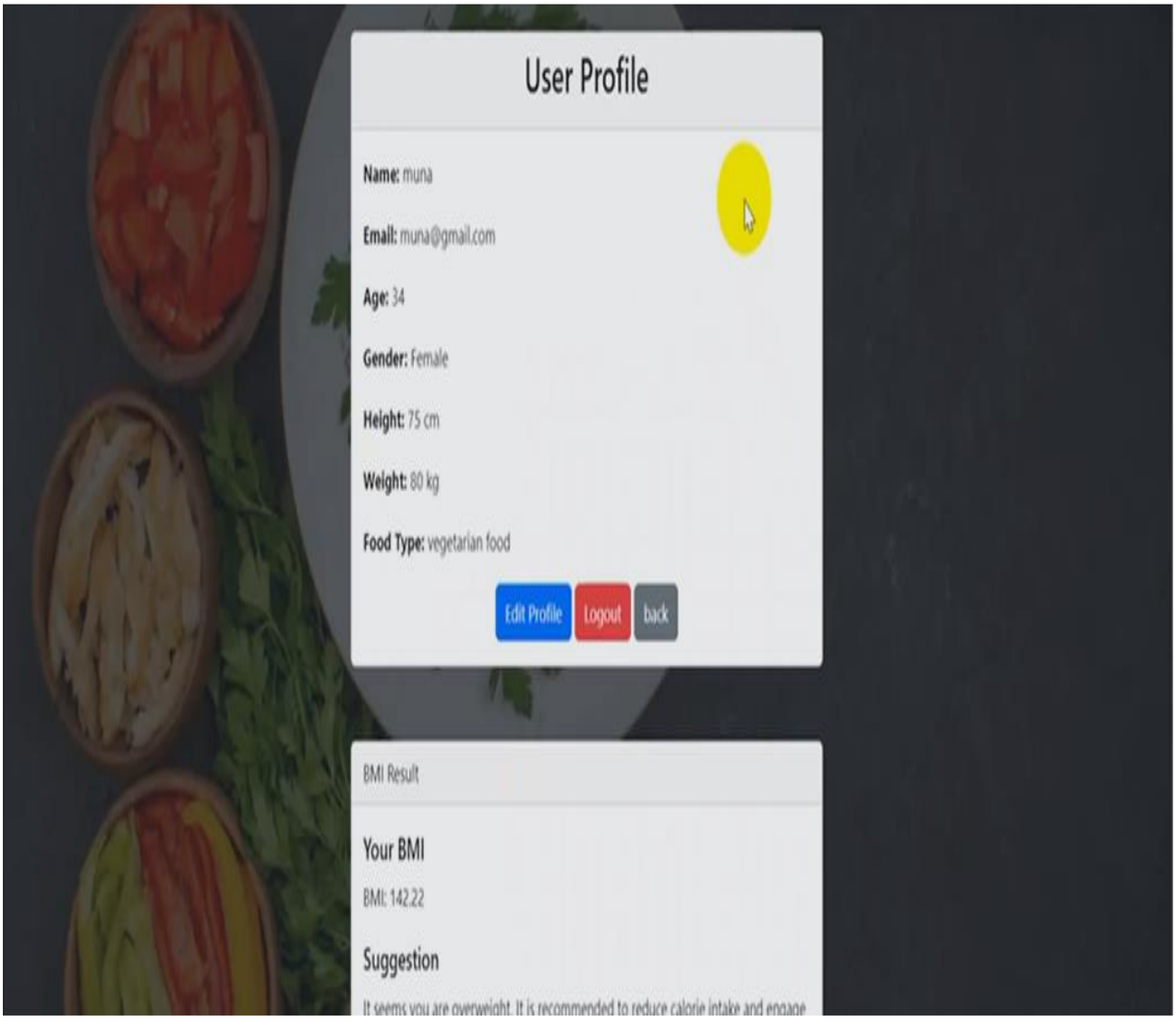




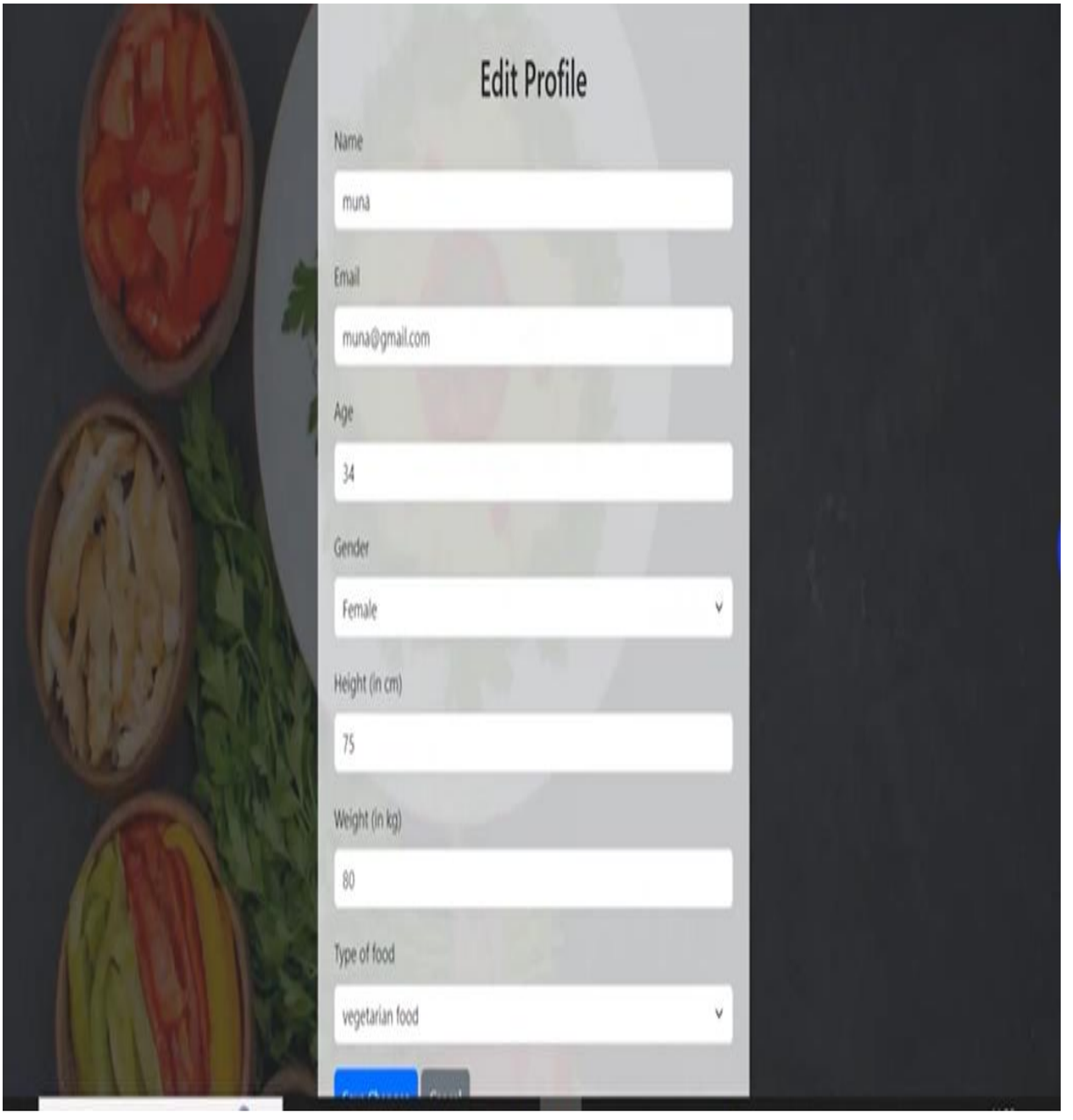
5.4.3 Login Page



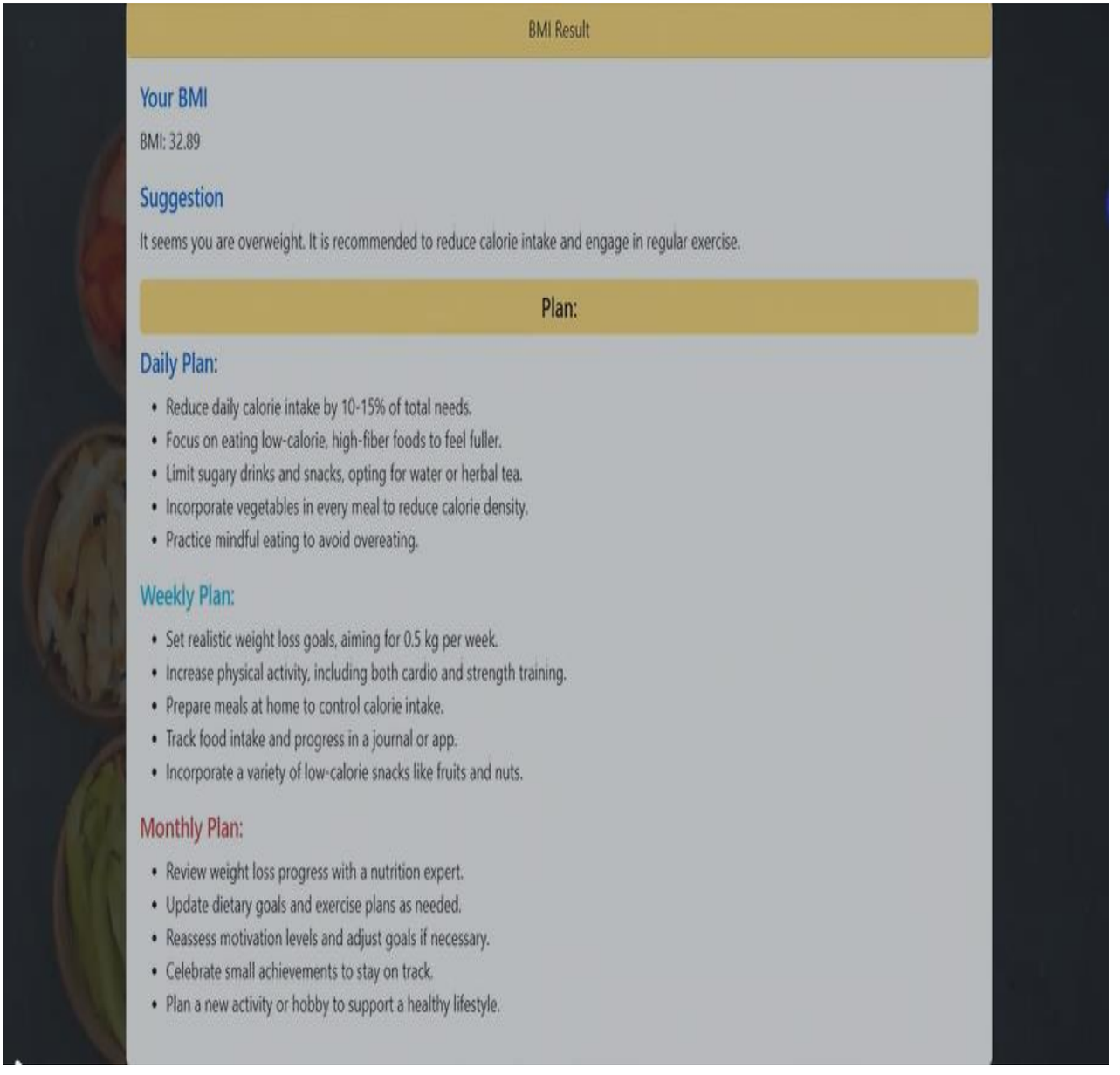
5.4.4 User Profile



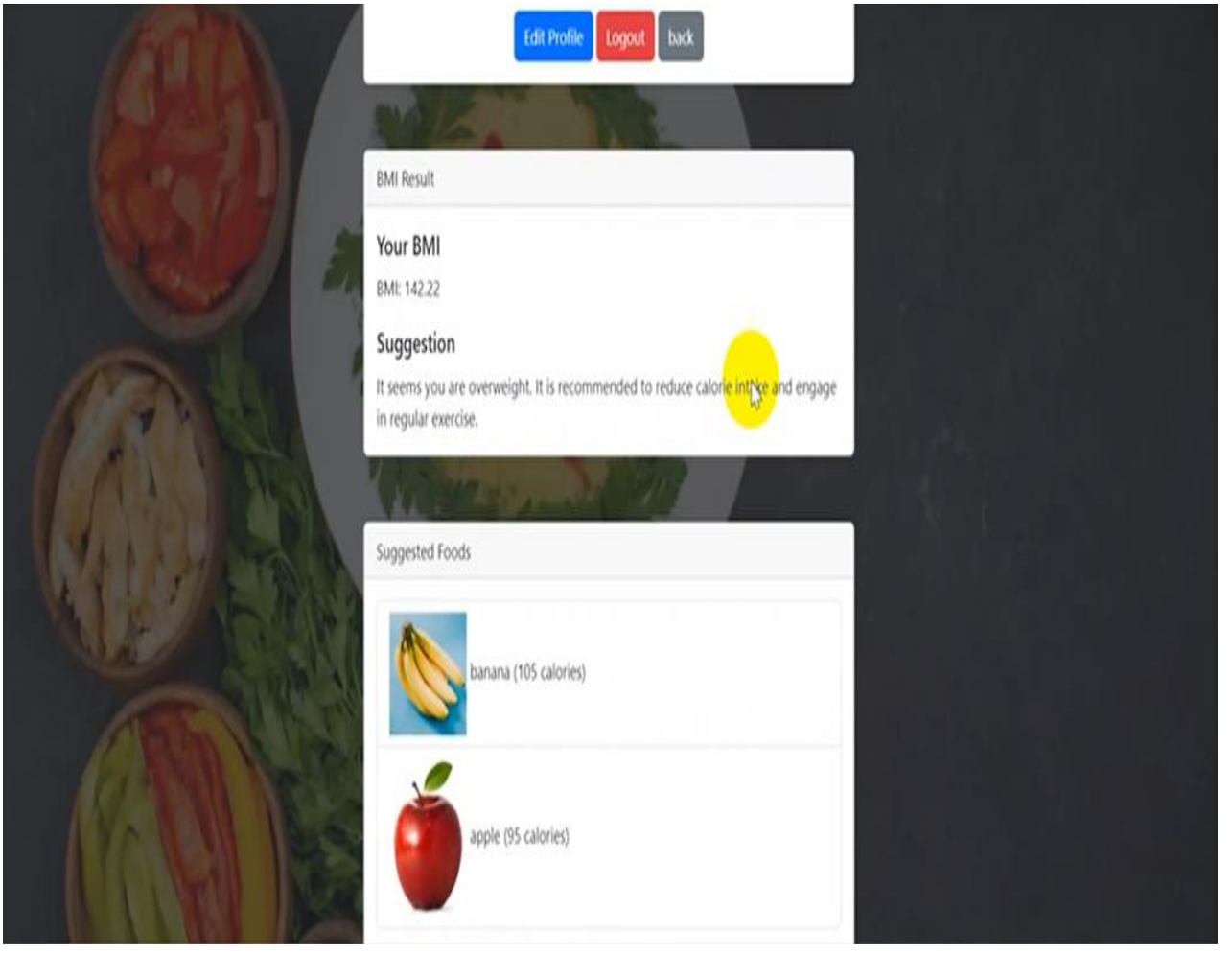
5.4.5 Edit Profile

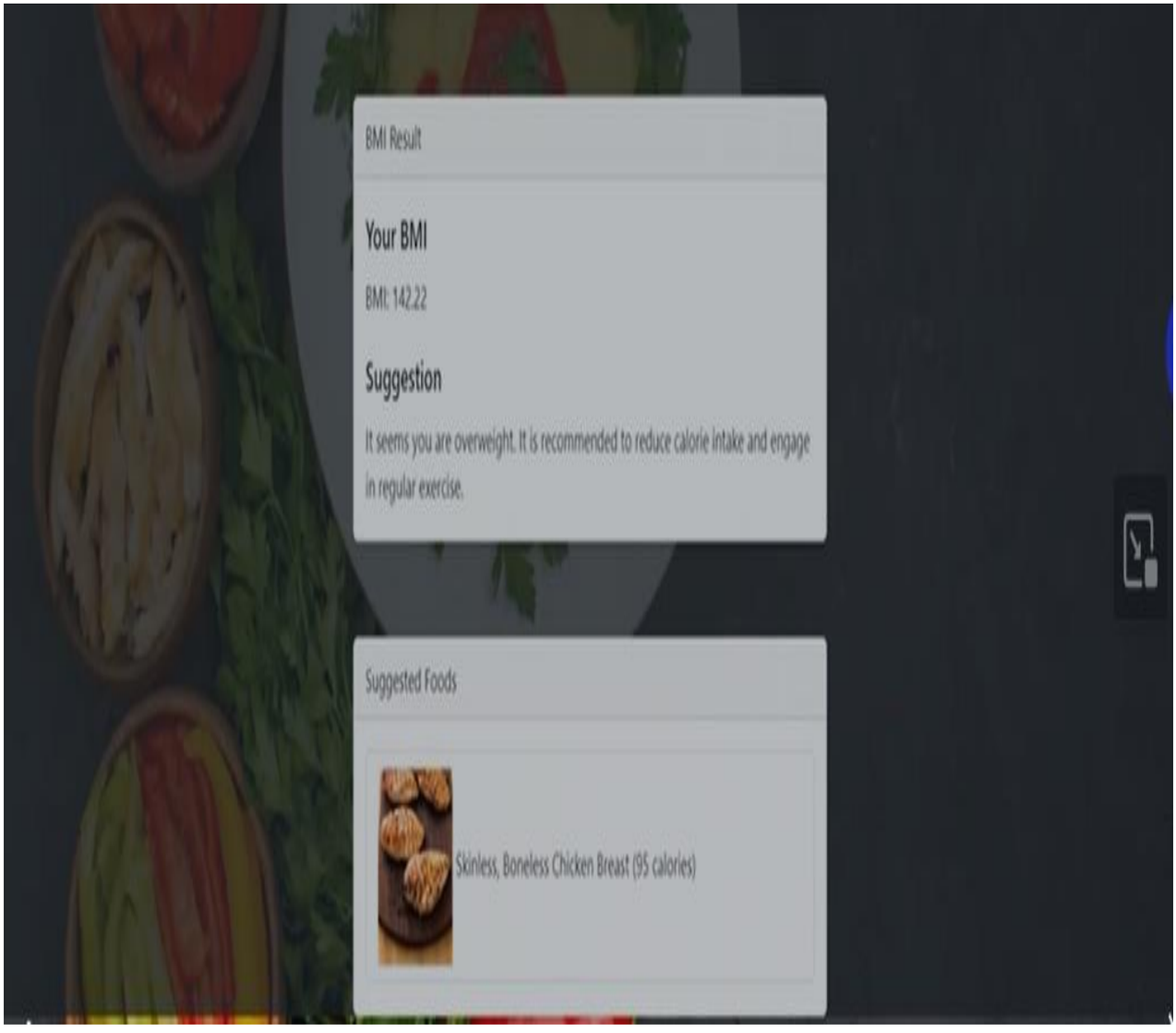

Edit Profile
Name
muna
Email
muna@gmail.com
Age
34
Gender
Female
Height (in cm)
75
Weight (in kg)
80
Type of food
vegetarian food

5.4.6 Plan

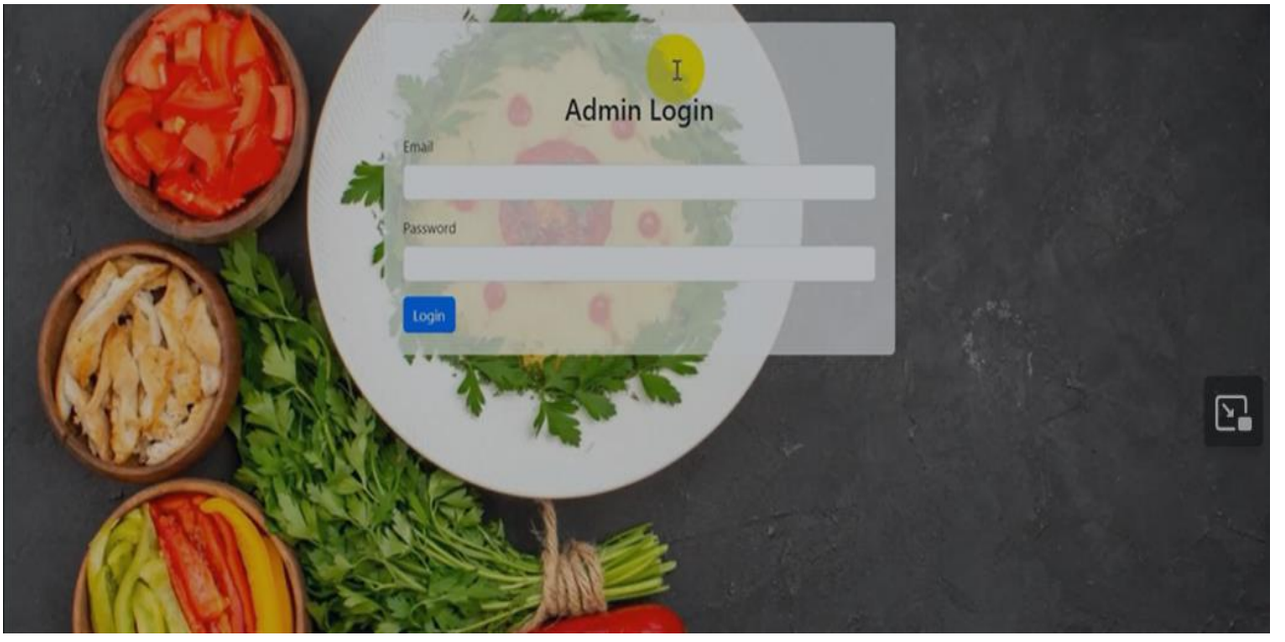


5.4.7 Food suggestion

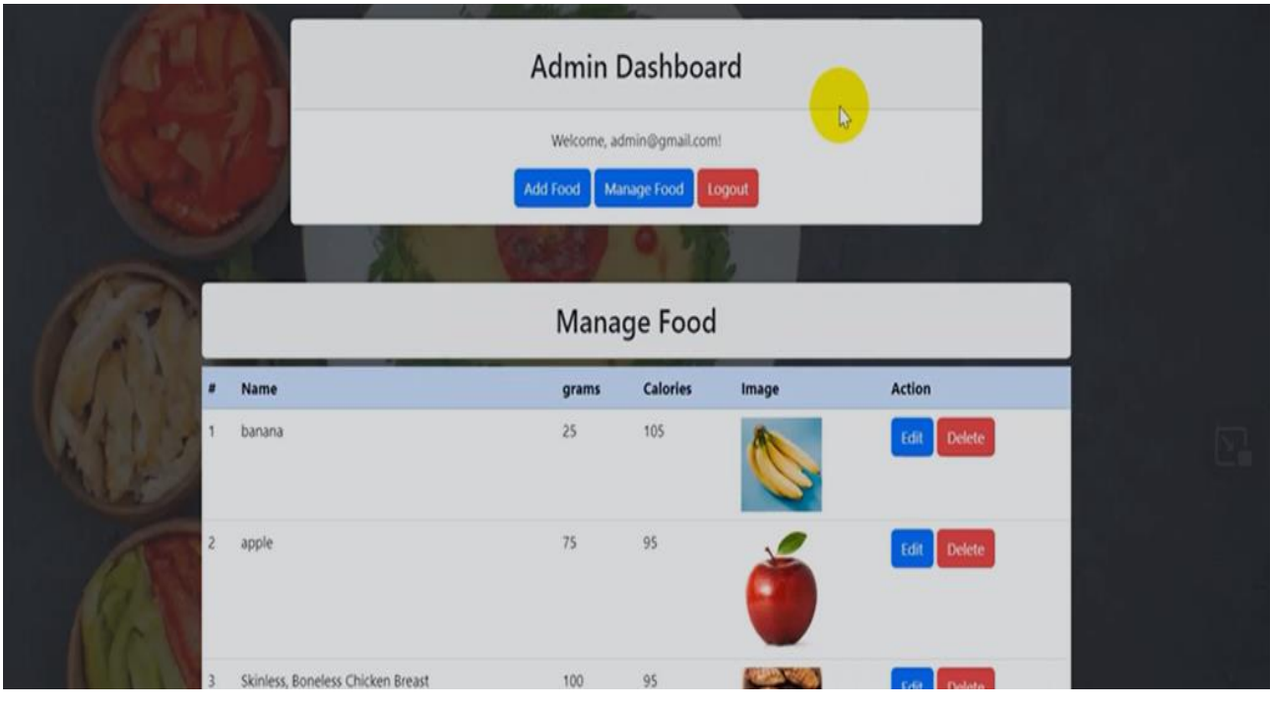

Edit Profile
Logout
back
BMI Result
Your BMI
BMI: 142.22
Suggestion
It seems you are overweight. It is recommended to reduce calorie intake and engage in regular exercise.
Suggested Foods
banana (105 calories)
apple (95 calories)



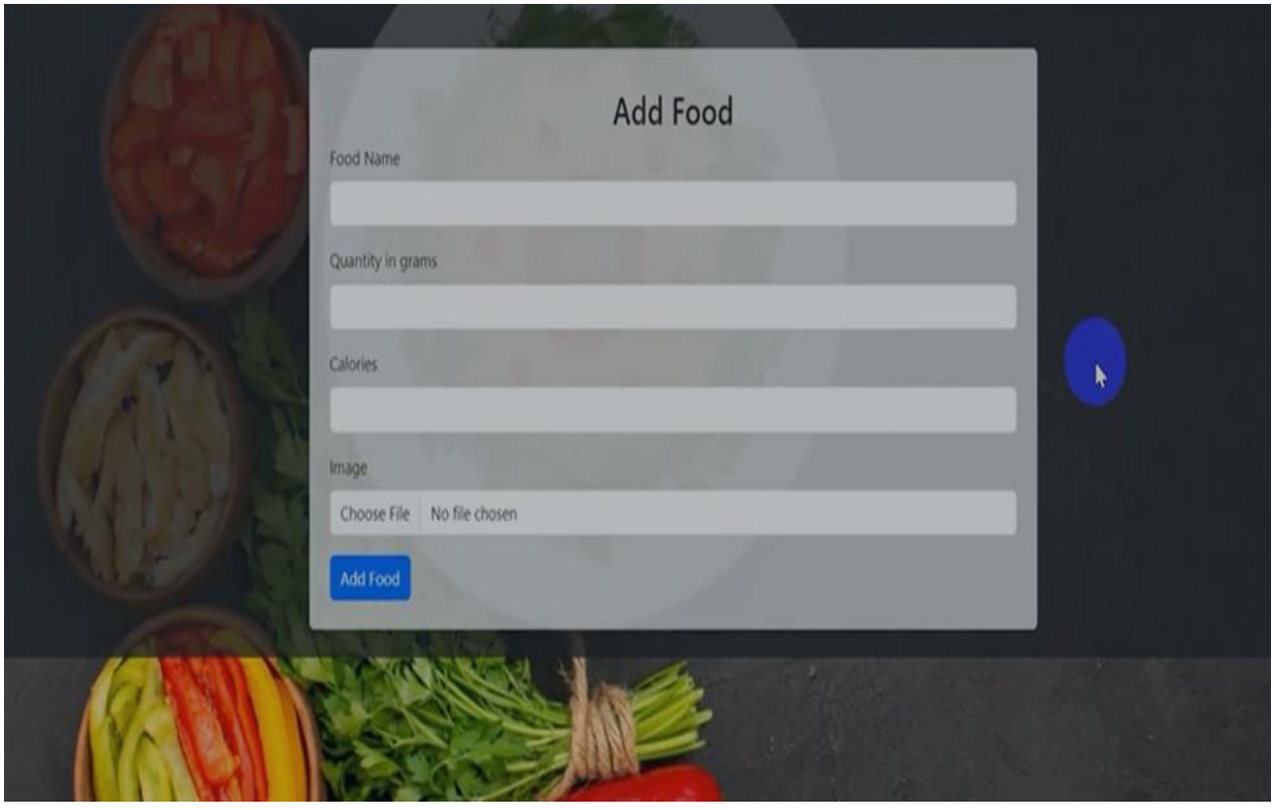
5.4.8 Admin Login

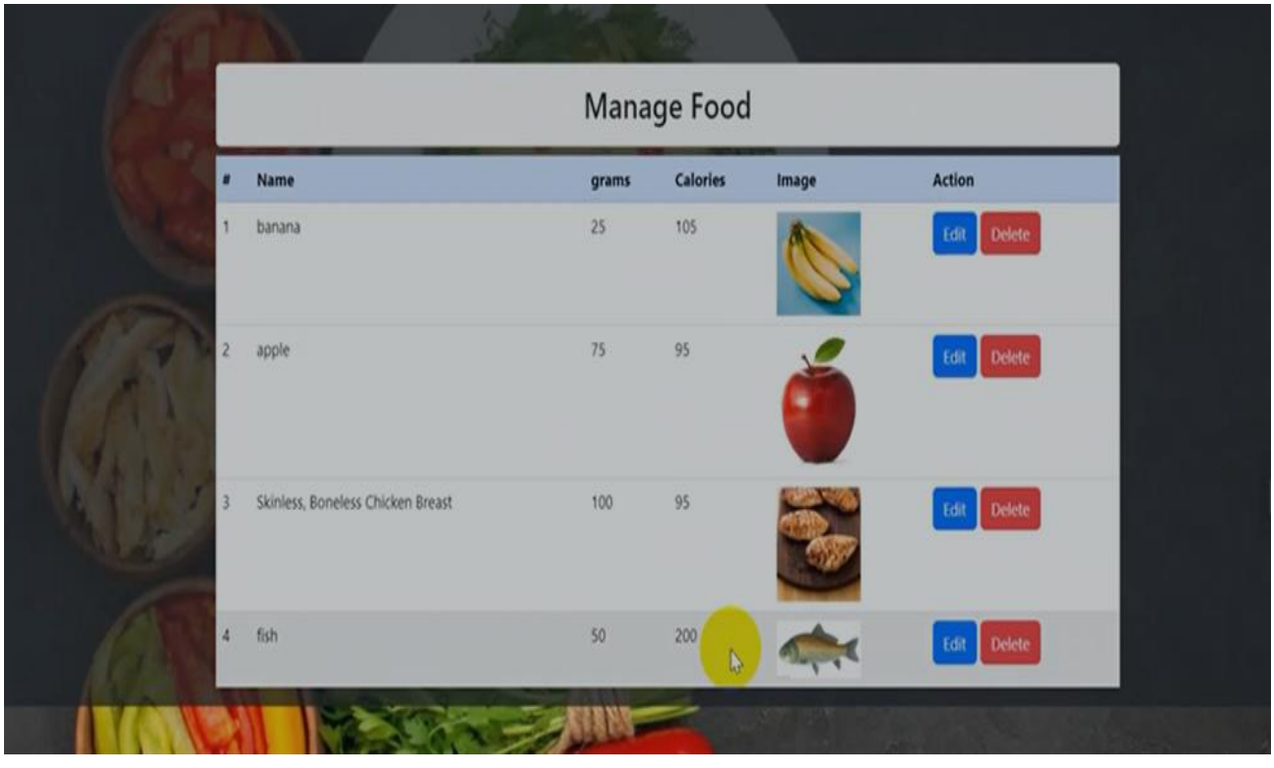


5.4.8 Admin Dashboard



5.4.9 Add or Remove Food


Add Food
Food Name
Quantity in grams
Calories
Image
Choose File
No file chosen
Add Food


Manage Food
Name grams Calories Image Action
1 banana 25 105 Edit Delete
2 apple 75 95 Edit Delete
3 Skinless, Boneless Chicken Breast 100 95 Edit Delete
4 fish 50 200 Edit Delete

6. Implementation Details

6.1 Task Description and Project Description

The calorie calculator project is designed to provide users with a personalized 30-day meal plan tailored to their dietary preferences, BMI, and health goals. The project is implemented using modern web development tools and practices, ensuring a responsive, scalable, and secure application.

Key Project Objectives:

- Allow users to register, log in, and manage profiles.
- Provide BMI calculations and tailored meal plans.
- Enable administrators to manage food databases.
- Ensure high performance, usability, and scalability.

Key Tasks:

1. Front-End Development: Building a responsive and intuitive user interface using React.
2. Back-End Development: Creating a scalable API using Node.js and Express.js.
3. Database Integration: Using MongoDB to store and manage user and food data.
4. Testing: Ensuring application reliability through Jest for unit and integration testing.
5. Deployment: Hosting the application on Google Cloud Platform (GCP).

6.2 Implementation Details

Tools and Technologies Used:

1. Integrated Development Environment (IDE):
   - Visual Studio Code: Chosen for its lightweight and extensive extensions for front-end and back-end development.
2. Version Control:
   - GitHub: Used as the primary version control system to track source code changes, enable collaboration, and maintain code history.
3. Front-End:
   - HTML, CSS (with Bootstrap): For structuring and styling the user interface.

- JavaScript (React): For building dynamic, component-based UIs.

4. Back-End:
    - Node.js and Express.js: To build a scalable and efficient server-side application.
    - Mongoose: For object modeling and connecting the MongoDB database with the backend.
5. Database:
    - MongoDB (NoSQL): Selected for its flexibility in handling diverse user and food data.
6. User Interface Design:
    - Figma, Proto.io, Canva: Used to design and prototype user interfaces before implementation.
7. Testing:
    - Jest: Utilized for automated unit and integration testing to ensure code quality and reliability.
8. Deployment and Hosting:
    - Google Cloud Platform (GCP): Chosen for its robust infrastructure and scalability to host the initial application build.

6.3 Operating Environment

The system operates in a web-based environment with the following requirements:

- Client-Side:
    - Modern web browsers (e.g., Chrome, Firefox, Safari).
    - Devices with at least 2GB RAM and a stable internet connection.
- Server-Side:
    - Google Cloud Platform (GCP) hosting Node.js and MongoDB instances.
    - Minimum specifications: 2vCPU, 8GB RAM, and 50GB SSD storage for the initial deployment.

6.4 Incremental Implementation

6.4.1 Strategy

The incremental implementation strategy divides the development process into three iterations. Each iteration focuses on delivering a specific set of features, ensuring continuous progress and integration.

6.4.2 Iteration One

Objective: Build the foundational structure of the application.

- Front-End:
  - Implement static pages for registration, login, and profile management.
  - Design templates using HTML, CSS, and React components.
- Back-End:
  - Set up the Node.js server and connect it to MongoDB using Mongoose.
  - Create APIs for user registration and login.
- Testing:
  - Validate initial APIs with unit tests using Jest.
- Deployment:
  - Deploy the initial static build on GCP.

6.4.3 Iteration Two

Objective: Develop core functionalities and dynamic features.

- Front-End:
  - Add dynamic components for BMI calculation and meal plan generation.
  - Ensure responsiveness using Bootstrap and React hooks.
- Back-End:
  - Build APIs for BMI calculations and personalized meal plan generation.
  - Implement CRUD operations for the admin panel (e.g., managing food data).
- Database:
  - Structure collections for user data, food items, and meal plans in MongoDB.
- Testing:
  - Conduct integration testing to ensure the seamless operation of the API endpoints.
- Deployment:
  - Update the deployment with dynamic features on GCP.

6.4.4 Iteration Three

Objective: Finalize the application and prepare for production.

- Front-End:
  - Refine the UI based on user feedback and implement advanced features.
  - Integrate animations and enhance user experience.
- Back-End:
  - Optimize APIs for performance and security.
  - Add logging and monitoring for admin activities.
- Database:
  - Perform data validation and indexing for improved query performance.
- Testing:
  - Conduct end-to-end testing to ensure system reliability.
  - Test for edge cases and error handling.
- Deployment:
  - Deploy the production-ready application on GCP with proper scaling configurations.

7. Software Testing

7.1 Purpose

The purpose of this section is to define the testing strategy, scope, and activities required to ensure the quality and reliability of the calorie calculator system. Testing ensures that all functional and non-functional requirements are met, and the system operates as intended in various environments and scenarios.

7.2 Test Plan

7.2.1 Test Items

The test items include all features and components of the calorie calculator system, such as:

- User registration and login functionality.
- BMI calculation logic and accuracy.

- Personalized meal plan generation.
- Admin panel for managing food data.
- Database integration for user and food data storage.
- Front-end responsiveness across devices and browsers.

7.2.2 Scope

This section describes the features and components to be tested in the calorie calculator system. The scope ensures comprehensive coverage of functional and non-functional requirements.

Table 12. System Features for Testing

Feature	Test Case Examples	Priority
User Registration	Verify form validations, data storage, and error handling for invalid inputs.	Must-Have
Login/Authentication	Check secure access, password validation, and error messages for incorrect inputs.	Must-Have
BMI Calculation	Validate accuracy of BMI formula and error handling for invalid data.	Must-Have
Meal Plan Generation	Verify tailored suggestions based on user preferences and allergies.	Must-Have

Feature	Test Case Examples	Priority
Admin Food Management	Test CRUD operations for managing food items and calorie data.	Should-Have
Responsive Design	Ensure proper layout and functionality across devices and browsers.	Should-Have

7.2.3 Approach

The testing process will follow these steps:

1. Unit Testing: Test individual modules, such as user registration or BMI calculation.
2. Integration Testing: Validate the interaction between front-end and back-end components.
3. System Testing: Assess the entire application against system requirements.
4. Acceptance Testing: Perform testing with stakeholders to verify that the system meets their expectations.

7.2.5 Item Pass/Fail Criteria

- Pass Criteria: A test case is marked as "Pass" if the actual result matches the expected result without errors.
- Fail Criteria: A test case is marked as "Fail" if the actual result deviates from the expected result or errors occur.

7.2.6 Suspension Criteria and Resumption Requirements

- Suspension Criteria: Testing will be suspended if critical issues such as server downtime or incomplete modules prevent test execution.

- Resumption Requirements: Testing will resume once the issues are resolved and approved by the testing team.

7.2.7 Test Deliverables

The following deliverables will be produced during the testing phase:

- Test Plan Document.
- Test Cases and Scripts.
- Test Execution Reports.
- Bug Reports and Logs.
- Final Test Summary Report.

7.2.8 Testing Tools

1. Jest

- Purpose: Used for unit and integration testing of front-end and back-end modules.
- Features:
    - Mocking capabilities for simulating dependencies.
    - Snapshot testing for UI components.
    - Fast and reliable execution.

2. Postman

- Purpose: To validate API functionality, performance, and security.
- Features:
    - Supports creating automated API test suites.
    - Allows load and performance testing of API endpoints.
    - Provides detailed test results and debugging insights.

7.2.9 Testing Tasks

1. Create and review test cases.
2. Set up the testing environment and tools.

3. Execute unit, integration, and system tests.
4. Log and resolve identified defects.
5. Generate test execution reports.

7.2.10 Testing Environment

- **Client-Side**: Modern browsers (e.g., Chrome, Firefox, Safari) on desktop and mobile devices.
- **Server-Side**: Testing on Google Cloud Platform (GCP) environment with Node.js and MongoDB.
- **Test Data**: Predefined sets of user profiles, food items, and system configurations.

7.2.11 Testing Team Organization

Team Member	Role	Responsibilities
Tester 1	Unit Tester	Develop and execute unit test cases for front-end and back-end modules.
Tester 2	Integration Tester	Validate interactions between API endpoints and database.
Tester 3	System Tester	Perform end-to-end testing of the entire application.
Tester 4	Automation Engineer	Create automated test scripts for UI and API testing.
QA Lead	Quality Assurance Lead	Oversee the testing process, analyze results, and ensure issue resolution.

7.2.12 Staffing and Training Needs

- **Staffing Needs**: Four testers and one QA lead.

- **Training Needs**: Testers will be trained in Jest, Selenium, and Postman for effective testing.

7.2.13 Schedule

Testing Activity	Start Date	End Date	Duration
Test Plan Creation	2024-11-01	2024-11-03	3 days
Unit Testing	2024-11-04	2024-11-10	7 days
Integration Testing	2024-11-11	2024-11-17	7 days
System Testing	2024-11-18	2024-11-24	7 days
Acceptance Testing	2024-11-25	2024-11-27	3 days
Final Test Report Submission	2024-11-28	2024-11-28	1 day

7.2.14 Risks Identified and Mitigations Planned

Risk	Likelihood	Impact	Mitigation Plan
Server Downtime	Medium	High	Schedule testing during non-peak hours and have a backup server.
Delays in Feature Completion	High	Medium	Focus on must-have features during testing.
Browser Compatibility Issues	Medium	Medium	Use BrowserStack for cross-browser testing early in the process.
Insufficient Test Data	Low	High	Generate diverse test datasets covering edge cases.

7.3 Test Specifications and Results

7.3.1 Unit Tests

7.3.1.1 Logging in to the portal (SF-1)

Unit tests for the login functionality ensure that the system accurately handles user credentials and manages session creation. Below is the detailed table for login test cases.

Table 18. Unit Test Cases for Logging In (SF-1)

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
UT-1	Valid login credentials	1. Enter a registered email. 2. Enter the correct password. 3. Click on "Login".	User is logged in successfully and redirected to the dashboard.	As expected	Pass

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
UT-2	Invalid password	1. Enter a registered email. 2. Enter an incorrect password. 3. Click on "Login".	System displays an error message: "Invalid credentials. Please try again."	As expected	Pass
UT-3	Unregistered email	1. Enter an unregistered email. 2. Enter a password. 3. Click on "Login".	System displays an error message: "Email not found. Please register."	As expected	Pass
UT-4	Empty email field	1. Leave the email field empty. 2. Enter a password. 3. Click on "Login".	System displays an error message: "Email is required."	As expected	Pass

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
UT-5	Empty password field	1. Enter a registered email. 2. Leave the password field empty. 3. Click on "Login".	System displays an error message: "Password is required."	As expected	Pass
UT-6	SQL injection attempt	1. Enter an SQL injection string (e.g., ' OR '1'='1) in the email or password field. 2. Click on "Login".	System rejects the input and displays an error message: "Invalid credentials. Please try again."	As expected	Pass

7.3.2 Integration Tests

Integration tests validate the interaction between components such as the front-end, back-end, and database. Below is a detailed table for integration test cases.

Table 19. Integration Test Cases

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
IT-1	User data retrieval on login	1. Log in with valid credentials. 2. System queries the database for user data.	User data (name, preferences) is retrieved and displayed on the dashboard.	As expected	Pass
IT-2	Password hashing during login	1. Log in with valid credentials. 2. Back-end hashes the password before database comparison.	Password is hashed and securely compared to the stored hash in the database.	As expected	Pass

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
IT-3	Database validation for new users	1. Register a new user. 2. Verify that the user data is stored in the MongoDB database.	New user data is correctly stored in the database with unique constraints enforced on email.	As expected	Pass
IT-4	Error handling for database disconnection	1. Simulate a database disconnection. 2. Attempt to log in with valid credentials.	System displays an error message: "Unable to connect to the database. Please try again later."	As expected	Pass

7.3.3 System Tests

System tests ensure that the entire application functions as intended across various scenarios. Below is the detailed table for system test cases.

Table 20. System Test Cases

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
ST-1	End-to-end registration flow	1. Open the registration page. 2. Enter valid details. 3. Submit the form.	User account is created, and the user is redirected to the login page.	As expected	Pass
ST-2	Personalized meal plan generation	1. Log in with a registered account. 2. Enter dietary preferences and allergies. 3. Submit form.	The system generates a 30-day meal plan excluding restricted food items.	As expected	Pass

Test Case ID	Test Description	Test Steps	Expected Result	Actual Result	Status
ST-3	Admin CRUD operations	1. Log in as an admin. 2. Add, edit, and delete a food item. 3. Check the updates.	Admin successfully manages food items, and changes are reflected in the user-facing meal plan.	As expected	Pass
ST-4	Cross-browser compatibility	1. Open the application on Chrome, Firefox, and Safari. 2. Perform key operations.	The application functions correctly and the layout is consistent across all browsers.	As expected	Pass
ST-5	Load testing for concurrent users	1. Simulate 500 concurrent users logging in and generating meal plans.	The application handles the load with minimal latency and no server crashes.	As expected	Pass

8. Constraints, Limitations, and Ethical Implications

8.1 Project Constraints and Limitations

1. Technical Constraints:
   - Database Type: The use of MongoDB restricts certain advanced relational database functionalities like JOIN operations.
   - Scalability: The initial deployment on Google Cloud Platform (GCP) may face scalability challenges with high user loads.
   - Front-End Framework: Limited to React and Bootstrap, which may restrict design complexity and flexibility.
2. Time Constraints:
   - The project has a strict timeline of six months, which restricts the inclusion of additional advanced features like AI-based meal recommendations.
3. Resource Constraints:
   - Team Size: A small team limits the capacity to handle parallel tasks efficiently.
   - Budget: Restricted budget affects the ability to adopt premium tools or services, such as advanced analytics platforms.
4. Functional Limitations:
   - Personalization: The system may not accommodate highly specific dietary preferences or rare allergies in the initial release.
   - Offline Access: The application requires internet connectivity for functionality, limiting accessibility in offline scenarios.
5. Ethical Implications:
   - Data Privacy: Ensuring user data, such as health metrics and dietary preferences, is securely stored and not misused.
   - Bias in Recommendations: Avoiding unintended biases in meal plans, which could misrepresent certain food options or cultures.
   - Accessibility: Designing an interface that accommodates users with disabilities to ensure inclusivity.

1. Mobile Application Development

- **Cross-Platform Support:** Develop mobile apps for iOS and Android to provide users with on-the-go access to the calorie calculator.
- **Push Notifications:** Implement reminders for meal times, water intake, and exercise sessions to keep users engaged.
- **Offline Functionality:** Enable offline calorie tracking and meal plan access, syncing with the server when online.

2. Integration with Wearable Devices

- **Fitness Tracker Syncing:** Integrate with popular wearable devices (e.g., Fitbit, Apple Watch, Garmin) to automatically import fitness data, such as step count and calories burned.
- **Heart Rate Monitoring:** Use heart rate data to provide more personalized health insights and recommendations.

3. AI-Powered Personalized Recommendations

- **Dynamic Meal Plan Adjustments:** Incorporate AI to adjust meal plans based on user feedback, progress, and dietary trends.
- **Recipe Suggestions:** Provide AI-driven recipe suggestions based on the user's available ingredients and calorie goals.
- **Health Insights:** Use AI to analyze user data over time, offering insights into health trends and suggestions for improvement.

4. Gamification

- **Achievements and Rewards:** Introduce badges, challenges, and rewards to motivate users to meet their health goals.
- **Community Challenges:** Allow users to participate in community-based challenges, such as steps walked or calories burned in a week, to foster engagement.

5. Enhanced Nutritional Database

- **Global Food Options:** Expand the food database to include a wider variety of international cuisines and dishes.
- **Barcode Scanner:** Add a barcode scanning feature to quickly log packaged foods into the system.
- **User Contributions:** Allow users to submit custom foods and recipes to the database for validation and sharing.

6. Advanced Analytics and Reporting

- **Progress Tracking:** Provide detailed reports and visualizations of user progress over time (e.g., weight loss, calorie intake trends).
- **Nutritional Breakdown:** Offer insights into macronutrient distribution (e.g., protein, carbohydrates, fats) for each meal plan.
- **Health Goal Analysis:** Show how close users are to their long-term goals and suggest strategies to achieve them.

7. Integration with Third-Party Services

- **Telemedicine Platforms:** Connect with healthcare providers for expert advice on diet and nutrition.
- **Fitness Apps:** Sync with fitness apps like MyFitnessPal, Strava, or Google Fit for a more holistic approach to health tracking.
- **E-commerce Integration:** Allow users to order groceries or meal kits based on their meal plans.

8. Multilingual and Accessibility Features

- **Multilingual Support:** Add support for multiple languages to cater to a broader audience.

- **Accessibility Enhancements:** Ensure the system complies with accessibility standards (e.g., WCAG) to accommodate users with disabilities.

9. Advanced Security Features

- **Two-Factor Authentication:** Enhance user data protection by implementing two-factor authentication.
- **Encrypted Data Storage:** Ensure all user data is stored securely with encryption.
- **Privacy Controls:** Allow users to control data sharing and visibility, ensuring compliance with regulations like Saudi Personal Data Protection Law.

10. AI Nutritionist Assistant

- **Interactive Assistant:** Implement an AI-based nutritionist that can answer user questions, suggest meal options, and guide users through setting and achieving their health goals.
- **Real-Time Feedback:** Provide instant feedback on logged meals or planned activities, helping users make healthier choices.

11. Integration of Mental Health and Wellbeing

- **Mood and Food Logging:** Allow users to log their mood and correlate it with eating habits to identify patterns.
- **Mindful Eating Suggestions:** Include mindfulness tips and techniques for healthier eating behaviors.

Conclusion

These future enhancements aim to elevate the calorie calculator system into a comprehensive, engaging, and user-friendly health management tool. By leveraging emerging technologies and focusing on user needs, the system can continue to evolve and support a larger audience in achieving their dietary and health goals.

11. References

1. Books and Articles:
    - Sommerville, I. (2016). Software Engineering (10th Edition). Pearson Education.
    - Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
2. Websites and Documentation:
    - React Documentation. https://reactjs.org/

- Node.js Official Website. https://nodejs.org/
- MongoDB Documentation. https://www.mongodb.com/docs/
- Google Cloud Platform. https://cloud.google.com/

3. Research Papers:
   - Martin, R. C. (2003). Agile Software Development: Principles, Patterns, and Practices. Prentice Hall.
   - Meyer, B. (1997). Object-Oriented Software Construction (2nd Edition). Prentice Hall.