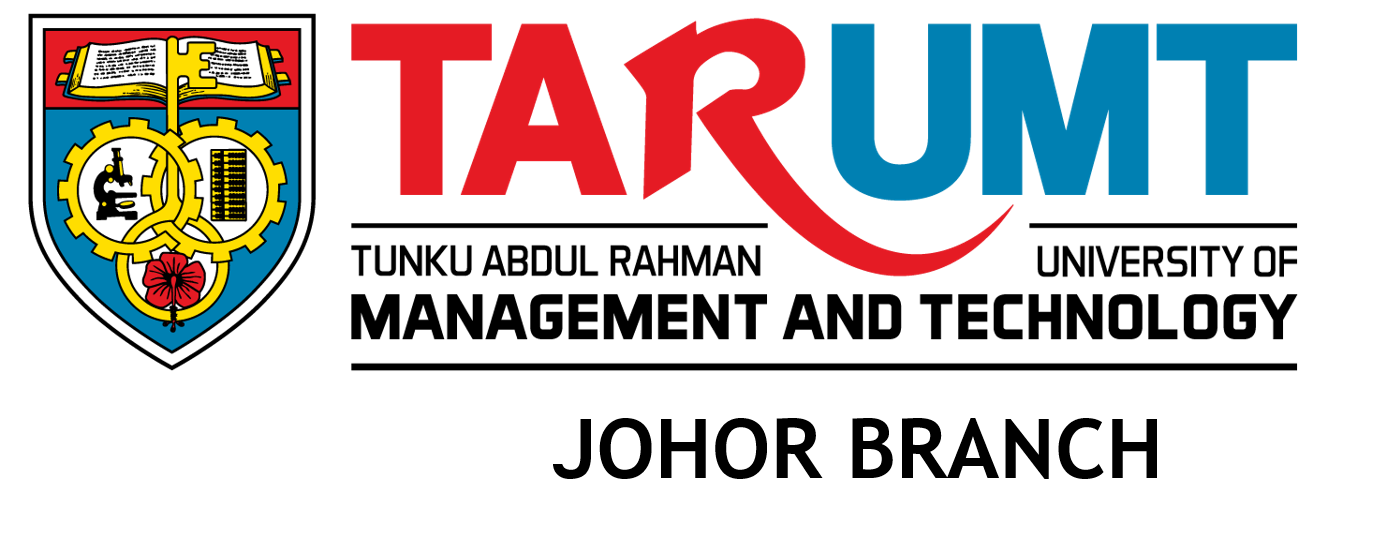
Individual Project Title

By

Wang Zi Zhen

****

FACULTY OF COMPUTING AND

INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

JOHOR BRANCH

ACADEMIC YEAR

**2024/25**

CalorieCare: Diet and Nutrition Management System

By

Wang Zi Zhen

Supervisor: Mr. Liow Thong Hwee

A project report submitted to the

Faculty of Computing and Information Technology

in partial fulfillment of the requirement for the

Bachelor of Information Technology (Honours)

**Department of Information and Communication Technology**

Faculty of Computing and Information Technology

Tunku Abdul Rahman University of Management and Technology

Johor Branch

Copyright by Tunku Abdul Rahman University of Management and Technology.

All rights reserved. No part of this project documentation may be reproduced, stored in retrieval system, or transmitted in any form or by any means without prior permission of Tunku Abdul Rahman University of Management and Technology.

# 

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University of Management and Technology rules and regulations and would be subjected to disciplinary actions.

\_\_\_\_\_\_\_\_\_\_Wang\_\_\_\_\_\_\_\_\_\_\_\_\_

Wang Zi Zhen

Bachelor of Information Technology (Honours) in Software Systems Development

ID:23JMR03485

Abstract

CalorieCare is a diet and nutrition management system developed to overcome common challenges found in traditional diet tracking applications. These challenges include the repetitive nature of manual food logging, negative emotional reactions caused by unmet intake targets, and irregular tracking behavior. The system promotes sustainable and healthy eating habits through intelligent features that improve convenience, motivation, and user engagement. To reduce the burden of manual data entry, CalorieCare incorporates AI-based image recognition capable of identifying food items and estimating calorie content. This functionality simplifies the meal logging process and supports more consistent dietary tracking. A dynamic calorie target adjustment mechanism is introduced to address emotional discouragement. When daily intake deviates from the goal, the system automatically adjusts the target for the following day or week to maintain nutritional balance. To encourage consistency in logging habits, a streak feature is implemented. Logging all three meals in a day earns a visual streak marker (e.g., 🔥), and consecutive daily entries build longer streaks. Missing a day resets the streak, reinforcing commitment. An additional Two-Person Supervision Mode allows two users to monitor each other’s logging progress. When both complete daily entries, a shared streak reward is granted. This feature enhances accountability and adds a cooperative motivation element. The system consists of six modules: User Management Module, Security Module, Smart Food Tracking Module, Dynamic Target Adjustment Module, Motivation Module, and Report Module. CalorieCare targets individuals who aim to maintain a healthy lifestyle by actively managing nutritional intake. The development process adopts the Incremental Software Model, with each increment delivering a functional module. Background research and interviews are used to gather system requirements, supported by overview and detailed use case diagrams to model system behavior and interactions.

Acknowledgement

I would like to express my deepest gratitude to my supervisor, Mr. Liow Thong Hwee, for his invaluable guidance, support, and encouragement throughout this project. His expertise and insightful feedback have been instrumental in shaping the direction and development of this work.

I would also like to extend my heartfelt appreciation to Tunku Abdul Rahman University of Management and Technology (TAR UMT) for providing the necessary resources and a conducive learning environment that made this project possible. The knowledge and experience gained during my time at TAR UMT have been invaluable to my academic and professional growth.

Lastly, I am sincerely grateful to my family for giving me the opportunity to pursue my education. Their unwavering support and sacrifices have enabled me to focus on my studies and reach this milestone.

Table of Contents

[Declaration ii](#_Toc196513510)

[Abstract iii](#_Toc196513511)

[Acknowledgement v](#_Toc196513512)

[1 Introduction 2](#_Toc196513513)

[1.1 Background 2](#_Toc196513514)

[1.2 Project Objective 3](#_Toc196513515)

[1.2.1 Research Question 4](#_Toc196513516)

[1.2.2 Research Objective 4](#_Toc196513517)

[1.3 Advantages & Contributions 4](#_Toc196513518)

[1.3.1 Advantages of CalorieCare 5](#_Toc196513519)

[1.3.2 Contributions of CalorieCare 5](#_Toc196513520)

[1.4 Project Plan 5](#_Toc196513521)

[1.4.1 Project Schedule 6](#_Toc196513522)

[1.4.2 Project Scope 7](#_Toc196513523)

[1.4.3 Module description 8](#_Toc196513524)

[1.5 Project Team & Organization 10](#_Toc196513525)

[1.6 Chapter Summary & Evaluation 11](#_Toc196513526)

[2 Literature Review 13](#_Toc196513527)

[2.1 Company Background 13](#_Toc196513528)

[2.2 Project Background 14](#_Toc196513529)

[2.3 Literature Review 15](#_Toc196513530)

[2.3.1 Existing Diet Tracking Applications 15](#_Toc196513531)

[2.3.2 Streak Function 20](#_Toc196513532)

[2.4 Feasibility Study 21](#_Toc196513533)

[2.4.1 Technical Feasibility 21](#_Toc196513534)

[2.4.2 Economic Feasibility 23](#_Toc196513535)

[2.4.3 Operational Feasibility 23](#_Toc196513536)

[2.4.4 Schedule Feasibility 24](#_Toc196513537)

[2.5 Development Environment 24](#_Toc196513538)

[2.5.1 Hardware Specifications 24](#_Toc196513539)

[2.5.2 Software and Development Tools 24](#_Toc196513540)

[2.5.3 Programming Language 25](#_Toc196513541)

[2.5.4 Database 25](#_Toc196513542)

[2.6 Chapter Summary & Evaluation 25](#_Toc196513543)

[3 Methodology and Requirements Analysis 27](#_Toc196513544)

[3.1 Software Process Methodology 27](#_Toc196513545)

[3.1.1 Phases of Incremental Model 28](#_Toc196513546)

[3.1.2 Advantages of Incremental Model 30](#_Toc196513547)

[3.2 Requirements Gathering Techniques 31](#_Toc196513548)

[3.2.1 Background Research 31](#_Toc196513549)

[3.2.2 Observation 32](#_Toc196513550)

[3.3 Requirement Recoding Techniques 32](#_Toc196513551)

[3.3.1 Overview Use Case Diagram 33](#_Toc196513552)

[3.3.2 Detailed Use Case Diagram 34](#_Toc196513553)

[3.4 Requirements Analysis 45](#_Toc196513554)

[3.4.1 Product Function 45](#_Toc196513555)

[3.4.2 Similar System Information 46](#_Toc196513556)

[3.4.3 User Problem Statement 47](#_Toc196513557)

[3.4.4 Users Objectives 47](#_Toc196513558)

[3.4.5 General Constraints 47](#_Toc196513559)

[3.4.6 External Interface Requirement 48](#_Toc196513560)

[3.5 Software Requirements Specification 48](#_Toc196513561)

[3.5.1 Functional Requirements 49](#_Toc196513562)

[3.5.2 Non-functional Requirements 53](#_Toc196513563)

[3.6 Chapter Summary & Evaluation 54](#_Toc196513564)

[4 System Design 56](#_Toc196513565)

[4.1 UI Design 56](#_Toc196513566)

[4.1.1 Register Module 57](#_Toc196513567)

[4.1.2 Set Target Module 58](#_Toc196513568)

[4.1.3 Calculate BMI Module 59](#_Toc196513569)

[4.1.4 Calculate Target Calorie Module 60](#_Toc196513570)

[4.1.5 Authenticate and Authorize User Module 61](#_Toc196513571)

[4.1.6 Recover Password Module 62](#_Toc196513572)

[4.1.7 Log Food Module 63](#_Toc196513573)

[4.1.8 Recognize Food Module 64](#_Toc196513574)

[4.1.9 Calculate Calorie Module 65](#_Toc196513575)

[4.1.10 Dynamic Target Adjustment Module 66](#_Toc196513576)

[4.1.11 Track Streak Module 67](#_Toc196513577)

[4.1.12 Invite Friend Module 68](#_Toc196513578)

[4.1.13 Alert Friend Module 69](#_Toc196513579)

[4.1.14 View Calorie Intake Module 70](#_Toc196513580)

[4.1.15 View Weight Report Module 71](#_Toc196513581)

[4.2 Data Design 72](#_Toc196513582)

[4.2.1 Entity Relationship Diagram (ERD) 73](#_Toc196513583)

[4.2.2 Data Dictionary 74](#_Toc196513584)

[4.3 Report Design 79](#_Toc196513585)

[4.3.1 Calorie Intake Report 79](#_Toc196513586)

[4.3.2 Weight Report 80](#_Toc196513587)

[4.4 Security Design 80](#_Toc196513588)

[4.4.1 Password Hashing 81](#_Toc196513589)

[4.4.2 Reset Password 81](#_Toc196513590)

[4.5 Process Design 81](#_Toc196513591)

[4.5.1 Activity Diagram 82](#_Toc196513592)

[4.6 Software Architecture Design 96](#_Toc196513593)

[4.7 Algorithm Design 97](#_Toc196513594)

[4.7.1 Recognize Food Function 97](#_Toc196513595)

[4.7.2 Dynamic Target Adjustment Function 98](#_Toc196513596)

[4.7.3 Track Streak Function 99](#_Toc196513597)

[4.8 Chapter Summary & Evaluation 100](#_Toc196513598)

[References 101](#_Toc196513599)

Chapter 1

Introduction

# Introduction

This chapter presents CalorieCare, a diet and nutrition management system created to overcome limitations commonly found in conventional diet tracking platforms. This section details the background and rationale behind the development, focusing on challenges such as the labor-intensive process of logging meals manually, demotivation caused by failure to meet nutritional targets, and irregular tracking behavior. Furthermore, the objectives, research questions, and major contributions of the system are clearly established. This chapter also outlines the project’s scope, essential modules, development timeline, and organizational framework, offering a complete overview of the system's strategic direction. This foundation supports subsequent research and development efforts.

## Background

Although the system of diet tracking enables the monitoring of calories and offers customized dietary advice, many issues still limit effectiveness in use. The main problem with personal diet tracking systems lies in the time-consuming nature of manual food logging, which is commonly perceived as a boring and repetitious chore. The process of documenting every meal or snack is often considered difficult, leading to disengagement. A lack of strong motivation to maintain long-term dietary tracking further contributes to the issue. Several studies have indicated that diet tracking systems are often abandoned due to the excessive time and effort required for manual meal logging, rendering the process tedious and demoralizing (Bruno et al., 2019; Chen et al., 2018). Continuous entry of meal data can lead to fatigue over time, eventually resulting in complete discontinuation of usage. Therefore, addressing the time-consuming aspect of diet tracking and enhancing engagement is essential for achieving sustained retention and success.

Abandonment of a diet tracking system can also result from consistently exceeding or failing to meet daily consumption goals. A methodical study highlights that "negative cognitive and emotional outcomes" interfere with the sustained implementation of nutritional interventions. Failure to achieve dietary goals often results in guilt, dissatisfaction, and anxiety, factors that hinder continued system interaction. In scenarios where specific intake targets are not met, the emotional pressure involved contributes to irregular usage patterns. These emotional responses can reduce the likelihood of following dietary recommendations over extended periods, thereby increasing the chances of discontinuing system usage (König et al., 2021).

One important element affecting the effectiveness of a diet tracking system is the absence of regular tracking practices. Studies have revealed that a lack of habit formation is one of the key reasons behind user desertion of different diet tracking systems. Many program users stop using the system in several tests simply because the user forgot to track the usual meal intake. Regarding long-term adherence to dietary recommendations, this lack of regular interaction with the system has unfavorable results (König et al., 2021).

In front of these challenges to improve long-term adherence, diet tracking systems must reduce the time required for meal entry, reduce emotional stress, and promote habit development. Maintaining user motivation and lowering system desertion relies on a simpler and user-friendly approach.

## Project Objective

CalorieCare is a diet and nutrition management system designed to solve the typical problems faced by users of traditional diet tracking apps, which are manually recording food consumption time, abandonment due to negative cognitive and emotional outcomes, and inconsistent tracking habits. The system improves user involvement through interactive and interesting features to inspire long-term commitment to good eating practices.

CalorieCare simplifies meal logging by reducing the effort required for manual data entry. Users can either input meals manually or use the system’s AI-powered image recognition, which automatically identifies dishes and estimates the calorie content. This intelligent functionality not only enhances user convenience but also encourages more consistent tracking by reducing the effort required for meal logging (Nadeem et al., 2020).

Either exceeding or under daily intake targets might cause users to give up the diet tracking system because of unpleasant emotional reactions, including guilt, disappointment, and worry. CalorieCare presents a dynamic daily calorie target adjustment feature to solve this problem. When the user’s daily intake goal is exceeded or falls short, the system automatically adjusts the next day’s calorie target to maintain balance. If intake is higher than the target, the following day’s target will be reduced, while if intake is lower, the following day’s target will be increased. The adjustment process respects the user’s goal type, whether weight loss, maintenance, or weight gain, while also adhering to safe health boundaries. Specifically, daily calorie intake never falls below the Basal Metabolic Rate (BMR) or 1,200 kcal for women and 1,500 kcal for men, and never exceeds Total Daily Energy Expenditure (TDEE) plus 500 kcal (Publishing, 2020; Larson-Meyer et al., 2022; NHS, 2021). This adaptive approach prevents discouragement and supports a sustainable and healthy way of dieting.

The streak feature is one key strategy to enhance user tracking habits to address the issue of inconsistent tracking habits. Inspired by Duolingo’s streak feature, which fosters intrinsic motivation rather than relying on external rewards (Bastiaan et al., 2024), the system will focus on personal satisfaction and accountability. By allowing users to track the progress and feel a sense of control over consistency, the streak feature promotes a sense of accomplishment, making meal tracking more engaging. Users who log all three meals in a day will receive a "🔥" streak marker, with continuous daily logging stacking additional "🔥" symbols (e.g., "3-Day Streak! 🔥🔥🔥"). However, missing a day will reset the streak, encouraging users to maintain momentum. Additionally, a Two-Person Supervision Mode will enable users to invite a friend or family member as a supervisor, allowing users to check each other’s meal logging status. If one forgets, the other can send a custom reminder message, such as "@User, you haven't logged your meals today! 🔥 Don't keep me waiting!". To further reinforce accountability, if both users complete meal logs for the day, the users will earn a shared "🔥" reward. However, if one user fails to log food, the other can send reminders, and once the missing log is completed, the streak reward will be granted. By integrating these features, the system fosters sustainable tracking habits, enhances personal satisfaction, and ultimately promotes long-term adherence to a healthy diet.

According to the proposed features above, the research questions and research objectives of this project can be summarized as follows:

### Research Question

RQ1: How to reduce the process of users manually logging meals？

RQ2: How can diet tracking apps help users meet dietary goals?

RQ3: How to remind users to continuously log meal information and long-term engagement in diet tracking apps?

### Research Objective

RO1: Implement image recognition technology for automatic meal identification and calorie estimation.

RO2: Implementing dynamic calorie target adjustment features to let users meet dietary goals.

RO3: Apply a streak function with streak-based incentives to reinforce meal logging habits and sustain user engagement.

## Advantages & Contributions

This section highlights the key advantages and unique contributions of CalorieCare in promoting healthier eating habits.

### Advantages of CalorieCare

CalorieCare tracks food consumption, so this system goes beyond just offering calorie tracking. The system improves odds of reaching dietary targets and aids in weight loss (Rusin et al., 2013). CalorieCare incorporates image recognition, allowing users to log meals simply by taking a photo. The system automatically identifies food items, estimates food calorie and nutrient content. This feature not only saves time but also reduces the likelihood of users abandoning the system due to tedious data entry. CalorieCare also offers dynamic goal changes that help users avoid negative emotional reactions, including guilt and discouragement, by preventing users from getting annoyed with surpassing daily intake restrictions. Two-person supervision and a streak-based tracking system allow users to be consistent in meal recording by inspiring responsibility. CalorieCare can tremendously inspire the end user.

### Contributions of CalorieCare

CalorieCare is designed for health-conscious people who maintain a good lifestyle by closely tracking meals and nutritional intake. The system enhances long-term engagement by addressing common challenges, such as inconsistent tracking habits and loss of motivation. With image recognition, users can effortlessly log meals, reducing the effort required for manual entry. Users are urged to create sustainable tracking habits, stay accountable, and remain dedicated to the user dietary objectives by means of streak-based motivation, dynamic goal modifications, and a Two-Person Supervisor Mode. Dynamic calorie target adjustment helps reduce stress from daily intake by automatically adapting intake goals based on eating patterns. This approach promotes flexibility and balance, making healthy eating more sustainable and accessible.

## Project Plan

This section outlines the overall timeline and scope of CalorieCare. The project schedule details key milestones and submission deadlines, while the project scope defines the system boundaries and major functionalities to be developed.

### Project Schedule

This section outlines the scheduled submission dates for the Proposal and each chapter of Project I and Project II.

Table 1.1: Project Schedule of CalorieCare

|  |  |  |
| --- | --- | --- |
| **ACTIVITIES** | **EXPECTED OUTCOME** | **COMPLETION DATE** |
| Approaching a supervisor | The supervisor was approached. | 19/11/2024 -  6/12/2024 |
| Submit Project Proposal | The Project Proposal was completed and submitted. | 03/01/2025 |
| Submit Chapter 1 Introduction | Chapter 1 Introduction was completed and submitted. | 21/02/2025 |
| Submit Chapter 2 Research Background | Chapter 2 Research Background was completed and submitted. | 6/03/2025 |
| Submit Chapter 3 Methodology and Requirements Analysis | Chapter 3 Methodology and Requirements Analysis was completed and submitted. | 28/03/2025 |
| Submit Chapter 4 System Design | Chapter 4 System Design was completed and submitted. | 11/04/2025 |
| Submit Project I Portfolio | All the updated chapters (1 to 4) with a  proper cover page was combined as Project I Portfolio and submitted. | 25 /04/2025 |
| Preparation of test plan/cases or experiment plan  System Preview with Supervisor | Test plan/cases were prepared and the system was previewed with supervisor. | 11/07/2025 |
| Final System Testing with Supervisor and Moderator | Final System was tested with Supervisor and Moderator. | 25/07/2025 |
| Submission of Draft FYP Report | Draft FYP Report was completed and submitted. | 22/08/2025 |
| Submission of Final FYP Report and all associated  deliverables (digital submission, source code, etc). | Final FYP Report and all associated deliverables (digital submission, source code, etc) were organised and submitted. | 3/09/2025 |

### Project Scope

CalorieCare comprises six main modules, each encompassing several sub-modules to support core functionalities. The system hierarchy chart, as illustrated in Figure 1.1, provides a visual representation of the project’s modular structure.



Figure 1.1: System Hierarchy Chart of CalorieCare

### Module description

#### User Management Module

The User Management Module is responsible for handling user-related operations, including registration, BMI calculation, and personalized target setting.

* **Register User Sub-module**

Allows new users to create an account by providing the necessary details such as name, email, and password. The system validates the input, ensures data security, and stores the information in the database.

* **Set Target Sub-module**

Enables users to set health goals, such as weight loss, maintenance, or muscle gain. Based on selected preferences, the system customizes the target nutritional intake.

* **Calculate BMI Sub-module**

Computes the user's Body Mass Index (BMI) using height and weight data. The BMI value is then used to determine the user's health status and provide appropriate dietary recommendations.

* **Calculate Target Calorie Sub-module**

Based on the user's BMI, activity level, and health goal, this sub-module calculates the daily calorie intake target to maintain a balanced diet.

#### Security Module

The Security Module ensures that user data is protected and access to the system is securely managed.

* **Authenticate User Sub-module**

This sub-module verifies user credentials during login, ensuring that only registered users can access accounts.

* **Authorize User Sub-module**

Manages access control by ensuring users can only perform actions permitted based on roles and permissions.

* **Recover Password Sub-module**

Allows users to reset passwords securely via email verification if users forget login credentials.

#### Smart Food Tracking Module

The Smart Food Tracking Module helps users log meals efficiently and analyze calorie intake using AI-powered food recognition.

* **Log Food Sub-module**

Users can manually enter meals. The system records the food intake.

* **Recognize Food Sub-module**

Uses AI-powered image recognition and OCR technology to identify food items from photos or scanned receipts. The system extracts nutritional values automatically.

* **Calculate Calorie Sub-module**

Computes the total calorie intake based on the logged food data, ensuring users stay within users target range.

#### Dynamic Target Adjustment Module

The Dynamic Target Adjustment Module automatically adapts user nutrition goals based on real-time dietary intake and progress.

* **Check Daily Intake Sub-Module**

Monitors user's daily food consumption and compares the data with target calorie and nutrient goals.

* **Compare Target Intake Sub-Module**

Analyzes whether the user’s actual intake aligns with recommended targets.

* **Auto Adjust Target Sub-Module**

If a user consistently exceeds or fails to meet daily intake goal, the system dynamically adjusts the next day's calorie target to restore balance. If this pattern continues for multiple days, the system gradually modifies the weekly calorie target to ensure a sustainable long-term approach to dieting.

#### Motivation Module

The Motivation Module encourages users to stay consistent with meal logging and health goals.

* **Track Streak Sub-module**

Keeps track of consecutive days the user logs meals, rewarding users with streak badges.

* **Invite Friend Sub-module**

Allows users to connect with friends and family for accountability and encouragement.

* **Alert Friend Sub-module**

Sends reminders or alerts to friends if a user forgets to log meals, promoting team motivation.

#### Report Module

The Report Module provides users with detailed insights into dietary habits and progress over time.

* **View Calorie Intake Sub-Module**

Displays a daily summary of consumed calories, helping users track nutritional balance.

* **View Weight Report Sub-Module**

Generates weight trend reports based on user input, allowing users to monitor progress toward the user's health goals.

## Project Team & Organization

This project is undertaken as an individual effort, allowing for complete autonomy in decision-making and implementation. Working independently ensures full control over direction, development pace, and vision without the need for compromises. This approach enhances problem-solving skills and self-discipline, as all aspects from planning and design to development and testing are managed by a single developer.

Additionally, an individual project fosters a deeper understanding of technologies and methodologies, as no tasks are divided. This comprehensive hands-on experience strengthens technical expertise and project management skills. While teamwork offers advantages, working alone eliminates potential conflicts and dependencies, enabling more efficient execution and a personalized approach to problem-solving.

## Chapter Summary & Evaluation

This chapter introduces the CalorieCare, addressing key limitations of traditional diet tracking systems, such as manual logging inefficiencies, emotional discouragement, and inconsistent user habits. To solve these issues, CalorieCare proposes AI-powered meal recognition, dynamic calorie target adjustments, and a streak-based motivation system to simplify tracking, reduce frustration, and encourage long-term adherence through social accountability and streak-based rewards. This chapter outlines clear objectives, advantages, and a structured development plan, positioning CalorieCare as a user-centric solution for long-term dietary management.

Chapter 2

Literature Review

# Literature Review

This section presents a review of online academic literature to identify proven solutions for challenges in diet and nutrition management, with the aim of integrating these approaches as functional components in CalorieCare. Existing systems for diet tracking and nutrition analysis are also examined, focusing on key features that demonstrate practical value and highlighting areas with potential for improvement. Through an evaluation of current systems’ strengths and limitations, this review offers insights into enhancing the functionality of CalorieCare to support a more effective, personalized, and engaging dietary management experience.

## Company Background

CalorieCare Tech is a technology-driven company focused on developing innovative solutions for diet and nutrition management. Based in Kuala Lumpur, operations run during standard business hours from 9:00 AM to 6:00 PM, Monday to Friday. With a commitment to improving dietary habits through AI-powered solutions, CalorieCare Tech strives to make nutrition tracking more accessible, engaging, and effective.

The vision is to revolutionize digital health by providing intelligent, user-friendly tools that promote long-term wellness and sustainable dietary habits. The mission is to develop advanced, AI-driven diet management systems that simplify meal tracking, enhance motivation, and encourage consistency in healthy eating.

To achieve these goals, CalorieCare Tech is dedicated to continuous research and innovation, ensuring products address real user challenges. Objectives include improving diet tracking accuracy, enhancing engagement through personalized AI recommendations, and integrating social and gamification features to boost long-term adherence. With a strong focus on user experience and scientifically backed methodologies, CalorieCare Tech aims to become a leader in digital nutrition management solutions.

## Project Background

CalorieCare Tech has previously developed similar diet tracking systems but encountered difficulties in attracting and retaining users. After conducting research, key problems preventing long-term engagement were identified.

One of the primary challenges in diet tracking systems is the tedious nature of manual meal logging. Many users found the process repetitive and time-consuming, leading to frustration and eventual disengagement. Studies have shown that the effort required for manual input discourages consistent tracking, resulting in high abandonment rates (Bruno et al., 2019; Chen et al., 2018). Over time, fatigue from constantly entering meal data increases the likelihood of discontinuing use.

To reduce the effort required for meal logging, integration of AI-powered image recognition and barcode scanning shall be implemented in CalorieCare. This functionality enables photographic input of meals, allowing automatic identification of dishes and estimation of nutritional content. By minimizing manual data entry, the solution enhances convenience and promotes consistent system usage.

Additionally, negative emotions such as guilt, disappointment, or anxiety often arise when daily intake targets are exceeded or not met. Such emotions discourage continued diet tracking and contribute to system abandonment (König et al., 2021). The pressure to strictly adhere to dietary goals can result in emotional stress, making long-term engagement difficult.

To address the issue of discouragement caused by unmet calorie goals, CalorieCare introduces a dynamic daily calorie target adjustment feature. When intake exceeds or falls short of the daily calorie goal, the system automatically adjusts the next day’s target to maintain balance. This prevents negative emotions such as guilt or anxiety and promotes a more adaptable and stress-free tracking experience. According to health guidelines, reducing daily calorie intake by approximately 500 kcal can support weight loss of around one pound per week, but intake should not fall below 1,200 kcal/day for women and 1,500 kcal/day for men (Publishing, 2020). For individuals aiming to gain weight, gradual increases of about 300–500 kcal/day are recommended to ensure healthy progression without excessive fat gain (Larson-Meyer et al., 2022; NHS, 2021). These scientifically supported boundaries are integrated into CalorieCare’s adjustment mechanism to ensure user safety while promoting steady and sustainable progress.

Another major reason for abandoning diet tracking systems is the lack of habit formation. Many users stop tracking meals simply due to forgetfulness, leading to irregular engagement and eventual discontinuation (König et al., 2021). Without consistency, integrating meal tracking into daily routines becomes challenging, reducing long-term effectiveness.

To encourage consistent usage, CalorieCare shall implement a streak tracking feature, inspired by Duolingo’s approach to habit reinforcement (Bastiaan et al., 2024). Users who log all three meals in a day shall receive a "🔥" streak marker, with additional streaks stacking for consecutive daily logging (e.g., "3-Day Streak! 🔥🔥🔥"). If a day is missed, the streak shall reset, providing motivation for regular tracking. Additionally, a Two-Person Supervision Mode shall allow users to invite a friend or family member as a supervisor. The supervisor shall be able to check meal logging status and send personalized reminders, such as "@User, you haven't logged meals today! 🔥 Don’t keep me waiting!" If both complete meal logs for the day, a shared "🔥" streak reward shall be earned. By reinforcing accountability and fostering a sense of accomplishment, this feature shall help establish long-term tracking habits.

Through these targeted solutions, CalorieCare shall enhance engagement, simplify the meal logging process, and promote sustainable dietary tracking habits. These improvements shall ensure long-term retention and maximize effectiveness in supporting healthy eating behaviours.

## Literature Review

This section reviews existing diet and nutrition tracking applications and relevant academic studies to identify effective solutions that address user engagement and tracking consistency. By analysing the key features, strengths, and limitations of three popular systems, Calorie Mama, FatSecret, and Track this section highlights useful elements for adoption and areas for improvement in CalorieCare. In addition, this section explores academic findings on the effectiveness of streak-based motivation systems in promoting long-term habit formation. Through this evaluation, valuable insights are gathered to guide the integration of AI-powered recognition, social accountability, and streak features into CalorieCare, with the aim of building a more user-centric, engaging, and sustainable diet management experience.

### Existing Diet Tracking Applications

Diet tracking applications play a crucial role in helping users maintain healthy eating habits. However, existing applications have limitations in terms of streak-based motivation and AI-powered meal recognition, which can affect user engagement and convenience. Unlike existing applications, CalorieCare integrates both AI-powered image recognition and a streak function, addressing the two major challenges faced by users effort-intensive meal logging and inconsistent tracking habits.

#### Calorie Mama

Calorie Mama is an AI-powered diet tracking application that simplifies calorie counting by allowing food intake logging through image recognition (Calorie Mama, 2017; Calorie Mama, 2021). Instead of manually entering meal details, a photo of the food can be taken, and Calorie Mama’s AI-driven system analyzes the image to estimate calorie content (Calorie Mama, 2021). Designed to make food tracking simple, fun, and efficient, deep learning algorithms and computer vision are leveraged to recognize a wide variety of food items (Calorie Mama, 2017; Calorie Mama, 2021). Various categories can be identified, including fruits, vegetables, meats, grains, and beverages, as well as complex dishes from different cuisines such as American, Western, European, Asian, and Latin American (Calorie Mama, 2021). Barcode scanning is also supported for logging packaged foods, particularly those available in the US and Canada (Calorie Mama, 2021). Since the system release on March 10, 2017, over 100,000 downloads have been recorded (Calorie Mama, 2017). Key features of Calorie Mama include:

* Food Diary
* Food Image Recognition
* Automatic Calorie Estimation
* Barcode Scanning
* Customizable Nutrition Goals
* Fitness & Health App Integration

Although Calorie Mama offers many useful features, Calorie Mama lacks features that encourage long-term user engagement, such as a streak-based motivation system. As shown in Figure 2.1, while the AI-powered food detection reduces the burden of manual logging, the app does not include mechanisms to reinforce habit formation or maintain user consistency. The absence of a streak feature means that users do not receive incentives for continuous tracking, which can lead to disengagement over time. Many users may struggle to maintain the habit of regular tracking, which ultimately reduces the long-term effectiveness of the app.

Compared to Calorie Mama, CalorieCare should have integrate both AI-powered image recognition and a streak feature to enhance user motivation. By incorporating a streak-based engagement system, users can receive visual rewards for consistent meal tracking, making the process more engaging. Additionally, the Two-Person Supervision Mode can allow users to track each other’s progress and send reminders, further reinforcing accountability. These enhancements can address the gaps in existing diet tracking applications, ensuring that users not only find food logging convenient but also stay motivated to maintain long-term healthy eating habits.

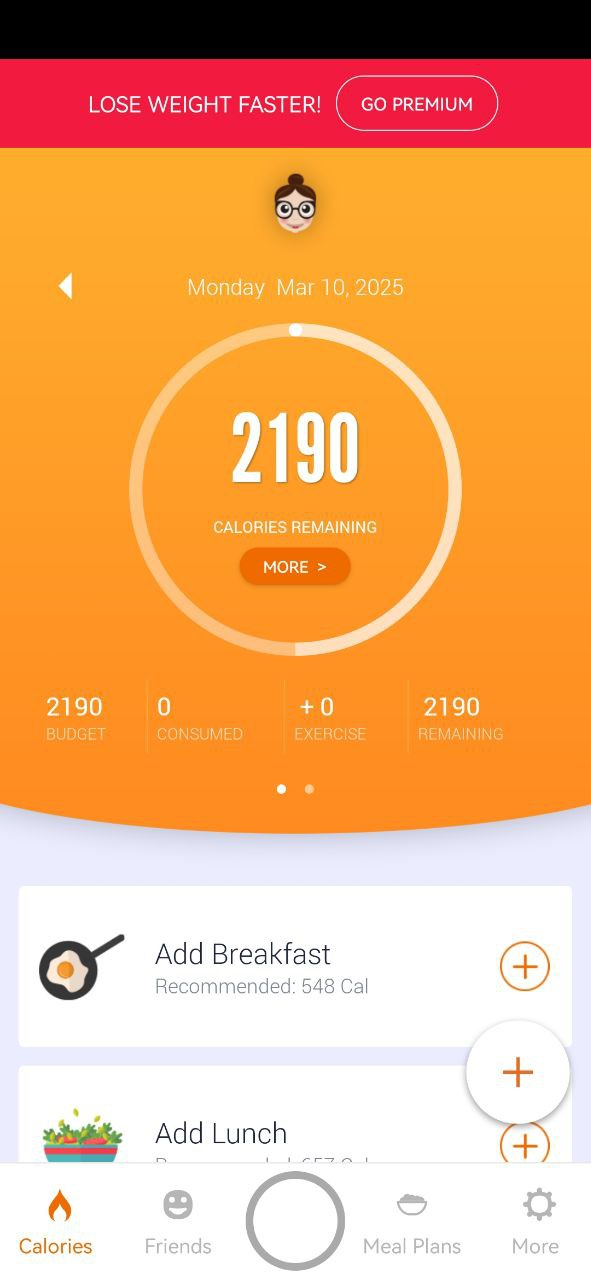
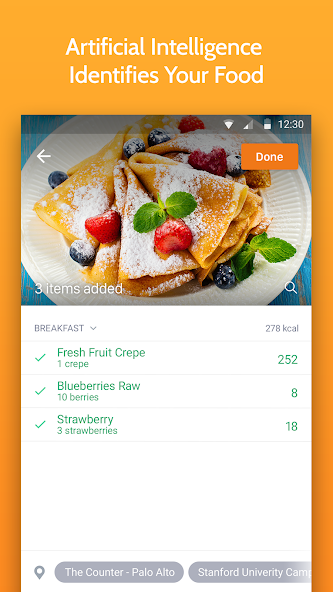
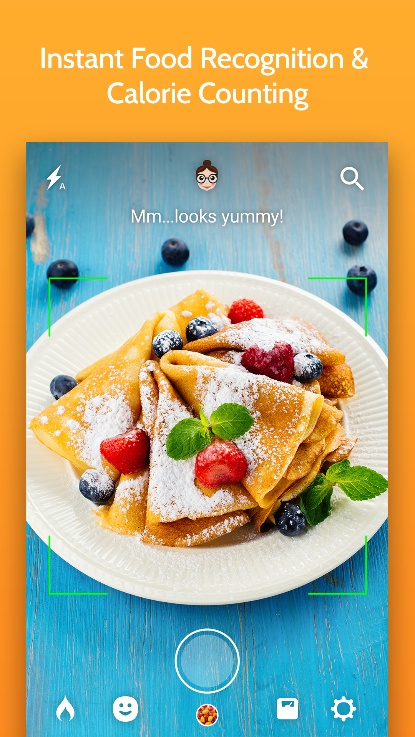


Figure 2.1: Calorie Mama App

#### FatSecret

FatSecret is a popular diet tracking application designed to help users monitor calorie intake, exercise, and weight loss progress. A comprehensive food diary is provided, allowing users to log meals and track nutritional intake using a vast food database (FatSecret, 2019; FatSecret, 2021). Additionally, the app integrates with external tools such as Google Fit, Samsung Health, and Fitbit to enhance activity tracking (FatSecret, 2021). FatSecret offers a barcode scanner for quick food logging and a meal planner to assist users in maintaining a structured diet (FatSecret, 2019; FatSecret, 2021). The system also includes a supportive community where users can share progress, seek motivation, and receive dietary advice (FatSecret, 2021). With over 50 million downloads, FatSecret remains one of the most popular diet tracking applications (FatSecret, 2021). Key features of FatSecret are listed as below:

* Food Diary
* Barcode Scanner
* Exercise & Weight Tracking
* Meal Planning & Reporting
* Community Support
* Integration with Health Apps

Although FatSecret offers many useful features, FatSecret lacks AI-powered image recognition for automatic meal logging. As shown in Figure 2.2, while FatSecret allows users to take food photos, the system does not include true image recognition. Instead of automatically identifying food items from images, users must manually input meal details after capturing a picture. This limitation means that meal logging still requires significant user effort, which may lead to inconsistent tracking over time. Without automated food recognition, users may find the process tedious, reducing the likelihood of sustained usage.

Compared to FatSecret, CalorieCare should have integrate both a streak function and AI-powered image recognition to enhance user experience and engagement. By allowing users to log meals simply by taking a photo, CalorieCare can minimize the effort required for manual data entry, making diet tracking more convenient. At the same time, streak-based motivation system can encourage consistent usage by rewarding users for maintaining daily meal logs. Additionally, the Two-Person Supervision Mode can further reinforce accountability by enabling users to track each other’s progress and send custom reminders. These enhancements can ensure that CalorieCare provides both ease of use and long-term engagement, addressing the limitations of existing diet tracking applications.

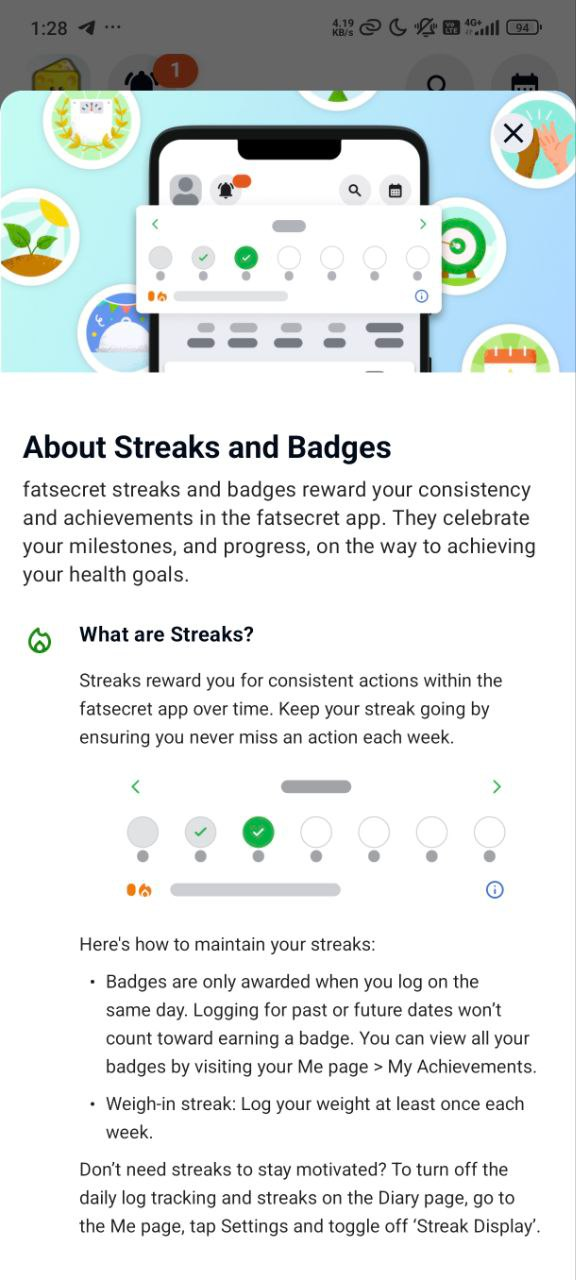
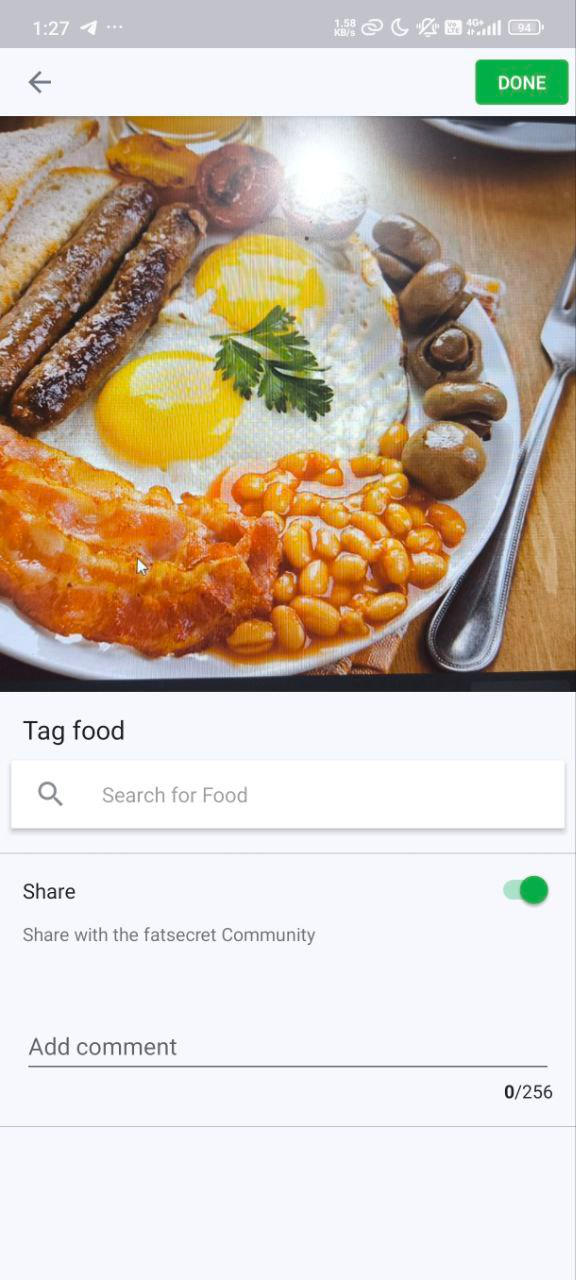


Figure 2.2: FatSecret App

#### Track

Track is a diet tracking application developed by Nutritionix, designed to help users monitor nutritional intake with a focus on accuracy and efficiency (Nutritionix, 2021). Unlike other diet tracking apps, Track is maintained by a team of registered dietitians and provides access to a comprehensive food database with over 800,000 unique food items, including 95% of grocery items in the US and Canada and 760+ US restaurant chain menus (Nutritionix, 2021). The app allows users to log food intake, track calories, macros, and nutrient totals, and monitor exercise and weight progress (Nutritionix, 2021). Track also supports barcode scanning and natural language processing (NLP) for faster food entry (Nutritionix, 2021). As of now, Track has over 1 million downloads since the system release in 2016, making the system a well-established option for users seeking detailed nutritional insights (Nutritionix, 2021). Key features of Track are listed as below:

* Food Diary
* Natural Language Processing
* Barcode Scanning
* Custom Recipe & Food Creation

As shown in Figure 2.3, one of the primary limitations of Track is the absence of both AI-powered image recognition and a streak-based motivation system. Unlike other applications that use computer vision to identify food items from photos, Track requires users to manually enter all meal details, which can be time-consuming and discouraging for long-term adherence. While the app excels in providing detailed nutritional breakdowns, the lack of a motivation system means that users do not receive reinforcement for maintaining a consistent tracking habit. Without engaging features such as streaks or reminders, users may lose interest over time, leading to a decline in adherence.

Compared to Track, CalorieCare should integrates both AI-powered image recognition and a streak-based motivation system to improve usability and user engagement. With AI recognition, users can log meals effortlessly by taking a photo, significantly reducing the manual effort required for food entry. This streamlines the tracking process and enhances accuracy. In addition, the streak feature rewards users for consistent tracking behaviour, encouraging habit formation. The Two-Person Supervision Mode further strengthens accountability by allowing users to monitor each other’s progress and provide support through reminders. These features address Track’s limitations by offering both convenience and long-term motivational support, making CalorieCare a more user-centric solution for diet tracking.

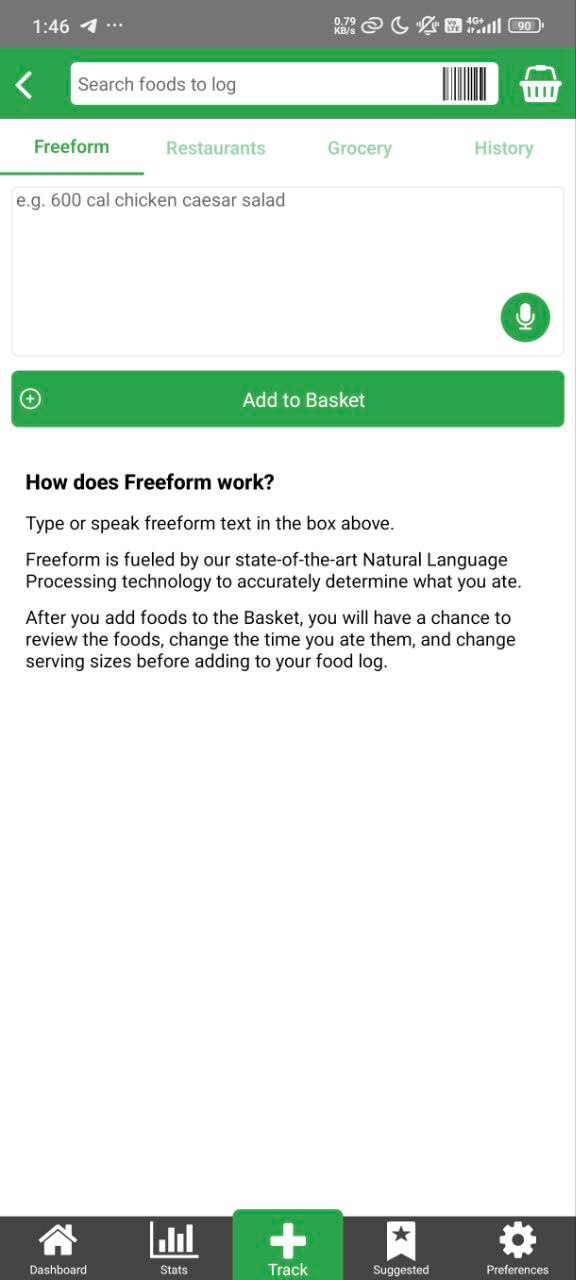
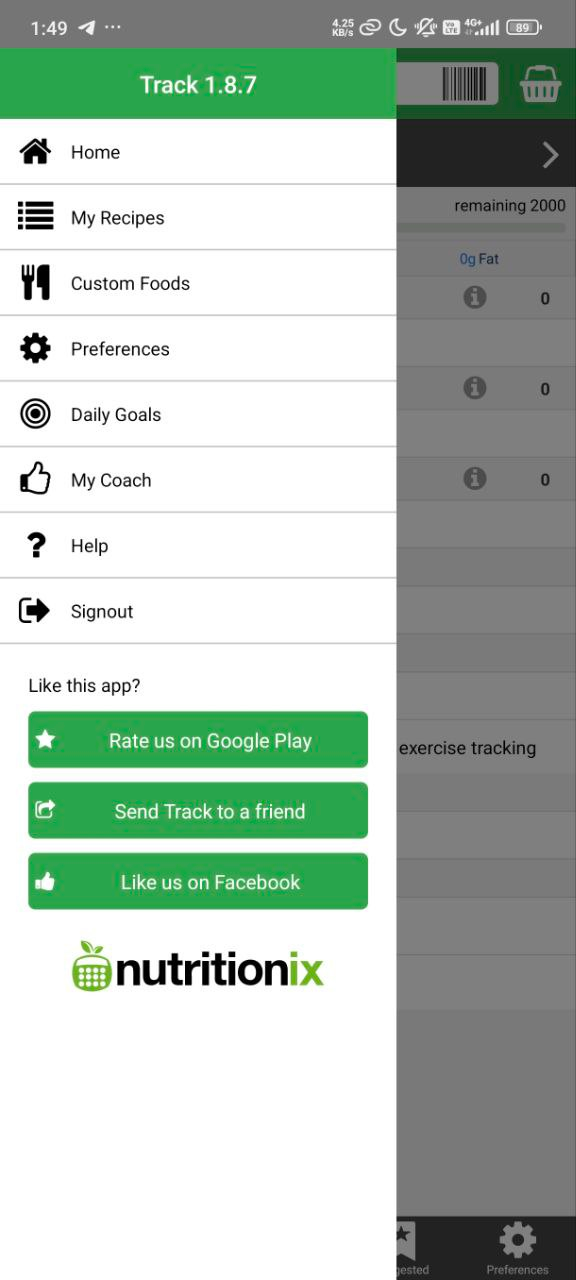
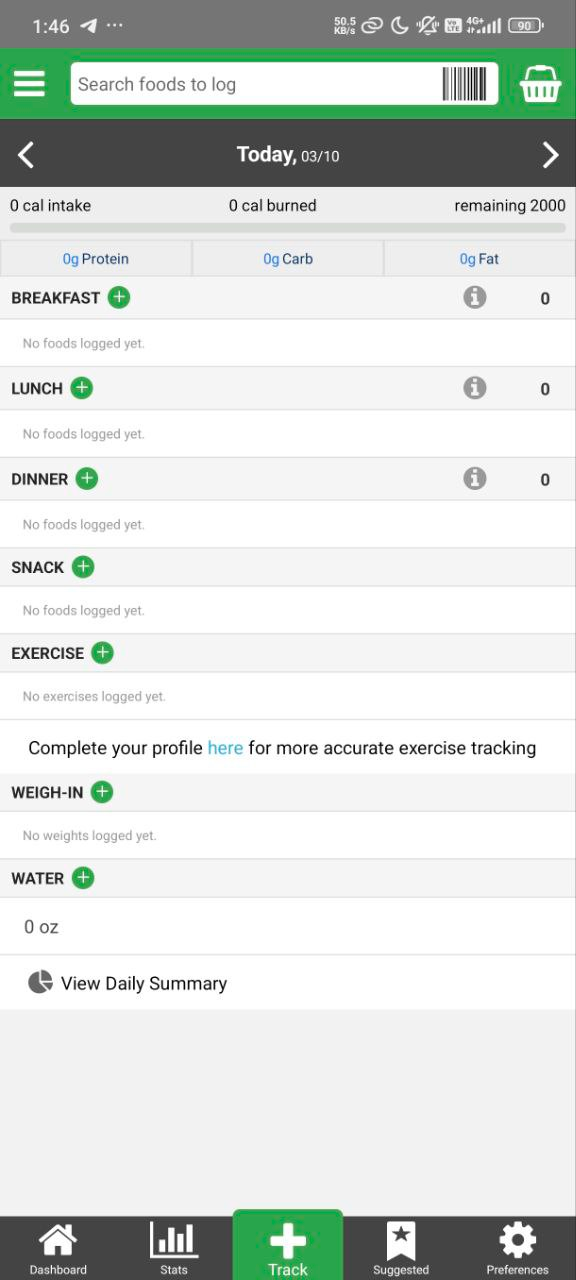


Figure 2.3: Track App

### Streak Function

The streak function has been widely studied in game-based learning and motivational psychology, particularly in educational applications such as Duolingo. Streak function is a mechanism that encourages habit formation by rewarding users for consistent engagement over time.

The term ‘Winning Streak’ was initially used in sports, referring to a consecutive number of games won, beginning with the third consecutive victory (Huynh et al., 2018). In Duolingo, a Winning Streak serves as a measurement of user consistency, increasing by one each day a learner completes the required lessons. If a user misses a lesson, the streak resets to zero (Huynh et al., 2018). This mechanic introduces a sense of progression and commitment, motivating users to maintain learning habit.

Research has demonstrated that streak-based mechanics significantly influence user motivation and engagement. Studies on game refinement theory indicate that streaking-users are more attracted to the application compared to normal users (HUYNH & IIDA, 2017). Moreover, streak function effectiveness increases over time, the longer the streak, the more valuable it becomes, as users do not want to lose accumulated progress (Huynh et al., 2018).

Furthermore, streak mechanics are known to foster intrinsic motivation, which is more sustainable than external rewards. Unlike traditional extrinsic incentives (e.g., points or monetary rewards), streaks focus on personal satisfaction and accountability, reinforcing consistent behavior (Bastiaan et al., 2024).

To maintain user engagement, applications employing streak functions often integrate reminders and social sharing features. Research by Freeman et al. (2023) highlights that streaks become a badge of pride for users, particularly in language and math learning applications. When users are aware of streak length, users exhibit a greater desire to maintain the streak length. Additionally, notifications play a crucial role in ensuring streak continuity by emphasizing salience and urgency, making the practice feel noteworthy and important (Freeman et al., 2023).

The streak function can be effectively integrated into CalorieCare to encourage long-term adherence to healthy eating habits. By leveraging streak-based motivation, users can stay engaged with dietary goals, much like how language learners are motivated to continue daily lessons in Duolingo. Features such as team streaks and supervised streak tracking can further enhance social accountability, ensuring users remain committed to nutrition plans.

The streak function is a proven motivational tool in educational and habit-forming applications. Research confirms that streak length positively correlates with user engagement and habit retention (HUYNH & IIDA, 2017; Huynh et al., 2018). Additionally, leveraging intrinsic motivation rather than external rewards enhances long-term adherence (Bastiaan et al., 2024). By implementing streak-based motivation, CalorieCare can significantly increase user retention and consistency, making a valuable component in habit-building systems.

## Feasibility Study

This section evaluates the feasibility of developing CalorieCare by analysing technical, economic, operational, and schedule aspects. The goal is to ensure the system can be successfully developed, implemented, and maintained within the available resources, time frame, and technical scope.

### Technical Feasibility

CalorieCare will be developed using Flutter (Flutter, 2025a) as the main development framework and Dart as the programming language. Development will be conducted using Android Studio (Android Studio, 2019b), an industry-standard Integrated Development Environment (IDE) that fully supports Flutter development.

The software tools used are all open-source and have extensive documentation and community support, which ensures effective troubleshooting and learning. This project will be developed on a personal Windows machine that meets all the required hardware and software specifications.

Table 2.1: Software Requirements

|  |  |
| --- | --- |
| **Software Tool** | **Requirements** |
| Android Studio | Version 2024.1.1 (Koala) or later, installed with Android SDK & emulator |
| Flutter | Requires Git, PowerShell 5+, and compatible with Windows 10 (64-bit) or later |
| Dart | Included with Flutter SDK |
| Git for Windows | Version 2.27 or later |

Table 2.1: Hardware Requirements

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirements** | **Recommended Requirements** |
| Operating System | 64-bit Windows 8 | Latest 64-bit version of Windows |
| RAM | 8 GB | 16 GB or more |
| Processor (CPU) | x86\_64 architecture, 2nd-gen Intel Core or newer | Latest Intel Core processor |
| Storage | 8 GB (for IDE & SDK) | SSD with 16 GB or more |
| Display Resolution | 1280 x 800 | 1920 x 1080 (Full HD) |
| Flutter Requirements | 4 CPU cores, 8 GB RAM, 11 GB disk space | 8 CPU cores, 16 GB RAM, 60 GB storage |

The personal laptop used for development meets and exceeds all the technical requirements. It runs on Windows 11 and is equipped with a 13th Gen Intel® Core™ i7-13620H processor (2.40 GHz, 10 cores, 16 logical processors), 16 GB of RAM, SSD storage of over 60 GB, and a Full HD display. The laptop also supports Wi-Fi network connectivity, which ensures smooth access to online documentation and tools during development. All necessary software tools, including Android Studio, Flutter, Dart, Git for Windows, and PowerShell, have been successfully installed and properly configured to support the development process.

From a skills perspective, the developer is proficient in Dart and has prior experience with Flutter and Android Studio through coursework and project assignments. Tutorials, documentation, and community forums provide ample resources to support development and address technical challenges efficiently.

In conclusion, since the hardware, software, and technical skills required for CalorieCare are all readily available and supported, the project is considered technically feasible.

### Economic Feasibility

The development of CalorieCare incurs minimal costs. The core technologies used, including Android Studio and Flutter, are free to download and use, eliminating the need for software licensing expenses (Android Studio, 2025; Flutter, 2025). Additionally, the system integrates the LogMeal API, which offers free access for a limited period (LogMeal, 2025), allowing for cost-free implementation during the development and testing phases.

Furthermore, the development process requires a computer for coding and testing; however, since the developer already owns a suitable computer, there is no additional hardware cost involved. This further reduces the financial burden and ensures that the project remains within the budget constraints of an FYP.

Since no paid infrastructure or services are required, the project can be completed without a financial burden, make a viable and cost-effective option.

### Operational Feasibility

This project aligns with the required technical and academic standards. This project involves developing a mobile application, and the developer possesses the necessary knowledge and skills in mobile app development, particularly using Flutter and Dart, ensuring smooth implementation.

Additionally, this project title has been approved by the university, confirming this project meets academic requirements and is suitable for the Final Year Project (FYP). Throughout the development phase, this project progresses smoothly under the supervision of an assigned supervisor, who provides guidance and ensures that the development stays on track.

With the necessary expertise, institutional approval, and continuous supervision, this project is deemed operationally feasible and can be successfully implemented.

### Schedule Feasibility

This project is progressing according to the scheduled dates outlined in the Milestone, ensuring that each stage aligns with the planned timeline and milestones. With a well-structured timetable, all tasks are systematically organized to facilitate smooth development. As a result, this project can be designed, developed, and implemented within the allotted time frame, ensuring timely completion without delays. Therefore, the schedule feasibility is feasible.

## Development Environment

The development environment for this project consists of both hardware and software components, ensuring efficient development and testing of the application.

### Hardware Specifications

The application is developed on a Windows 11 machine with the following hardware specifications:

Operating System (OS): Windows 11

Processor (CPU): 13th Gen Intel® Core™ i7-13620H, 2.40 GHz, 10 Cores, 16 Logical Processors

Memory (RAM): 16 GB

Network Connectivity: Wi-Fi

These specifications provide sufficient processing power and memory for running Android Studio and Flutter, ensuring smooth application development and testing.

### Software and Development Tools

The primary software and development tools used in this project include:

Android Studio: The official Integrated Development Environment (IDE) for Android development, used for writing, debugging, and testing the application.

Flutter SDK: A UI toolkit for building cross-platform applications with a single codebase.

### Programming Language

The application is developed using Dart, the programming language specifically designed for Flutter. Dart offers:

Cross-platform compatibility, enabling deployment on both Android and iOS.

Fast development cycles with hot reload for instant UI updates.

Null safety features, reducing runtime errors and improving code reliability.

### Database

The application requires a database:

Firebase Firestore: A cloud-based NoSQL database for real-time data synchronization.

## Chapter Summary & Evaluation

This chapter provided an in-depth analysis of the Diet and Nutrition Management System developed by CalorieCare Tech. The discussion began with an overview of the company background, highlighting vision, mission, and objectives. CalorieCare aims to revolutionize digital health by integrating AI-powered solutions to simplify diet tracking and enhance user engagement.

This project background explored the challenges faced by traditional diet tracking systems, including manual meal logging fatigue, emotional stress from rigid calorie goals, and lack of habit formation mechanisms. To address these issues, the system integrates AI-powered image recognition, barcode scanning, dynamic calorie target adjustment, and streak-based motivation features.

A comprehensive literature review examined existing diet tracking applications, such as Calorie Mama, FatSecret, and Track, identifying limitations in AI automation and long-term engagement. The review also discussed the streak function as a proven method to reinforce user habits, drawing insights from game-based learning and psychological research.

The feasibility study assessed the project’s technical, economic, and operational viability. The assessment concluded that system development is feasible, supported by accessible tools such as Flutter and Android Studio, cost-effective implementation, and compatibility with established user engagement strategies.

Chapter 3

Methodology and Requirements Analysis

# Methodology and Requirements Analysis

This chapter outlines the methodology used in developing CalorieCare, covering the software process model, requirement gathering techniques, and analysis methods. This chapter details how system requirements are collected, documented, and analyzed to ensure alignment with user needs. Additionally, this chapter presents the Software Requirements Specification (SRS), including functional and non-functional requirements, which define the system's expected behavior and performance criteria.

## Software Process Methodology

This project will utilize the Incremental Model, which will divide this project into multiple increments, each of which will deliver a functional component (Alshamrani & Bahattab, 2015). The initial increment focuses on implementing the core functionality, while subsequent increments introduce additional features until the complete system is developed.

For this project, the initial increment will focus on implementing core features such as dynamic calorie target adjustment, food image recognition, and the streak function. These essential functionalities will form the foundation of the system, ensuring that users can track nutrition effectively. Subsequent increments will introduce additional features.

Each increment in this project undergoes requirement analysis, design, implementation, and testing before integration with previous increments (TpointTech, 2023). This structured approach ensures that functional components are delivered in an operational state, allowing for continuous assessment and iterative improvements throughout the development process. In addition, this project is continuously improved at each stage of development based on Stakeholder feedback, thus reducing uncertainty and aligning the final product with changing requirements (GeeksforGeeks, 2018).

### Phases of Incremental Model

Figure 3.1 shows the phases of the incremental model which are requirement analysis, design & development, testing and implementation.

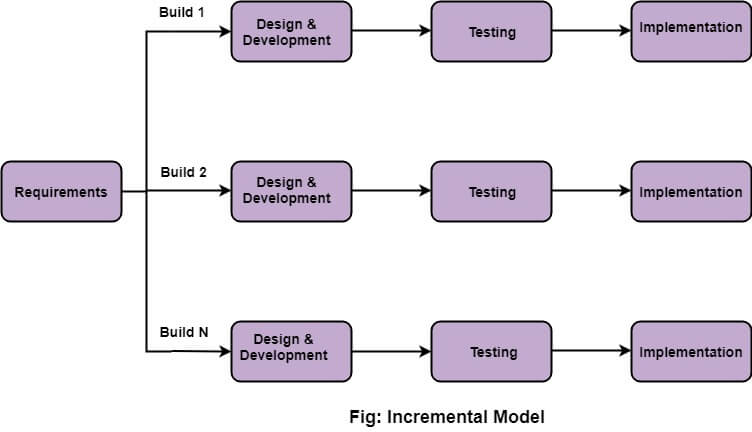


Figure 3.1: Phases of incremental model

* **Requirement Analysis**

The initial phase of this project involves gathering and analysing key system requirements. Unlike traditional models that define all requirements upfront, the Incremental Model allows planning for only the next increment, providing flexibility for modifications based on stakeholder feedback (GeeksforGeeks, 2018).

This project will first focus on identifying and prioritizing core functionalities essential for the initial increment, including dynamic calorie target adjustment, food image recognition, and the streak function. These features will form the foundation of the system to ensure essential functions are implemented early. Stakeholder feedback, such as user expectations and dietary tracking needs, will be collected and analysed to refine the requirements. Additionally, non-functional requirements, including system performance, usability, and data security, will be considered to enhance the system’s reliability. This iterative approach allows for continuous improvements in later increments, ensuring the system adapts to evolving user needs. This flexibility reduces the risk of requirement changes disrupting development while enabling an efficient and iterative improvement process (Sachan, 2023).

* **Design and Development**

Once the requirements for an increment are defined, the system architecture, components, and user interface will be designed. This phase follows an iterative approach, where core functionalities are developed first, ensuring that essential services operate independently before integrating additional features in subsequent increments (GeeksforGeeks, 2018). Each increment builds upon the previous one, refining and expanding the software’s capabilities. The use of iterative waterfall principles within each increment ensures a structured and systematic development process (Sachan, 2023).

In this stage, this project will first focus on designing and developing the core system architecture, including the BMI-based calorie target adjustment, food image recognition, and streak tracking features. The system will be designed with modular components to allow seamless integration of future enhancements. The user interface will be designed to provide an intuitive experience, ensuring that users can easily log meals and track progress. Each increment will undergo development and testing before integration, ensuring a stable and scalable system as new features are introduced.

* **Testing**

After the development of each increment, rigorous testing is conducted to evaluate the system’s performance, functionality, and integration with previous increments. Testing methods include unit testing, integration testing, and user acceptance testing to ensure that new features do not introduce defects or disrupt existing functionalities (TpointTech, 2023). Early detection of errors in each increment helps in reducing overall development risks and ensures that the final product meets quality standards (Sachan, 2023).

In this stage, this project will implement a structured testing approach to validate core functionalities, including dynamic calorie target adjustment, food image recognition, and streak tracking. Unit testing will be performed to verify individual components, while integration testing will ensure seamless interaction between different system modules. Acceptance testing will be conducted to verify whether the system meets the functional requirements and to identify and fix any bugs. Testing for each increment will be carried out iteratively to ensure early detection and resolution of any issues, thereby minimizing potential risks in later development phases.

* **Implementation**

Upon successful testing, the completed increment is deployed for user evaluation or operational use. The implementation phase involves integrating the newly developed features into the existing system while ensuring compatibility and stability (GeeksforGeeks, 2018). Once all planned increments are completed and integrated, the final version of the software is fully deployed at the client site (TpointTech, 2023). This phased deployment strategy allows stakeholders to gain access to critical system functionalities early, providing valuable feedback that can influence future increments.

In this stage, this project will implement and deploy each completed increment in phases, starting with core features such as dynamic calorie target adjustment, food image recognition, and streak tracking. These features will be made available for early user evaluation, allowing for real-world testing and feedback collection. Subsequent increments will be integrated gradually, ensuring smooth transitions and maintaining system stability. Continuous monitoring will be conducted post-deployment to address any issues and optimize system performance based on user feedback.

### Advantages of Incremental Model

* **Early Delivery of Functional Components**

Each increment delivers a working version of the software, allowing users to access core functionalities at an early stage (GeeksforGeeks, 2018). This project will first implement essential features, such as dynamic calorie target adjustment, food image recognition, and streak tracking, ensuring that users can start utilizing key functionalities while additional features are still under development.

* **Reduced Development Risk**

By dividing the project into smaller increments, risks are distributed rather than concentrated in a single-phase delivery. Any issues can be identified and resolved within individual increments, minimizing the likelihood of large-scale failures (Sachan, 2023). This project will adopt an incremental approach to mitigate development risks. Any issues detected in early increments will be addressed immediately, preventing major disruptions in later stages.

* **Continuous Feedback Integration**

Stakeholders have the opportunity to review each increment, providing valuable feedback that can be used to refine future increments. This iterative process enhances user satisfaction and ensures that the final product meets functional and usability requirements (Sachan, 2023). This project will incorporate user feedback at each development stage, ensuring that adjustments can be made based on user needs and preferences, ultimately improving the overall user experience.

* **Simplified Testing and Debugging**

Errors are easier to identify and resolve since testing is conducted for each increment separately. This reduces the complexity of debugging compared to a monolithic development approach (TpointTech, 2023). This project will implement testing at each increment, allowing for systematic debugging and ensuring that defects are resolved before integrating new features.

* **Efficient Resource Management**

The model facilitates better allocation of resources by focusing on smaller, manageable increments. Development teams can prioritize high-risk or critical functionalities in earlier increments, ensuring that essential components are developed first (Alshamrani & Bahattab, 2015). This project will prioritize the implementation of high-impact features in the initial increments, ensuring that core functionalities are well-developed before focusing on additional enhancements.

## Requirements Gathering Techniques

This section outlines the requirements gathering techniques used in the development of CalorieCare. Requirements gathering is an essential process to identify existing problems, user needs, system requirements, and potential solutions before system design and implementation.

### Background Research

Background research was conducted to gather relevant information on diet tracking applications, AI-powered nutrition analysis, and motivational techniques such as streak-based engagement. This research approach involved reviewing journal articles, conference papers, industry reports, and online resources to gain a comprehensive understanding of current trends, best practices, and common challenges in the field of diet and nutrition management. Special focus was placed on the integration of artificial intelligence in personalized dietary systems, as well as behavioural science strategies used to enhance user adherence.

### Observation

In addition to academic research, observation of existing diet tracking applications was carried out to gather practical insights from real-world implementations. Applications such as Calorie Mama, FatSecret, and Track were analysed to identify key system features, user engagement mechanisms, and technological capabilities. These systems were reviewed to determine their strengths and limitations, particularly in areas such as food recognition accuracy, customization of dietary goals, and the use of gamification elements. By observing these applications, the project aims to incorporate effective design patterns while addressing the gaps and usability issues identified in current solutions.

## Requirement Recoding Techniques

This section describes the methods used to document the clarified requirements obtained during the requirements gathering phase for CalorieCare. The system requirements are recorded using an Overview Use Case Diagram and Detailed Use Case Diagrams. These diagrams serve as visual representations of the system's functional requirements, illustrating the interactions between users and the system. This structured approach ensures a clear and organized understanding of the system functionalities.

### Overview Use Case Diagram

The Overview Use Case Diagram outlines the core functionalities of CalorieCare, capturing key interactions between users and system.

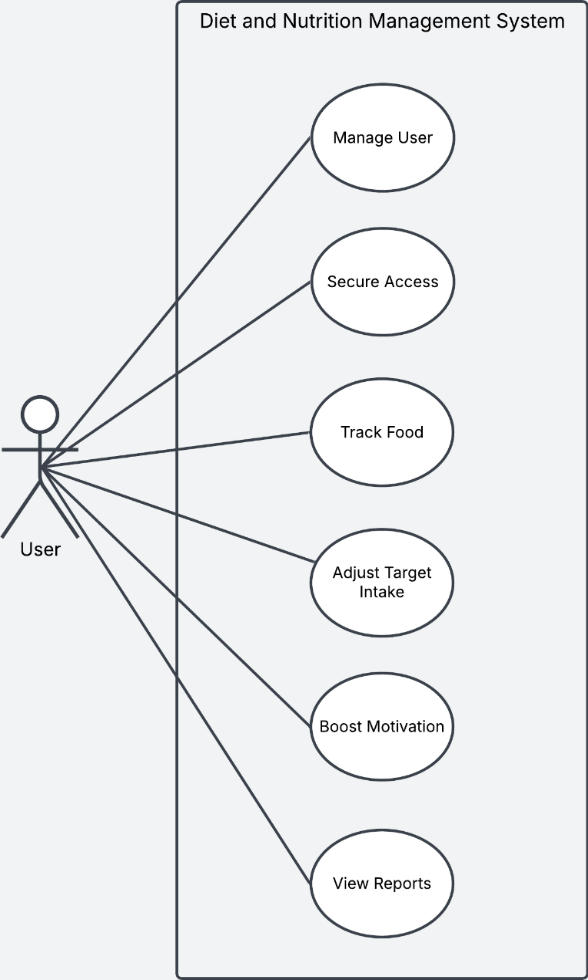


Figure 3.1: Overview Use Case Diagram

### Detailed Use Case Diagram

The Detailed Use Case Diagrams specify the functional requirements of individual modules within CalorieCare, breaking down high-level interactions into granular workflows. Each diagram is accompanied by a Use Case Description to describe how a system will function.

#### Manage User Use Case Diagram



Figure 3.2: Detailed Use Case Diagram for Manage User

Table 3.1: Use case description for Register User

|  |  |
| --- | --- |
| Name of Use Case: Register User | |
| Brief Description: This use case allows users to register | |
| Actors: User | |
| Precondition: The user does not have an account yet | |
| Actor action | System Response |
| 1. The user enters personal details (name, email, password) | 2. The system validates the input and creates a new account  3. The system confirms registration |
| Alternative Flows:  A1. Step 2: If the email is already registered or the input is invalid (e.g., incorrect email format, weak password), the system displays an error message and prompts the user to re-enter details | |
| Postcondition: The user successfully registers in the system. | |

Table 3.2: Use case description for Set Target

|  |  |
| --- | --- |
| Name of Use Case: Set Target | |
| Brief Description: This use case allows users to set the target | |
| Actors: User | |
| Precondition: The user has not set the target yet | |
| Actor action | System Response |
| 1. The user selects a weight goal: Lose Weight, Maintain Weight, or Increase Weight | 2. The system saves the selected goal  3. The system sets the Total Daily Energy Expenditure based on different health goals |
| Alternative Flows:  A1. Step 2: If the user does not select a goal, the system prompts them to choose before proceeding | |
| Postcondition: The user successfully set the target | |

Table 3.3: Use case description for Calculate BMI

|  |  |
| --- | --- |
| Name of Use Case: Calculate BMI | |
| Brief Description: This use case allows for calculating BMI based on user information | |
| Actors: User | |
| Precondition: The user has not input their weight and height | |
| Actor action | System Response |
| 1. The user inputs their weight and height | 2. The system calculates the BMI  3. The system saves the user's BMI  4. The system categorizes the BMI result (e.g., underweight, normal, overweight) |
| Alternative Flows:  A1. Step 2: If the user enters invalid values (e.g., negative numbers, non-numeric inputs), the system prompts for valid input | |
| Postcondition: The user receives their BMI result and classification | |

Table 3.4: Use case description for Calculate Target Calorie

|  |  |
| --- | --- |
| Name of Use Case: Set Target | |
| Brief Description: This use case allows for calculating the user’s daily calorie target based on BMI, activity level, and selected weight goal | |
| Actors: User | |
| Precondition: The user has set a weight goal | |
| Actor action | System Response |
| 1. The user provides additional details (age, activity level, weight goal) | 2. The system calculates the daily recommended calorie intake based on the selected weight goal  3. The system saves the user's daily recommended calorie intake  4. The system displays the daily recommended calorie intake |
| Alternative Flows:  A1. Step 2: If the user has not set a weight goal before this step, the system prompts them to first complete the Set Target step.  A2. Step 2: If invalid input is provided (e.g., unrealistic age or activity level), the system requests correction. | |
| Postcondition: The user receives their daily calorie intake recommendation | |

#### Secure Access Use Case Diagram

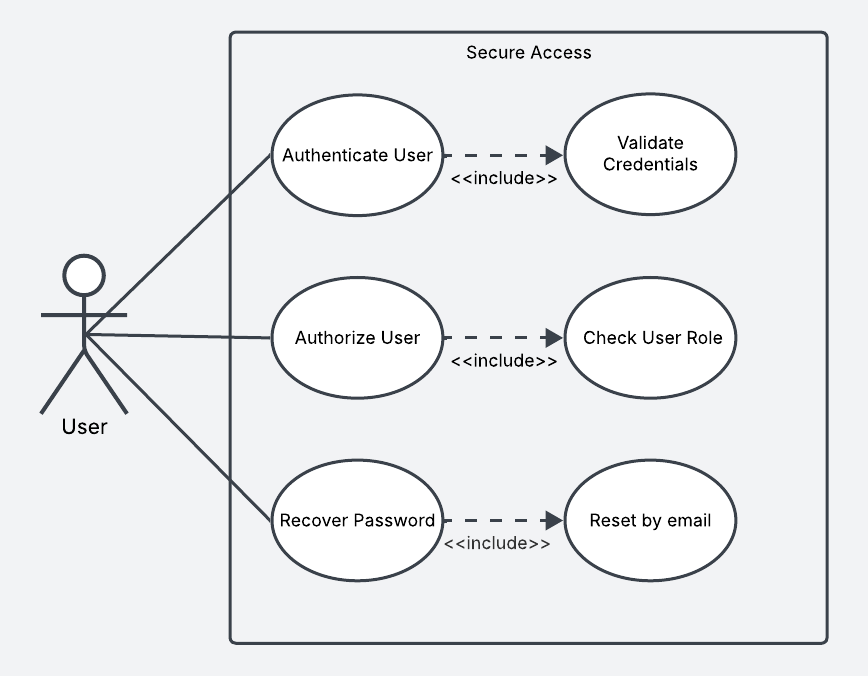


Figure 3.3: Detailed Use Case Diagram for Secure Access

Table 3.5: Use case description for Authenticate User

|  |  |
| --- | --- |
| Name of Use Case: Authenticate User | |
| Brief Description: This use case allows for verifying the user's identity by validating login credentials | |
| Actors: User | |
| Precondition: The users already have an account | |
| Actor action | System Response |
| 1. The user enters their email and password | 2. The system validates the credentials  3. The user is granted access to the system |
| Alternative Flows:  A1. Step 3: If the credentials are invalid, the system displays an error message and the user cannot access the system | |
| Postcondition: Users can securely log in to their accounts | |

Table 3.6: Use case description for Authorize User

|  |  |
| --- | --- |
| Name of Use Case: Authorize User | |
| Brief Description: This use case allows for checking the user's role to determine access rights | |
| Actors: User | |
| Precondition: The user must be authenticated | |
| Actor action | System Response |
| 1. The user enters their email and password | 2. The system retrieves the user's role and associated permissions  5. The user is granted access to the requested resources |
| Alternative Flows:  A1. Step 2: If the user lacks privileges, access is denied | |
| Postcondition: Users successfully gain access | |

Table 3.7: Use case description for Recover Password

|  |  |
| --- | --- |
| Name of Use Case: Recover Password | |
| Brief Description: This use case allows the user to reset their password | |
| Actors: User | |
| Precondition: The user must have a registered email | |
| Actor action | System Response |
| 1. The user clicks "Forgot Password"  3. The user enters their registered email  5. The user receives an email to reset the password  7. The user inputs a new password | 2. The system prompts for email input  4. The system sends a password reset link to the user's email  6. The system navigates reset password page to let the user reset the password  7. The system resets the user’s password |
| Alternative Flows:  A1. Step 4: If the email is not registered, the system displays an error message | |
| Postcondition: Users successfully reset their password | |

#### Track Food Use Case Diagram

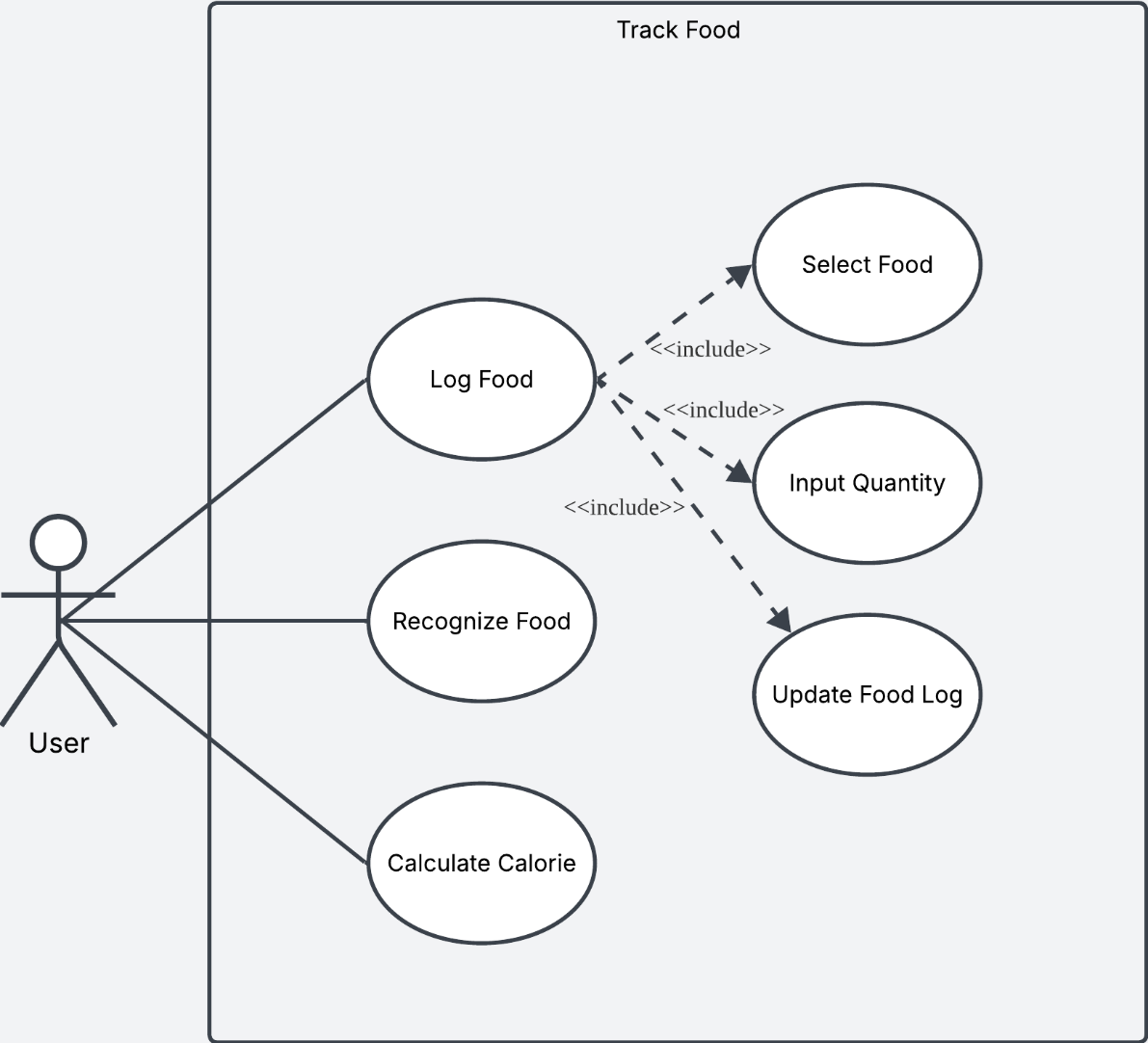


Figure 3.4: Detailed Use Case Diagram for Track Food

Table 3.8: Use case description for Log Food

|  |  |
| --- | --- |
| Name of Use Case: Log Food | |
| Brief Description: This use case enables users to log food | |
| Actors: User | |
| Precondition: The user must be logged into the system | |
| Actor action | System Response |
| 1. The user selects the "Log Food" option  3. The user selects a food item  5. The user inputs the quantity of food consumed  7. The user confirms and saves the log | 2. The system displays a list of available food items  4. The system records the selected food  6. The system updates the food log with the quantity  8. The system create the food log successfully |
| Alternative Flows:  A1. Step 4: If the food item is not in the database, the system allows the user to manually enter food details  A2. Step 6: If the user enters invalid values (e.g., negative numbers, non-numeric inputs), the system prompts for valid input | |
| Postcondition: The user's food intake is recorded in the system. | |

Table 3.9: Use case description for Recognize Food

|  |  |
| --- | --- |
| Name of Use Case: Recognize Food | |
| Brief Description: This use case allows for calculating the total calorie intake based on the logged food data | |
| Actors: User | |
| Precondition: The user must have at least one logged food entry | |
| Actor action | System Response |
| 1. The user selects the "Recognize Food" option  3. The user captures or uploads an image of the food | 2. The system activates the camera or allows image upload  4. The system processes the image using food recognition API  5. The system suggests a detected food item |
| Alternative Flows:  A1. Step 5: If the system cannot recognize the food, the user can manually input the food details | |
| Postcondition: The food item was successfully detected | |

Table 3.10: Use case description for Calculate Calorie

|  |  |
| --- | --- |
| Name of Use Case: Calculate Calorie | |
| Brief Description: This use case enables users to log food | |
| Actors: User | |
| Precondition: The user must be logged into the system | |
| Actor action | System Response |
| 1. The user selects food item and inputs the food quantity | 2. The system calculates the calorie intake based on food type and quantity |
| Alternative Flows: | |
| Postcondition: The system displays the calorie intake to the user | |

#### Adjust Target Use Case Diagram

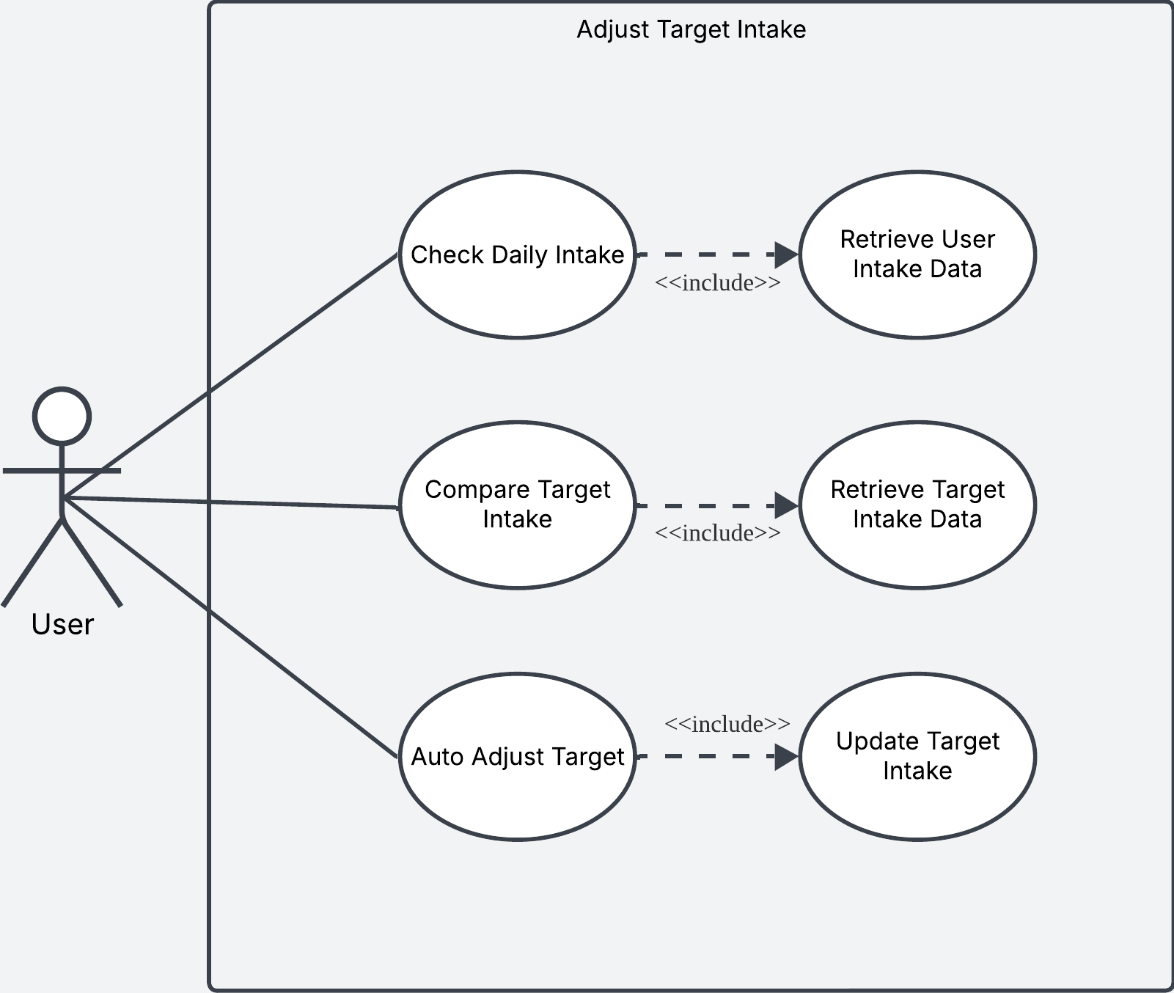


Figure 3.5: Detailed Use Case Diagram for Adjust Target Intake

Table 3.11: Use case description for Check Daily Intake

|  |  |
| --- | --- |
| Name of Use Case: Check Daily Intake | |
| Brief Description: This use case allows the system to check the user's daily intake | |
| Actors: User | |
| Precondition: The user must be logged into the system and user intake data and target intake data must be available. | |
| Actor action | System Response |
| 1. The user finishes logging food | 2. The system retrieves the user's daily intake data |
| Alternative Flows: | |
| Postcondition: The system successfully retrieves the user’s daily intake data | |

Table 3.12: Use case description for Compare Target Intake

|  |  |
| --- | --- |
| Name of Use Case: Compare Target Intake | |
| Brief Description: This use case allows the system to compare the user's daily intake and the user's target intake | |
| Actors: User | |
| Precondition: The user must be logged into the system and user intake data and target intake data must be available. | |
| Actor action | System Response |
| 1. The user finishes logging food | 2. The system retrieves the target intake data for comparison  3. The system compares the daily intake with the target intake |
| Alternative Flows: | |
| Postcondition: The system successfully compares | |

Table 3.13: Use case description for Auto Adjust Target

|  |  |
| --- | --- |
| Name of Use Case: Auto Adjust Target | |
| Brief Description: This use case allows the system to compare the user's daily intake and the user's target intake to automatically adjust the daily intake target | |
| Actors: User | |
| Precondition: The user must be logged into the system and user intake data and target intake data must be available. | |
| Actor action | System Response |
| 1. The user finishes logging food | 2. The system automatically adjusts the target intake  3. The system updates the target intake to reflect the new recommended value |
| Alternative Flows: | |
| Postcondition: The Target Calorie Intake is updated based on the user’s actual daily intake | |

#### Boost Motivation Use Case Diagram



Figure 3.6: Detailed Use Case Diagram for Boost Motivation

Table 3.14: Use case description for Track Streak

|  |  |
| --- | --- |
| Name of Use Case: Track Streak | |
| Brief Description: This use case allows the system to monitor and update the user's streak based on their daily food logging activity | |
| Actors: User | |
| Precondition: The user must be logged into the system and the user must finish the daily food log | |
| Actor action | System Response |
| 1. The user finishes logging food | 2. The system checks the user's streak data from the database  3. The system increments the streak count and adds a 🔥 (fire) icon  4. The system displays the updated streak information to the user |
| Alternative Flows:  A1. Step 3: If the user does not have an existing streak, the system starts the streak from zero and adds the first 🔥 (fire) icon | |
| Postcondition: The user's streak count is updated and saved in the database | |

Table 3.15: Use case description for Invite Friend

|  |  |
| --- | --- |
| Name of Use Case: Invite Friend | |
| Brief Description: This use case allows a user to invite a friend to become a supervisor | |
| Actors: User | |
| Precondition: The user must have at least one friend | |
| Actor action | System Response |
| 1. The user selects the "Invite Friend" option  3. The user selects a friend  5. The user’s friend accepts the invitation | 2. The system prompts the friend list  4. The system sends an invitation link to the friend  6. The system adds the friend to become the user’s supervisor |
| Alternative Flows:  A1. Step 6: If the invited friend declines the invitation, the system notifies the user | |
| Postcondition: The user's friend successfully became the supervisor | |

Table 3.16: Use case description for Alert Friend

|  |  |
| --- | --- |
| Name of Use Case: Alert Friend | |
| Brief Description: This use case allows a user to alert a friend when the friend hasn‘t logged food | |
| Actors: User | |
| Precondition: The users and friends must be each other's supervisor | |
| Actor action | System Response |
| 1. The user finishes logging food  4. The user selects "Remind Friend" | 2. The system checks if the friend has logged their food  3. The system allows the user to send a reminder  5. The system sends a notification to the friend, reminding them to log food |
| Alternative Flows:  A1. Step 3: If the friend has also logged food, the system notifies both users that their streak continues | |
| Postcondition: The user's friend successfully receives the reminder notification | |

#### View Reports Use Case Diagram

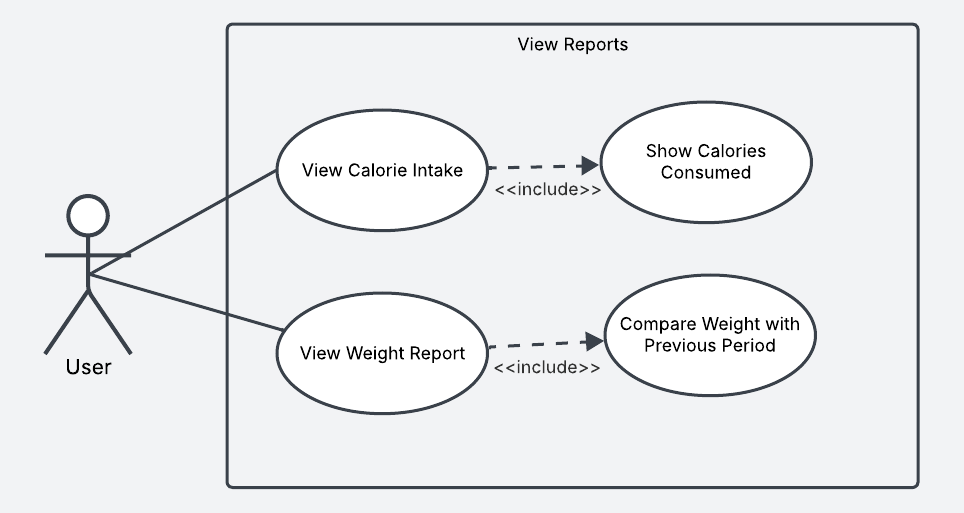


Figure 3.7: Detailed Use Case Diagram for View Reports

Table 3.17: Use case description for View Calorie Intake

|  |  |
| --- | --- |
| Name of Use Case: View Calorie Intake Report | |
| Brief Description: This use case allows a user to view their calorie intake based on logged meals | |
| Actors: User | |
| Precondition: The user must have logged at least one meal for the day | |
| Actor action | System Response |
| 1. The user selects the "View Daily Calorie Intake" option | 2. The system retrieves the total calorie intake of logged meals for the current day  3. The system displays the total calorie count along with meal details |
| Alternative Flows:  A1. Step 3: If no meals are logged, the system prompts the user to add meals | |
| Postcondition: The user successfully views their total calorie intake for the selected day | |

Table 3.18: Use case description for View Weight Reports

|  |  |
| --- | --- |
| Name of Use Case: View Weight Report | |
| Brief Description: This use case allows the user to track their weight changes | |
| Actors: User | |
| Precondition: The user must be logged into the system and must have stored data weight history | |
| Actor action | System Response |
| 1. The user selects "View Weight Report" | 2. The system retrieves weight data and compares it with the previous period  3. The system displays the weight trend and analysis |
| Alternative Flows:  A1. Step 3: If there is no previous weight data, the system prompts the user to input their initial weight. | |
| Postcondition: The user successfully views the weight report. | |

## Requirements Analysis

This section provides a detailed analysis of the system requirements for CalorieCare. This section begins by outlining the key product functions derived from the fact-gathering process, followed by an evaluation of similar existing systems to identify potential improvements. The analysis also includes a clear user problem statement and outlines specific user objectives that the system aims to fulfil. In addition, general constraints such as technical limitations and platform requirements are considered to ensure realistic development planning. Finally, external interface requirements are examined to define how CalorieCare will interact with third-party systems and hardware.

### Product Function

This system shall simplify meal logging by providing users with two convenient options: manual input and AI-powered image recognition. Users can manually enter meals, including detailed nutritional information, or take advantage of the system’s intelligent image recognition feature, which automatically identifies food items and estimates calorie content. This reduces the effort required for meal tracking, simplifying and enhancing the intuitiveness of logging daily intake.

To ensure a balanced and sustainable approach to dieting, the system incorporates dynamic daily calorie target adjustment. Whenever a user exceeds or falls short of the daily calorie goal, the system automatically recalculates the next day’s target to restore balance. By making adjustments responsive to real dietary behavior, CalorieCare helps users maintain steady progress without feeling overly restricted or demotivated. At the same time, all changes remain within safe nutritional boundaries, as calorie targets never go below the Basal Metabolic Rate (BMR) or 1,200 kcal/day for women and 1,500 kcal/day for men, and never exceed TDEE plus 500 kcal (Publishing, 2020; Larson-Meyer et al., 2022; NHS, 2021). This evidence-based framework ensures the adaptation is both flexible and safe, preventing health risks while supporting long-term adherence.

To encourage consistency, the system introduces a streak-based habit reinforcement feature. Users who successfully log all three meals in a day will receive a 🔥 streak marker. With each consecutive day of meal logging, the streak count increases, motivating users to maintain tracking habits. However, missing a day will reset the streak, creating an incentive for users to stay engaged and consistent in meal tracking. This feature is designed to foster a sense of achievement and personal accountability, making the process of diet tracking more engaging and rewarding.

Furthermore, the system offers a Two-Person Supervision Mode, which enhances accountability and motivation through social engagement. Users can invite a friend or family member to act as a supervisor, allowing them to monitor each other’s meal logging progress. If one user forgets to log a meal, the supervisor can send a custom reminder message, making the process interactive and supportive. When both users successfully complete daily meal logs, the users earn a shared 🔥 streak reward. This feature not only reinforces consistency but also creates a collaborative and encouraging environment for users striving to maintain healthy eating habits.

### Similar System Information

Several diet-tracking applications are available in the market, each offering unique features.

Table 3.19: Comparison of CalorieCare with Similar Diet-Tracking Apps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **CalorieCare** | **Calorie MAMA** | **FatSecret** | **Track** |
| AI Food Image Recognition | Yes | Yes | No | No |
| Automatic Calorie Adjustment | Yes | No | No | No |
| Streak-Based Motivation | Yes | No | Yes | No |
| Two-Person Supervision Mode | Yes | No | No | No |
| User Data Protection | Yes | Yes | Yes | Yes |

One of the key differences is AI-powered food recognition. Both CalorieCare and Calorie MAMA allow users to log meals using AI image recognition, making meal tracking faster and more convenient. In contrast, FatSecret and Track rely on manual food entry, requiring users to search for and input meal details themselves.

Another major distinction is automatic calorie adjustment, a feature unique to CalorieCare. Unlike other apps that provide static calorie targets, CalorieCare dynamically adjusts daily and weekly calorie goals based on the user's daily calorie intake.

Additionally, CalorieCare and FatSecret introduces a streak-based motivation system to encourage consistent meal logging. This includes visual streak tracking and rewards to help users stay committed to dietary goals. While Calorie MAMA and Track lack such a feature, CalorieCare goes a step further by offering a Two-Person Supervision Mode, allowing users to track each other's meal logging and send reminders when necessary. In terms of data privacy and security, all four apps ensure user data protection through encryption and secure storage.

Overall, CalorieCare ability to adapt calorie goals based on user intake and foster accountability through supervision and streak tracking gives CalorieCare a competitive edge in the diet management app market.

### User Problem Statement

Traditional diet tracking apps are often time-consuming and repetitive, making users feel disengaged and leading to abandonment. Manual meal logging is perceived as tedious, while failing to meet dietary goals can cause negative emotions like guilt and anxiety, discouraging users from continuing. Additionally, inconsistent tracking habits further reduce long-term adherence. To improve user retention, diet tracking systems must simplify meal logging, reduce emotional stress, and encourage habit formation.

### Users Objectives

CalorieCare addresses these issues by automating meal logging with AI-powered image recognition, adjusting calorie targets dynamically to prevent discouragement, and introducing a streak-based motivation system. Users can track progress through streak rewards and engage in Two-Person Supervision Mode, where friends can remind and motivate each other. By making diet tracking easier, engaging, and habit-forming, CalorieCare promotes long-term adherence to healthy eating.

### General Constraints

The development timeline for CalorieCare is limited, requiring completion within the project deadline, which impacts feature scope and implementation complexity. Additionally, budget constraints restrict the use of premium APIs and cloud services, requiring efficient resource allocation and reliance on open-source tools where possible.

Furthermore, hardware limitations must be considered, ensuring the app runs smoothly on a range of smartphones without requiring high-end processing power. Data privacy and security regulations must also be followed, ensuring user information, including meal logs and personal health data, is protected.

### External Interface Requirement

CalorieCare’s external interface requirements ensure seamless system functionality and user interaction. The User Interface (UI) is designed specifically for Android devices, providing a responsive and intuitive experience that allows users to log meals effortlessly, track progress, and interact with AI-based meal recognition features. The interface prioritizes ease of use, ensuring that users can efficiently access key features without unnecessary complexity.

For data management, the system relies on Firebase as primary database solution, ensuring secure storage and efficient retrieval of user data. This allows users to access dietary records, progress, and AI-generated recommendations anytime, maintaining data consistency across sessions.

The communication interface is essential for real-time data synchronization. The app requires internet connectivity to sync user data, retrieve nutritional information, and update AI-generated recommendations.

## Software Requirements Specification

This section defines both the functional and non-functional requirements of CalorieCare. The functional requirements describe the core features and services that the system must provide to meet user needs, while the non-functional requirements specify quality attributes such as performance, usability, and security. Together, these specifications ensure CalorieCare is both operationally effective and user-friendly.

### Functional Requirements

CalorieCare shall provide the following functional capabilities

#### User Management Module

1. **Register User**

The system shall allow new users to register by providing their name, email address, and password. During the registration process, the system will validate all user input to ensure data integrity and compliance with required formats. Upon successful validation, the system will securely store the user credentials in Firebase using industry-standard encryption methods to protect sensitive information. A valid email address meeting standard email format requirements and a password that meets minimum security criteria are mandatory for successful registration completion.

1. **Set Target**

The system shall enable users to establish their health objectives by selecting from predefined options including weight loss, weight maintenance, or muscle gain. This selection of a specific health goal is mandatory for all users during the initial setup process.

1. **Calculate BMI**

The user must provide accurate weight and height measurements in the specified units for the BMI calculation to be valid. The system shall automatically compute the Body Mass Index using the standard formula (weight divided by height squared) based on the user's input. Following the calculation, the system shall classify the user's health status according to established BMI categories (underweight, normal weight, overweight, or obese). The user should review the calculated BMI result and corresponding health classification to gain awareness of their current health status and inform their nutritional goals.

1. **Calculate Target Calorie**

The system shall automatically compute the user's daily caloric requirements by integrating multiple physiological and lifestyle parameters, including their calculated BMI, self-reported activity level, and selected health goals (weight loss, maintenance, or muscle gain). For the calorie calculation to be accurate, the user must provide their current activity level through the system's standardized activity scale, which ranges from sedentary to highly active. This comprehensive calculation will utilize established metabolic equations (such as the Mifflin-St Jeor Formula) to generate personalized daily calorie targets that align with the user's specified objectives.

#### Security Module

1. **Authenticate User**

The system shall implement secure user authentication by verifying credentials against stored encrypted data before granting access, while enforcing strict access control measures to prevent unauthorized system entry. For successful authentication, users must provide valid credentials consisting of their registered email address and correct password combination.

1. **Authorize User**

The system shall manage user permissions to ensure that access to various functionalities is granted only to authorized users. Each user shall be assigned a specific role, and the system shall enforce access control rules based on the permissions associated with that role. Users should adhere to the defined access restrictions and must not attempt to bypass or manipulate the system's security protocols.

1. **Recover Password**

The system shall provide a secure password reset mechanism through email verification protocols. To initiate password recovery, users must supply the email address associated with their account to receive an authenticated reset link.

#### Smart Food Tracking Module

1. **Log Food**

The system shall support manual food logging by accepting user-inputted food items with corresponding portion sizes. Users must accurately specify both the food name and quantity for proper nutritional tracking.

1. **Recognize Food**

The system shall use AI-based image recognition to identify food from photos and automatically identify meal contents, subsequently estimating nutritional values from recognized items. For optimal recognition accuracy, users must upload clear, well-lit photographs of their meals.

1. **Calculate Calorie**

The system shall calculate the total calorie intake based on the food items logged by the user, taking into account any portion size adjustments made during the logging process. When a user selects or enters a food item, the system shall display the default calorie value based on a standard serving size. The user should be able to modify the portion size according to their actual consumption, and the system shall dynamically recalculate the corresponding calorie value based on the updated quantity.

#### Dynamic Target Adjustment Module

1. **Check Daily Intake**

The system shall monitor the user’s daily food consumption to dynamically adjust their calorie intake based on their logged meals and nutritional goals. This feature ensures that the system continuously evaluates the user's caloric needs and updates their target intake accordingly. The user must log their meals first for the system to track their daily intake and make the necessary adjustments. Without meal logs, the system shall not be able to assess or modify the user's calorie intake, preventing any dynamic adjustments from taking place.

1. **Compare Target Intake**

The system should analyse the user's daily nutritional intake to determine whether the daily intake meets, exceeds, or falls short of the recommended target values based on their health profile. The user must have already logged at least one meal for the day. Without any logged meals, the system shall not be able to generate a meaningful comparison regarding the user’s intake status.

1. **Auto Adjust Target**

The system shall dynamically adjust the user's daily calorie targets based on long-term dietary and health trends observed over time. The user should follow the updated calorie recommendations provided by the system to achieve optimal results and maintain a balanced nutritional intake aligned with their personal objectives.

#### Motivation Module

1. **Track Streak**

The system shall track the number of consecutive days the user logs their meals to promote consistency and build healthy tracking habits over time. To visually reinforce progress, the system shall reward users with streak indicators, such as a fire icon (🔥) that increases with each additional consecutive day of logging. For example, logging meals for two days in a row would result in a 🔥🔥 streak. The user must log all required meals each day to maintain their streak. If a day is missed, the streak shall reset to zero, encouraging users to stay engaged and consistent with their meal tracking behaviour.

1. **Invite Friend**

The system shall allow users to invite friends to join the platform for accountability, enabling them to track and support each other’s progress. This feature aims to foster a sense of community and motivation through mutual encouragement and reminders. To initiate the invitation process, the user must send an invitation request to a friend via the system, providing the friend with an option to accept or decline the request. Once accepted, both users shall be able to view each other’s meal logging progress and offer encouragement, enhancing their commitment to their nutritional goals.

1. **Alert Friend**

If a user forgets to log their meals for the day, the system must enable friends to send reminders to encourage meal logging. This feature allows friends to play an active role in helping each other stay on track with their nutritional goals. The user’s friend shall be notified of the missed meal log and be able to send a customized reminder message to the user, urging them to complete their logging for the day. By enabling peer support, the system fosters accountability and motivates users to stay consistent with their meal tracking.

#### Report Module

1. **View Calorie Intake**

The system shall display a summary of the user’s consumed calories for the day, providing an overview of their current intake in relation to their calorie goals. This summary shall include details such as the total calories consumed, along with a breakdown by meal (e.g., breakfast, lunch, dinner, and snacks). In order for this summary to be generated, the users must log their meals first. Without meal logs, the system shall not be able to calculate or display the calorie intake, ensuring the accuracy and relevance of the information presented.

1. **View Weight Report**

The system shall generate weight trend reports based on the user’s input, providing visual insights into their weight progress over time. These reports shall include graphical representations such as charts or graphs, allowing users to track their weight changes and identify patterns or trends. The user must input weight data regularly, preferably on a daily or weekly basis, to ensure the accuracy and relevance of the trend report. Without consistent weight entries, the system shall not be able to generate a complete or accurate weight report, limiting the ability to track progress effectively.

### Non-functional Requirements

CalorieCare shall meet the following quality attributes

#### Product Requirements

1. **Usability**

The system should provide an intuitive and user-friendly interface, ensuring that users can easily navigate and log meals with minimal effort. The meal logging process must be simplified, allowing users to either manually enter meals or use AI-powered image recognition to reduce the burden of tracking food intake.

1. **Efficiency**

The app must process meal recognition and retrieve nutritional data within 5 seconds to ensure a seamless experience. Data storage and retrieval should be optimized to minimize delays in updating meal logs and calorie tracking, ensuring real-time responsiveness for users.

#### Organizational Requirements

1. **Delivery**

The system must be developed and deployed within the project’s designated timeline and budget. Regular updates and maintenance should be planned post-launch to enhance functionality, fix bugs, and incorporate user feedback for continuous improvement.

#### External Requirements

1. **Safety**

The system must prioritize user safety by ensuring that all nutritional recommendations are based on credible dietary guidelines. The system should also provide clear disclaimers that the app is not a substitute for professional medical advice, preventing users from relying on it for critical health decisions.

## Chapter Summary & Evaluation

This chapter outlined the methodology and requirements analysis for CalorieCare. This chapter began by explaining the use of qualitative research methodology, which helps understand user behaviors and motivations behind dietary tracking. Next, the Incremental Model was introduced as the selected software development methodology, allowing for iterative improvements and stakeholder feedback integration throughout the development cycle.

Various requirements gathering techniques were explored, including online research, which provided insights into existing diet-tracking applications, AI-powered nutrition analysis, and motivational techniques. The chapter also discussed requirement recording techniques and requirements analysis, which defined the system's core functionalities and user expectations. A comparison with similar diet-tracking applications highlighted the competitive advantages of CalorieCare, such as AI-powered food recognition, automatic calorie adjustments, and the Two-Person Supervision Mode.

Additionally, the Software Requirements Specification (SRS) was detailed, categorizing the system’s functional and non-functional requirements. Functional requirements covered essential modules, including user management, security, smart food tracking, dynamic calorie adjustment, motivation features, and reporting capabilities. Non-functional requirements addressed usability, efficiency, security, and compliance with dietary safety standards.

Chapter 4

System Design

# System Design

This chapter describes the overall system design of CalorieCare, aiming to translate the system requirements into a clear technical structure to support the development and implementation of the system. In this chapter, the following major design aspects will be covered: UI Design, Data Design, Report Design, Security Design, Process Design, Software Architecture Design, and Algorithm Design. Architecture Design, and Algorithm Design. Together, these designs ensure the integrity and viability of the system in terms of functionality, efficiency and user experience.

## UI Design

This section presents the user interface (UI) design of CalorieCare. The UI design aims to visually demonstrate how users interact with the system's core features such as food logging, target tracking, streak monitoring, and supervisor interaction. The prototype was created using Figma, focusing on delivering a clean, intuitive, and user-friendly experience that encourages daily engagement and promotes healthy habits.

### Register Module

The following shows the user prototype interface design for the Register User Module.

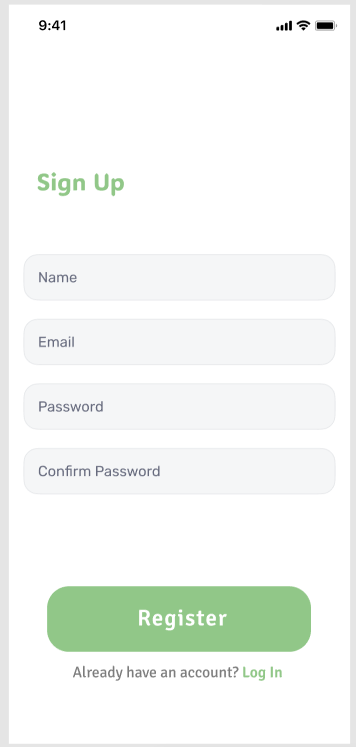


Figure 4.1: Register User Interface

### Set Target Module

The following shows the user prototype interface design for the Set Target Module.



Figure 4.2: Set Target Interface

### Calculate BMI Module

The following shows the user prototype interface design for the Calculate BMI Module.



Figure 4.3: Calculate BMI Interface

### Calculate Target Calorie Module

The following shows the user prototype interface design for the Calculate Target Calorie Module.

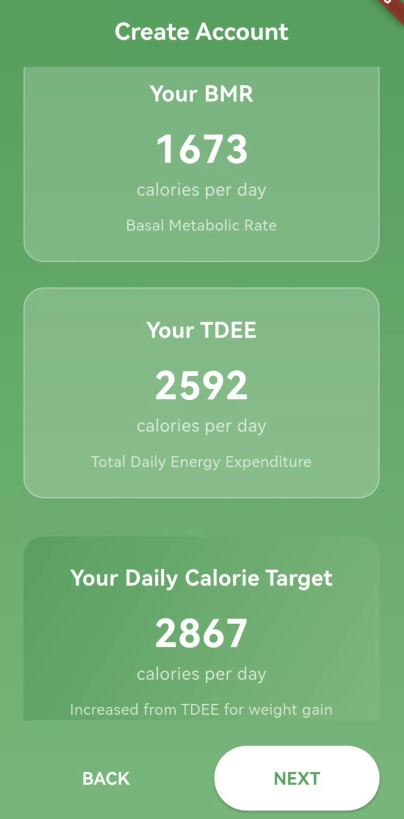


Figure 4.4: Calculate Target Calorie Interface

### Authenticate and Authorize User Module

The following shows the user prototype interface design for the Authenticate and Authorize User Module.

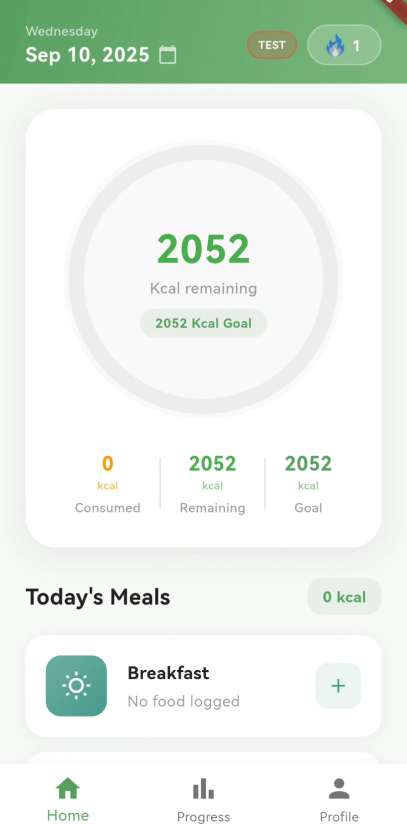
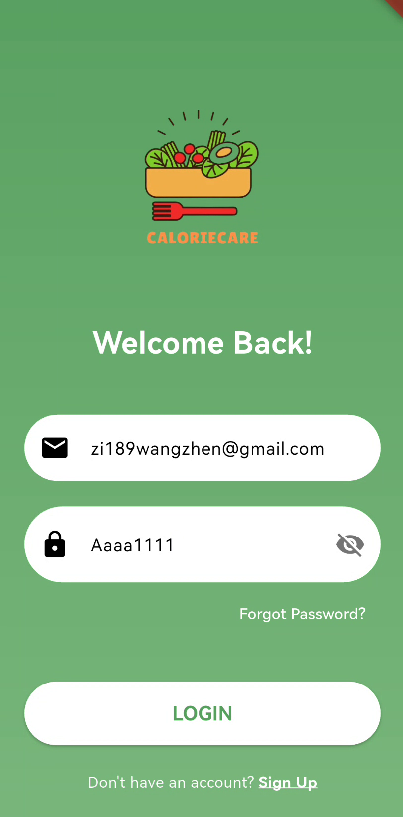


Figure 4.5: Authenticate and Authorize User Interface

### Recover Password Module

The following shows the user prototype interface design for the Recover Password Module.



Figure 4.6: Recover Password Interface

### Log Food Module

The following shows the user prototype interface design for the Log Food Module.

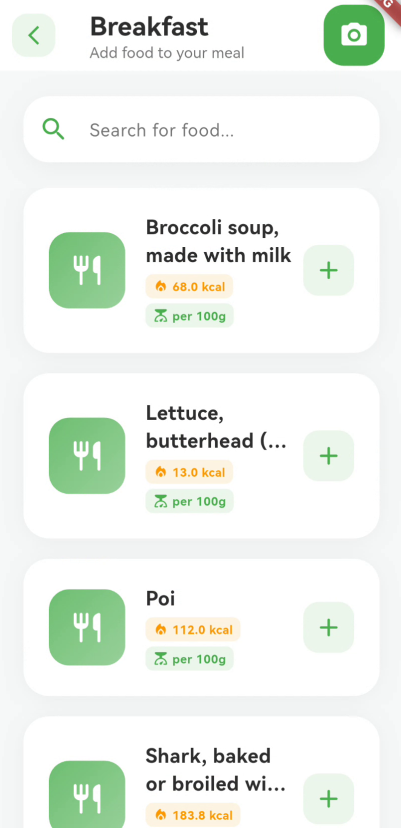


Figure 4.7: Log Food Interface

### Recognize Food Module

The following shows the user prototype interface design for the Recognize Food Module.

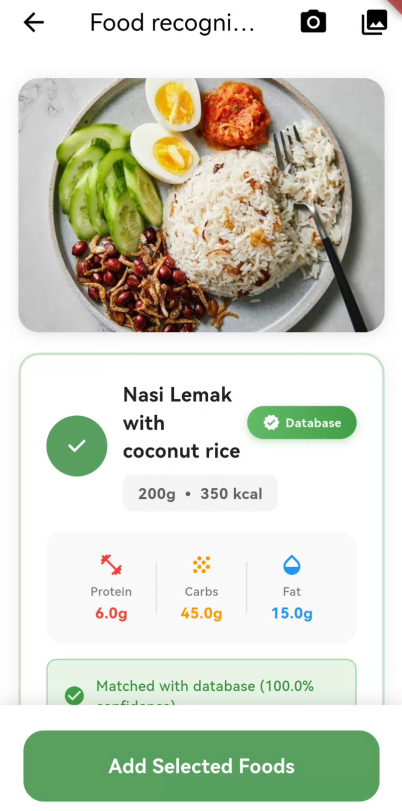


Figure 4.8: Recognize Food Interface

### Calculate Calorie Module

The following shows the user prototype interface design for the Calculate Calorie Module.

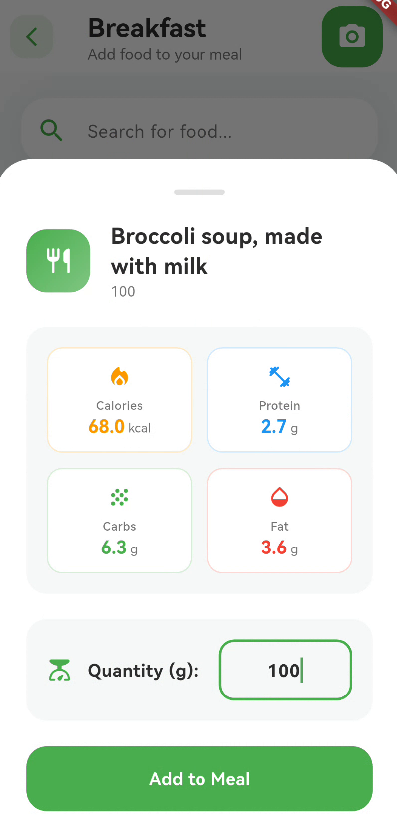


Figure 4.9: Calculate Calorie Interface

### Dynamic Target Adjustment Module

The Dynamic Target Adjustment Module ensures that nutrition goals adapt in real time to the user’s eating behavior. If a user exceeds or fails to meet the daily intake goal, the system automatically adjusts the next day’s calorie target to restore balance. These adjustments are sensitive to the user’s overall goal type, whether weight loss, maintenance, or weight gain, and are always constrained within health and safety boundaries. Specifically, daily intake is never allowed to drop below the BMR or 1,200 kcal for women and 1,500 kcal for men, and never allowed to exceed TDEE plus 500 kcal (Publishing, 2020; Larson-Meyer et al., 2022; NHS, 2021). By embedding these constraints, the system promotes a sustainable approach to dieting that avoids both nutrient deficiency and excessive intake.



Figure 4.10: Dynamic Target Adjustment Interface

### Track Streak Module

The following shows the user prototype interface design for the Track Streak Module.

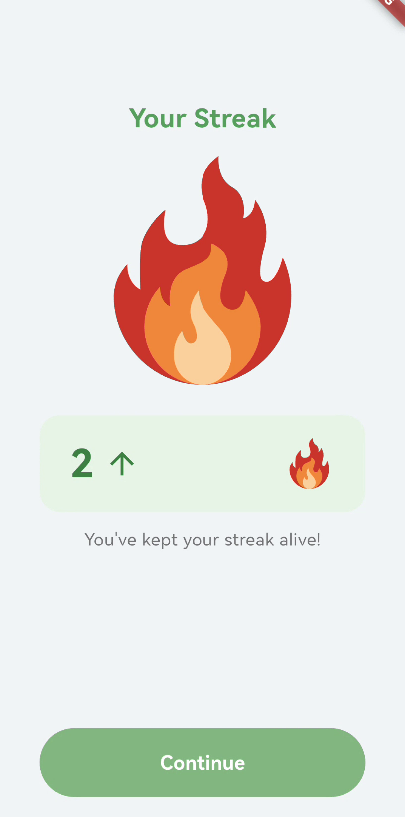


Figure 4.11: Track Streak Interface

### Invite Friend Module

The following shows the user prototype interface design for the Invite Friend Module.

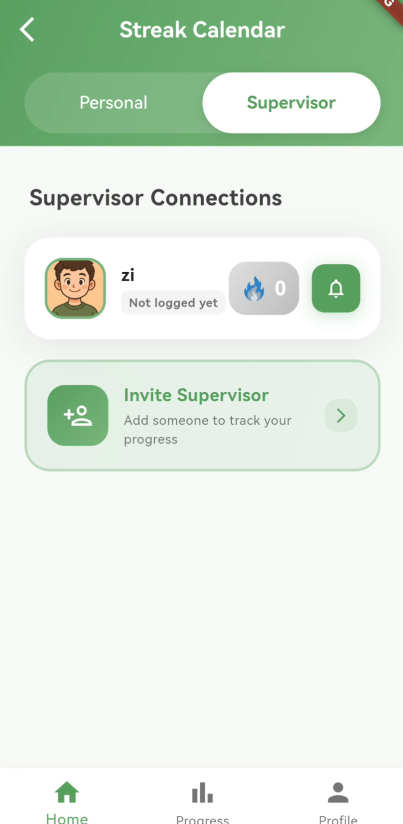


Figure 4.12: Invite Friend Interface

### Alert Friend Module

The following shows the user prototype interface design for the Alert Friend Module.

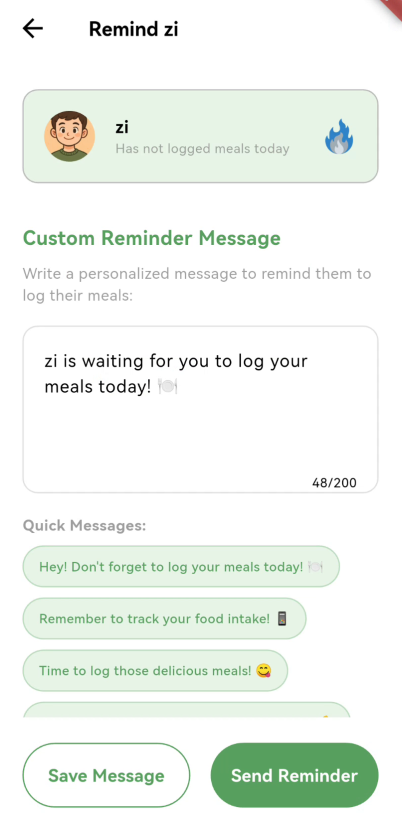


Figure 4.13: Alert Friend Interface

### View Calorie Intake Module

The following shows the user prototype interface design for the View Calorie Intake Module.

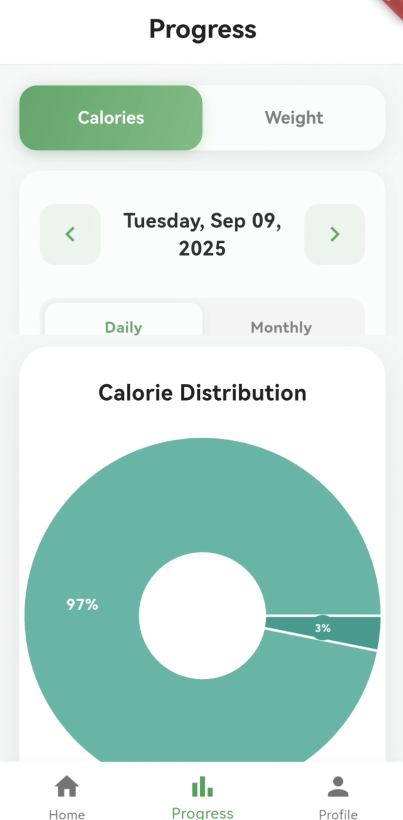


Figure 4.14: View Calorie Intake Interface

### View Weight Report Module

The following shows the user prototype interface design for the View Weight Report Module.

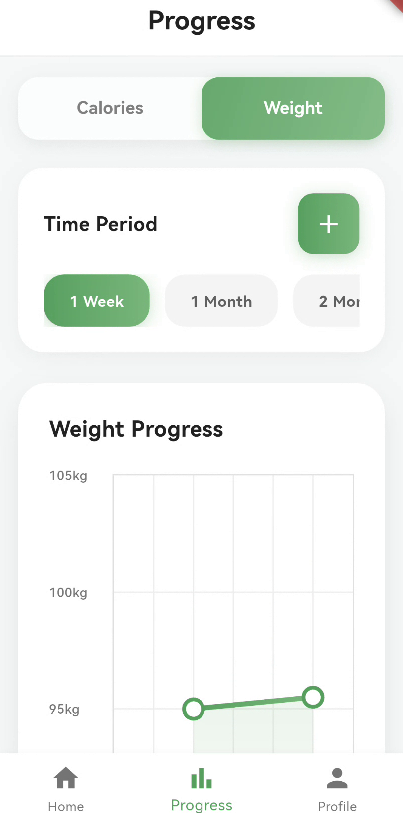


Figure 4.15: View Weight Report Interface

## Data Design

This section describes the data design of CalorieCare. This section includes the Entity Relationship Diagram (ERD), which outlines the relationships between various entities such as users, meal logs, food items, calorie targets, streak records, and supervision connections. The data design ensures data integrity, supports core system functionalities such as food tracking, target adjustment, and streak monitoring, and enables efficient data retrieval and management. In addition, this section also includes the data dictionary, which defines the attributes of each entity, including data types, field lengths, constraints, formats, default values, and descriptions. This helps ensure consistency and clarity in the database implementation.

### 

### Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) illustrates the overall data structure of CalorieCare. The ERD shows the key entities in the system and the relationships among them. ERD provides a solid foundation for implementing the system’s core functionalities and ensuring reliable, consistent, and scalable data management.

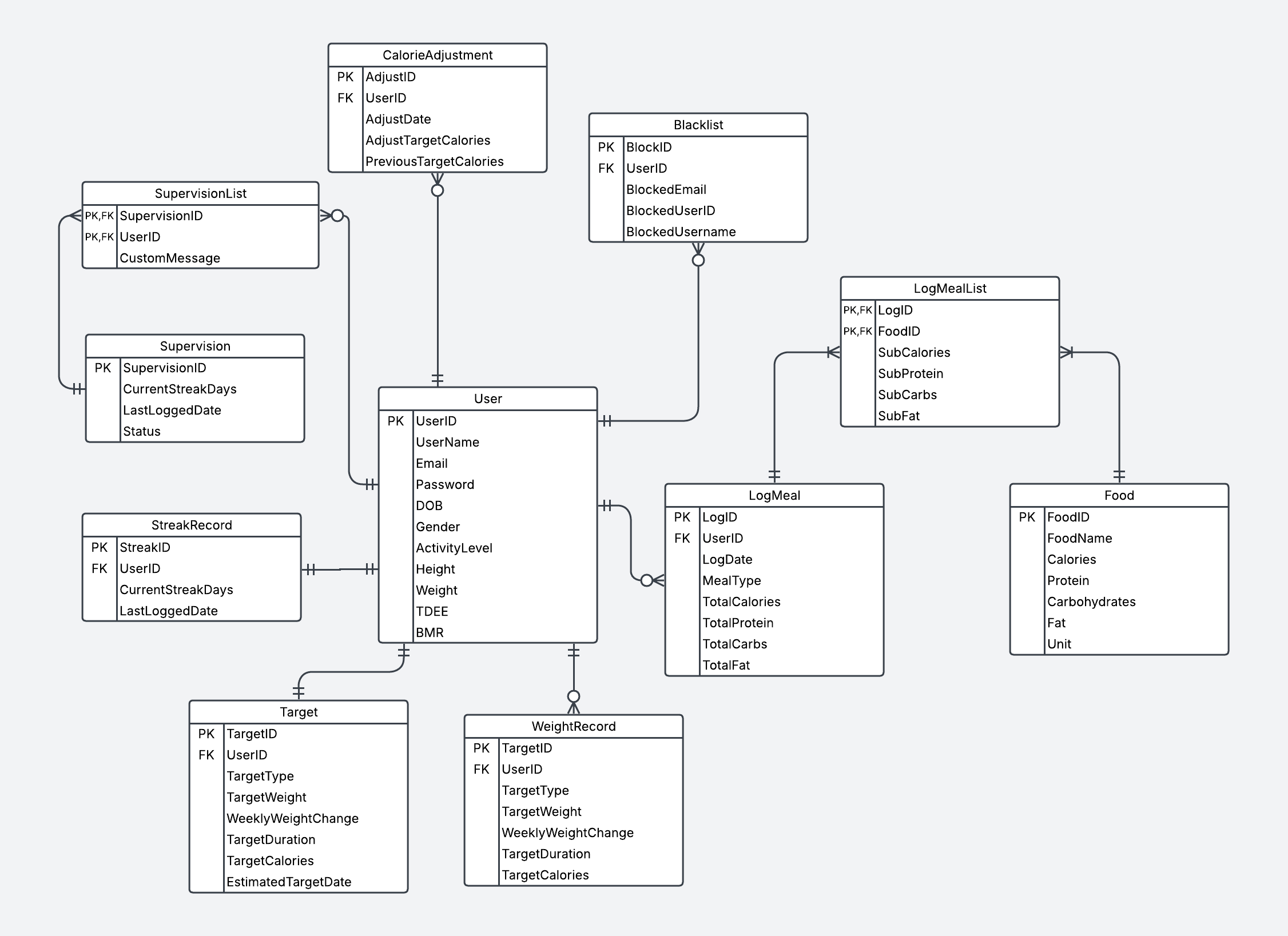


Figure 4.16: ERD Diagram

### Data Dictionary

This section presents the data dictionary of the CalorieCare, which defines the structure and attributes of each entity in the system. The data dictionary includes details such as data type, field length, constraints, format, default values, and descriptions for each field.

1. User

Table 4.1: Data Dictionary of User

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| UserID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique identifier for each user |
| UserName | VARCHAR | 20 | NOT NULL | X (20) | None | Name of the user |
| Email | VARCHAR | 50 | UNIQUE, NOT NULL | X (50) | None | User's email address |
| Password | VARCHAR | 20 | NOT NULL | X (20) | None | Encrypted user password |
| DOB | DATE | - | NOT NULL | DD/MM/YYYY | None | Date of birth |
| Gender | CHAR | 1 | NOT NULL | X | M / F | Gender: M=Male, F=Female |
| ActivityLevel | VARCHAR | 20 | NOT NULL | X (20) | Sedentary/  Lightly/  Moderately/  Active/  Extremely | Physical activity level of the user |
| Height | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Height in centimetres |
| Weight | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Current weight in kilograms |
| TDEE | DECIMAL | 6,2 | NULLABLE | 9999.99 | None | Total Daily Energy Expenditure |
| BMR | DECIMAL | 6,2 | NULLABLE | 9999.99 | None | Basal Metabolic Rate |

1. Target

Table 4.2: Data Dictionary of Target

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| TargetID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique target record ID |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| TargetType | VARCHAR | 20 | NOT NULL | X (20) | LoseWeight/  MaintainHealth/  GainWeight | Type of target (e.g., weight or calorie goal) |
| TargetWeight | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Desired target weight (optional) |
| WeeklyWeightChange | DECIMAL | 3,2 | NOT NULL | 9.99 | 0.50 | Weekly target weight loss/gain (e.g., 0.5 = -+0.5kg/week) |
| TargetDuration | INT | - | NOT NULL | 99 | None | Estimated duration (days) |
| TargetCalories | INT | 4 | NULLABLE | 9999 | None | Daily calorie target |
| EstimatedTargetDate | DATE | - | NULLABLE | DD/MM/YYYY | None | Estimated completion date |

1. LogMeal

Table 4.3: Data Dictionary of LogMeal

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| LogID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Meal log unique identifier |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| LogDateTime | DATETIME | - | NOT NULL | DD/MM/YYYY HH:MM | None | Timestamp of the logged meal |
| MealType | VARCHAR | 20 | NOT NULL | X (20) | Breakfast/Lunch/Dinner/Snack | Type of meal |
| TotalCalories | INT | 4 | NOT NULL | 9999 | None | Total calories in the meal |
| TotalProtein | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Total protein content |
| TotalCarbs | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Total carbohydrates |
| TotalFat | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Total fat content |

1. LogMealList

Table 4.4: Data Dictionary of LogMealList

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| LogID | VARCHAR | 10 | PK, FK, NOT NULL | X (10) | None | Meal log ID, foreign key |
| FoodID | VARCHAR | 10 | PK, FK, NOT NULL | X (10) | None | Food ID, foreign key |
| SubCalories | INT | 4 | NOT NULL | 9999 | None | Calories contributed by this food item |
| SubProtein | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Protein from this food item |
| SubCarbs | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Carbohydrates from this item |
| SubFat | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Fat from this food item |

1. Food

Table 4.5: Data Dictionary of Food

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| FoodID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique food identifier |
| FoodName | VARCHAR | 50 | NOT NULL | X (50) | None | Name of the food item |
| Calories | INT | 4 | NOT NULL | 9999 | None | Calories per unit |
| Protein | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Protein per unit |
| Carbohydrates | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Carbohydrates per unit |
| Fat | DECIMAL | 5,2 | NOT NULL | 999.99 | None | Fat per unit |
| Unit | VARCHAR | 10 | NOT NULL | X (10) | g/ml/serving | Measurement unit (e.g., gram, ml) |

1. CalorieAdjustment

Table 4.6: Data Dictionary of CalorieAdjustment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| AdjustID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Adjustment record ID |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| AdjustDate | DATE | - | NOT NULL | DD/MM/YYYY | None | Date of adjustment |
| AdjustTargetCalories | INT | 3 | NOT NULL | 999 | None | New calorie target |
| PreviousTargetCalories | INT | 3 | NULLABLE | 999 | None | Previous calorie target |

1. Supervision

Table 4.7: Data Dictionary of Supervision

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| SupervisionID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Supervision record ID |
| CurrentStreakDays | INT | 3 | NOT NULL | 999 | None | Number of current streak days |
| LastLoggedDate | DATE | - | NOT NULL | DD/MM/YYYY | None | Last logged date |
| Status | VARCHAR | 20 | NOT NULL | X(10) | None | Active / Inactive |

1. SupervisionList

Table 4.8: Data Dictionary of SupervisionList

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| SupervisionID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Supervision ID, foreign key |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | User ID, foreign key |
| CustomMessage | VARCHAR | 255 | NULLABLE | X (255) | Null | Custom motivational or reminder message |

1. StreakRecord

Table 4.9: Data Dictionary of StreakRecord

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| StreakID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique streak record ID |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| CurrentStreakDays | INT | 3 | NOT NULL | 999 | None | Number of consecutive days logged |
| LastLoggedDate | DATE | - | NOT NULL | DD/MM/YYYY | None | Most recent log date |

1. WeightRecord

Table 4.10: Data Dictionary of WeightRecord

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| StreakID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique streak record ID |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| CurrentStreakDays | INT | 3 | NOT NULL | 999 | None | Number of consecutive days logged |
| LastLoggedDate | DATE | - | NOT NULL | DD/MM/YYYY | None | Most recent log date |
| MaxStreakDays | INT | 3 | NOT NULL | 999 | None | Longest recorded streak |

1. Blacklist

Table 4.11: Data Dictionary of Blacklist

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Length** | **Constraint** | **Format** | **Default Value** | **Description** |
| StreakID | VARCHAR | 10 | PK, NOT NULL | X (10) | None | Unique streak record ID |
| UserID | VARCHAR | 10 | FK, NOT NULL | X (10) | None | Foreign key referencing user |
| CurrentStreakDays | INT | 3 | NOT NULL | 999 | None | Number of consecutive days logged |
| LastLoggedDate | DATE | - | NOT NULL | DD/MM/YYYY | None | Most recent log date |
| MaxStreakDays | INT | 3 | NOT NULL | 999 | None | Longest recorded streak |

## Report Design

This section describes the report design of CalorieCare. Reports are essential for helping users monitor health progress and gain insights into dietary habits. The system generates meaningful reports based on logged data, which are presented in a clear and understandable format to support user decision-making.

### Calorie Intake Report

The Calorie Intake Report provides a clear and visual summary of the user's calorie consumption. This report helps the user track how much has been eaten throughout the intake. This report is designed to help the user stay aware of eating habits and remain on track toward achieving a balanced and healthy lifestyle.

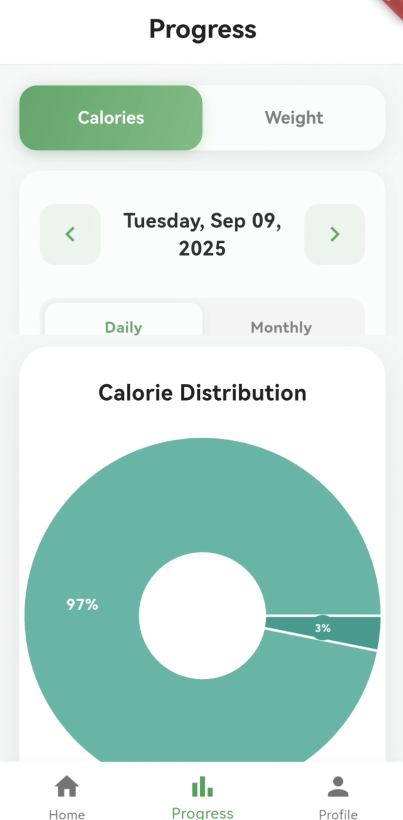


Figure 4.17: Calorie Intake Report Interface

### Weight Report

The Weight Report provides a visual overview of the user's weight progress over time. This report allows the user to monitor weight changes through a line graph and a timeline view, helping track progress toward health or fitness goals. This report supports the user in staying motivated and aware of long-term progress, making it easier to evaluate the effectiveness of lifestyle habits or dietary adjustments.

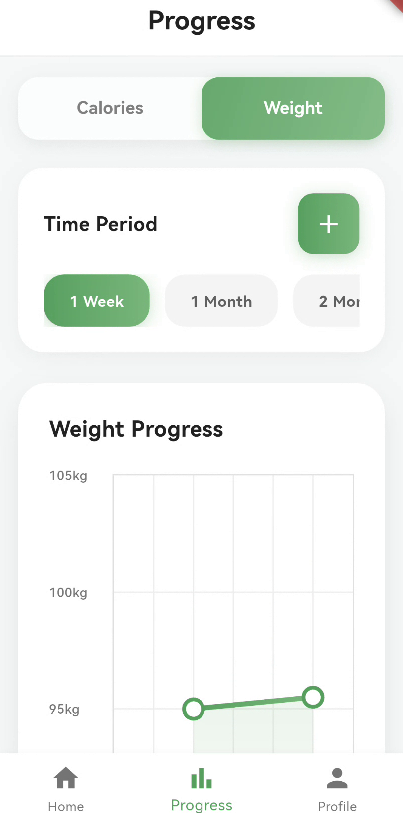


Figure 4.18: Weight Report Interface

## Security Design

This section outlines the security design of CalorieCare to ensure the confidentiality, integrity, and availability of user data. The system implements security measures to protect sensitive information such as login credentials, personal data, and dietary logs to prevent unauthorized access and data breaches.

### Password Hashing

Password hashing is a widely used technique to enhance the security of user credentials. Rather than storing user passwords in plain text, the system uses a cryptographic hash function to convert each password into an irreversible string of characters. This means that even if the database is compromised, the original password cannot be easily recovered from the stored hash, thus significantly reducing the risk of exposing sensitive user information.

In CalorieCare, password hashing is applied to protect user accounts. When a user registers, the password users provide is hashed before being saved to the database. During login, the system hashes the entered password again and compares it with the existing hashed value in the database. If the two hashes match, the user is granted access; otherwise, access is denied. This process ensures that passwords are never stored or transmitted in plain text, providing a secure authentication mechanism for the system.

### Reset Password

To enhance user account security and accessibility, CalorieCare provides a secure password reset mechanism through email. This feature ensures that users who have forgotten passwords can regain access to accounts without compromising the integrity or confidentiality of user data.

When a user requests a password reset, the system initiates a verification process by sending a unique, time-limited reset link to the registered email address associated with the account. This link contains a secure token generated by the system, which is validated upon access to prevent unauthorized resets. The token is valid only for a short period (e.g., 15 minutes), and the token becomes invalid after use or expiration to prevent misuse.

After clicking the reset link, the user is redirected to a secure interface to enter and confirm a new password. The new password is then hashed using the same cryptographic hash function employed in the registration and login processes, and the updated hash is stored in the database.

## Process Design

The process design outlines the workflow of key system functions CalorieCare. This process design helps visualize how users interact with the system and how the system processes different user actions. By mapping out the processes, developers and stakeholders can better understand the system's logic, ensure smooth user experiences.

### Activity Diagram

This section presents activity diagrams to illustrate the sequence of activities for core features in the system. Each diagram represents the flow of actions from the user’s perspective and the system's responses, highlighting decision points, parallel processes, and outcomes.

#### Register User Activity Diagram

The following activity diagram illustrates the user interactions within the Register User Function.

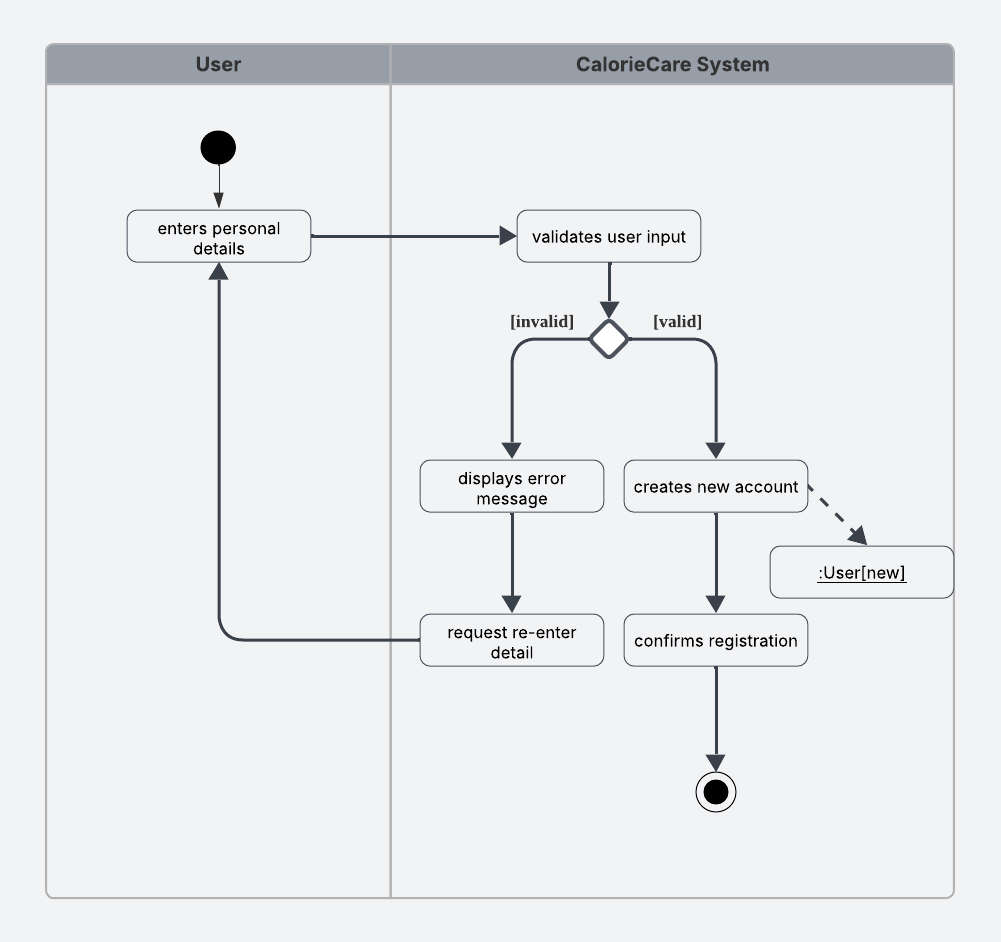


Figure 4.19: Activity Diagram for Register User Function

#### Set Target Activity Diagram

The following activity diagram illustrates the user interactions within the Set Target Function.

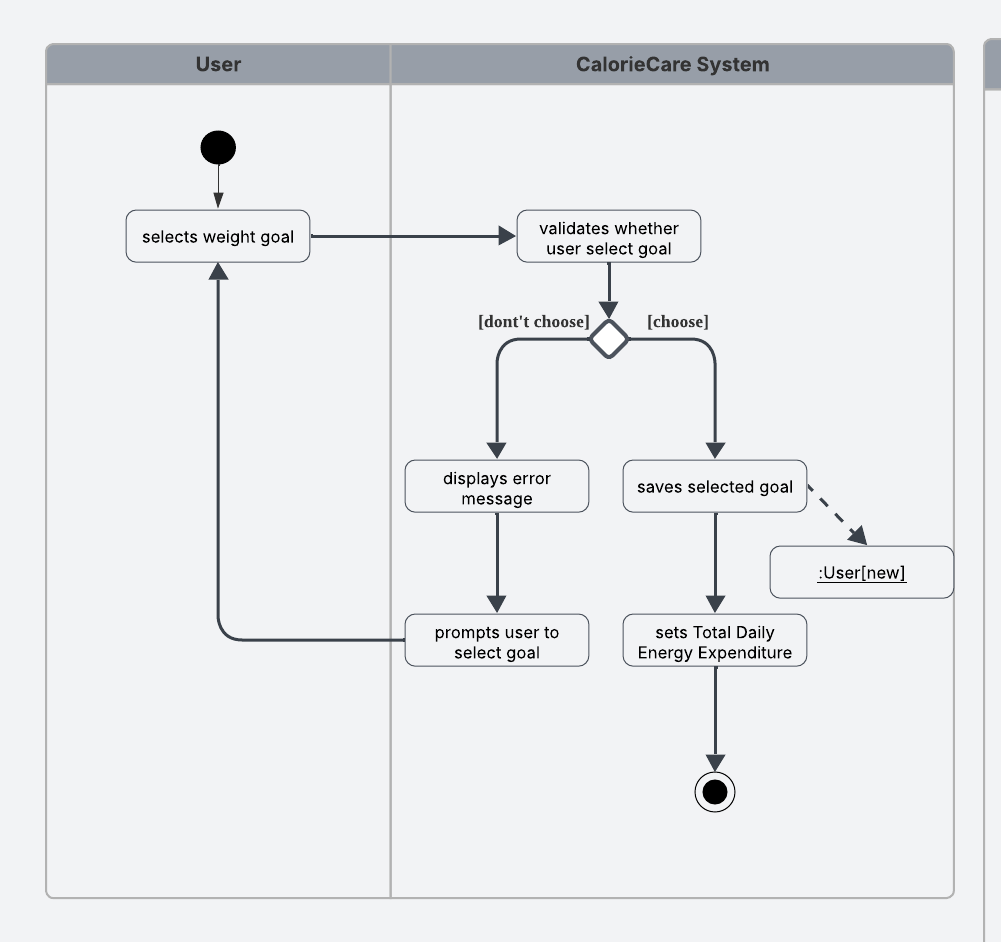


Figure 4.20: Activity Diagram for Set Target Function

#### Calculate BMI Activity Diagram

The following activity diagram illustrates the user interactions within the Calculate BMI Function.

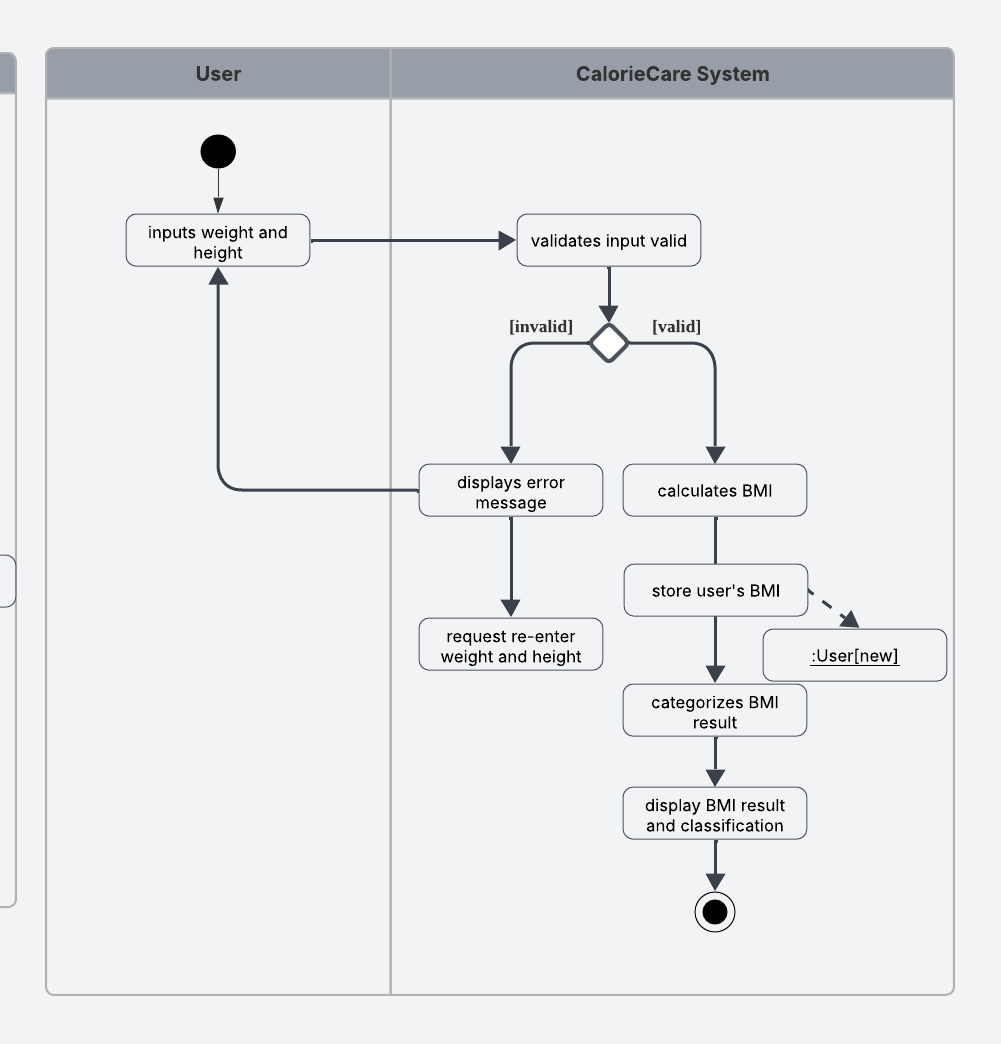


Figure 4.21: Activity Diagram for Calculate BMI Function

#### Calculate Target Calorie Activity Diagram

The following activity diagram illustrates the user interactions within the Calculate Target Calorie Function.

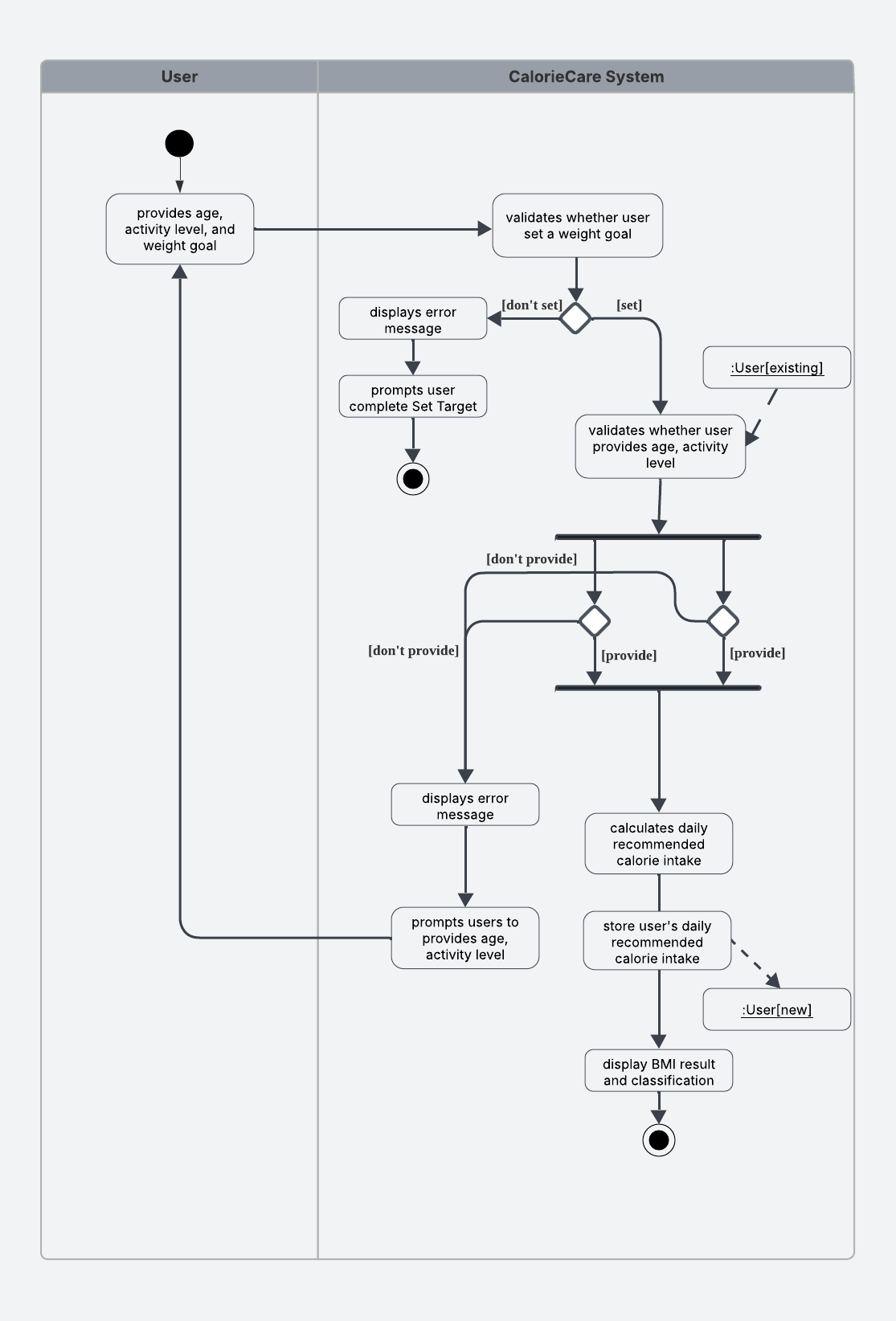


Figure 4.22: Activity Diagram for Calculate Target Calorie Function

#### Authenticate and Authorize User Activity Diagram

The following activity diagram illustrates the user interactions within the Authenticate and Authorize User Function.

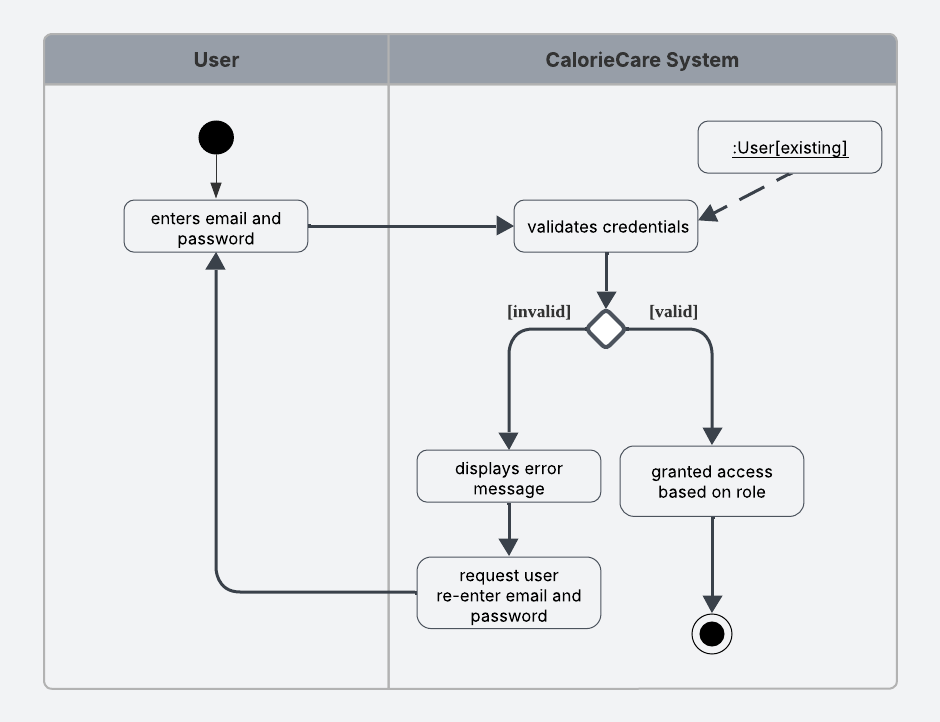


Figure 4.23: Activity Diagram for Authenticate and Authorize User Function

#### Recover Password Activity Diagram

The following activity diagram illustrates the user interactions within the Recover Password Function.

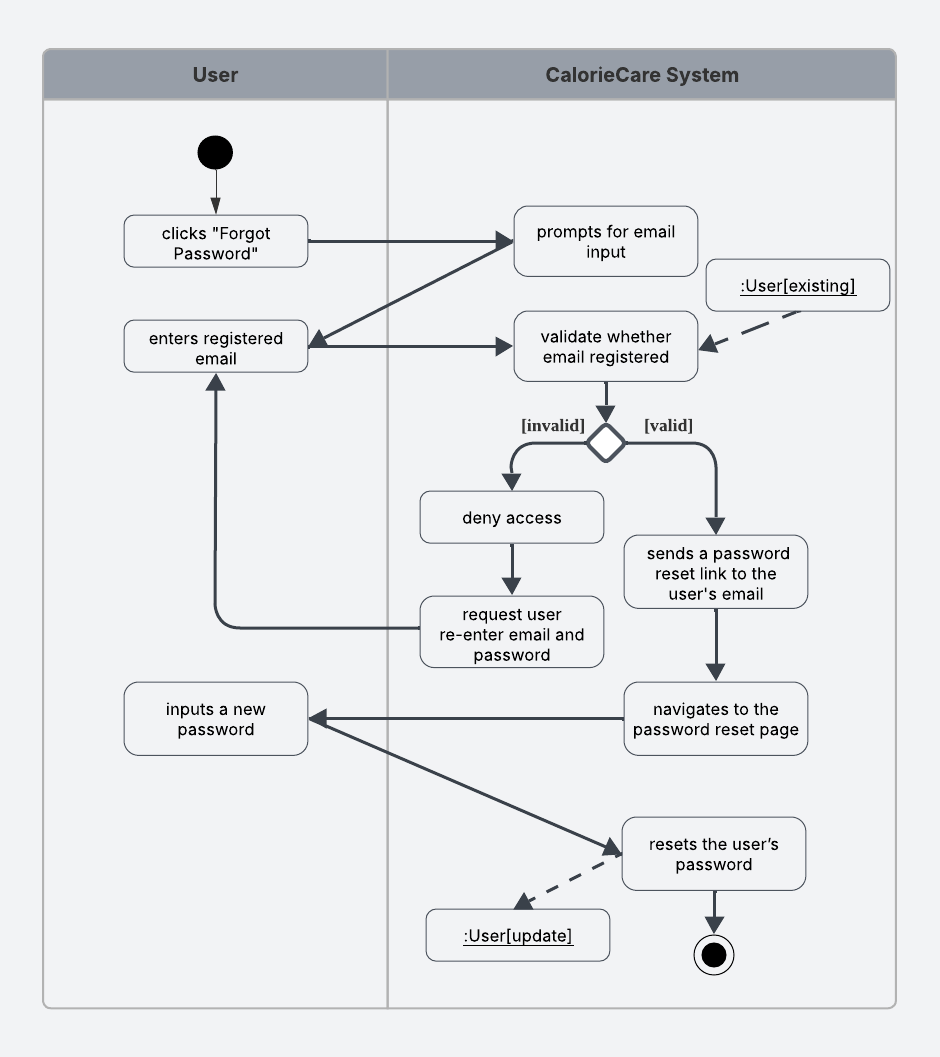


Figure 4.24: Activity Diagram for Recover Password Function

#### Log Food Activity Diagram

The following activity diagram illustrates the user interactions within the Log Food Function.

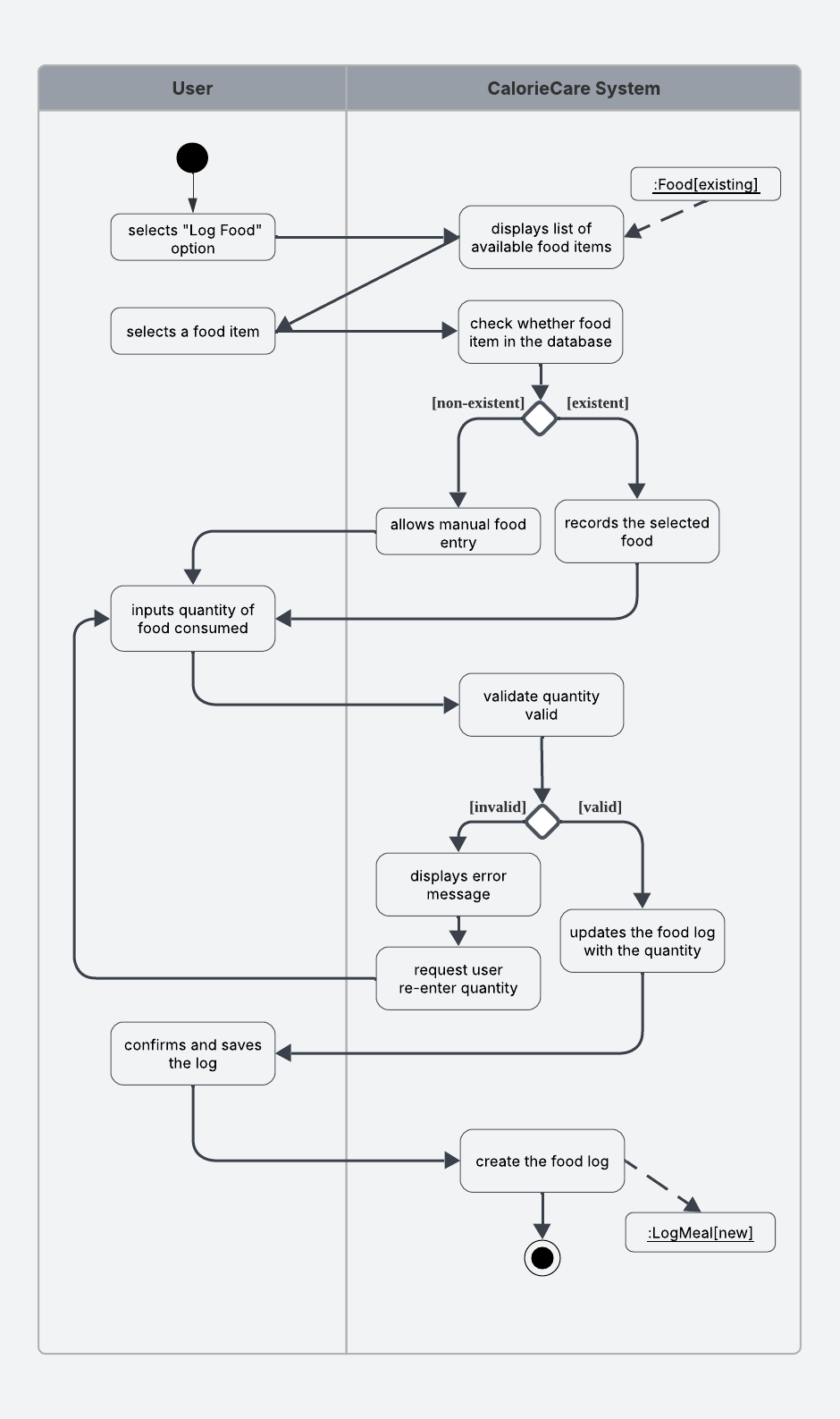


Figure 4.25: Activity Diagram for Log Food Function

#### Recognize Food Activity Diagram

The following activity diagram illustrates the user interactions within the Recognize Food Function.

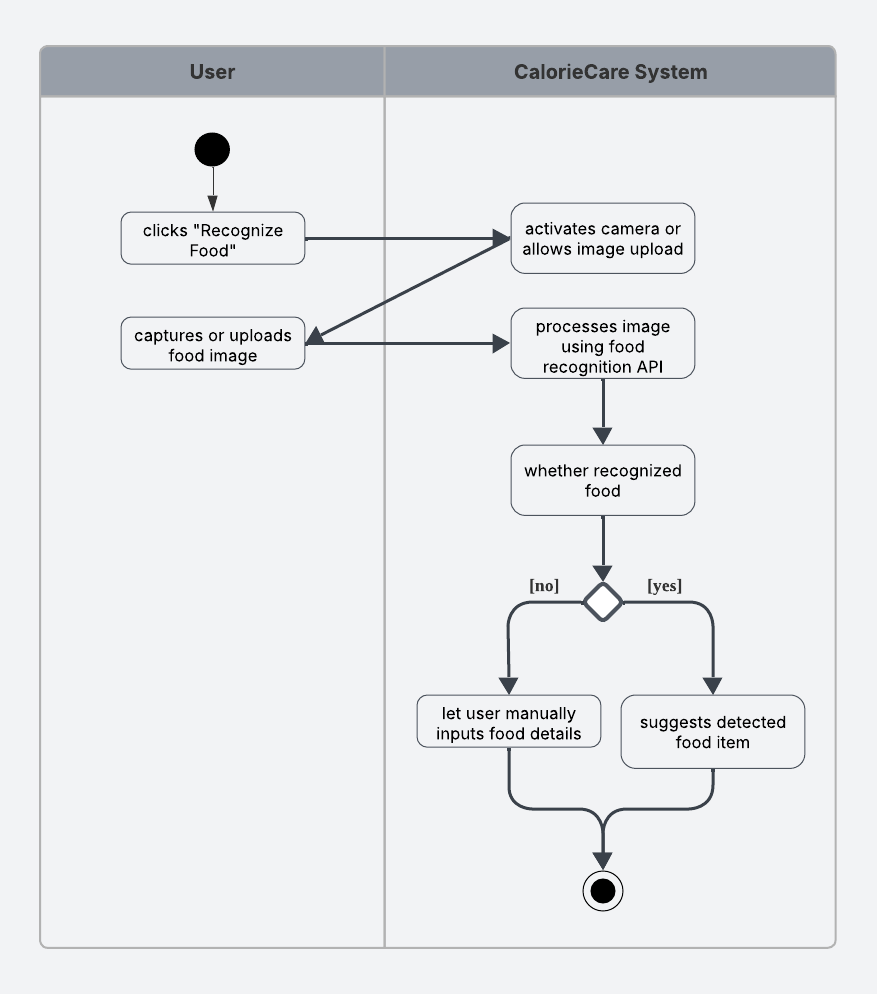


Figure 4.26: Activity Diagram for Recognize Food Function

#### Calculate Calorie Activity Diagram

The following activity diagram illustrates the user interactions within the Calculate Calorie Function.

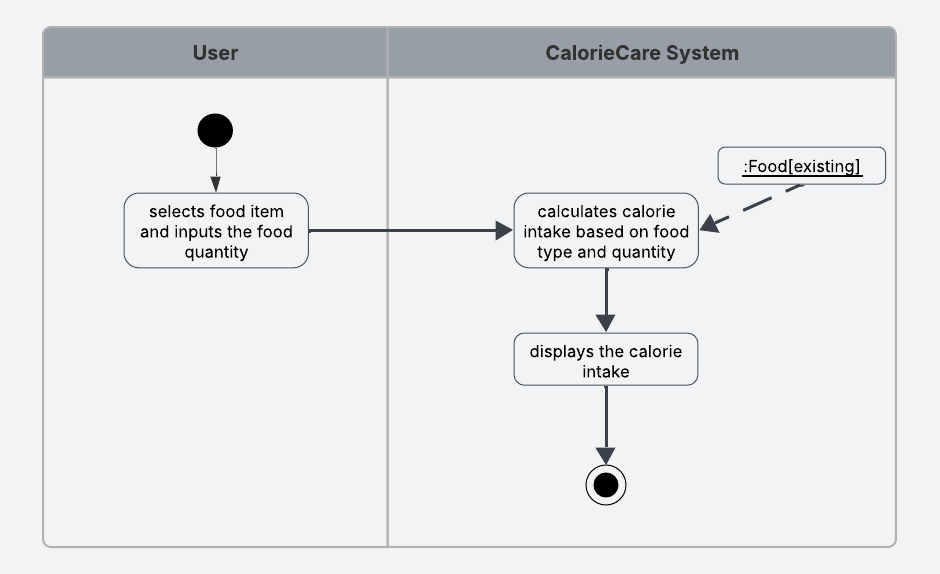


Figure 4.27: Activity Diagram for Calculate Calorie Function

#### Dynamic Calorie Target Adjustment Activity Diagram

The following activity diagram illustrates the user interactions within the Dynamic Calorie Target Adjustment Function.

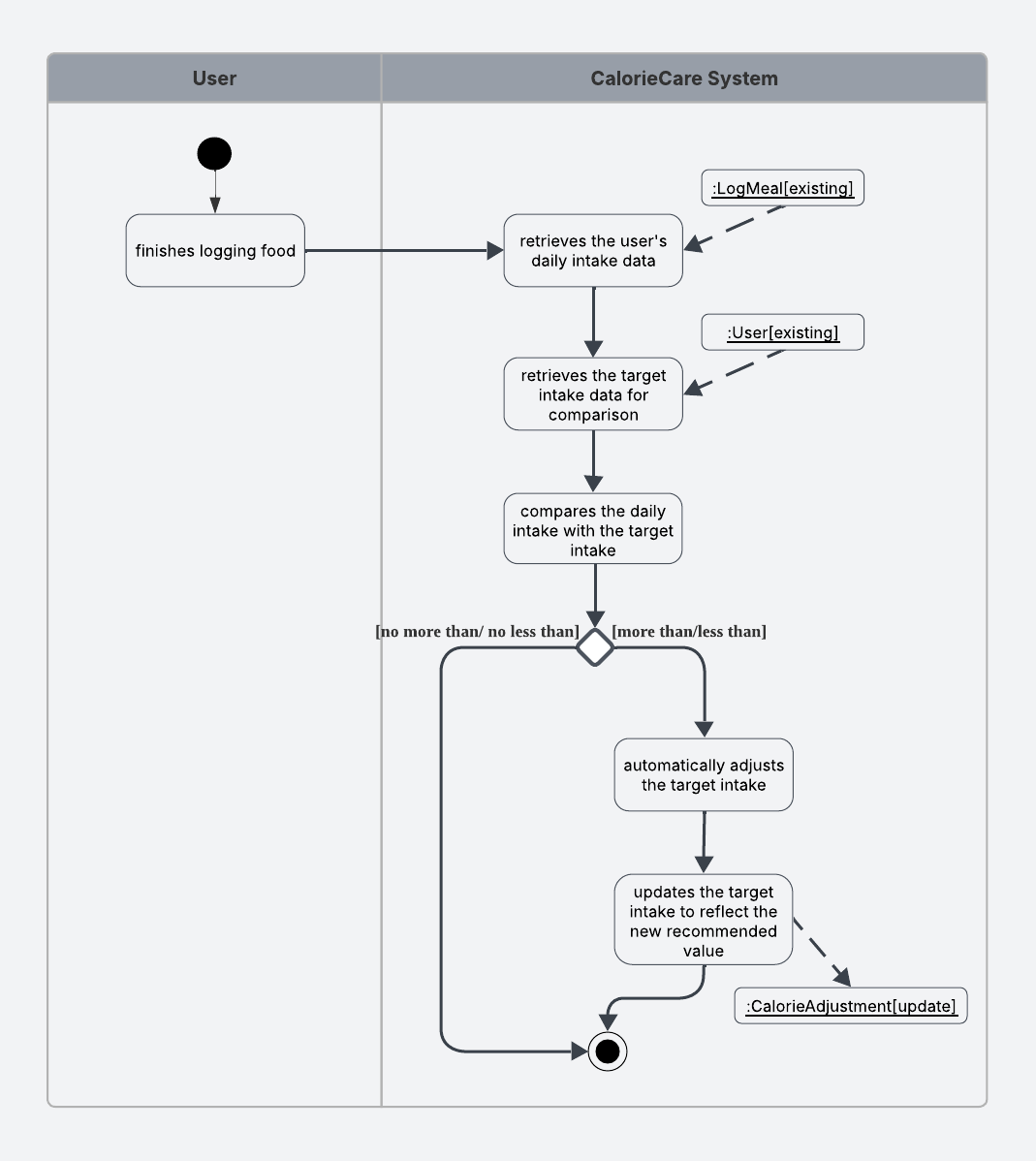


Figure 4.28: Activity Diagram for Dynamic Calorie Target Adjustment Function

#### Track Streak Activity Diagram

The following activity diagram illustrates the user interactions within the Track Streak Function.

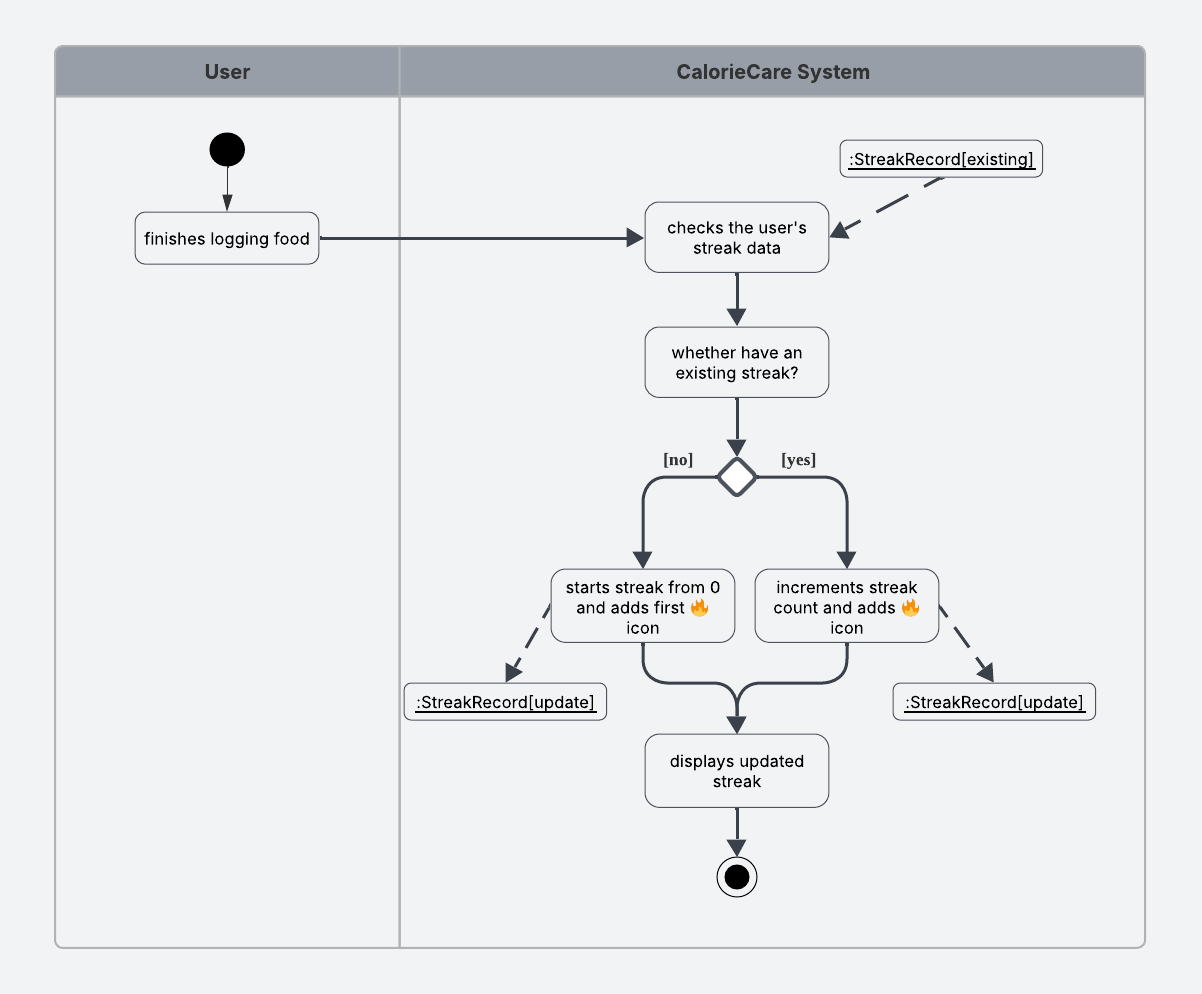


Figure 4.29: Activity Diagram for Track Streak Function

#### Invite Friend Activity Diagram

The following activity diagram illustrates the user interactions within the Invite Friend Function.

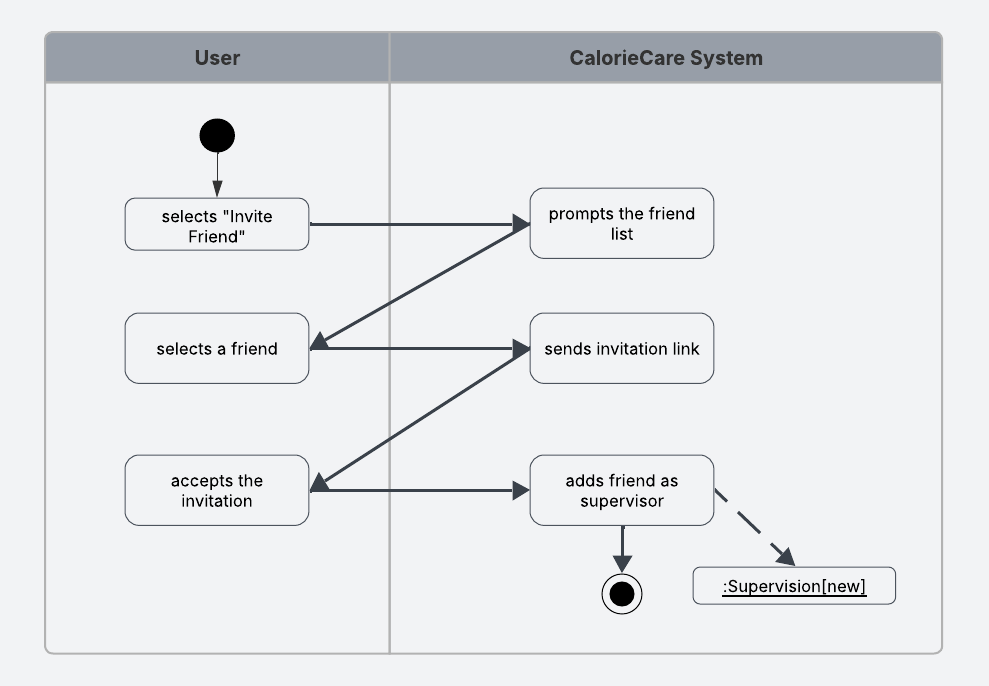


Figure 4.30: Activity Diagram for Invite Friend Function

#### Alert Friend Activity Diagram

The following activity diagram illustrates the user interactions within the Alert Friend Function.

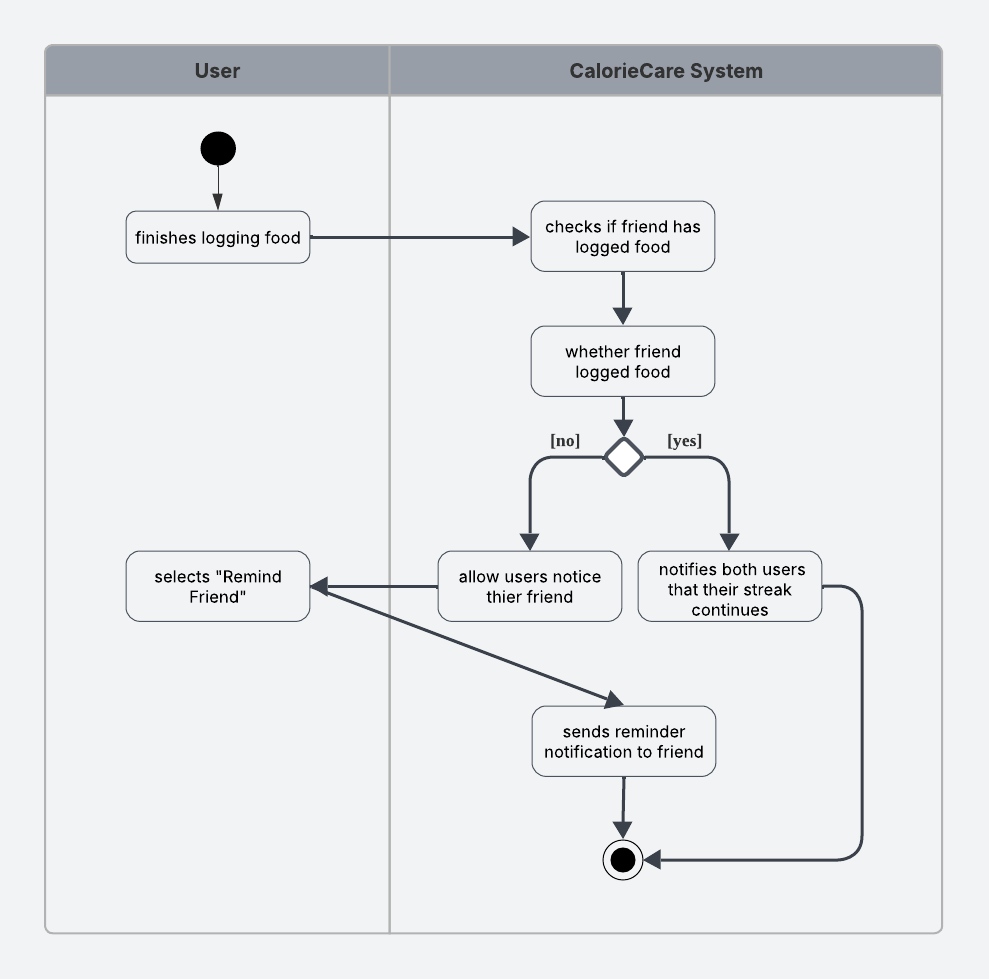


Figure 4.31: Activity Diagram for Alert Friend Function

#### View Calorie Intake Activity Diagram

The following activity diagram illustrates the user interactions within the View Calorie Intake Function.

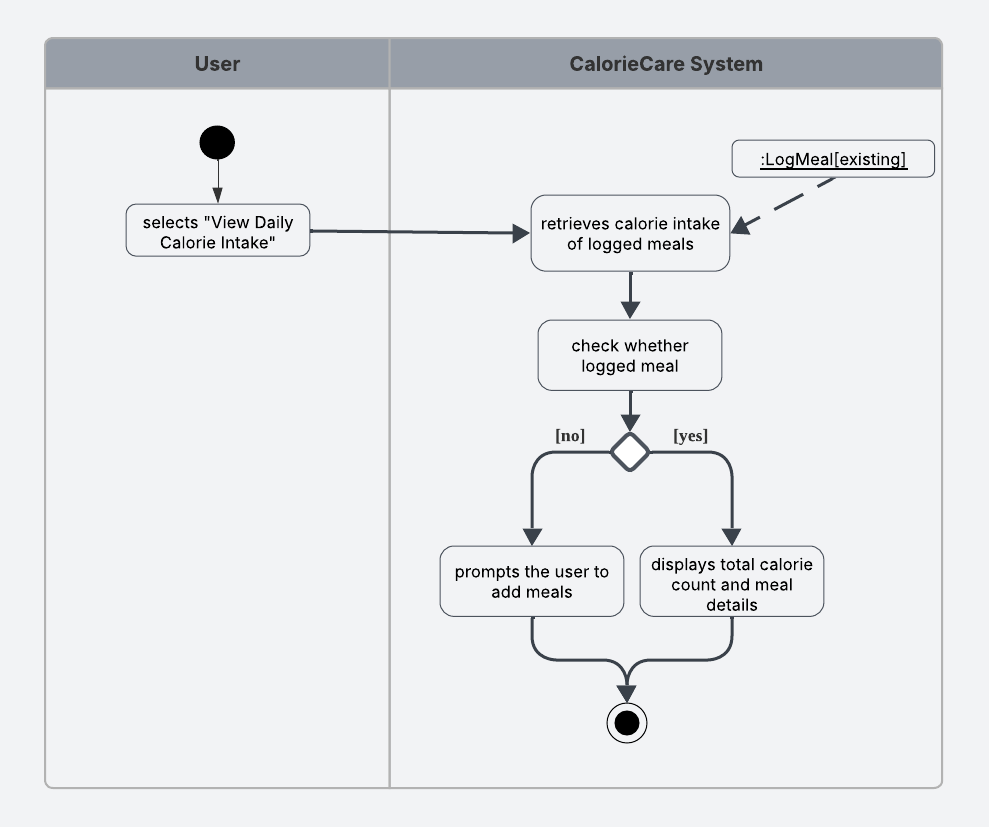


Figure 4.32: Activity Diagram for View Calorie Intake Function

#### View Weight Report Activity Diagram

The following activity diagram illustrates the user interactions within the View Weight Report Function.

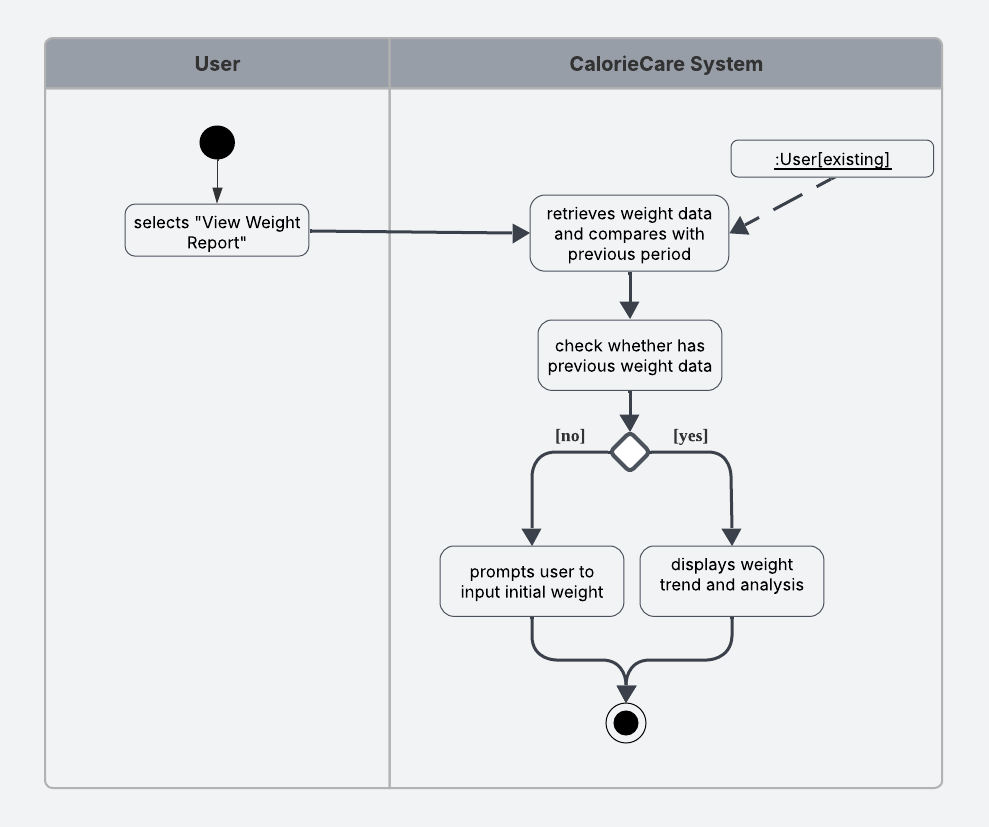


Figure 4.33: Activity Diagram for View Weight Report Function

## Software Architecture Design

This system adopts a three-layer architecture model that includes the User Interface (UI) Layer, Business Logic Layer, and Data Layer. The UI Layer handles all user interface components and interactions. This layer includes ViewModels which prepare data for display, Controllers that manage user input, and User Interaction Logic governing how users engage with the system. This layer focuses solely on presentation without containing business rules.

The Business Layer forms the core logic of CalorieCare. This layer contains Use Cases defining nutritional operations, Interactors coordinating workflows, Business Rules for diet-specific calculations, Validation for checking meal plans or user inputs, and State Management to track dietary progress or user goals. This layer ensures proper application functionality while remaining independent from UI or data details.

The Data Layer manages all nutrition-related data persistence. Repositories provide abstract access to stored information, while the Database handles actual storage of user profiles, meal plans, and nutritional data. This separation allows easy future changes to storage mechanisms without affecting other layers.

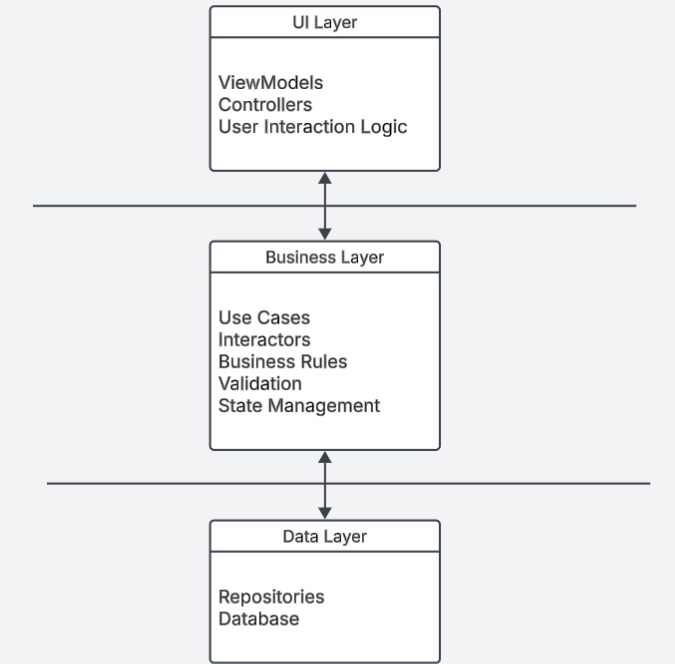


Figure 4.34: Software Architecture Design

## Algorithm Design

This section outlines the key algorithms used in CalorieCare. The algorithms are designed to support essential system functionalities such as food image recognition, calorie target adjustment, and streak tracking. Each algorithm is presented in pseudocode form to illustrate the logical steps and flow of operations.

### Recognize Food Function

This section provides the pseudocode for the Recognize Food Function, which allows users to identify food items using images captured via camera or selected from the gallery.

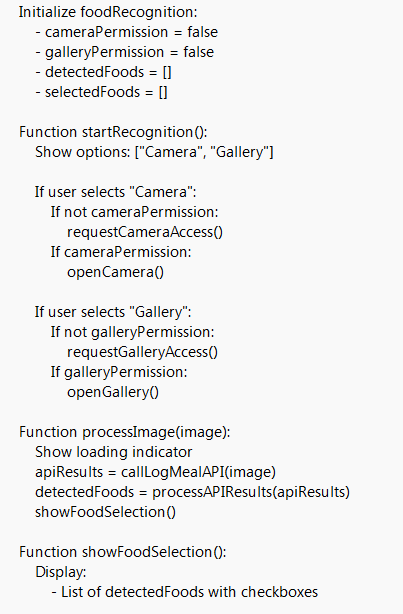


Figure 4.35: Algorithm Design for Recognize Food Function

### Dynamic Target Adjustment Function

The algorithm evaluates the difference between the user’s actual daily intake and the set target. If intake is greater than the target, the following day’s target is reduced proportionally; if intake is less than the target, the following day’s target is increased. The adjustment magnitude depends on the deviation, but the system ensures that the recalculated target always falls within safe health limits: never below BMR or 1,200 kcal/day for women and 1,500 kcal/day for men, and never above TDEE plus 500 kcal (Publishing, 2020; Larson-Meyer et al., 2022; NHS, 2021). This ensures the calorie goals adapt to user behavior while maintaining nutritional safety.

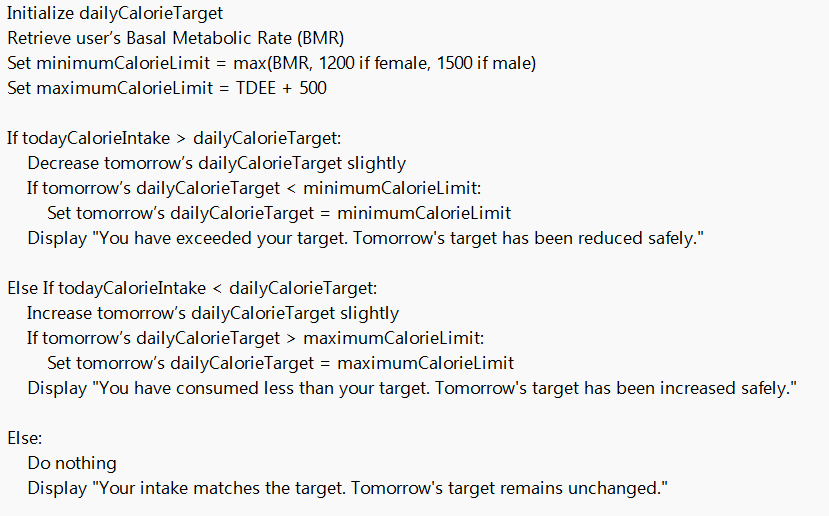


Figure 4.36: Algorithm Design for Adjust Target Function

### Track Streak Function

This section describes the pseudocode for the Track Streak Function, which monitors the user's daily food logging activity to maintain motivation through streaks.

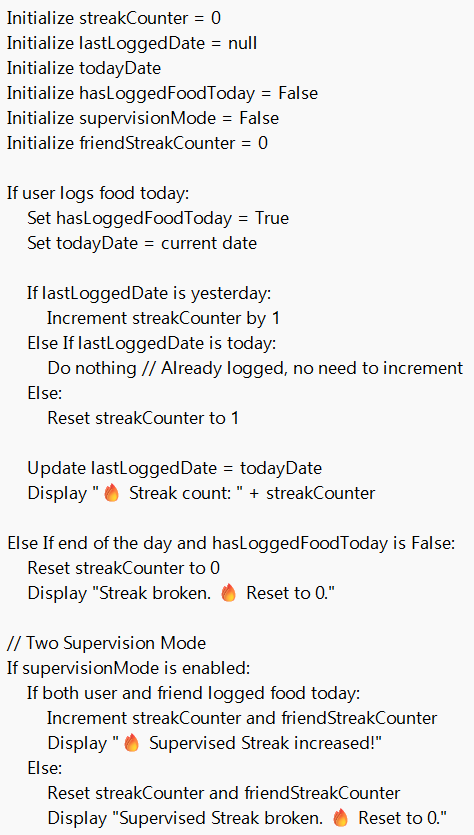


Figure 4.37: Algorithm Design for Track Streak Function

## Chapter Summary & Evaluation

This chapter outlines the overall system design of CalorieCare, encompassing the user interface layout, data structure, report generation, security mechanisms, process workflows, software architecture, and algorithm logic. The UI design aims to provide a user-friendly and seamless experience, while the data design, supported by an ERD and detailed data dictionary, ensures reliable and efficient data management. The report design offers meaningful summaries to help users monitor nutritional intake and progress. Security is addressed through password hashing, ensuring user credentials are stored securely. The three-layer software architecture promotes modularity by separating the interface, business logic, and data layers. Key algorithms are presented in pseudocode, covering essential modules such as food recognition, calorie adjustment, and streak tracking with supervision support. In conclusion, this chapter has successfully established a solid foundation for building a secure, scalable, and health-oriented platform. The structured and modular design supports future enhancements, making the system adaptable to evolving user needs and new health tracking features.

Chapter 5

**Implementation and Testing**

1. **Implementation and Testing**

This section explains the implementation of the CalorieCare system. The description covers the main components of the Flutter framework, the integration with the Firebase Firestore database, and the logic applied in the core modules such as AI-based food recognition, dynamic calorie target adjustment, and streak tracking. Code snippets and explanations are provided to present a clear view of the system development.

* 1. **Implementation / Coding**

This part presents information on the implementation of the CalorieCare system. The focus is on the coding of the main features, including the Smart Food Tracking module, Dynamic Target Adjustment module, and Motivation module. Explanations are supported with code snippets and detailed descriptions of the underlying logic.

* + 1. **Smart Food Tracking module**

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

* + 1. **Dynamic Target Adjustment module**

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

* + 1. **Motivation module**

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

## Testing Strategies / Approaches

To ensure the reliability, functionality, and quality of the CalorieCare system, a multi-layered testing strategy was employed. This strategy is composed of three distinct phases: Unit Testing, Integration Testing, and Acceptance Testing. Each phase focuses on a different level of the system to progressively identify and resolve issues.

### Unit Testing

Unit testing checks individual components of the system in isolation to ensure each part performs its intended function correctly. In CalorieCare, each module was tested separately by executing its main functions and checking the expected results in the database and user interface.

For the Smart Food Tracking module, tests were conducted by uploading various meal images to verify that the AI-powered recognition returned the correct food type and calorie value. Manual entries were also tested to ensure that the data was stored correctly in Firebase Firestore.

For the Dynamic Target Adjustment module, unit tests simulated scenarios where daily intake was above, below, or equal to the calorie target. The tests confirmed that the system recalculated the next day’s target correctly and respected safety boundaries (not below BMR or 1,200 kcal/day for women and 1,500 kcal/day for men, and not above TDEE + 500 kcal).

For the Motivation module (Streak tracking), tests checked whether logging all three meals in a day increased the streak count correctly and displayed the 🔥 marker. Additional tests confirmed that streaks reset when a day’s logging was incomplete. Furthermore, the Supervisor Streak feature was tested by pairing two user accounts. The system was verified to correctly grant a shared streak reward when both users completed their daily logs, and to send reminder notifications if one user failed to log meals on time.

These unit tests ensured that the critical functions of each module operated correctly before moving on to integration.

### Integration Testing

Integration testing in the CalorieCare system was conducted after unit testing to ensure all modules worked together as a whole. The focus was on verifying data flow and interactions across modules.

For the Smart Food Tracking module, integration testing confirmed that recognized meals were automatically stored in Firebase and displayed in the user’s calorie history. The interaction between the image recognition API and Firestore was validated to ensure consistent data accuracy.

For the Dynamic Target Adjustment module, tests confirmed that daily intake data retrieved from Firestore was correctly compared against the user’s target. Updated values for the next day’s target were stored back into the database and reflected in the user interface, ensuring end-to-end consistency.

For the Motivation module, tests verified that meal logs recorded in Firestore triggered streak updates automatically. Integration tests confirmed that both individual streaks and supervisor streaks worked correctly. Shared streak rewards were validated when both users logged meals successfully, and the reminder mechanism was confirmed to function when one user missed logging.

Overall, integration testing confirmed smooth data flow between modules, consistent profile updates, and accurate tracking of calorie intake, goals, and streak rewards.

### User Acceptance Testing

User Acceptance Testing (UAT) was carried out to verify that the CalorieCare system met user expectations and project requirements, while delivering a practical and usable experience.

A group of target users was invited to test the system in realistic scenarios, including food logging, image recognition, calorie adjustment, and streak tracking. Testers were asked to complete typical daily activities, such as recording meals with photos, checking updated calorie targets the next day, and maintaining streaks with or without a supervisor partner.

Feedback collected during UAT focused on ease of use, system responsiveness, and overall usability. Users reported that the image recognition feature significantly reduced the effort of manual logging, while the automatic calorie adjustment provided reassurance that their targets adapted safely to their eating patterns. The streak and supervisor streak features were particularly highlighted as motivating and engaging, encouraging users to remain consistent in their logging habits.

Based on the UAT feedback, minor refinements were applied to improve notification clarity and user interface responsiveness. The overall results confirmed that the system provided a smooth, reliable, and user-friendly experience.

## Test Plan & Test Cases

This section presents the test plan and detailed test cases developed to validate the functionalities of the StudyBuddy system, covering all six modules, from the Gamification Module to the Study Report Module. The test plan establishes a structured approach to testing, while the test cases define the steps, input data, and expected outcomes required to verify that each StudyBuddy feature meets its specified requirements and operates correctly.

### User Management Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1001 |
| **Module:** | User Management Module | | |
| **Actor(s):** | New User, Existing User | | |
| **Pre-requisites:** | * Internet connection available * Firebase services are operational * Application is installed and launched | | |
| **Dependencies:** | Firebase Authentication and Firestore services must be running | | |

**Test Case 1001:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | Register User: Verify new user registration with valid data for different goals. | 1. Launch the app and navigate to the registration page. 2. Select goal: "Lose Weight", "Gain Weight", or "Maintain Weight". 3. Enter valid user details (gender, DOB, height, weight, target weight if applicable, activity level). 4. Enter a unique email and a strong password. 5. Submit the registration form. | 1. User is successfully created in Firebase Authentication. 2. A new document is created in the 'User' collection with correct data. 3. A new document is created in the 'Target' collection with appropriate TargetType ('lose'/'gain'/'maintain'), TargetWeight, and WeeklyWeightChange. 4. Daily calorie target is calculated correctly (TDEE±calories based on goal). 5. User is redirected to the home page. | 1. User is successfully created in Firebase Authentication. 2. A new document is created in the 'User' collection with correct data. 3. A new document is created in the 'Target' collection with appropriate TargetType ('lose'/'gain'/'maintain'), TargetWeight, and WeeklyWeightChange. 4. Daily calorie target is calculated correctly (TDEE±calories based on goal). 5. User is redirected to the home page. | P |  |
|  | Register User (Existing Email): Verify registration with an existing email. | 1. Navigate to the registration page. 2. Enter an email that is already registered. 3. Fill in other fields with valid data. 4. Submit the form. | An error message "This email address is already registered" is displayed. Registration fails. | An error message "This email address is already registered" is displayed. Registration fails. | P |  |
|  | Register User (Goal Validation): Verify goal validation for users with different BMI categories selecting inappropriate goals. | 1. Start registration with different BMI scenarios:   Underweight (BMI < 18.5): height 170cm, weight 50kg, select "Lose Weight"  Overweight (BMI > 25): height 170cm, weight 85kg, select "Gain Weight"  Obese (BMI ≥ 30): height 160cm, weight 95kg, select "Gain Weight"  Normal BMI (18.5-24.9): height 170cm, weight 65kg, select any goal   1. Proceed to the next step. | 1. For inappropriate goal selections, a warning dialog appears with goal recommendation. 2. User can accept recommendation (goal changes to 'lose'/'gain' as appropriate). 3. For normal BMI, no warning appears and selected goal is accepted. 4. User proceeds to next step with validated goal. | 1. For inappropriate goal selections, a warning dialog appears with goal recommendation. 2. User can accept recommendation (goal changes to 'lose'/'gain' as appropriate). 3. For normal BMI, no warning appears and selected goal is accepted. 4. User proceeds to next step with validated goal. | P |  |
|  | Set Target Weight: Verify target weight selection for different goals. | 1. During registration, test different goal scenarios:   Gain Weight: current weight 60kg, target weight 65kg  Lose Weight: current weight 80kg, target weight 70kg   1. Complete registration for each scenario | 1. Target document has correct TargetType ('gain'/'lose') and TargetWeight. 2. Gain goal: slider allows selection above current weight, weekly change 0.25kg. 3. Lose goal: slider allows selection below current weight only, weekly change 0.5kg. 4. Maintain goal: no target weight required. | 1. Target document has correct TargetType ('gain'/'lose') and TargetWeight. 2. Gain goal: slider allows selection above current weight, weekly change 0.25kg. 3. Lose goal: slider allows selection below current weight only, weekly change 0.5kg. 4. Maintain goal: no target weight required. | P |  |
|  | Target Weight Validation (Unhealthy BMI): Verify warning when target weight results in unhealthy BMI. | 1. During registration, enter height: 170cm, current weight: 70kg. 2. Select "Lose Weight" goal. 3. Set target weight to 45kg (BMI < 18.5). | A warning message appears indicating the target weight would result in an unhealthy BMI. The slider shows visual feedback (red color) for unhealthy range. | A warning message appears indicating the target weight would result in an unhealthy BMI. The slider shows visual feedback (red color) for unhealthy range. | P |  |
|  | Target Weight Constraints: Verify target weight constraints for different goals. | 1. During registration, test constraint scenarios:   Lose Weight: current weight 70kg, try to set target above 70kg  Gain Weight: current weight 60kg, try to set target below 60kg   1. Attempt to save invalid target weights. | 1. Lose goal: prevents selection above current weight (max = current weight). 2. Gain goal: prevents selection below current weight (min = current weight). 3. Appropriate validation messages are shown for invalid selections. | 1. Lose goal: prevents selection above current weight (max = current weight). 2. Gain goal: prevents selection below current weight (min = current weight). 3. Appropriate validation messages are shown for invalid selections. |  |  |
|  | Calculate BMI: Verify BMI calculation during registration. | 1. During registration, enter height: 175 cm and weight: 70 kg. 2. Proceed to the BMI calculation step. | The BMI is calculated correctly as 22.9 and displayed on the BMI screen. | The BMI is calculated correctly as 22.9 and displayed on the BMI screen. | P |  |
|  | Calculate Target Calorie: Verify target calorie calculation for different goals. | Complete the registration process with specific data for different goals (lose/gain/maintain weight). | The daily calorie target is calculated correctly: TDEE-550 for loss, TDEE+275 for gain, TDEE for maintain, displayed on summary screen and homepage. | The daily calorie target is calculated correctly: TDEE-550 for loss, TDEE+275 for gain, TDEE for maintain, displayed on summary screen and homepage. | P |  |
|  | Calculate Target Date: Verify target date calculation based on goal and weight difference. | Complete registration with lose weight goal (current: 80kg, target: 70kg, 0.5kg/week loss rate). | The estimated target date is calculated as 20 weeks from registration date (10kg ÷ 0.5kg/week). | The estimated target date is calculated as 20 weeks from registration date (10kg ÷ 0.5kg/week). | P |  |
|  | Activity Level Impact on TDEE: Verify different activity levels affect calorie calculations. | 1. Register two users with identical data except activity level (one sedentary, one very active). 2. Compare their TDEE and daily calorie targets. | The very active user should have significantly higher TDEE and daily calorie target compared to the sedentary user. | The very active user should have significantly higher TDEE and daily calorie target compared to the sedentary user. | P |  |
|  | Edit Profile: Verify profile information updates and goal changes. | 1. Log in as existing user and navigate to Edit Profile. 2. Test different update scenarios:   Change height to 180cm and activity level to 'light'  Change goal from "Maintain Weight" to "Weight Loss" with new target weight   1. Save changes for each scenario. | 1. User document in Firestore is updated with new information. 2. TDEE and Target Calories are recalculated when relevant fields change. 3. Goal changes update TargetType and adjust TargetCalories appropriately. 4. User session reflects updated information. | 1. User document in Firestore is updated with new information. 2. TDEE and Target Calories are recalculated when relevant fields change. 3. Goal changes update TargetType and adjust TargetCalories appropriately. 4. User session reflects updated information. | P |  |
|  | Weight Record with Goal Check: Verify weight recording triggers goal achievement check. | 1. Log in as user with active weight loss goal. 2. Navigate to weight record page. 3. Enter new weight that meets the target. 4. Save the record. | 1. Weight is saved to WeightRecord collection. 2. Goal achievement check is automatically triggered. 3. If goal is met, achievement is recorded and user is notified. | 1. Weight is saved to WeightRecord collection. 2. Goal achievement check is automatically triggered. 3. If goal is met, achievement is recorded and user is notified. | P |  |
|  | View Profile Information: Verify display of current user profile data. | 1. Navigate to Profile page. 2. Check display of user information (username, height, weight, goal, etc.). | All user profile information is displayed correctly with current values from database. | All user profile information is displayed correctly with current values from database. | P |  |
|  | Profile Validation: Verify validation for profile field updates. | 1. Attempt to enter invalid data (negative height, unrealistic target weight). 2. Try to save changes. | Form validation prevents saving invalid data with appropriate error messages. | Form validation prevents saving invalid data with appropriate error messages. | P |  |
|  | Change Password: Verify password change functionality. | 1. Navigate to change password section. 2. Enter current password and new password. 3. Confirm new password and submit. | 1. Current password is verified. 2. New password meets strength requirements. 3. Password is updated successfully. | 1. Current password is verified. 2. New password meets strength requirements. 3. Password is updated successfully. | P |  |
|  | Auto-Adjustment Settings: Verify calorie auto-adjustment toggle functionality. | 1. Navigate to calorie adjustment settings. 2. Toggle auto-adjustment on/off. 3. Check impact on daily target calculations. | 1. Auto-adjustment setting is saved correctly. 2. When disabled, targets remain static. 3. When enabled, targets adjust based on progress. | 1. Auto-adjustment setting is saved correctly. 2. When disabled, targets remain static. 3. When enabled, targets adjust based on progress. | P |  |

### Security Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1002 |
| **Module:** | Security Module | | |
| **Actor(s):** | New User, Existing User | | |
| **Pre-requisites:** | * User has an account * Internet connection is available | | |
| **Dependencies:** | Firebase Authentication | | |

**Test Case 1002:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | Authenticate User: Verify successful login with valid credentials. | 1. Launch the app and navigate to the login page. 2. Enter a registered email and the correct password. 3. Tap the "LOGIN" button. | The user is successfully authenticated, user data is fetched from Firestore, the session is saved, and the user is redirected to the homepage. | The user is successfully authenticated, user data is fetched from Firestore, the session is saved, and the user is redirected to the homepage. | P |  |
|  | Authenticate User (Incorrect Password): Verify unsuccessful login with an incorrect password. | 1. Navigate to the login page. 2. Enter a registered email and an incorrect password. 3. Tap the "LOGIN" button. | An error message "Invalid email or password." is displayed. The user remains on the login page. | An error message "Invalid email or password." is displayed. The user remains on the login page. | P |  |
|  | Authenticate User (Account Lockout): Verify account lockout after multiple failed attempts. | On the login page, enter a registered email and an incorrect password 5 times consecutively. | After the 5th failed attempt, an error message "Account locked for 5 seconds..." is displayed. The "LOGIN" button becomes disabled or unresponsive for 5 seconds. | After the 5th failed attempt, an error message "Account locked for 5 seconds..." is displayed. The "LOGIN" button becomes disabled or unresponsive for 5 seconds. | P |  |
|  | Authorize User: Verify that a logged-in user can access protected pages. | 1. Log in successfully. 2. Attempt to navigate to the Profile Page or Progress Page. | The user can access these pages without being prompted to log in again. | The user can access these pages without being prompted to log in again. | P |  |
|  | Authorize User (Logged Out): Verify that a logged-out user cannot access protected pages. | 1. Ensure the user is logged out. 2. Attempt to directly access the Profile Page URL/route. | The user is redirected to the login page. | The user is redirected to the login page. | P |  |
|  | Recover Password: Verify the password recovery flow. | 1. On the login page, tap "Forgot Password?". 2. Enter a registered email address and submit. 3. Open the email and get the verification code/link. 4. Enter the code on the verification page. 5. Enter and confirm a new password on the reset page. | 1. A password reset email is sent successfully. 2. The verification code is accepted. 3. The password is changed in Firebase Authentication. 4. The user can log in with the new password. | 1. A password reset email is sent successfully. 2. The verification code is accepted. 3. The password is changed in Firebase Authentication. 4. The user can log in with the new password. | P |  |
|  | Recover Password (Invalid Code): Verify entering an invalid reset code. | 1. Initiate the password reset process and receive a code. 2. On the verification page, enter an incorrect or expired code. | An error message "Invalid verification code or link" is displayed. The user cannot proceed to the password reset page. | An error message "Invalid verification code or link" is displayed. The user cannot proceed to the password reset page. | P |  |
|  | Login with Unregistered Email: Verify login attempt with non-existent email. | 1. Navigate to login page. 2. Enter an email that has never been registered. 3. Enter any password and tap "LOGIN". | An error message "Invalid email or password" is displayed. Login fails. | An error message "Invalid email or password" is displayed. Login fails. | P |  |
|  | Login with Empty Fields: Verify validation for empty login fields. | 1. Navigate to login page. 2. Leave email field empty and enter a password, or vice versa. 3. Attempt to login. | Form validation prevents login attempt. Error messages indicate which fields are required. | Form validation prevents login attempt. Error messages indicate which fields are required. | P |  |
|  | Session Persistence: Verify user remains logged in after app restart. | 1. Login successfully and close the app. 2. Reopen the app after some time. | User remains logged in and is directed to homepage without needing to login again. Session data is preserved. | User remains logged in and is directed to homepage without needing to login again. Session data is preserved. | P |  |
|  | Password Reset (Unregistered Email): Verify password reset with non-existent email. | 1. Navigate to "Forgot Password" page. 2. Enter an email that is not registered. 3. Submit the request. | An error message indicates the email is not found, or a generic message is shown for security ("If email exists, reset link sent"). | An error message indicates the email is not found, or a generic message is shown for security ("If email exists, reset link sent"). | P |  |

### Smart Food Tracking Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1003 |
| **Module:** | Smart Food Tracking Module | | |
| **Actor(s):** | Existing User | | |
| **Pre-requisites:** | * User is logged in * Internet connection is available | | |
| **Dependencies:** | * Firebase Firestore * Gemini API for food recognition | | |

**Test Case 1003:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | Log Food: Verify that a user can search for and log a food item manually. | 1. On the homepage, select a meal type (e.g., "Breakfast"). 2. On the Log Food page, search for "Apple". 3. Select "Apple" from the results. 4. In the modal, enter a quantity of 150g. 5. Tap "Add to Meal". | 1. The food item is added to the 'LogMealList' collection. 2. The 'LogMeal' document for today's breakfast is created or updated with the correct total calories. 3. The user is navigated back, and the homepage UI reflects the updated calorie intake. | 1. The food item is added to the 'LogMealList' collection. 2. The 'LogMeal' document for today's breakfast is created or updated with the correct total calories. 3. The user is navigated back, and the homepage UI reflects the updated calorie intake. | P |  |
|  | Recognize Food: Verify food recognition using a clear image from the camera. | 1. From the Log Food page, tap the camera icon. 2. Take a clear picture of a banana. 3. The app analyzes the image. | 1. The system correctly identifies the food as "Lemon". 2. It matches it with an entry from the 'Food' database. 3. The nutritional information for the estimated portion size is displayed. | 1. The system correctly identifies the food as "Lemon". 2. It matches it with an entry from the 'Food' database. 3. The nutritional information for the estimated portion size is displayed. | P |  |
|  | Recognize Food (Multiple Items): Verify handling of an image with multiple food items. | 1. Take a picture containing the food. 2. The app analyzes the image. | The system identifies all three distinct food items and displays them in a list, each with its estimated nutritional information, allowing the user to select which ones to log. | The system identifies all three distinct food items and displays them in a list, each with its estimated nutritional information, allowing the user to select which ones to log. | P |  |
|  | Calculate Calorie: Verify calorie calculation based on user-entered quantity. | 1. Search for a food with known nutrition (e.g., "Chicken Breast", 182 kcal per 100g). 2. In the modal, enter a quantity of 200g. | The modal correctly calculates and displays the calories for the entered quantity (e.g., 364 kcal). | The modal correctly calculates and displays the calories for the entered quantity (e.g., 364 kcal). | P |  |
|  | AI Nutrition Fetching: Verify AI is used when a food is not found in the database. | 1. On the Log Food page, search for a rare or unique food item not in the database (e.g., "Durian Crepe"). 2. When no results are found, tap the "Get AI Nutrition" button. | 1. The Gemini API is called with the food name. 2. A confirmation dialog appears with the AI-generated nutritional info. 3. Upon confirmation, the new food is added to the 'Food' database and then logged. | 1. The Gemini API is called with the food name. 2. A confirmation dialog appears with the AI-generated nutritional info. 3. Upon confirmation, the new food is added to the 'Food' database and then logged. | P |  |
|  | Log Food (Zero Quantity): Verify validation for zero or negative quantity input. | 1. Search and select a food item. 2. In the quantity modal, enter 0 or negative value. 3. Attempt to add to meal. | Form validation prevents adding food with invalid quantity. | Form validation prevents adding food with invalid quantity. | P |  |
|  | Recognize Food (No Food in Image): Verify handling of images without food. | 1. Take a photo of non-food items (e.g., table, person, landscape). 2. Submit for recognition. | System detects no food items and prompts user to take a photo containing food or use manual search instead. | System detects no food items and prompts user to take a photo containing food or use manual search instead. | P |  |
|  | Edit Logged Food: Verify ability to modify previously logged food entries. | 1. Log a food item with specific quantity. 2. Navigate to meal details or daily log. 3. Edit the quantity or remove the food item. | 1. Food quantity can be updated and calories recalculate automatically. 2. Food items can be removed from meals. 3. Total daily calories update to reflect changes. | 1. Food quantity can be updated and calories recalculate automatically. 2. Food items can be removed from meals. 3. Total daily calories update to reflect changes. | P |  |

### Dynamic Target Adjustment Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1004 |
| **Module:** | Dynamic Target Adjustment Module | | |
| **Actor(s):** | Existing User | | |
| **Pre-requisites:** | * User is logged in * User has logged food intake for the day | | |
| **Dependencies:** | Firebase Firestore | | |

**Test Case 1004:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | Check Daily Intake: Verify the system correctly sums daily calorie intake. | Log multiple food items for a single day (e.g., Breakfast: 300 kcal, Lunch: 500 kcal, Dinner: 700 kcal). | The homepage UI correctly displays the total consumed calories for the day as 1500 kcal. | The homepage UI correctly displays the total consumed calories for the day as 1500 kcal. | P |  |
|  | Compare Target Intake: Verify the comparison between consumed and target calories. | 1. Assume the user's daily target is 2000 kcal. 2. The user has consumed 1500 kcal. | The homepage UI correctly shows that the user has 500 kcal remaining for the day. | The homepage UI correctly shows that the user has 500 kcal remaining for the day. | P |  |
|  | Auto Adjust Target: Verify target adjustment when intake is higher than a "Weight Loss" target. | 1. User's goal is "Weight Loss" with a base target of 1800 kcal. 2. User logs 2200 kcal for the previous day. 3. The app is launched the next day, triggering the auto-adjustment service. | 1. A new 'CalorieAdjustment' document is created for the current day. 2. The new `AdjustTargetCalories` is calculated to be lower than the base target (e.g., 1800 - (2200-1800) = 1400 kcal). 3. The homepage displays the newly adjusted target. | 1. A new 'CalorieAdjustment' document is created for the current day. 2. The new `AdjustTargetCalories` is calculated to be lower than the base target (e.g., 1800 - (2200-1800) = 1400 kcal). 3. The homepage displays the newly adjusted target. | P |  |
|  | Auto Adjust Target (No Adjustment): Verify no adjustment when intake is lower than a "Weight Loss" target. | 1. User's goal is "Weight Loss" with a base target of 1800 kcal. 2. User logs 1750 kcal for the previous day. 3. The app is launched the next day. | No adjustment is performed because the user's intake was below their target, which aligns with the weight loss goal. The target on the homepage remains at 1800 kcal. | No adjustment is performed because the user's intake was below their target, which aligns with the weight loss goal. The target on the homepage remains at 1800 kcal. | P |  |
|  | Auto Adjust Target (Safety Boundaries): Verify safety boundaries prevent dangerously low targets. | 1. User's goal is "Weight Loss" with a base target of 1800 kcal. 2. User logs an extremely high intake of 4000 kcal. 3. The app is launched the next day. | The new target is clamped by the safety boundary (e.g., BMR or a minimum of 1200/1500 kcal) and does not drop to a dangerously low value (e.g., -400 kcal). | The new target is clamped by the safety boundary (e.g., BMR or a minimum of 1200/1500 kcal) and does not drop to a dangerously low value (e.g., -400 kcal). | P |  |
|  | Missed Adjustment Check: Verify that a missed adjustment is performed on startup. | 1. Do not open the app for a full day where an adjustment should have occurred. 2. Launch the app the following day. | The `\_checkMissedAdjustmentOnStartup` function is triggered, and the missed daily adjustment is calculated and applied for the previous day. | The `\_checkMissedAdjustmentOnStartup` function is triggered, and the missed daily adjustment is calculated and applied for the previous day. | P |  |
|  | Auto Adjust Target (Maintain Weight): Verify adjustment for maintain weight goal based on intake. | 1. User's goal is "Maintain Weight" with target 2000 kcal. 2. User logs 2500 kcal (over target) or 1500 kcal (under target). 3. App is launched the next day. | 1. Calorie adjustment performed for maintain weight goal. 2. If user consumed 2500 kcal (over target), new target should be adjusted downward 3. If user consumed 1500 kcal (under target), new target should be adjusted upward 4. Target is adjusted to help user maintain their weight more effectively | 1. Calorie adjustment performed for maintain weight goal. 2. If user consumed 2500 kcal (over target), new target should be adjusted downward 3. If user consumed 1500 kcal (under target), new target should be adjusted upward 4. Target is adjusted to help user maintain their weight more effectively | P |  |
|  | Adjustment Calculation Accuracy: Verify mathematical accuracy of adjustment formulas. | 1. Set up specific test case: Base target 2000 kcal, actual intake 2300 kcal, weight loss goal. 2. Trigger adjustment calculation. | New target is calculated correctly: 2000 - (2300 - 2000) = 1700 kcal, respecting any safety boundaries implemented. | New target is calculated correctly: 2000 - (2300 - 2000) = 1700 kcal, respecting any safety boundaries implemented. | P |  |
|  | Goal Achievement Display: Verify celebration page appears when goal is achieved. | 1. User achieves weight goal (reaches target weight). 2. Record the achieving weight. | 1. Goal Achievement page is displayed with celebration animation. 2. Achieved weight and new TDEE are shown. 3. Congratulatory message is displayed. | 1. Goal Achievement page is displayed with celebration animation. 2. Achieved weight and new TDEE are shown. 3. Congratulatory message is displayed. | P |  |
|  | Achievement Animation: Verify celebration animations and visual effects. | 1. Trigger goal achievement. 2. Observe page animations and effects. | 1. Lottie celebration animation plays smoothly. 2. Fade and scale animations work correctly. 3. Visual effects enhance user experience. | 1. Lottie celebration animation plays smoothly. 2. Fade and scale animations work correctly. 3. Visual effects enhance user experience. | P |  |
|  | TDEE Recalculation: Verify new TDEE calculation for achieved weight. | 1. Achieve weight goal. 2. Check new TDEE calculation on achievement page. | 1. New TDEE is calculated based on achieved weight. 2. TDEE value is accurate according to user's profile. 3. New maintenance calories are displayed. | 1. New TDEE is calculated based on achieved weight. 2. TDEE value is accurate according to user's profile. 3. New maintenance calories are displayed. | P |  |

### Motivation Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1005 |
| **Module:** | Motivation Module | | |
| **Actor(s):** | Existing User | | |
| **Pre-requisites:** | User is logged in | | |
| **Dependencies:** | * Firebase Firestore * Firebase Cloud Messaging (FCM) | | |

**Test Case 1005:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | Track Streak: Increment streak on first log of the day. | 1. Ensure the user has not logged any food today. 2. Log a food item via Log Food page (any meal). | 1. A 'StreakRecord' exists for the user and CurrentStreakDays increments by 1 (or set to 1 if first ever). 2. LastLoggedDate is set to today (YYYY-MM-DD). 3. App navigates to the Streak Page showing a celebration animation and the updated streak. | 1. A 'StreakRecord' exists for the user and CurrentStreakDays increments by 1 (or set to 1 if first ever). 2. LastLoggedDate is set to today (YYYY-MM-DD). 3. App navigates to the Streak Page showing a celebration animation and the updated streak. | P |  |
|  | Track Streak (Consecutive Days): Maintain streak on consecutive days. | 1. Day 1: Log any food item (streak becomes N). 2. Day 2: Log any food item again. | CurrentStreakDays becomes N+1. The streak calendar shows two consecutive-colored days. | CurrentStreakDays becomes N+1. The streak calendar shows two consecutive-colored days. | P |  |
|  | Track Streak (Reset): Reset after missed day. | 1. Day 1: Log any food item (streak becomes N). 2. Day 2: Do not log any item. 3. Day 3: Log any food item. | CurrentStreakDays resets to 1 on Day 3. LastLoggedDate updates to Day 3. | CurrentStreakDays resets to 1 on Day 3. LastLoggedDate updates to Day 3. | P |  |
|  | Track Streak (Update): Verify that the streak is updated on the first log of a new day. | 1. Ensure no food has been logged for the current day. 2. Log any food item successfully. | 1. The user's streak is incremented. 2. The user is navigated to the Streak Page, which shows a celebration animation and the updated streak count. | 1. The user's streak is incremented. 2. The user is navigated to the Streak Page, which shows a celebration animation and the updated streak count. | P |  |
|  | Track Streak (Maintenance): Verify that the streak is not updated again on subsequent logs the same day. | 1. After the first log of the day, log a second food item. 2. Observe the navigation after logging. | The food is logged successfully, but the user is navigated back to the homepage directly, without showing the Streak Page again. | The food is logged successfully, but the user is navigated back to the homepage directly, without showing the Streak Page again. | P |  |
|  | Invite Friend: Send supervision invitation. | 1. Navigate to Invite Supervisor page. 2. Search by username or email of another registered user. 3. Tap "Invite". | 1. A 'Supervision' document with Status='pending' is created and a unique SupervisionID is assigned. 2. Two 'SupervisionList' records are created (for inviter and invitee). 3. An RTDB notification is sent to the invitee; the invitee sees an invitation in the app. | 1. A 'Supervision' document with Status='pending' is created and a unique SupervisionID is assigned. 2. Two 'SupervisionList' records are created (for inviter and invitee). 3. An RTDB notification is sent to the invitee; the invitee sees an invitation in the app. | P |  |
|  | Invite Friend (Blacklist): Respect blacklist and existing relationships. | 1. From search results, tap block on a user and confirm. 2. Try searching/inviting that blocked user again. 3. If a supervision already exists (accepted/pending), try to invite again. | 1. A 'Blacklist' record is created; blocked user is excluded from search results and cannot be invited. 2. If an accepted/pending supervision exists, inviting the same user is prevented with an appropriate message. | A 'Blacklist' record is created; blocked user is excluded from search results and cannot be invited. | P |  |
|  | Alert Friend: Supervisor views user's streak after acceptance. | 1. Accept the pending supervision invitation (Status changes to 'accepted'). 2. As the supervisor, open the supervised streak page. | Supervisor can view the user's current streak count and streak calendar. | Supervisor can view the user's current streak count and streak calendar. | P |  |
|  | Supervisor Streak: Update supervisor streak when both log today. | 1. With an 'accepted' supervision, both users log food on the same day. 2. Trigger streak update (log action).P | 'Supervision' record CurrentStreakDays increments by 1 and LastLoggedDate is today. If only one logs, supervisor streak does not increment. | 'Supervision' record CurrentStreakDays increments by 1 and LastLoggedDate is today. If only one logs, supervisor streak does not increment. | P |  |
|  | Invite Friend (Self Invitation): Verify prevention of self-supervision. | 1. Navigate to Invite Supervisor page. 2. Search for and attempt to invite your own username/email. | System prevents self-invitation by excluding current user from search results (current user does not appear in search results). | System prevents self-invitation by excluding current user from search results (current user does not appear in search results). | P |  |
|  | Invite Friend (Duplicate Invitation): Verify handling of duplicate invitations. | 1. Search for a user who already has a pending or accepted supervision relationship with you. 2. Attempt to send an invitation to that user. | System prevents duplicate invitations by excluding users with existing supervision relationships (accepted & pending) from search results. | System prevents duplicate invitations by excluding users with existing supervision relationships (accepted & pending) from search results. | P |  |
|  | Supervision Acceptance/Rejection: Verify invitation response handling. | 1. Receive a supervision invitation. 2. Accept or reject the invitation. | 1. Accepting changes status to 'accepted', initializes CurrentStreakDays to 0 and LastLoggedDate to null, enables supervision features. 2. Rejecting changes status to 'rejected' (records remain in database for potential reinvitation). 3. Database status is updated accordingly for both acceptance and rejection. | 1. Accepting changes status to 'accepted', initializes CurrentStreakDays to 0 and LastLoggedDate to null, enables supervision features. 2. Rejecting changes status to 'rejected' (records remain in database for potential reinvitation). 3. Database status is updated accordingly for both acceptance and rejection. | P |  |
|  | Notification Delivery: Verify FCM notifications are sent and received. | 1. Perform actions that trigger notifications (invitations, etc.). 2. Check notification delivery on recipient device. | 1. Notifications are sent via FCM successfully. 2. Recipients receive notifications in real-time. 3. Notification content is accurate and actionable. | 1. Notifications are sent via FCM successfully. 2. Recipients receive notifications in real-time. 3. Notification content is accurate and actionable. | P |  |
|  | Calendar Display: Verify calendar shows logged days and current streak. | 1. Navigate to Streak Calendar page. 2. View calendar with logged meal days. | 1. Calendar displays with logged days highlighted. 2. Current streak count is shown prominently. 3. Today's status (logged/not logged) is clearly indicated. | 1. Calendar displays with logged days highlighted. 2. Current streak count is shown prominently. 3. Today's status (logged/not logged) is clearly indicated. | P |  |
|  | Calendar Navigation: Verify ability to navigate between months and select dates. | 1. Navigate to previous/next months. 2. Select different dates on calendar. 3. Check date selection and display. | 1. Month navigation works smoothly. 2. Date selection updates focused day. 3. Calendar maintains state during navigation. | 1. Month navigation works smoothly. 2. Date selection updates focused day. 3. Calendar maintains state during navigation. | P |  |
|  | Reminder Notifications: Verify reminder system for supervised users. | 1. Supervised user misses logging for a day. 2. Check if reminder notifications are sent. | 1. Reminder notifications are sent to supervised user. 2. Supervisor may receive notification about missed logging. 3. Notifications are timely and actionable. | 1. Reminder notifications are sent to supervised user. 2. Supervisor may receive notification about missed logging. 3. Notifications are timely and actionable. | P |  |

### Report Module

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Details** | | | |
| **Student Name** | Wang Zi Zhen | **Programme:** | RSD3S1 |
| **Project Title:** | CalorieCare: Diet and Nutrition Management System | **Test Case No:** | 1006 |
| **Module:** | Report Module | | |
| **Actor(s):** | Existing User | | |
| **Pre-requisites:** | * User is logged in * User has logged calorie intake and weight data over several days | | |
| **Dependencies:** | Firebase Firestore | | |

**Test Case 1006:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Description** | **Test actions/ inputs** | **Expected** | **Actual Results** | **Pass(P)/ Fail(F)** | **Remarks** |
|  | View Calorie Intake: Display daily calorie intake chart. | 1. Ensure multiple days have logged meals (different totals). 2. Navigate to Progress/Report page and select Calorie Intake view. | A chart (e.g., bar chart) shows daily calorie intake for the selected period (e.g., past week). The values match sums in 'LogMeal' per day. | A chart (e.g., bar chart) shows daily calorie intake for the selected period (e.g., past week). The values match sums in 'LogMeal' per day. | P |  |
|  | View Weight Report: Display weight progress chart. | 1. Ensure the user has multiple weight entries (from registration and subsequent logs). 2. Navigate to the Weight Report view. | A line chart displays weight over time using saved weight entries; the trend matches stored data. | A line chart displays weight over time using saved weight entries; the trend matches stored data. | P |  |
|  | Log New Weight: Add current weight and update chart. | 1. In the weight report page, tap to add a new weight entry. 2. Enter valid weight (e.g., 68.5 kg) and save. | 1. A new weight entry is saved with today's date. 2. The weight chart updates immediately to include the new data point. | 1. A new weight entry is saved with today's date.   The weight chart updates immediately to include the new data point. | P |  |
|  | Weight Validation: Verify validation for unrealistic weight entries. | 1. Attempt to enter extremely low weight (e.g., 10 kg) or high weight (e.g., 500 kg). 2. Try to save the entry. | Form validation prevents saving unrealistic weights with appropriate error messages indicating acceptable weight ranges. | Form validation prevents saving unrealistic weights with appropriate error messages indicating acceptable weight ranges. | P |  |
|  | Report Time Range Selection: Verify different time period views for reports. | 1. Navigate to calorie intake report. 2. Select different time ranges (week, month, 3 months, year). 3. Check data display for each range. | Charts update to show data for selected time period with appropriate scaling and data points. Empty periods show zero or no data indicators. | Charts update to show data for selected time period with appropriate scaling and data points. Empty periods show zero or no data indicators. | P |  |

## Chapter Summary and Evaluation

This chapter presented the final development of the StudyBuddy system, including full implementation and a complete testing process to confirm system reliability. Section 5.1 described the coding of the main modules using the Django framework and the connection to the SQLite database. The Gamification module was implemented to adjust the virtual pet’s mood and award Capy Coins based on student activity, the Module Practice module was developed to handle quiz creation and grading, and the Pomodoro Timer module was integrated with front-end JavaScript and back-end Django logic to manage study sessions. Section 5.2 explained the testing strategy, which included unit testing of individual functions, integration testing to check module interaction, and acceptance testing that confirmed the StudyBuddy system met project requirements during a demonstration. Section 5.3 provided proof of correctness through a detailed test plan and test cases, showing successful outcomes across all modules from the Gamification Module to the Study Report Module. The results confirmed that the StudyBuddy system is fully functional and working properly.

Chapter 6

**Discussions and Conclusion**

1. **Discussions and Conclusion**

This section presents the evaluation of the CalorieCare project. The discussion highlights the technical problems encountered during development, the solutions applied, and the issues that remain for future enhancement. Achieved objectives and completed modules are summarized, while incomplete parts are identified with suggestions for improvement. The conclusion provides an overall reflection on the project outcomes and contributions.

## Summary

The CalorieCare system was developed to address the growing need for effective diet and nutrition management tools in today's health-conscious society. The project tackles three major challenges: difficulty in accurately tracking daily calorie intake, lack of motivation to maintain consistent dietary habits, and the absence of personalized, adaptive nutrition guidance. To overcome these issues, the system integrates six core modules: User Management for comprehensive profile and goal setting, Security for robust authentication and data protection, Smart Food Tracking with AI-powered food recognition, Dynamic Target Adjustment for personalized calorie management, Motivation through streak tracking and social supervision, and comprehensive Reporting for progress visualization.

The development process leveraged modern mobile technologies to create a cross-platform solution that operates seamlessly across different devices. Flutter was selected as the primary framework, offering excellent performance and native-like user experience across iOS and Android platforms. Firebase served as the backend infrastructure, providing reliable authentication, real-time database capabilities, and cloud storage for user data. The integration of Google's Gemini AI API enables advanced food recognition from images, while Firebase Cloud Messaging (FCM) ensures real-time notifications and social features. The system architecture follows a modular design pattern, allowing for easy maintenance and future expansion.

The project demonstrates how artificial intelligence, real-time data processing, and social features can be combined to create an engaging and effective nutrition management platform. CalorieCare shows the potential to transform traditional calorie counting into an intelligent, adaptive, and socially-supported health management experience that encourages consistent healthy eating habits through gamification and personalized guidance.

## Achievements

Overall, the CalorieCare system successfully met the project objectives outlined at the beginning of development. The following highlights how each goal was accomplished through the system's core modules:

### Develop a comprehensive User Management System with personalized goal setting and profile management

The user management system features a multi-step registration process that captures detailed user information including physical characteristics, health goals, and activity levels. The system automatically calculates BMI, BMR, TDEE, and personalized daily calorie targets based on user goals (weight loss, weight gain, or maintenance). Advanced validation ensures appropriate goal selection based on BMI categories, preventing unhealthy target setting. The profile management system allows users to update their information and recalculate targets dynamically. This module was fully developed and provides a solid foundation for personalized nutrition guidance.

### Implement a Smart Food Tracking System with AI-powered food recognition and comprehensive nutrition database

The system integrates Google's Gemini AI API to analyze food images and identify nutritional content automatically. Users can either search from an extensive food database or capture photos for AI analysis. The system performs intelligent matching between AI-identified foods and the existing database, ensuring accurate nutritional information. When foods are not found in the database, the AI-generated nutrition data is automatically added, expanding the system's knowledge base. The food logging interface supports multiple meal types and provides real-time calorie calculations based on portion sizes. This module significantly improves the user experience by reducing manual input requirements and increasing accuracy.

### Create a Dynamic Target Adjustment System that adapts daily calorie goals based on user behavior and progress

The auto-adjustment service monitors daily calorie intake and automatically adjusts targets for the following day based on goal adherence. For weight loss goals, exceeding targets results in reduced targets for compensation, while for weight gain goals, under-consumption triggers increased targets. The system includes safety boundaries to prevent dangerously low or high targets, ensuring user health and safety. The adjustment logic considers the user's base metabolic rate and maintains reasonable limits. This adaptive approach helps users stay on track with their goals while providing flexibility for real-world eating patterns.

### Build a comprehensive Motivation System with streak tracking and social supervision features

The motivation module implements a streak tracking system that rewards consistent daily food logging with visual celebrations and progress indicators. Users can invite friends or family members as supervisors to monitor their progress, creating accountability and social support. The system includes a streak calendar that visualizes logged days and current streaks. FCM notifications keep users engaged with reminders and achievement celebrations. The social features enable users to share their progress and receive encouragement from their support network, significantly enhancing motivation and adherence to healthy eating habits.

### Develop a robust Security System with authentication, session management, and data protection

The security module implements Firebase Authentication with email/password login, password recovery, and session persistence. Account lockout mechanisms prevent brute force attacks, while secure data transmission ensures user privacy. The system includes comprehensive input validation and error handling to prevent security vulnerabilities. User data is encrypted and stored securely in Firebase, with proper access controls and authentication requirements for all sensitive operations.

### Create a comprehensive Reporting System with progress visualization and data analytics

The reporting module provides detailed charts and graphs showing calorie intake trends, weight progress, and goal achievement over time. Users can view their progress across different time periods and track their journey toward health goals. The system includes weight recording functionality with validation and historical tracking. Goal achievement detection automatically recognizes when users reach their targets and provides celebration animations and new maintenance recommendations.

The CalorieCare achieved all key objectives by combining artificial intelligence, real-time data processing, social features, and adaptive algorithms into a comprehensive mobile health platform. These modules were successfully implemented and tested to ensure functionality, reliability, and user satisfaction.

## Contributions

The CalorieCare system offers an innovative approach to nutrition management by combining artificial intelligence, adaptive algorithms, and social features to create a comprehensive health management platform. By integrating AI-powered food recognition with personalized goal setting and social supervision, the platform transforms traditional calorie counting into an intelligent, engaging, and supportive experience.

One of the key innovations is the AI-powered food recognition system that uses Google's Gemini API to analyze food images and provide accurate nutritional information. This feature significantly reduces the manual effort required for food logging while improving accuracy through intelligent database matching. The system's ability to automatically expand its food database with AI-generated nutrition data ensures comprehensive coverage of diverse food items, making it accessible to users with varied dietary preferences.

The dynamic target adjustment system represents another significant contribution, providing personalized calorie management that adapts to user behavior and progress. Unlike static calorie targets found in traditional apps, CalorieCare's adaptive system considers daily intake patterns and automatically adjusts goals to help users stay on track with their health objectives. The inclusion of safety boundaries ensures that adjustments remain within healthy ranges, preventing potentially harmful target modifications.

The social supervision feature introduces a novel approach to health motivation by enabling users to invite friends or family members to monitor their progress. This accountability system leverages social support to enhance motivation and adherence to healthy eating habits. The streak tracking system with visual celebrations and progress indicators gamifies the nutrition tracking experience, making healthy eating more engaging and rewarding.

From a technical perspective, CalorieCare demonstrates effective integration of multiple complex systems including Firebase services, AI APIs, real-time notifications, and cross-platform mobile development. The modular architecture ensures maintainability and scalability, while the comprehensive error handling and validation systems provide robust user experience.

The system contributes to public health by making nutrition tracking more accessible, accurate, and engaging. CalorieCare supports both extrinsic motivation through social features and streak rewards, as well as intrinsic motivation by providing clear progress visualization and goal achievement recognition. The adaptive nature of the system helps users develop sustainable healthy eating habits rather than relying on rigid, one-size-fits-all approaches.

From a market perspective, CalorieCare addresses the growing demand for personalized health management tools in an increasingly health-conscious society. The system can be positioned as a comprehensive nutrition management solution for individuals seeking to improve their dietary habits, healthcare providers looking for patient monitoring tools, or wellness programs requiring engagement and tracking capabilities.

Overall, CalorieCare addresses common challenges in nutrition management such as tedious food logging, lack of motivation, and generic guidance while offering an intelligent, adaptive, and socially-supported approach to healthy eating.

## Limitations and Future Improvements

While the CalorieCare system successfully delivers a comprehensive and functional nutrition management platform, several limitations were encountered during development that offer opportunities for future improvement.

One major limitation is the scope of food recognition accuracy. While the AI-powered system performs well with common foods, it may struggle with complex dishes, regional cuisines, or foods with unusual presentations. The current system relies on a single AI model (Gemini) and could benefit from ensemble approaches or specialized nutrition-focused AI models. Future development could integrate multiple AI services for improved accuracy and implement user feedback mechanisms to continuously improve recognition quality.

Additionally, the social supervision features are currently limited to basic streak monitoring and simple notifications. More advanced social features such as group challenges, recipe sharing, meal planning collaboration, or nutritionist consultation integration could significantly enhance user engagement and provide more comprehensive support for healthy eating habits.

The reporting and analytics capabilities, while functional, could be enhanced with more sophisticated data analysis. Future versions could include predictive analytics for weight trends, personalized meal recommendations based on eating patterns, integration with fitness trackers for comprehensive health monitoring, or advanced nutrition analysis including micronutrient tracking and dietary deficiency identification.

From a technical perspective, certain limitations remain in the current CalorieCare system. The offline functionality is limited, preventing users from logging food when internet connectivity is unavailable. The system currently uses Firebase as the primary backend, which may have scalability limitations for very large user bases. Security features could be strengthened with advanced authentication methods such as biometric login, two-factor authentication, or integration with health data standards for medical-grade security.

The user interface, while functional, could benefit from more advanced personalization options, accessibility features for users with disabilities, or integration with smart home devices for automated meal logging. The current system focuses primarily on calorie tracking but could expand to include comprehensive nutrition education, meal planning tools, or integration with grocery shopping and meal delivery services.

All in all, while the current version meets the core objectives effectively, future improvements can expand functionality, increase accuracy, enhance user engagement, and improve the system's value in real-world health management scenarios.

## Issues and Solutions

Throughout the development of the CalorieCare system, several technical issues were encountered that provided valuable learning experiences and contributed to significant growth in mobile app development and AI integration skills.

The Food Recognition module posed significant difficulties in managing API rate limits from Google's Gemini API, handling image processing performance, and ensuring accurate nutritional data. To solve this, comprehensive error handling, image compression algorithms, and intelligent caching mechanisms were implemented to optimize and reduce API calls. The system now includes fallback mechanisms for API failures and a user feedback loop to improve recognition accuracy over time.

The Real-time Data Synchronization module involved complex state management to keep dynamic targets and streaks updated across multiple data sources. Integrating Firebase and local app state created challenges in maintaining data consistency, especially in offline scenarios. A centralized state management system using Stream controllers was developed, which included conflict resolution mechanisms and automatic synchronization once a connection is restored.

The Notification System presented challenges in managing multiple notification types and ensuring reliable delivery across different devices, particularly when handling background messages. This was addressed by creating a unified notification manager to handle all alerts through a single service, which prevents duplicate notifications and manages user preferences.

Developing with Flutter for Cross-platform Compatibility introduced challenges in ensuring consistent performance on both iOS and Android, especially with camera integration and native notifications. These issues were resolved by implementing platform-specific code where necessary, optimizing image processing algorithms, and conducting comprehensive testing across various devices and operating systems.

The Database Design required careful planning to support complex relationships between users, meals, and supervision records while maintaining real-time performance. The database schema was redesigned with proper indexing and efficient data access patterns to ensure the system could handle large datasets efficiently and remain responsive.

All of these challenges were addressed through systematic problem-solving, extensive testing, and iterative development. This approach not only resolved immediate issues but also improved the overall system architecture, making it more maintainable and scalable for the future.

## Conclusion

The CalorieCare was successfully completed within the planned timeframe, achieving all primary objectives and delivering a comprehensive, mobile-based nutrition management platform that integrates artificial intelligence, adaptive algorithms, and social features to address common challenges in diet and nutrition tracking. Users can log food through AI-powered image recognition, receive personalized calorie targets that adapt to their behavior, track their progress through gamified streak systems, and receive social support through supervision features, all of which encourage consistent and effective healthy eating habits.

All core modules were fully implemented and tested to ensure functionality, reliability, and a smooth user experience. The user management module provides comprehensive profile and goal setting capabilities, the smart food tracking system offers accurate nutrition logging through AI integration, the dynamic target adjustment system adapts to user behavior for personalized guidance, the motivation module gamifies healthy eating through streaks and social features, the security system ensures robust data protection and authentication, and the reporting module provides detailed progress visualization and analytics.

Challenges encountered during development, including AI integration complexity, real-time data synchronization, cross-platform compatibility, and notification management, were successfully addressed through systematic problem-solving, comprehensive testing, and iterative development approaches. The solutions implemented not only resolved immediate technical issues but also enhanced the overall system architecture and user experience.

While the current version effectively meets the intended goals, future enhancements could expand AI recognition accuracy, implement advanced social features, add comprehensive offline functionality, integrate with fitness trackers and health devices, enhance security with biometric authentication, and develop advanced analytics for predictive health insights. The modular architecture and comprehensive foundation established in this project provide an excellent base for these future developments.

Overall, CalorieCare demonstrates a practical and innovative approach to nutrition management, offering an intelligent, adaptive, and socially-supported environment for effective healthy eating habit formation. The project successfully combines cutting-edge technologies with user-centered design to create a valuable tool for individuals seeking to improve their dietary habits and overall health.

# 

# References

Abul Hasnat, Nikhil Mahato, & Halder, S. (2023). *Basics of Android Application Development*. *IV*(1), 35–53. https://www.researchgate.net/publication/385358171\_Basics\_of\_Android\_Application\_Development

Alias Sri Swathiga, U., Vinodhini, & Sasikala (Eds.). (2021, October 3). *AN INTERPRETATION OF DART PROGRAMMING LANGUAGE*. https://www.researchgate.net/publication/358661479\_AN\_INTERPRETATION\_OF\_DART\_PROGRAMMING\_LANGUAGE

Alshamrani , A., & Bahattab, A. (Eds.). (2015). *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model* . https://www.academia.edu/download/36637147/SDLC.pdf

Android Studio. (2019a). *Android Studio and SDK tools*. Android Developers. https://developer.android.com/studio

Android Studio. (2019b). *Install Android Studio  |  Android Developers*. Android Developers. https://developer.android.com/studio/install

Bastiaan, R. G., Lawren Julio Rumokoy, & Maria. (2024). WHAT LIES BEYOND STREAKS AND BADGES? DUOLINGO’S GAMIFICATION STRATEGY AND ITS LIMITATIONS. *Jurnal EMBA : Jurnal Riset Ekonomi, Manajemen, Bisnis Dan Akuntansi*, *12*(03), 1275–1284. https://doi.org/10.35794/emba.v12i03.58183

Bruno, V., Resende, S., & Juan, C. (2017). A Survey on Automated Food Monitoring and Dietary Management Systems. *Journal of Health & Medical Informatics*, *8*(3), 272. https://doi.org/10.4172/2157-7420.1000272

Calorie Mama. (2017). *Calorie Mama Food AI - Food Image Recognition and Calorie Counter using Deep Learning*. Caloriemama.ai. https://www.caloriemama.ai/#Database

Calorie Mama. (2021). *Calorie Mama AI: Meal Planner*. Google.com. https://play.google.com/store/apps/details?id=com.azumio.android.caloriesbuddy&hl=en

Chen, J., Berkman, W., Bardouh, M., Ng, C. Y. K., & Allman-Farinelli, M. (2019). The use of a food logging app in the naturalistic setting fails to provide accurate measurements of nutrients and poses usability challenges. *Nutrition*, *57*, 208–216. https://doi.org/10.1016/j.nut.2018.05.003

Dart. (2024). *Dart overview*. Dart.dev. https://dart.dev/overview

Deniz-Garcia, A., Fabelo, H., Rodriguez-Almeida, A., Zamora-Zamorano, G., Castro-Fernandez, M., Alberiche-Ruano, M. del P., Solvoll, T., Bartnæs, C., Schopf, T. R., Callico, G. M., Soguero-Ruiz, C., & Wagner, A. M. (2022). Quality, Usability and Effectiveness of mHealth Applications and the Role of Artificial Intelligence: Current Scenario and Challenges (Preprint). *Journal of Medical Internet Research*, *25*. https://doi.org/10.2196/44030

Fat Secret. (2019). *FatSecret - Calorie Counter and Diet Tracker for Weight Loss*. Fatsecret.com. https://www.fatsecret.com/

fatsecret. (2021). *Calorie Counter by fatsecret*. Google.com. https://play.google.com/store/apps/details?id=com.fatsecret.android&hl=en

Flutter. (2025a). *Start building Flutter Android apps on Windows*. Docs.flutter.dev. https://docs.flutter.dev/get-started/install/windows/mobile

Flutter. (2025b, January 31). *Flutter documentation*. Docs.flutter.dev. https://docs.flutter.dev/

Freeman, C., Kittredge, A., Wilson, H., & Pajak, B. (2023). *The Duolingo Method for App-based Teaching and Learning*. http://duolingo-papers.s3.amazonaws.com/reports/duolingo-method-whitepaper.pdf

GeeksforGeeks. (2018, May 28). *Incremental Process Model - Software Engineering*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-incremental-process-model/#advantages-of-incremental-process-model

HUYNH , D., & IIDA, H. (2017). *JAIST Repository: An Analysis of Winning Streak’s Effects in Language Course of “Duolingo.”* Jaist.ac.jp. https://dspace.jaist.ac.jp/dspace/handle/10119/15292

Huynh, D., Zuo, L., & Iida, H. (2018). An Assessment of Game Elements in Language-Learning Platform Duolingo. *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*. https://doi.org/10.1109/iccoins.2018.8510568

JAVA. (2023). *What is Java technology and why do I need it?* Java.com. https://www.java.com/en/download/help/whatis\_java.html

König, L. M., Attig, C., Franke, T., & Renner, B. (2021). Barriers to and Facilitators for Using Nutrition Apps: Systematic Review and Conceptual Framework. *JMIR MHealth and UHealth*, *9*(6), e20037. https://doi.org/10.2196/20037

Larson-Meyer, D. E., Krason, R. K., & Meyer, L. M. (2022). Weight Gain Recommendations for Athletes and Military Personnel: a Critical Review of the Evidence. *Current Nutrition Reports*, *11*(2), 225–239. https://doi.org/10.1007/s13668-022-00395-3

LogMeal. (2025). *LogMeal Food AI - Image API for Food Detection - Food API: recognition and nutritional tracking*. Logmeal.com. https://logmeal.com/api/

Mohamed, E., Nashed, E., Roshdy, M., Abdelsalam, R., Sayed, S., Elsayed, S., & Hassan, M. (2024). New Technique of Mathematics Exams Based on Java and Dart. *البحوث التطبيقية في العلوم والانسانيات*, *1*(1), 49–76. https://doi.org/10.21608/aash.2024.368290

Muhammad Shoaib Farooq, Riaz, S., Alvi, A., Ali, A., & Rehman, I. U. (2022). Cross-Platform Mobile Development Approaches and Frameworks. *VFAST Transactions on Software Engineering*, *10*(2), 79–93. https://doi.org/10.21015/vtse.v10i2.978

Nadeem, M., Shen, H., Choy, L., & Barakat, J. M. H. (2023). Smart Diet Diary: Real-Time Mobile Application for Food Recognition. *Applied System Innovation*, *6*(2), 53. https://doi.org/10.3390/asi6020053

NHS. (2021, November 4). *Healthy ways to gain weight*. Nhs.uk. https://www.nhs.uk/live-well/healthy-weight/managing-your-weight/healthy-ways-to-gain-weight/

Nutritionix. (2021). *Track - Calorie Counter*. Google.com. https://play.google.com/store/apps/details?id=com.nutritionix.nixtrack&hl=en

Oranga, J., & Matere, A. (2023). Qualitative Research: Essence, Types and Advantages. *Open Access Library Journal*, *10*(12), 1–9. https://doi.org/10.4236/oalib.1111001

Publishing, H. H. (2020, July 11). *Calorie counting made easy*. Harvard Health. https://www.health.harvard.edu/staying-healthy/calorie-counting-made-easy

Rusin, M., Årsand, E., & Hartvigsen, G. (2013). Functionalities and input methods for recording food intake: A systematic review. *International Journal of Medical Informatics*, *82*(8), 653–664. https://doi.org/10.1016/j.ijmedinf.2013.01.007

Sachan, D. (2023, September 21). *Incremental Model in Software Engineering*. Scaler Topics. https://www.scaler.com/topics/incremental-model-in-software-engineering/

Sardorbek Rustam Ugli, I., & Woo, G. (2024). *Comparative Analysis of Cross-Platform and Native Mobile App Development Approaches* . https://koreascience.or.kr/article/CFKO202422572532969.pdf

TpointTech. (2023). *Incremental Model (Software Engineering) - javatpoint*. Www.javatpoint.com. https://www.tpointtech.com/software-engineering-incremental-model

Woo;, U. (2024). Comparative Analysis of Cross-Platform and Native Mobile App Development Approaches. *Annual Conference of KIPS*, 53–56. https://doi.org/10.3745/PKIPS.y2024m05a.53

Zazz. (2022, November 8). *Explore the Best Languages for Android App Development in 2025*. Zazz. https://www.zazz.io/blog/top-programming-languages-for-developing-android-apps/