EatFit Requirements Specifications

(Final Deliverable – Week 8)

SDEV-520: Software Engineering

Instructor: Prof. Nathan Braun

Submitted: **April 21, 2025**

Team 2 – EatFit

Brian Richmond – Project-Manager

Jonathan Nguyen – QA & Use-Cacse Design

Sydney Isaacs – Technical Lead & Compliance

Kurt Mason – Backend & Integration Specialist

Revision History (Table 1)

Version	Date	Author	Modifications
1.0	Apr 4, 2025		Initial Design Draft
1.1	Apr 6, 2025	William Richmo…	Document Proofing
2.0	Apr 14, 2025	Kurt Mason / Jonathan Nguy…	Use Case Descriptive Activity Diagrams
2.6	Apr 23, 2025	Sydney Isaacs	Table/Figure Labels
3.0	Apr 25, 2025	William Richmo…	Proof and Final Release

Executive Summary

This document presents the comprehensive requirements specification for EatFit, a nutrition tracking application designed for home cooks. EatFit addresses the need for accessible nutrition analysis of home-cooked meals through flexible input methods and personalized guidance. The system integrates verified USDA nutritional data, supports multiple user roles (general users, nutritionists, and administrators), and complies with HIPAA and GDPR regulations.

The specification includes a vision statement, stakeholder analysis, detailed use cases, supporting requirements, and a technical approach built on React Native and Microsoft Azure services. This document serves as the foundation for development, quality assurance, and acceptance testing of the EatFit application, with particular emphasis on security, usability, and data accuracy.

Table of Contents

Revision History (Table 1)....2
Executive Summary....3
Table of Contents....4
1. Glossary of Terms....10
Glossary (Table 2)....10
2. Document Approvals....11
Document Approvals (Table 3)....11
3. Introduction....12
4. Problem Statement....12
5. Vision Statement....12
6. System Boundaries....13
7. Stakeholders....13
a. Stakeholder Engagement Strategy....14
Collaboration Mechanisms (Table 4)....14
Stakeholder Input on Requirements Prioritization....14
Ongoing Stakeholder Involvement....15
8. Key Features & Scope....15
Requirements Prioritization System....15
Version 1 - Core Features (Must-Have):....16
Version 2 - Future Enhancements (Nice-to-Have):....16
Trade-Off: Unified App vs. Companion App Role-Based Capabilities:....16
Capabilities Comparison (Table 5)....16
9. User Personas and Scenarios....17
Personas....17
Robert (Health-Focused Individual)....17
Michael (System Administrator)....18
Key Scenarios....18
Scenario 1: Quick Weeknight Meal Logging....18
Scenario 2: Tracking Macronutrients for Fitness Goals....18
Scenario 3: Nutritionist Client Monitoring....18
Scenario 4: Database Maintenance....18
10. Technical Approach....19
Design Architecture (Figure 1)....19
11. Domain Model....20
Key Entities (Table 6)....20
Entity Relationships....20
Data Dictionary (Table 7)....21

12. Constraints & Risks..... 22
13. Risk Management..... 24
Assessed Risks (Table 8)..... 24
Risk Monitoring and Review Process..... 27
Risk Priority Matrix (Table 9)..... 28
13.1 Risk-Driven Iteration Planning..... 28
Iterative Risk Assessments (Table 10)..... 28
14. Agile Iteration Planning & Work Items..... 29
Iteration Plan Table(Table 11)..... 30
Work Items Table (Table 12)..... 31
15. Compliance Mapping Table (HIPAA / GDPR)..... 33
Compliance Mapping Table (Table 13)..... 33
16. Product Roadmap (Post-MVP Enhancements)..... 34
17. Test Plan..... 34
17.1 Testing Objectives..... 34
17.2 Testing Levels..... 35
Unit Testing..... 35
Integration Testing..... 35
System Testing..... 35
User Acceptance Testing..... 35
17.3 Specialized Testing Types..... 36
Performance Testing..... 36
Security Testing..... 36
Accessibility Testing..... 36
Compatibility Testing..... 36
17.4 Test Environment..... 37
Hardware Environment Specifications (Table 14)..... 37
17.5 Test Data Management..... 37
17.6 Defect Management..... 37
17.7 Test Deliverables..... 38
17.8 Specific Test Cases..... 38
Test Cases (Table 15)..... 38
17.9 Test-Driven Development Approach..... 39
Unit Test Specifications (Table 16)..... 39
Continuous Testing Strategy..... 40
18. Milestones..... 40
Milestones (Table 17)..... 40
19. Next Steps & Team Assignments..... 41
Upcoming Actions:..... 41

Team Responsibilities (Table 18)..... 41
21. Use-Case Model..... 42
Actors (Table 19)..... 42
Use Case Diagram (Figure 2)..... 43
Key Use Case Relationships (Table 20)..... 44
Core Feature Relationships (Table 21)..... 44
20. Use-Case Specifications..... 45
UC-001: Log Meal Ingredients..... 45
Alternative Flows:..... 46
Exception Flows:..... 47
Frequency of Use:..... 48
Special Requirements:..... 48
Log Meal Ingredients (Figure 3) UML Diagram..... 48
Log Meal Ingtredients (Figure 4) Activity Diagram..... 49
UC-002: View Nutrition Report..... 49
Main Success Scenario:..... 50
Alternative Flows:..... 50
Exception Flows:..... 51
Frequency of Use:..... 52
Special Requirements:..... 52
UC-002: UML diagram..... 53
View Nutrition Report (Figure 5) UML Diagram..... 53
View Nutrition Report (figure 6) Activity Diagram..... 54
UC-003: Set Diet Goals..... 54
Main Success Scenario:..... 55
Alternative Flows:..... 55
Exception Flows:..... 56
Frequency of Use:..... 57
Special Requirements:..... 57
UC-003: UML diagram..... 57
Set Diet Goals (Figure 7) UML Diagram..... 57
Personalized Dietary Goals (Figure 8) Activity Diagram..... 58
UC-004: Send Reminders..... 59
Main Success Scenario:..... 59
Alternative Flows:..... 60
Exception Flows:..... 61
Frequency of Use:..... 61
Special Requirements:..... 61
Send Remineders (Figure 9) UML Diagram..... 62
Send Reminders (Figure10) Activity Diagram..... 63

UC-005: Maintain USDA Nutrition Database....63
Main Success Scenario:.... 64
Exception Flows:....65
Frequency of Use:....66
Special Requirements:....66
Maintain USDA Database (Figure 11) UML Diagram....67
Maintain USDA Nutrition Database (Figure 12) Activity Diagram....68
UC-006: User Authentication....68
UC-006.1: Register Account....69
Main Success Scenario:.... 69
Alternative Flows:.... 70
Exception Flows:....70
UC-006.2: Log in to Account.... 71
Main Success Scenario:.... 71
Alternative Flows:.... 71
Exception Flows:....72
UC-006.3: Log Out from Account....73
Main Success Scenario:.... 73
Exception Flows:....73
UC-006.4: Reset Password....74
Main Success Scenario:.... 74
Alternative Flows:.... 74
Exception Flows:....75
Frequency of Use:....75
Special Requirements:....76
User Authentication (Figure 13) UML Diagram....76
User Authentication (Figure 14) Activity Diagram....77
UC-007: Update Profile....77
UC-007.1: Edit Account Information....78
Main Success Scenario:.... 78
Alternative Flows:.... 79
Exception Flows:....79
UC-007.2: Manage Communication Preferences....80
Main Success Scenario:.... 80
Alternative Flows:.... 81
Exception Flows:....81
UC-007.3: Manage Privacy Settings.... 81
Main Success Scenario:.... 81
Alternative Flows:.... 82

Exception Flows:.....83
Frequency of Use:.....83
Special Requirements:.....83
Update Profile (Figure 15) UML Diagram.....84
Update Profile (Figure 16) Activity Daigram.....85
Account Management Relationships (Table 22).....86
21. User Interface Design.....87
UI Design Principles.....87
Key Screens.....87
22. Supporting Requirements.....88
Security Requirements (Table 23).....88
Availability Requirements (Table 24).....89
Usability Requirements (Table 25).....90
Maintainability Requirements (Table 26).....91
Performance Requirements (Table 27).....91
Legal & Compliance (Table 28).....92
Global Error Handling Strategy (Table 29).....93
23. Additional Requirements.....93
User Roles & Permissions Matrix (Table 30).....93
Assumptions & Dependencies (Table 31).....94
Data Requirements (Table 32).....96
Interfaces & Integration Requirements (Table 33).....97
Backup & Recovery Requirements (Table 34).....98
Logging & Audit Trail Requirements (Table 35).....99
24. External Interface Specifications.....99
USDA FoodData Central API.....99
Mobile Device Camera API.....100
Push Notification Services.....101
Authentication Providers.....101
Future Third-Party Integrations (Version 2).....102
25. Business Rules.....102
Business Rules (Table 36).....102
26. Deployment Requirements.....103
27. System Quality Attributes.....104
Quality Attributes (Table 38).....104
28. OpenUP Methodology Alignment.....105
29.1 OpenUP Principles Applied.....105
OpenUp Applications (Table 39).....105
29.2 OpenUP Practices Implemented.....105
29.3 OpenUP Work Products.....106

Document Finalization Checklist.....106
Consistency Check.....106
Completeness Verification.....107
OpenUP Alignment..... 107
Stakeholder Review..... 107
Appendix A: Traceability Matrix.....108
Core Use Cases to Business Needs (Table 40)..... 108
Security Requirements to Compliance Needs (Table 41)..... 109
Performance Requirements to Technical Components (Table 42)..... 109
A.1 Requirements to Use Cases Traceability.....110
Use Case Security Requirements (Table 43).....110
A.2 Requirements to Work Items Traceability.....110
Work Items Traceability Requirements (Table 44)..... 110
A.3 Bidirectional Traceability Example.....111
Appendix B: Decision Log..... 112
Decision Log (Table 45)..... 112
Appendix C: Architecture Notebook (Summary)..... 113
A.1 System Structure..... 113
A.2 Security Considerations.....113
A.3 Integration Strategy..... 113
A.4 Design Patterns & Principles..... 114
Appendix D: Architecture Evolution Plan..... 115
D.1 Architectural Decisions Points..... 115
Iterative Decision Points(Table 46).....115
D.2 Technical Debt Management..... 115
D.3 Architecture Validation.....116

1. Glossary of Terms

Glossary (Table 2)

Term	Definition
BMI	Body Mass Index – body fat measurement based on height and weight, calculated as weight(kg)/height(m)².
Client	A user who receives nutritional guidance from a nutritionist within the system.
Companion App	A separate mobile-first application designed for optimized smartphone user interaction that syncs with the main platform.
Diet Plan	A structured eating regimen designed to achieve specific nutritional or health goals.
GDPR	General Data Protection Regulation; EU data protection law governing personal data collection and processing.
HIPAA	Health Insurance Portability and Accountability Act; U.S. law governing data privacy and security for health applications.
Macros	Macronutrients include carbohydrates, proteins, and fats – the primary components of food that provide energy.
Meal Log	A record of consumed foods, including ingredients, amounts, and timestamps.
Micronutrients	Essential dietary elements, including vitamins and minerals, are required in small quantities.
MySQL	Relational database system for structured data storage and retrieval.
React Native	Cross-platform frontend framework for building native mobile/web apps using JavaScript.
REST API	Representational State Transfer Application Programming Interface; an architecture for web services allowing systems to communicate.

Smart Home Devices	Internet-connected devices like Alexa or Google Home enable voice commands and home automation.
SSL/TLS	Secure Sockets Layer/Transport Layer Security: Cryptographic protocols that provide secure communication.
Two-factor authentication (2FA)	A security process requiring two distinct forms of identification to access resources.
USDA	U.S. Department of Agriculture; source of verified nutritional data used as the primary database for food nutritional information.
Voice Input	A method of inputting data using spoken language via microphone-enabled devices.

2. Document Approvals

Document Approvals (Table 3)

Role	Name	Signature	Date
Project Manager	Brian Richmond	*William Richmond*	04/20/2025
Technical Lead	Sydney Isaacs	*Sydney Isaacs*	04/20/2025
Requirements Analyst	Kurt Mason	**Kurt Mason**	04/20/2025
Client Representative	Nathan Braun		04/20/2025
Quality Assurance	Jonathan Nguyen	*Jonathan Nguyen*	04/20/2025

3. Introduction

Project Name: EatFit
Team Members: Brian Richmond, Kurt Mason, Jonathan Nguyen, Sydney Isaacs
Document Purpose:
This document presents the Project Vision Statement for EatFit, a user-friendly nutrition tracking app designed for individuals who prepare meals at home. It aims to define the project's direction, outline key features, and demonstrate how our solution aligns with the needs of EatFit Inc. and its future users.
The app will empower users to understand the nutritional content of their meals using verified USDA data and flexible input options such as recipes, ingredients, photos, and voice commands. This document also outlines our technical approach, stakeholder needs, system boundaries, and a roadmap for growth. We aim to deliver a secure, scalable, and forward-thinking solution that adapts to the evolving health tech landscape.

4. Problem Statement

EatFit Inc. is a startup that wants to target the nutrition and diet segment of the mobile health (mHealth) app market. One opportunity identified through market research is to serve home cooks who lack the tools or knowledge to assess the nutritional quality of their meals. Recipes often include packaged products (e.g., oil, meat, bread) and fresh produce (e.g., vegetables, fruit), making nutrition tracking more complex.

This application addresses that gap by enabling users to analyze meals using various input methods—raw ingredients, custom recipes, photos, or voice. It will provide accessible, real-time feedback on nutritional value and allow users to easily track their diets across devices. The deeper problem we are solving is empowering individuals to make healthier food choices with clear, trustworthy nutritional insights.

5. Vision Statement

What EatFit Aims to Achieve:

- Provide an intuitive platform for tracking nutrition in home-cooked

Meals

- Helps users log, analyze, and monitor their dietary intake over time
- Offer flexibility for future feature additions and integrations

6. System Boundaries

In Scope:

- Ingredient and recipe analysis
- Integration with USDA data
- Nutritional breakdown and suggestions
- Data visualization, voice/photo input

Out of Scope:

- Medical advice or clinical diagnosis
- In-depth fitness tracking
- Third-party advertising or promotional features

7. Stakeholders

Primary Stakeholders:

- Home Cooks (Users): Individuals who want better control over their nutritional intake.
- EatFit Inc. (Client): Seeking a scalable, compliant, and engaging Solution.
- Team 2 (Developers): Responsible for system design, build, testing, and delivery.
- Nutritionists: Provide expert insights and manage health programs Data.
- Admins: Oversee system operations, user roles, and nutritional Content.

- External Entities: USDA for data, regulators for HIPAA/GDPR Compliance.

Stakeholder Profiles:

- Users want flexibility, easy input methods, and personalized Suggestions.
- Nutritionists need to track clients, send reminders, and adjust plans.
- Admins require tools to maintain data integrity and manage users Access.

a. Stakeholder Engagement Strategy

Stakeholder collaboration drives our development process. We've implemented the following mechanisms to ensure ongoing stakeholder involvement:

Collaboration Mechanisms (Table 4)

Mechanism	Frequency	Participants	Purpose
Requirements Workshops	Bi-weekly	All stakeholders	Elicit and validate requirements
Use Case Reviews	End of iterations 2-4	Users, Nutritionists, Product Owner	Validate use case accuracy
Architecture Reviews	End of iteration 4	Technical stakeholders	Validate technical approach
UI Prototype Walkthroughs	Weekly during iterations 3-5	Users, Client representatives	Gather feedback on usability
Risk Assessment Sessions	Start of each iteration	All stakeholders	Identify and evaluate new risks

Stakeholder Input on Requirements Prioritization

Requirements prioritization resulted directly from stakeholder workshops where the MoSCoW method (Must, Should, Could, Won't) was applied to features. Key prioritization decisions included:

i. User logging capabilities identified as highest priority by home cooks (primary users)

ii. USDA database integration prioritized based on nutritionist stakeholder input

iii. Smart device integration deferred to Version 2 based on technical stakeholder risk assessment

Ongoing Stakeholder Involvement

Stakeholders will remain engaged throughout implementation via:

1. Weekly progress demonstrations
2. User acceptance testing for each delivered increment
3. Issue triage and prioritization sessions
4. Regular review of analytics and user feedback once deployed

8. Key Features & Scope

Requirements Prioritization System

Requirements in this document are prioritized using the following three-level system:

- High Priority: Essential features required for the minimum viable product (MVP). These requirements must be implemented for the initial release.
- Medium Priority: Important features that provide significant value but are not critical for initial release. These may be implemented in early post-MVP iterations.
- Low Priority: Desirable features that enhance the product but can be deferred to later releases.

This prioritization system is used consistently throughout all requirements sections to guide development planning and resource allocation.

Version 1 - Core Features (Must-Have):

- USDA Database Integration
- Manual Recipe & Ingredient Entry
- Meal Logging & Nutrient Analysis
- Basic Reporting & Progress Tracking

Version 2 - Future Enhancements (Nice-to-Have):

- AI-powered Meal Recommendations
- Grocery List Generator
- Fitness App Integration (e.g., MyFitnessPal, Fitbit)
- Mobile-Optimized Companion App: A streamlined app optimized explicitly for smartphones to enhance the user experience on smaller screens and support features like barcode scanning and simplified logging.

Why the Companion App is Version 2: Our initial release will use React Native to deliver a unified cross-platform experience across web and mobile. This approach reduces development time and cost while maximizing market reach. However, a companion app in Version 2 would provide a more refined, mobile-first experience for daily logging and notifications.

Trade-Off: Unified App vs. Companion App Role-Based Capabilities:

Capabilities Comparison (Table 5)

Feature	**Unified Cross-Platform App**	**Mobile Companion App**
Development Cost	Lower	Higher
Time to Market	Faster	Slower
Code Maintenance	Easier	Requires dual support

Feature	**Unified Cross-Platform App**	**Mobile Companion App**
User Experience (Mobile)	Moderate	Optimized
Feature Specialization	General-purpose	Focused, mobile-tuned

- Admins
    - Manage user roles and permissions
    - Maintain nutrition database accuracy
    - Monitor system performance
    - Configure system settings
    - Generate usage analytics
- Nutritionists
    - Create diet programs, send reminders, and track subscribers
- Users
    - Log ingredients, track nutrition, receive health insights

9. User Personas and Scenarios

Personas

Maria (Home Cook)
Maria is a 35-year-old working parent who prepares meals for her family of four. She is health-conscious but has limited nutritional knowledge. Maria wants to ensure her family's meals are balanced but finds traditional calorie-counting apps tedious.

Robert (Health-Focused Individual)

Robert is a 28-year-old fitness enthusiast who prepares most of his meals at home. He tracks macronutrients carefully to support his training goals but

struggles with accurately measuring nutrition in home-cooked meals.

Dr. Sarah (Nutritionist)

Dr. Sarah is a certified nutritionist who works with clients managing various health conditions. She needs a reliable way to monitor her clients' actual food intake between appointments.

Michael (System Administrator)

Michael is the IT administrator responsible for maintaining the EatFit system. He needs to ensure data accuracy, system security, and compliance with health data regulations.

Key Scenarios

Scenario 1: Quick Weeknight Meal Logging

Maria makes a pasta dinner with jarred sauce, whole wheat pasta, ground turkey, and vegetables. She opens EatFit, selects "Log Meal," and quickly adds each ingredient using the search function and barcode scanner for packaged items. The app calculates the nutritional breakdown and shows that the meal is high in protein but could use more vegetables.

Scenario 2: Tracking Macronutrients for Fitness Goals

Robert prepares his weekly meal prep of chicken, rice, and vegetables. He logs each ingredient with precise measurements, saves it as a recipe, and checks if the macronutrient ratio meets his current goals. The system shows he's below his protein target for the day, so he adds a protein shake to his meal plan.

Scenario 3: Nutritionist Client Monitoring

Dr. Sarah logs into the professional dashboard to review her client's food logs before their upcoming appointment. She notices that the client has been consistently exceeding sodium intake. She sets up an automatic reminder to suggest lower-sodium alternatives when the client logs similar meals in the future.

Scenario 4: Database Maintenance

Michael receives a notification that a new USDA database update is available. He logs into the admin panel, initiates the database update process, reviews the changes summary, and approves the update during off-peak hours to minimize user disruption.

10.Technical Approach

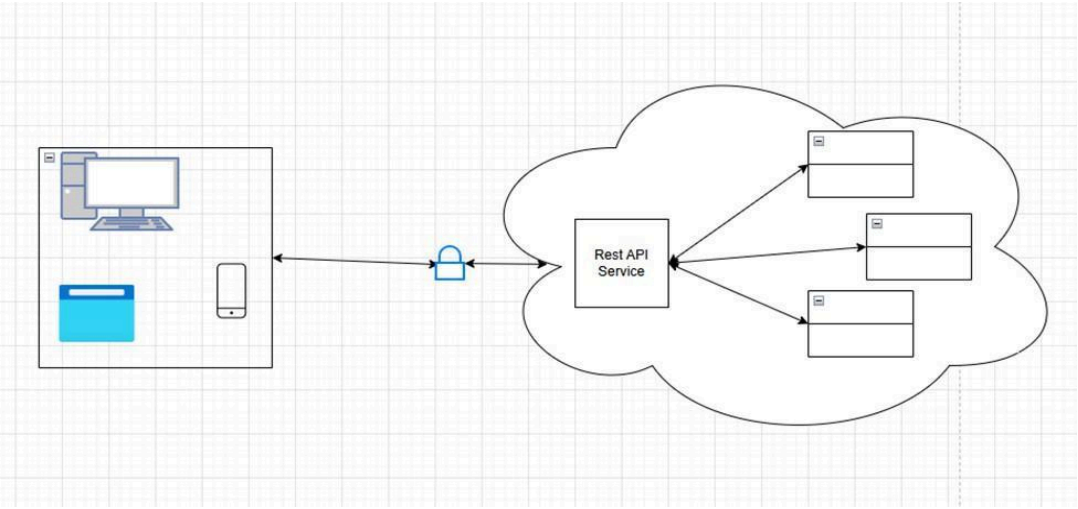
Tech Stack:

- Frontend: The team will use React Native to create a deployable user interface across multiple platforms. This will allow us to deploy compatible code without additional development.
- Backend: Microsoft Azure API applications will provide a security-managed environment for API endpoints developed to interact with client device applications.
- Database: Cosmos DB will store all client information. Client-specific information will be secured using row-level security, which will secure database information only and expose it to the respective users.
- Cloud: Microsoft Azure offers several HIPAA-compliant products. The platform is recognized for meeting the stringent requirements of HIPAA databases and applications. Authorization and authentication can be integrated and deployed across the suite of products that we intend to use

Compliance:

- Data handling will adhere to HIPAA and GDPR requirements to ensure user privacy and secure storage of sensitive nutritional data.

Design Architecture (Figure 1)



11.Domain Model

The following domain model presents the primary entities in the EatFit system and their relationships:

Key Entities (Table 6)

Entity	Description	Primary Attributes
User	Core user of the system	userID, name, email, password, height, weight, age, gender, activityLevel
Nutritionist	Professional providing nutritional guidance	nutritionistID, credentials, specializations, clientList
Admin	System administrator	adminID, permissions
Meal	Collection of ingredients consumed at a specific time	mealID, name, timestamp, mealType, totalNutrients
Ingredient	Food item used in meal preparation	ingredientID, name, servingSize, nutrientContent
Recipe	Reusable collection of ingredients with preparation instructions	recipeID, name, ingredients, instructions, totalNutrients
NutrientProfile	Nutritional composition of ingredients, recipes, and meals	macronutrients, micronutrients, calories
DietGoal	User's nutritional targets	goalID, targetCalories, targetNutrients, timeframe
NutritionReport	Analysis of user's nutritional intake	reportID, timeframe, actualVsTarget, recommendations

Entity Relationships

- User creates Meals
- User sets DietGoals
- User views NutritionReports

- Nutritionist manages multiple Users (as clients)
- Nutritionist sets DietGoals for clients
- Meals contain multiple Ingredients
- Recipes contain multiple Ingredients
- Admin maintains the USDA database
- Admin manages User and Nutritionist accounts

The domain model supports the key use cases while maintaining clear separation of concerns between different user roles and system functionalities.

Data Dictionary (Table 7)

Entity	Attribute	Data Type	Constraints	Description
User	userID	UUID	Primary Key, Not Null	Unique identifier for each user
User	email	String(255)	Unique, Not Null	User's email address used for login
User	password	String(64)	Not Null	Bcrypt hashed password
User	height	Decimal(5,2)	Nullable	User's height in centimeters
User	weight	Decimal(5,2)	Nullable	User's weight in kilograms
User	activityLevel	Enum	Not Null, Default='Moderate'	Activity level (Sedentary, Light, Moderate, Very Active, Extra Active)
Nutritionist	nutritionistID	UUID	Primary Key, Not Null	Unique identifier for each nutritionist
Nutritionist	credentials	JSON	Not Null	JSON object storing certification information
Meal	mealID	UUID	Primary Key, Not Null	Unique identifier for each meal
Meal	userID	UUID	Foreign Key, Not Null	Reference to the User who created the meal
Meal	timestamp	DateTime	Not Null	When the meal was

				consumed
Meal	mealType	Enum	Not Null	Type of meal (Breakfast, Lunch, Dinner, Snack)
Ingredient	ingredientID	UUID	Primary Key, Not Null	Unique identifier for each ingredient
Ingredient	name	String(255)	Not Null	Name of the ingredient
Ingredient	fdcID	String(20)	Nullable	USDA FoodData Central ID reference
MealIngredient	mealID	UUID	Foreign Key, Not Null	Reference to the Meal
MealIngredient	ingredientID	UUID	Foreign Key, Not Null	Reference to the Ingredient
MealIngredient	quantity	Decimal(8,2)	Not Null	Amount of ingredient used
MealIngredient	unitOfMeasure	String(50)	Not Null	Unit of measurement (g, oz, cup, etc.)
DietGoal	goalID	UUID	Primary Key, Not Null	Unique identifier for each goal
DietGoal	userID	UUID	Foreign Key, Not Null	Reference to the User
DietGoal		Integer	Nullable	Daily calorie target
DietGoal	targetNutrients	JSON	Nullable	JSON object storing nutrient targets

12.Constraints & Risks

- Performance: Potential lag from large datasets (USDA).
- Scalability: Data on the user base's initial size and growth projections should be used to stress-test the product's scalability.
- Accessibility: As a user-facing product, EatFit should consider

compliance with accessibility guidelines (e.g., WCAG 2.1, ADA) to ensure inclusivity for users with disabilities. Future versions may include screen reader support, voice navigation, and visual contrast optimization.

- Privacy: Compliance with HIPAA and GDPR is essential.
- Cost: Cloud services and third-party APIs may introduce expenses.
- Security: Nutrition data must be securely stored and encrypted.
- Technical Feasibility: Integration with devices and fitness platforms must be evaluated
- Timeline: Time to market is essential; team member availability and familiarity with the tech stack should be considered when setting Deadlines.
- Dependency Throughput: It will be necessary to verify the calls allotted to the USDA endpoint and manage that risk with a back-off strategy or possibly local caching to reduce the number of calls if the number is too limiting. This can also be mitigated by implementing an exponential backoff to limit user downtime if this does occur.

13.Risk Management

This risk management section includes a comprehensive analysis of potential risks, their impacts, and both mitigation and contingency strategies.

Assessed Risks (Table 8)

Risk ID	Description	Likelihood	Impact	Risk Score	Trigger Conditions	Mitigation Strategy	Contingency Plan	Owner
R-01	USDA API rate limits may throttle requests during peak hours	Medium	High	High	When API response times exceed 1500ms or when error rates exceed 5% on USDA API calls	Implement local caching and exponential backoff logic; maintain a local copy of frequently accessed data	Switch to offline mode with cached data; display "last updated" timestamp	Sydney Isaacs
R-02	Users may submit incomplete or invalid recipe data	High	Medium	High	When validation errors exceed 10% of submission attempts	Validate inputs in real-time and provide user prompts; implement field-level validation	Allow partial data submission with flagging for completion later	Jonathan Nguyen

R-03	HIPAA compliance breach through logging or data leakage	Low	High	Medium	Upon detection of any unencrypted PII in logs or any unauthorized data access attempt	Mask PII in logs, enable audit trails, and review access controls; conduct regular security audits	Breach notification protocol; immediate containment procedures; incident response team activation	Sydney Isaacs
R-04	React Native deployment issues due to team inexperience	Medium	Medium	Medium	When build failures exceed 20% or deployment time exceeds 60 minutes	Assign spike tasks and conduct early build/deployment trials; engage mentor with React Native expertise	Fallback to web-only for initial release with mobile following in phase 2	Brian Richmond
R-05	Azure cloud pricing spikes or regional outages	Low	Medium	Low	When monthly costs increase by 30%+ or when primary region availability drops below 99%	Monitor usage/budget alerts; scope AWS fallback if needed; implement multi-region deployment	Automatic failover to secondary region; temporary service degradation protocol	Kurt Mason

R-06	Accessibility barriers (e.g., screen reader compatibility)	Medium	High	High	When accessibility audit score falls below 85% compliance	Follow WCAG 2.1 during design; include in QA checklist; conduct targeted accessibility testing	Rapid-response accessibility fixes; alternative workflows for affected users	Jonathan Nguyen
R-07	Smart device integration (e.g., Alexa, Google Home) delayed	Medium	Low	Low	If integration testing reveals >10% failure rate in smart device integration	Modular API design; deprioritize as post-launch enhancement; clear separation of core and extended features	Communicate timeline changes to stakeholders; focus on core mobile/web experience	Brian Richmond
R-08	Timeline disruption due to academic or personal conflicts	Medium	Medium	Medium	When sprint velocity decreases by >20% for two consecutive sprints	Use early milestone delivery and buffer time to reduce risk; cross-train team members on critical components	Reprioritize backlog; adjust scope while maintaining core functionality	Kurt Mason

R-09	Data privacy regulations change during development	Low	High	Medium	Upon announcement of major privacy regulation changes with <6 months implementation deadline	Design flexible consent and privacy controls; parameterize privacy settings	Rapid compliance update protocol; temporary feature restrictions if needed	Sydney Isaacs
R-10	User adoption lower than expected after initial launch	Medium	High	High	When user acquisition falls 25% below projections or churn exceeds 8% monthly	Early user testing; incorporate feedback cycles; implement engaging features like progress tracking	Pivot marketing strategy; introduce incentives for early adopters; analyze and address drop-off points	Brian Richmond

Risk Monitoring and Review Process

The team will review risk status bi-weekly during sprint planning meetings. New risks identified during development will be added to the risk register and assigned appropriate responses. Risk owners will report on mitigation progress, and risk scores will be re-evaluated based on changing conditions or implemented controls.

Risk Priority Matrix (Table 9)

	High Impact	**Medium Impact**	**Low Impact**
High Likelihood	R-02, R-10	-	-
Medium Likelihood	R-06	R-04, R-08	R-07
Low Likelihood	R-03, R-09	R-05	-

13.1 Risk-Driven Iteration Planning

The iterations are organized to address highest-priority risks earliest. The connection between risk reduction and iteration planning is shown below, with particular focus on high-impact and high-likelihood risks:

Iterative Risk Assessments (Table 10)

Iteration	**Primary Risks Addressed**	**Risk Reduction Strategy**	**Success Metrics**
Iteration 1	R-02: Users submitting incomplete data (High Likelihood, High Impact)	Define clear data validation requirements; prototype input interfaces	Invalid data submission rate <10%
Iteration 2	R-10: Low user adoption (High Likelihood, High Impact)	Early user testing; incorporate user feedback loops	Positive user feedback on initial prototypes >80%

Iteration 3	R-06: Accessibility barriers (Medium Likelihood, High Impact)	Implement WCAG standards; conduct accessibility testing	WCAG compliance score >90%
Iteration 4	R-03, R-09: HIPAA/GDPR compliance (Low Likelihood, High Impact)	Review by compliance experts; implement security controls	Compliance assessment passing score
Iteration 5	R-04, R-08: React Native and timeline issues (Medium Likelihood, Medium Impact)	Technical spike on React Native; contingency planning	Successful prototype deployment
Iteration 6	R-01: USDA API limitations (Medium Likelihood, High Impact)	Prototype API integration; develop caching strategies	API response success rate >95%

14.Agile Iteration Planning & Work Items

The EatFit team adopted an Agile-inspired approach rooted in OpenUP’s evolutionary and incremental development model to structure our project and manage deliverables effectively. While we did not follow a full Scrum framework, our iterations were designed to deliver valuable increments, reduce key risks, and align with stakeholders' feedback cycles—each iteration built on the last, incorporating stakeholder input and addressing emerging requirements. The tables below summarize our iteration plan and list of tracked work items, illustrating team coordination and progress throughout the project lifecycle.

Iteration Plan Table(Table 11)

teration	Duration	Goals	Primary Output	Risk Reduction	Acceptance Criteria
Iteration 1	Mar 11 – Mar 18	Define project vision, outline features, assign initial roles	Vision Statement (Draft), Feature Backlog	Reduces scope uncertainty	Stakeholder agreement on vision; Feature priorities established
Iteration 2	Mar 18 – Mar 25	Create glossary, stakeholder list, initial use-case draft	Use-Case Model (Text + Initial Diagrams)	Reduces requirement ambiguity	Complete use case list; Stakeholder roles defined
Iteration 3	Mar 25 – Apr 1	Finalize use-cases, define supporting requirements	Functional Requirements, UML Refinements	Reduces technical uncertainty	Validated use cases; Technical approach approved
Iteration 4	Apr 1 – Apr 8	Complete UI/UX draft, define tech stack and integration points	Tech Stack Architecture, System Boundaries	Reduces integration risks	Architecture review complete; Integration points defined
Iteration 5	Apr 8 – Apr 15	Add validation (Vision merge, Risk List, Glossary updates)	Final Draft w/ Enhancements	Reduces documentation risks	All sections peer-reviewed; Traceability verified

Iteration 6	Apr 15 – Apr 23	Activity diagrams, review, finalize PDF submission	Final Submission Package	Addresses remaining quality risks	Final stakeholder approval; Document consistency verified

Each iteration followed OpenUP's micro-incremental approach, emphasizing daily team synchronization and continuous integration of evolving work products. At the conclusion of each iteration, stakeholders reviewed deliverables and provided feedback, which directly informed planning and priorities for subsequent iterations.

Work Items Table (Table 12)

ID	Work Item	Owner	Use Case Reference	Dependencies	Effort Estimate	Status
WI-01	Merge Vision Statement into Requirements Doc	Brian	N/A	None	2 hours	✅ Completed
WI-02	Define Use-Case Model	Kurt	UC-001 through UC-007	WI-01	8 hours	✅ Completed
WI-03	Create Activity Diagram – UC-01	Jonathan	UC-001	WI-02	3 hours	⏳ In Progress
WI-04	Create Activity Diagram – UC-03	Kurt	UC-003	WI-02	3 hours	⏳ In Progress
WI-05	Number Figures and	Sydney	N/A	All diagrams complete	2 hours	⏳ In Progress

	Table					
WI-06	Create Version History Table	Jonathan	N/A	None	1 hour	🔜 Scheduled
WI-07	Add Risk Management Section	Brian	N/A	WI-01	4 hours	✅ Completed
WI-08	Final PDF Packaging	Brian	N/A	All other items	2 hours	🔜 Scheduled
WI-09	Final Proofreading	Entire Team	N/A	WI-08	4 hours (1h per member)	🔜 Scheduled
WI-10	Create Unit Test Specifications for Meal Logging	Kurt	UC-001	WI-02	6 hours	🔜 Scheduled
WI-11	Define Integration Test Scenarios	Sydney	UC-006	WI-02	6 hours	🔜 Scheduled
WI-12	Define Integration Test Scenarios	Jonathan	UC-001, UC-002, UC-005	WI-02, WI-10	8 hours	🔜 Scheduled

15.Compliance Mapping Table (HIPAA / GDPR)

Compliance Mapping Table (Table 13)

Requirement	Regulation	Implementation in EatFit
Data encryption in transit & at rest	HIPAA / GDPR	SSL/TLS for all API traffic; Azure-encrypted Cosmos DB
Consent for data processing	GDPR	Privacy policy acceptance required at signup
Data access audit trail	HIPAA	Logging of user and nutritionist activity
Role-based access control	HIPAA / GDPR	Defined roles for Admins, Nutritionists, and Users
Right to be forgotten / data deletion	GDPR	Account deletion feature planned for post-launch
Minimal data collection	GDPR	Only essential personal and health-related data is collected
Breach notification process	HIPAA	Email notifications and logging for security incidents (future implementation)

16. Product Roadmap (Post-MVP Enhancements)

To ensure long-term relevance and extensibility, EatFit's roadmap includes several high-impact features for future releases. These enhancements align with evolving user needs and technology trends.

- Fitness Tracker Integration (Fitbit, Apple Health)
- Voice-Enabled Logging via Smart Home Devices (Alexa, Google Home)
- Advanced Grocery Planning Tools (Ingredient substitution, meal prep lists)
- AI-Powered Nutritionist Dashboard (Recommendations, trend insights)
- Recipe Sharing and Social Integration
- Data Export Features for Medical Professionals
- Internationalization (Localized food databases and multi-language support)
- Gamification of Healthy Habits (Badges, streaks, challenges)

17. Test Plan

This section outlines the testing approach for EatFit, combining test plans, methodologies, and specific test requirements to ensure quality and compliance.

17.1 Testing Objectives

- Verify the system meets all functional and non-functional requirements

- Validate that the application satisfies user needs and delivers value
- Ensure compliance with HIPAA and GDPR regulations
- Identify and resolve defects early in the development lifecycle
- Confirm the system works across all supported platforms and devices

17.2 Testing Levels

Unit Testing

- Scope: Individual components and functions
- Approach: Automated tests using Jest
- Requirement: 80% code coverage minimum (TST-001)
- Responsibility: Developers

Integration Testing

- Scope: Component interactions and API endpoints
- Approach: Automated API tests and interface verification
- Requirement: All interfaces covered (TST-002)
- Responsibility: Development team with QA oversight

System Testing

- Scope: End-to-end functionality and workflows
- Approach: Manual and automated testing of complete scenarios
- Requirement: All use cases verified
- Responsibility: QA team

User Acceptance Testing

- Scope: Core features and critical workflows
- Approach: User representatives test against real-world scenarios

- Requirement: Key stakeholders must approve (TST-010)
- Responsibility: QA team with stakeholder participation

17.3 Specialized Testing Types

Performance Testing

- Approach: Load, stress, and endurance testing
- Tools: JMeter, Firebase Test Lab
- Key Metrics: Response times, throughput, resource utilization
- Requirement: 200% expected load handling (TST-004)

Security Testing

- Approach: Vulnerability scanning, penetration testing, code analysis
- Frequency: Quarterly assessments
- Requirement: OWASP Top 10 compliance (TST-005)

Security testing will specifically include test scenarios for SQL injection, XSS, CSRF, broken authentication, sensitive data exposure vulnerabilities, and other OWASP Top 10 risks, with detailed test cases for each vulnerability type.

Accessibility Testing

- Standard: WCAG 2.1 AA
- Tools: Axe, screen readers, manual testing
- Requirement: All core functions accessible (TST-006)

Compatibility Testing

- Platforms: Web browsers, iOS, Android

- Coverage: Last two major versions of each platform
- Devices: At least 5 different mobile device models
- Requirement: Consistent experience across platforms (TST-007, TST-008)

17.4 Test Environment

Hardware Environment Specifications (Table 14)

Environment	Purpose	Refresh Cycle	Data
Development	Developer testing	Continuous	Synthetic
QA	Formal testing	Weekly	Anonymized
Staging	Pre-production validation	Bi-weekly	Production-like
Production	Live system	N/A	Real user data

17.5 Test Data Management

- Synthetic data generation for development and testing
- Anonymized data for performance testing
- Data masking for any production data used in non-production environments
- Automated test data cleanup after test execution

17.6 Defect Management

- Classification: Critical, High, Medium, Low
- Lifecycle: New → Assigned → Fixed → Verified → Closed
- Exit Criteria: No open Critical or High defects for release

- Regression Testing: Required after fixing defects

17.7 Test Deliverables

- Test plans for each release
- Test cases mapped to requirements
- Automated test scripts
- Test execution reports
- Defect reports and resolution status
- Test coverage analysis

17.8 Specific Test Cases

Test Cases (Table 15)

Test Case ID	Feature	Test Description	Expected Result
TC-01	Log Meal	User logs a meal using manual input	Meal is saved, nutrients are calculated
TC-02	Log Meal	User logs a meal using voice input	Voice is transcribed; nutrients are displayed
TC-03	Analyze Nutrients	User opens meal details	Macronutrients and micronutrients appear
TC-04	Role Permissions	Nutritionist attempts to access admin dashboard	Access is denied with appropriate message
TC-05	Accessibility	User navigates with screen reader	All UI components are screen-reader compatible
TC-06	USDA API	API rate limit is reached	System retries or fails gracefully with message

TC-07	Account Deletion	User requests account deletion	Account and associated data are deleted and confirmed
TC-08	Invalid Recipe Input	User submits incomplete ingredient list	Error prompt asks for correction
TC-09	Data Export	User requests data download	Complete data package provided in requested format
TC-10	Performance	System under peak load	Response times remain under specified thresholds

17.9 Test-Driven Development Approach

EatFit will implement test-driven development throughout the project lifecycle. Each requirement will have associated test cases developed before implementation begins.

Unit Test Specifications (Table 16)

Requirement ID	**Test Specification**	**Implementation Guidance**
DAT-001	Test_MealEntry_RequiredFields	Verify meal entries cannot be saved without name, quantity, and unit
DAT-002	Test_Ingredient_USDAValidation	Verify ingredients are properly mapped to USDA database entries
SEC-004	Test_Password_BcryptHashing	Verify passwords are properly hashed using bcrypt with appropriate salt
PRF-001	Test_NutritionReport_GenerationTime	Measure report generation time under various data loads

Continuous Testing Strategy

1. Automated Testing Pipeline: All code commits will trigger automated unit tests
2. Test Coverage Monitoring: Maintain minimum 80% code coverage across all components
3. Regression Test Suite: Automated regression testing before each iteration completion
4. Performance Test Benchmarks: Continuous monitoring of performance against baseline metrics

The test-driven approach ensures requirements are testable and verified throughout development, reducing defects and ensuring alignment with stakeholder expectations. Test cases evolve alongside the implementation, with feedback from testing informing subsequent iterations.

18.Milestones

Milestones (Table 17)

Phase	**Milestone Description**	**Duration**
Analysis	Requirements gathering	10 days
Planning	Scope definition and acceptance criteria	5 days
Design	UI/UX and system architecture	5 days
Development	Build core functionalities	20 days
Integration	Component testing and system build	10 days
Installation	Deployment to the test environment	5 days
Testing	End-to-end testing and QA	10 days

19.Next Steps & Team Assignments

Upcoming Actions:

- Finalize stakeholder feedback in Sunday's meeting
- Assign remaining project roles
- Finalize and submit Vision Document by March 30

Team Responsibilities (Table 18)

Role	**Team Member**	**Responsibilities**
Project Manager	Brian Richmond	Timeline oversight, coordination
Technical Lead	Sydney Isaacs	Tech stack decisions, HIPAA compliance
Requirements Analyst	Kurt Mason	Document requirements from stakeholders
Frontend Developer	Brian Richmond	Build and style the user interface
Backend Developer	Kurt Mason	Build a database and server-side logic
QA Specialist	Jonathan Nguyen	Create test plans and execute QA

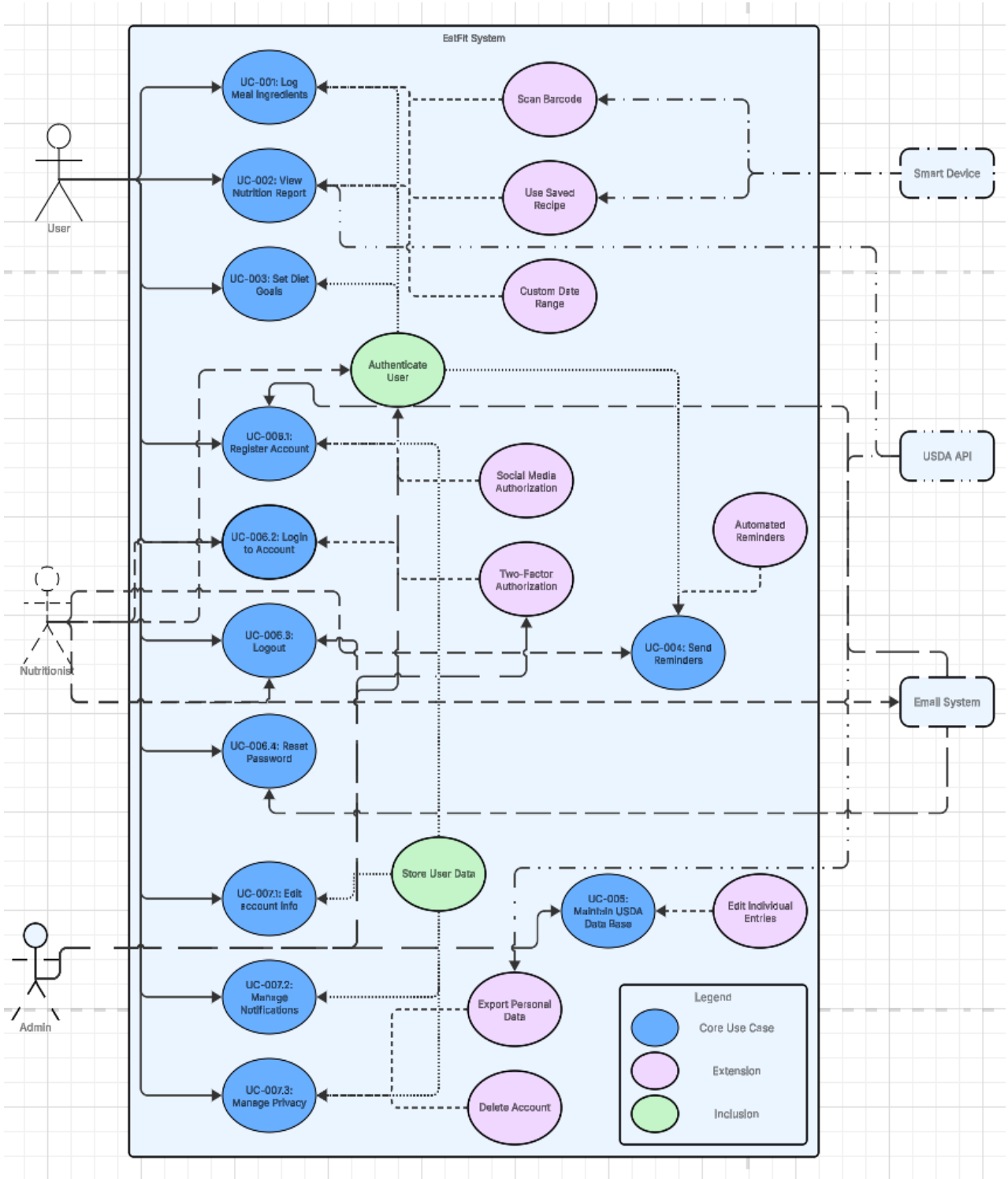
21. Use-Case Model

Actors (Table 19)

Actor ID	Actor Name	Description
ACT-001	User	The primary user of the application is the one who logs meals and views nutritional information

Actor ID	Actor Name	Description
ACT-002	Nutritionist	A professional who provides dietary guidance and sets up meal plans for clients
ACT-003	Admin	The system administrator who manages the application and database
ACT-004	USDA API	External system providing nutritional data
ACT-005	Smart Device	Optional external hardware for data input (planned for version 2)

Use Case Diagram (Figure 2)



Key Use Case Relationships (Table 20)

UC-006.1: Register Account	Prerequisite for	All other system functions
UC-006.2: Log in to Account	Prerequisite for	All protected system functions
UC-003: Set Diet Goals	Impacts	UC-002: View Nutrition Report
UC-005: Maintain USDA Database	Impacts	UC-001/UC-002: Meal/Nutrition Data
UC-007.2: Manage Notifications	Affects	UC-004: Send Reminders
UC-007.3: Manage Privacy	Governs	Data access across all functions

Core Feature Relationships (Table 21)

Source Use Case ID	**Relationship Type**	**Target Use Case ID**	**Description**
UC-001	prerequisite for	UC-002	Log Meal Ingredients (UC-001) is a prerequisite for View Nutrition Report (UC-002)
UC-003	affects	UC-002	Set Diet Goals (UC-003) affects the presentation of data in View Nutrition Report (UC-002)
UC-004	prompts	UC-001	Send Reminders (UC-004) may prompt users to perform Log Meal Ingredients (UC-001)
UC-005	supports	UC-002	Maintain USDA Nutrition Database (UC-005) to ensure accurate data for

Source Use Case ID	Relationship Type	Target Use Case ID	Description
			View Nutrition Report (UC-002)

20.Use-Case Specifications

UC-001: Log Meal Ingredients

ID: UC-001
Priority: High
Primary Actor: User
Stakeholders: User, Nutritionist
Goal: Allow users to input one or more ingredients from a home-cooked meal for nutritional analysis

Preconditions:

- The user is authenticated in the system
- The user has an active account
- An internet connection is available

Success Guarantee:

- Ingredient data is stored in the database
- The system acknowledges successful logging
- Data is available for nutritional analysis

Main Success Scenario:

1. The user selects "Log Ingredients" from the dashboard
2. The system displays the ingredient input form
3. The user enters the ingredient name in the search field

4. The system displays matching ingredients from the USDA database
5. The user selects the correct ingredient from the list
6. The user enters the quantity and unit of measurement
7. The user can add additional ingredients by repeating steps 3-6
8. The user selects "Save Meal" when all ingredients are entered
9. The system validates data input
10. The system stores meal data with a timestamp
11. The system displays a confirmation message

Alternative Flows:

A1. The user selects a saved recipe

1. At step 2, the user selects the "Use Saved Recipe" option
2. The system displays a list of the user's saved recipes
3. The user selects a recipe from the list
4. The System populates the ingredient form with saved recipe data
5. The user can modify quantities if needed
6. Continue at step 8 of the main flow

A2. The user scans the product barcode

1. At step 3, the user selects the "Scan Barcode" option
2. The system activates the device's camera
3. The user scans the product barcode
4. The system retrieves product information from the database
5. Continue at step 6 of the main flow

A3. Ingredient not found in the database

1. At step 4, if the ingredient is not found
2. The system displays the "Ingredient not found" message
3. The system offers an "Add Custom Ingredient" option
4. The User enters a custom ingredient name and known nutritional values
5. The system saves custom ingredients to the user's database
6. Continue at step 6 of the main flow

Exception Flows:

E1. Network connection lost

1. The system detects connection loss during any operation
2. The system saves the current data to local storage
3. The system displays the "Connection lost" message
4. The system attempts to reconnect
5. When the connection is restored, the system syncs local data with the server
6. The user continues from where they left off

E2. Invalid quantity input

1. At step 6, the user enters an invalid quantity (negative, zero, or non-numeric)
2. The system displays the error message "Please enter a valid quantity."
3. The user corrects the input
4. Continue at step 7 of the main flow

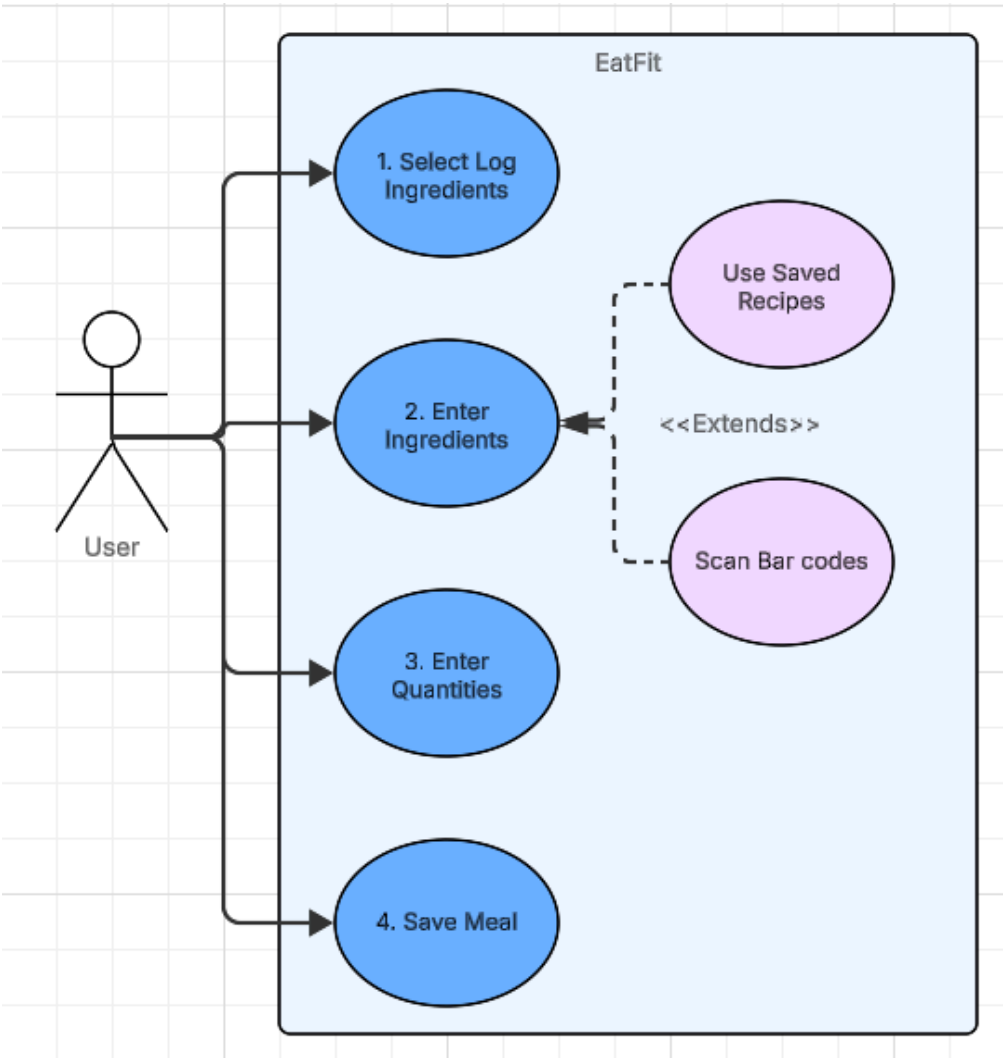
Frequency of Use:

Multiple times daily per user

Special Requirements:

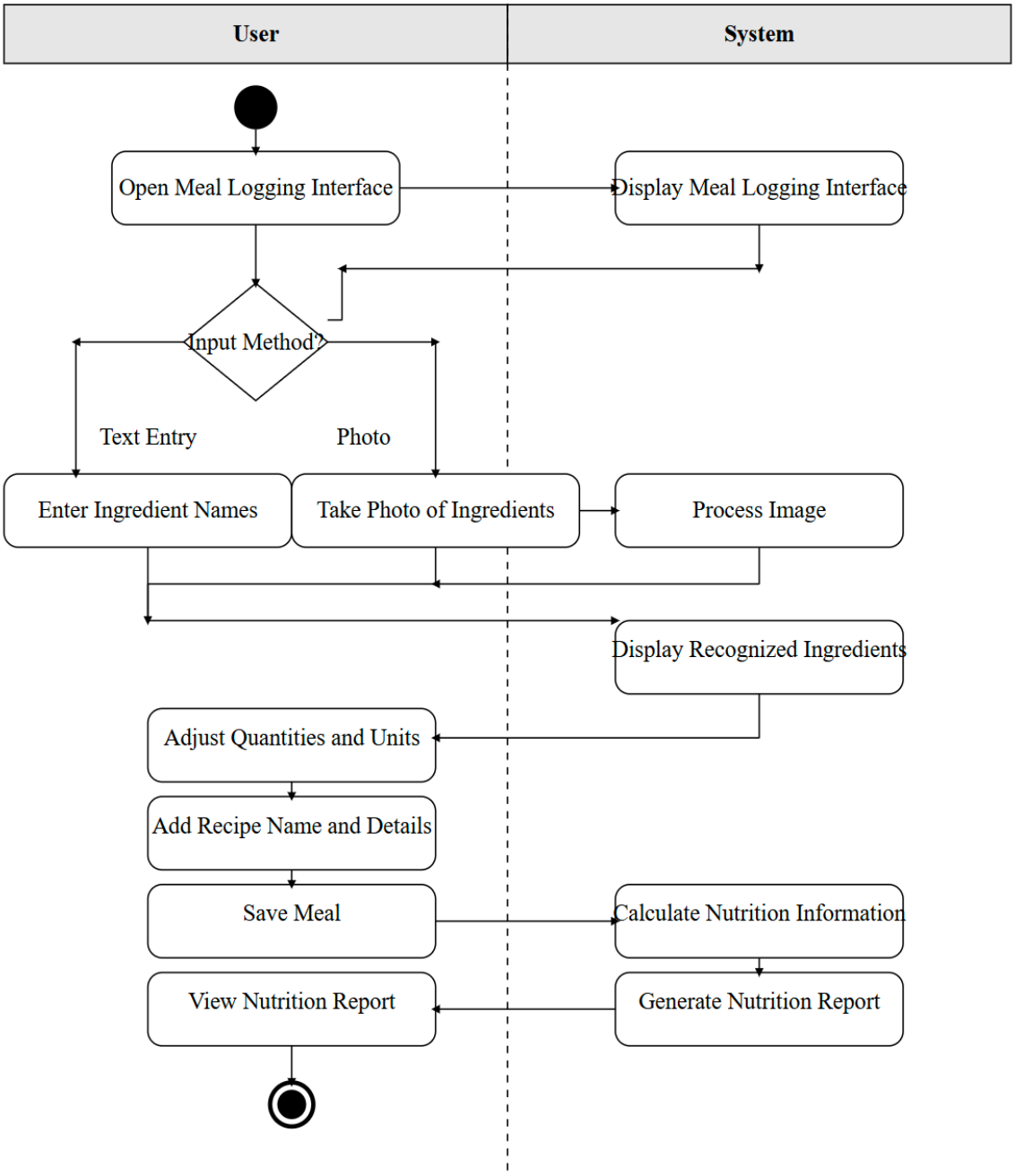
- Must work offline with data syncing when the connection is restored
- Must support voice input for accessibility
- Must accept metric and imperial measurements

Log Meal Ingredients (Figure 3) UML Diagram



Log Meal Ingtredients (Figure 4) Activity Diagram

Activity Diagram: Log Meal Ingredients (UC-001)



UC-002: View Nutrition Report

ID: UC-002
Priority: High
Primary Actor: User
Stakeholders: User, Nutritionist
Goal: Display a comprehensive nutritional breakdown of logged meals

Preconditions:

- The user is authenticated in the system
- At least one meal has been logged
- The nutritional databases are accessible

Success Guarantee:

- The nutritional data is accurately calculated
- The report is displayed with appropriate visualizations
- The users can interact with the report to explore details

Main Success Scenario:

1. The user selects "View Report" from the dashboard
2. The system presents report type options (daily, weekly, monthly)
3. The user selects the desired report timeframe
4. The system fetches meal log data for the selected timeframe
5. The system queries the USDA database for nutritional information
6. The system calculates nutritional totals and averages
7. The system compares results with the user's dietary goals (if set)
8. The system generates appropriate visualizations (charts, graphs)
9. The system displays the complete nutritional report to the user
10. The users can view breakdowns by meal, day, or nutrient

Alternative Flows:

A1. The user selects a specific date range

1. At step 2, the user selects the "Custom Date Range" option
2. The system displays a date picker interface

3. The user selects start and end dates
4. Continue at step 4 of the main flow

A2. User requests report export

1. After step 9, the user selects the "Export Report" option
2. The system offers format options (PDF, CSV, JSON)
3. The user selects the desired format
4. The system generates an export file
5. The system initiates a download or shares a file
6. Return to step 10 of the main flow

A3. User filters by nutrient type

1. After step 9, the user selects the "Filter" option
2. The system displays filter options (macros, vitamins, minerals)
3. The user selects the desired filter categories
4. The system updates the report to show only selected nutrients
5. Return to step 10 of the main flow

Exception Flows:

E1. No meals logged in the selected timeframe

1. At step 4, if no meal data is found
2. The system displays the "No meal data available for the selected period" message.
3. The system offers options to choose a different time frame or log a meal
4. The user selects the desired action

E2. Error accessing nutritional database

1. At step 5, if the USDA database cannot be accessed
2. The system attempts to use cached nutritional data
3. If cached data is available, the system displays a report with a "Using cached data" notice
4. If no cached data, the system displays the error message and suggests trying later.

Frequency of Use:

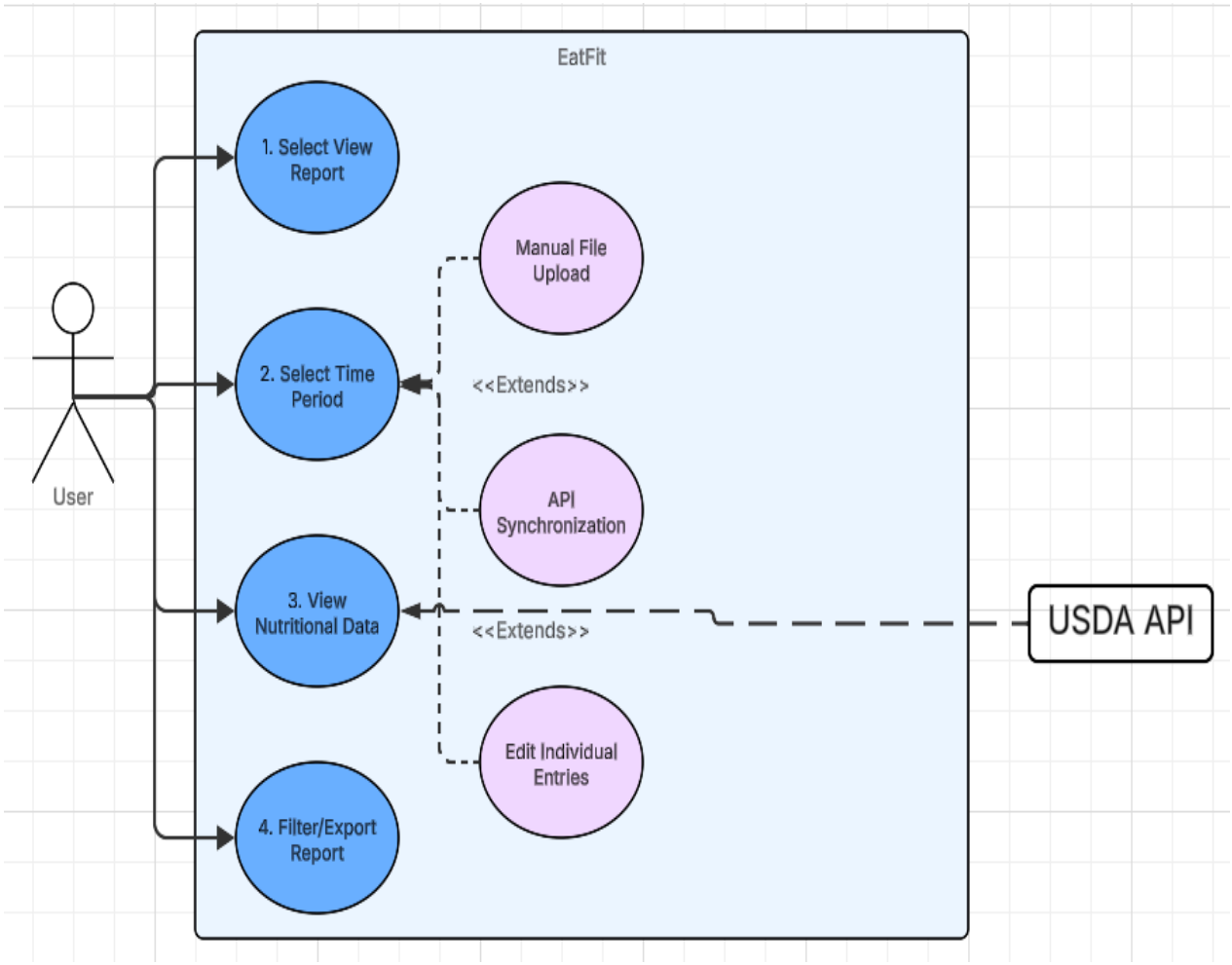
Daily per user

Special Requirements:

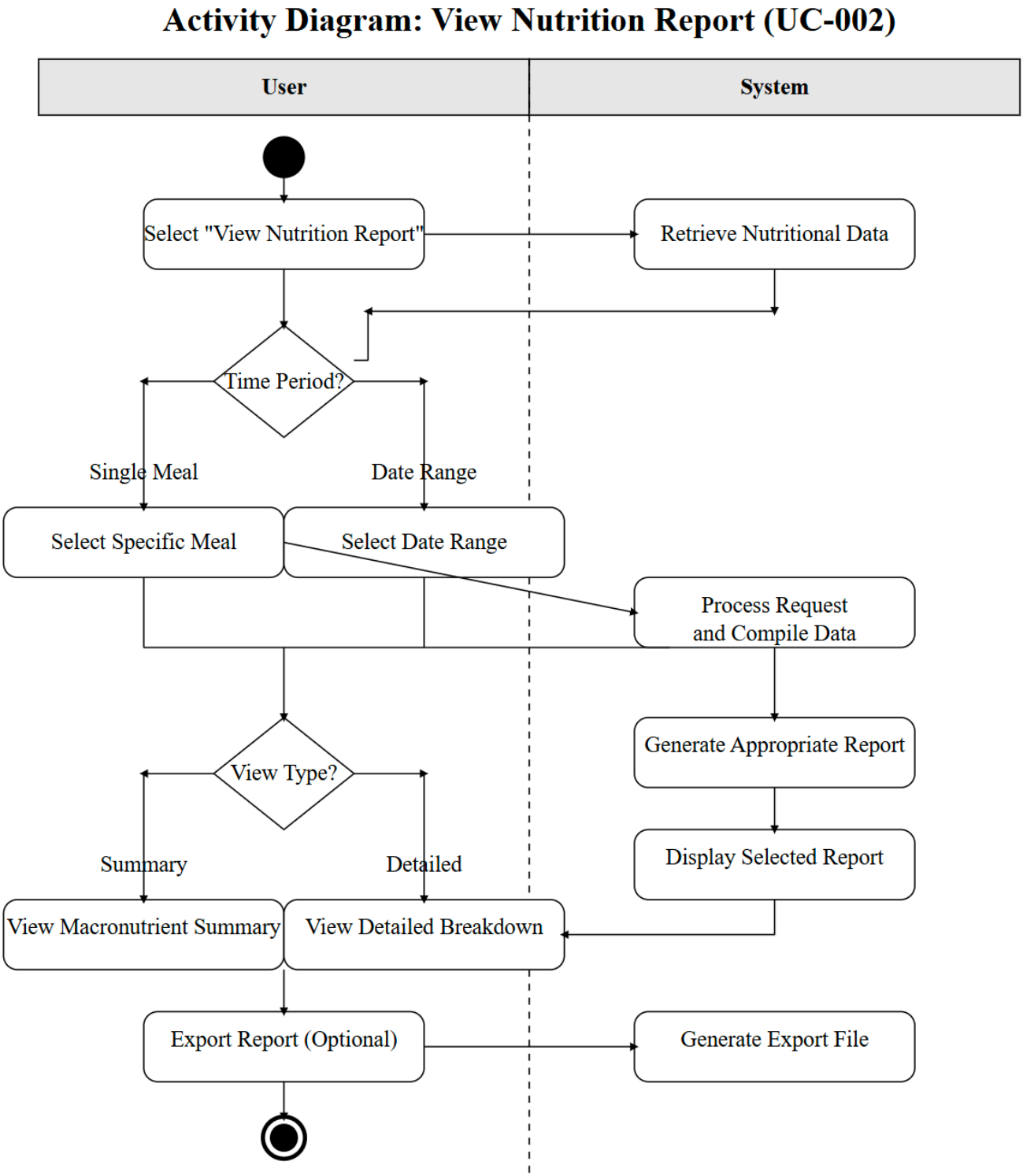
- Reports must be generated in under 2 seconds
- Must support colorblind-friendly visualization options
- Must support data export in multiple formats

UC-002: UML diagram

View Nutrition Report (Figure 5) UML Diagram



View Nutrition Report (figure 6) Activity Diagram



UC-003: Set Diet Goals

ID: UC-003

Priority: Medium

Primary Actor: User

Stakeholders: User, Nutritionist
Goal: Allow users to set personalized dietary goals for tracking and recommendations

Preconditions:

- The user is authenticated in the system
- The user has completed initial profile setup (age, height, weight, activity level)

Success Guarantee:

- Goals are saved to the user profile
- Goals are used for nutritional recommendations
- Goals are visible in reports for comparison

Main Success Scenario:

1. The user navigates to the "Diet Goals" section from profile settings
2. The system displays current goals (if any) and the goal-setting form
3. The system suggests recommended values based on the user profile
4. The user enters or adjusts the daily calorie target
5. The user enters or adjusts macronutrient targets (protein, carbs, fat)
6. The user optionally sets specific nutrient targets (sodium, fiber, etc.)
7. The user selects the "Save Goals" button
8. The system validates input values against healthy ranges
9. The system saves goals to the user's profile
10. The system confirms a successful update with a notification

Alternative Flows:

A1. The user chooses a preset diet plan

1. At step 2, the user selects the "Use Preset Plan" option
2. The system displays a list of preset diet plans (keto, vegan, etc).

3. The user selects a plan from the list
4. The system populates the goal form with preset values
5. The user can make adjustments if desired
6. Continue at step 7 of the main flow

A2. The nutritionist sets goals for a client

1. The nutritionist logs into a professional dashboard
2. The nutritionist selects a client from the client list
3. The nutritionist navigates to the client's "Diet Goals" section
4. The system displays the goal-setting form
5. The nutritionist enters appropriate values based on the client's needs
6. The nutritionist adds optional notes/explanations
7. Continue at step 7 of the main flow, with the system notifying the client of the update

Exception Flows:

E1. Invalid goal values

1. At step 8, if values are outside healthy ranges
2. The system displays a warning message with recommended ranges
3. The user can choose to adjust values or override the warning
4. If the user overrides, the system logs the exception and records the goals with a warning flag

E2. Conflicting nutrient targets

1. At step 8, if the system detects a mathematically impossible combination
2. The system displays a specific conflict explanation
3. The system suggests possible adjustments
4. The user adjusts values to resolve the conflict

5. Continue at step 9 of the main flow

Frequency of Use:

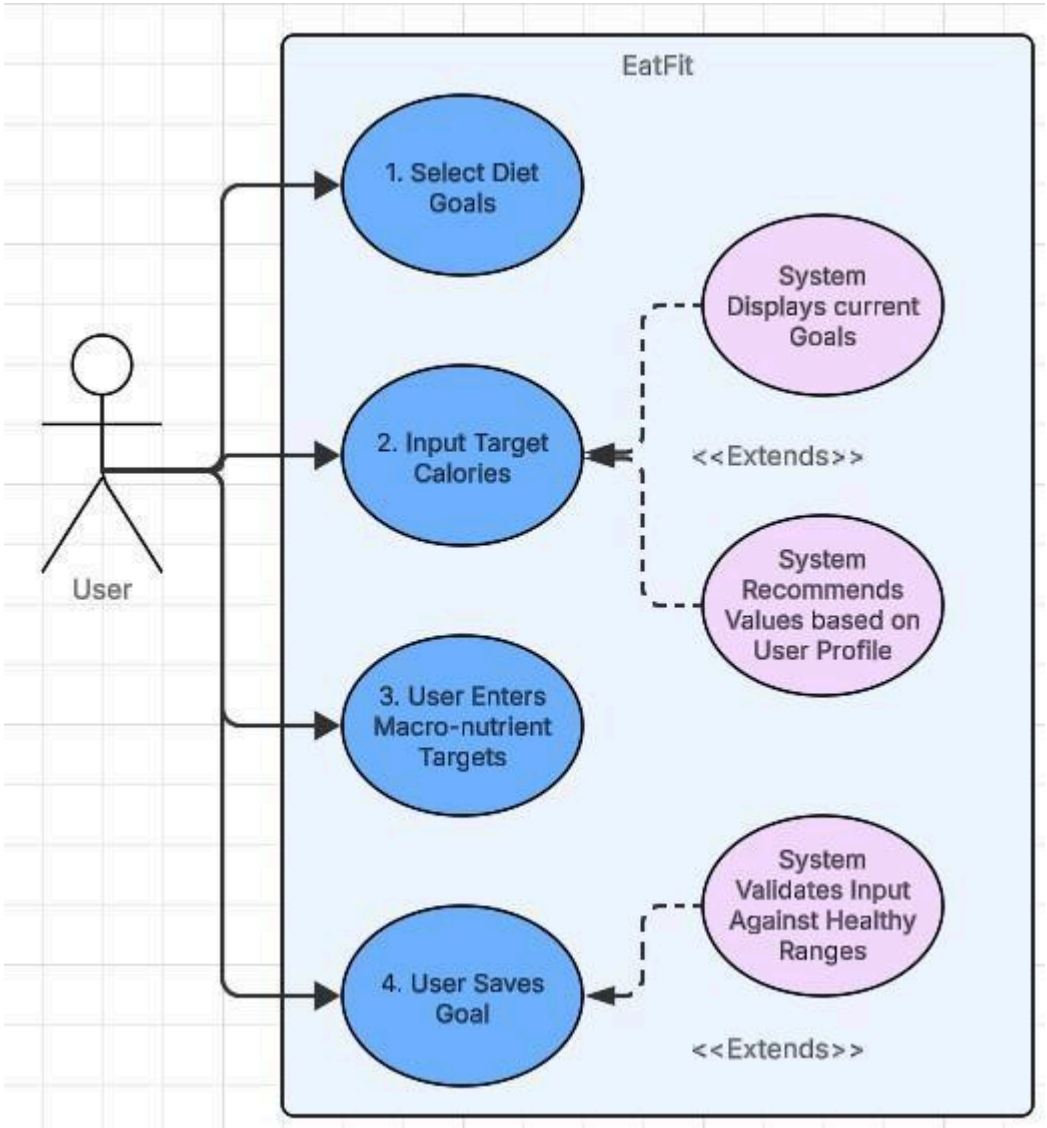
Weekly per user

Special Requirements:

- Must include educational information about healthy ranges
- Must support different goal types (weight loss, maintenance, gain)
- Must account for special health conditions (diabetes, hypertension)

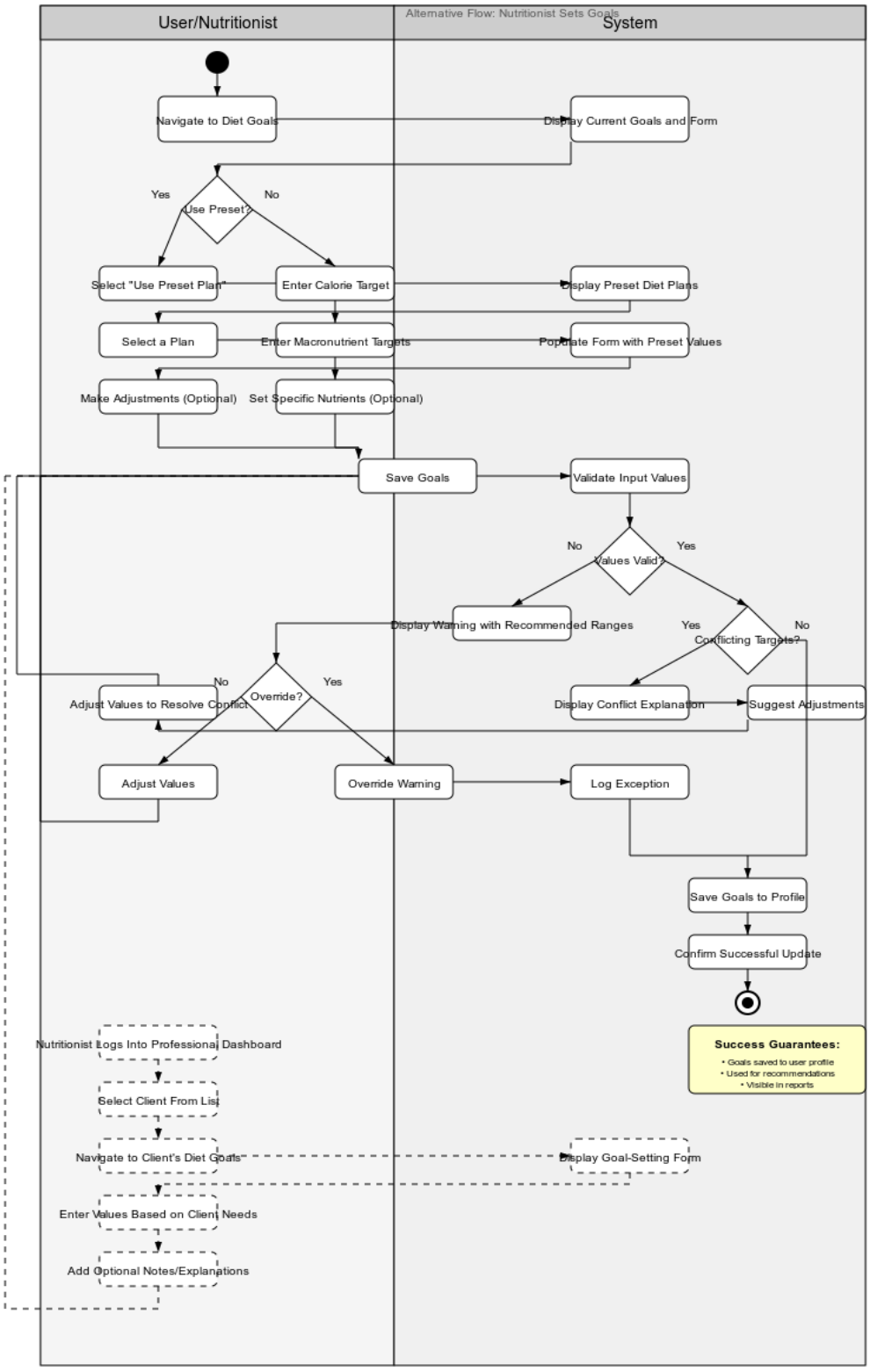
UC-003: UML diagram

Set Diet Goals (Figure 7) UML Diagram



Personalized Dietary Goals (Figure 8) Activity Diagram

Activity Diagram: Setting Personalized Dietary Goals



UC-004: Send Reminders

ID: UC-004
Priority: Medium
Primary Actor: Nutritionist
Stakeholders: Nutritionists, User
Goal: Enable nutritionists to send timely reminders to clients about dietary goals or logging meals

Preconditions:

- The nutritionist is authenticated with appropriate permissions
- The nutritionist has at least one client assigned
- Notification services are operational

Success Guarantee:

- Reminders are scheduled according to the specified timing
- Notifications are delivered to intended recipients
- The system tracks delivery and read status

Main Success Scenario:

1. The nutritionist navigates to the "Client Management" section
2. The nutritionist selects the "Send Reminders" option
3. The system displays the client selection interface (individuals or groups)
4. The nutritionist selects target recipients
5. The system presents reminder template options
6. The nutritionist selects or creates a template
7. The nutritionist customizes the message content and adds attachments if needed

8. The nutritionist sets the delivery schedule (immediate, recurring, or specific time)
9. The nutritionist submits a reminder for delivery
10. The system validates the reminder configuration
11. The system queues reminders for delivery
12. The system confirms successful scheduling
13. The system delivers notifications according to schedule
14. The system logs delivery status for reporting

Alternative Flows:

A1. Automated reminders based on user activity

1. The nutritionist navigates to the "Automation Rules" section
2. The nutritionist creates a rule (e.g., "Send reminder if no meals logged for 2 days")
3. The nutritionist configures rule parameters and message content
4. The nutritionist activates the rule
5. The system monitors user activity based on rule conditions
6. The system automatically sends reminders when conditions are met
7. The system logs all automated notifications

A2. Client reminder frequency preferences

1. At step 8, the system displays client communication preferences
2. The nutritionist adjusts schedules to align with client preferences
3. Continue at step 9 of the main flow

Exception Flows:

E1. Notification service unavailable

1. At step 13, if the notification service is down
2. The system queues messages for delayed delivery
3. The system alerts the administrator about a service issue
4. The system attempts delivery when service is restored
5. The system updates the delivery status and notifies the nutritionist

E2. The client has disabled notifications

1. At step 10, the system identifies clients with disabled notifications
2. The system warns the nutritionist about delivery limitations
3. The nutritionist can choose to proceed or remove those clients
4. The system only delivers to clients with enabled notifications
5. The system logs exceptions in the delivery report

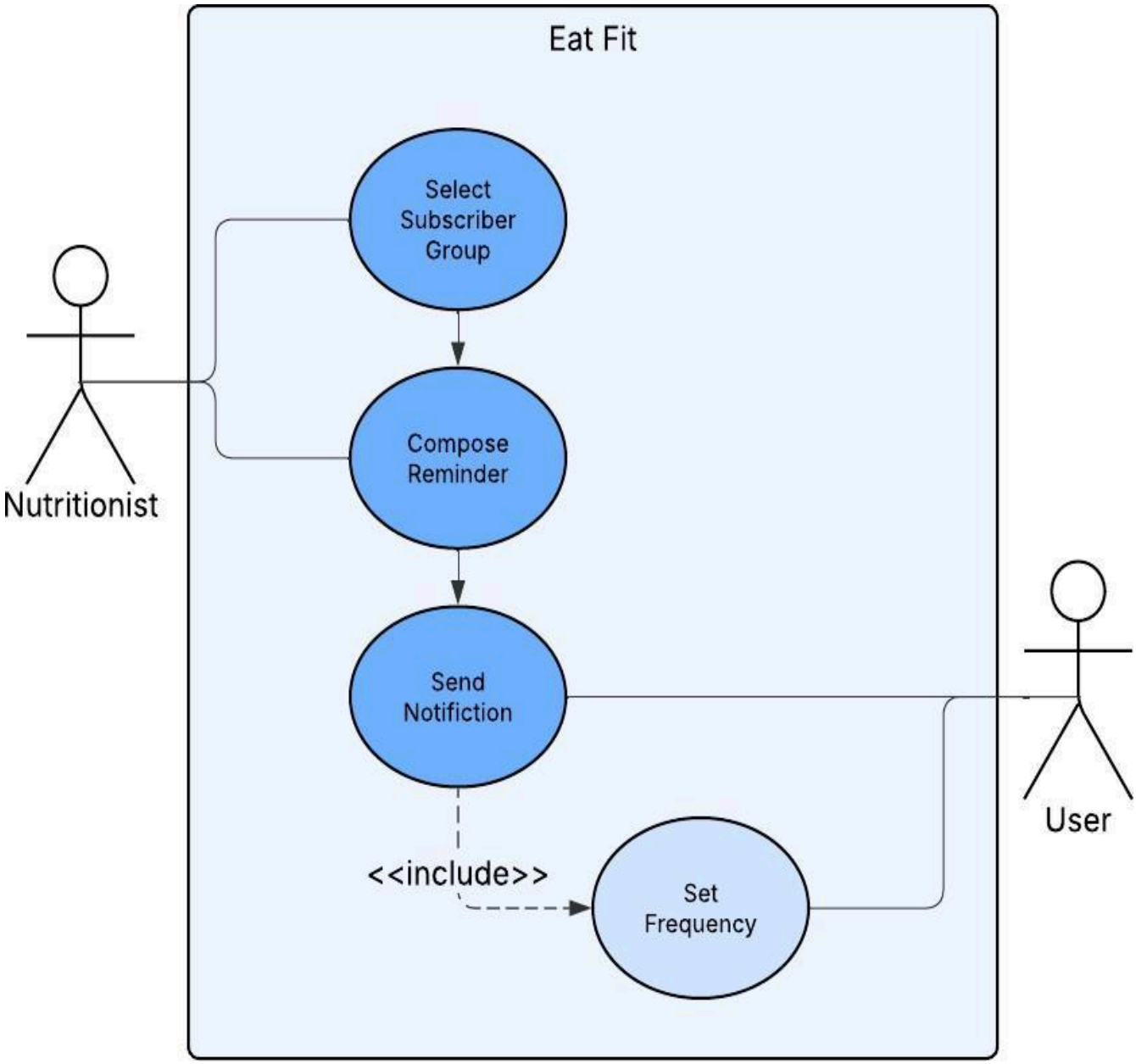
Frequency of Use:

Daily per nutritionist

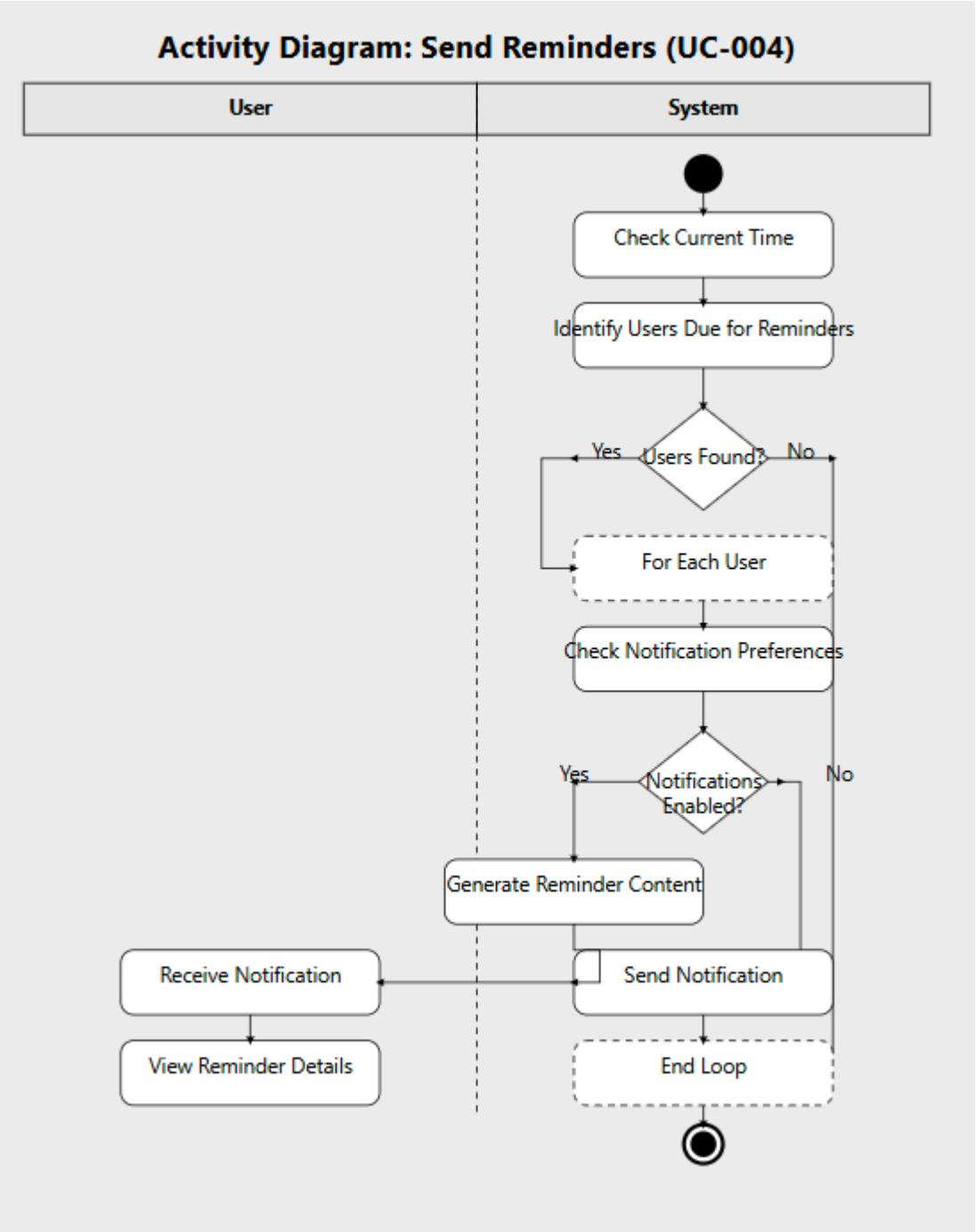
Special Requirements:

- Must support multiple notification channels (app, email, SMS)
- Must comply with communication opt-out regulations
- Must support scheduling in different time zones

Send Remineders (Figure 9) UML Diagram



Send Reminders (Figure10) Activity Diagram



UC-005: Maintain USDA Nutrition Database

ID: UC-005
Priority: High
Primary Actor: Admin
Stakeholders: Admin, All Users
Goal: Keep nutrition database current, accurate, and optimized for application use

Preconditions:

- Admin is authenticated with database management privileges
- Admin has secure access to the USDA API or data sources
- Database backup is available before modifications
- Success Guarantee:
- The nutritional database is updated with the latest information
- Data integrity is maintained
- Updates are logged for audit purposes
- The system uses updated data for calculations

Main Success Scenario:

1. Admin logs into the system administration dashboard
2. Admin navigates to the "Database Management" section
3. Admin selects the "Update Nutrition Database" option
4. The system displays the current database status and the last update timestamp
5. Admin chooses update method (automatic sync or manual upload)
6. If automatic, the admin initiates USDA API synchronization
7. The system connects to the USDA API and requests the latest dataset
8. The system compares the received data with the existing database
9. The system identifies new, modified, and deprecated entries
10. The system displays a summary of proposed changes
11. Admin reviews and approves changes
12. The system applies updates to the production database
13. The system verifies data integrity after the update
14. The system logs all changes with a timestamp and admin ID
15. The system updates the cache and notifies services of the

refresh requirement

16. Alternative Flows:

A1. Manual database update

1. At step 5, the admin chooses the "Manual Upload" option
2. The system displays the file upload interface
3. Admin uploads the USDA database file in a supported format
4. The system validates file structure and data format
5. Continue at step 8 of the main flow

A2. Admin edits individual entries

1. Admin navigates to the "Database Search" section
2. Admin searches for a specific food item
3. The system displays matching database entries
4. Admin selects the entry to edit
5. The system displays an editable entry form
6. Admin makes necessary modifications
7. Admin saves changes
8. The system validates input data
9. The system applies changes to the database
10. The system logs specific changes for the audit trail

Exception Flows:

E1. API connection failure

1. At step 7, if the connection to the USDA API fails
2. The system displays an error message with details
3. The system offers retry options or alternative methods

4. Admin selects the desired action
5. If the retry succeeds, continue at step 8
6. If the retry fails, the system suggests a manual update

E2. Data validation errors

1. At step 9 or during manual editing, if data validation fails
2. The system identifies specific validation issues
3. The system generates a detailed error report
4. Admin reviews issues and corrects problems
5. Admin resubmits for validation
6. If validation passes, continue with the update process
7. The system maintains the database in the current state until a valid update completes

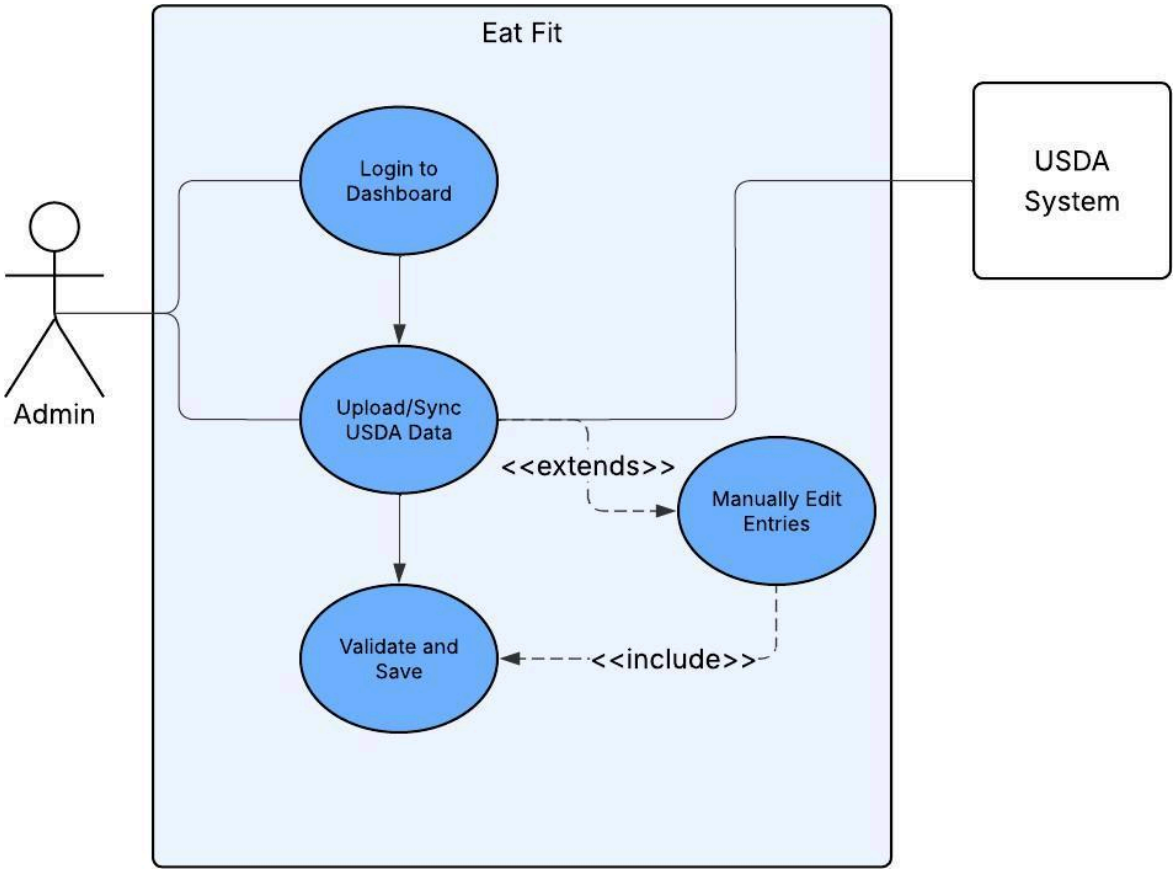
Frequency of Use:

Monthly or as needed when USDA updates occur

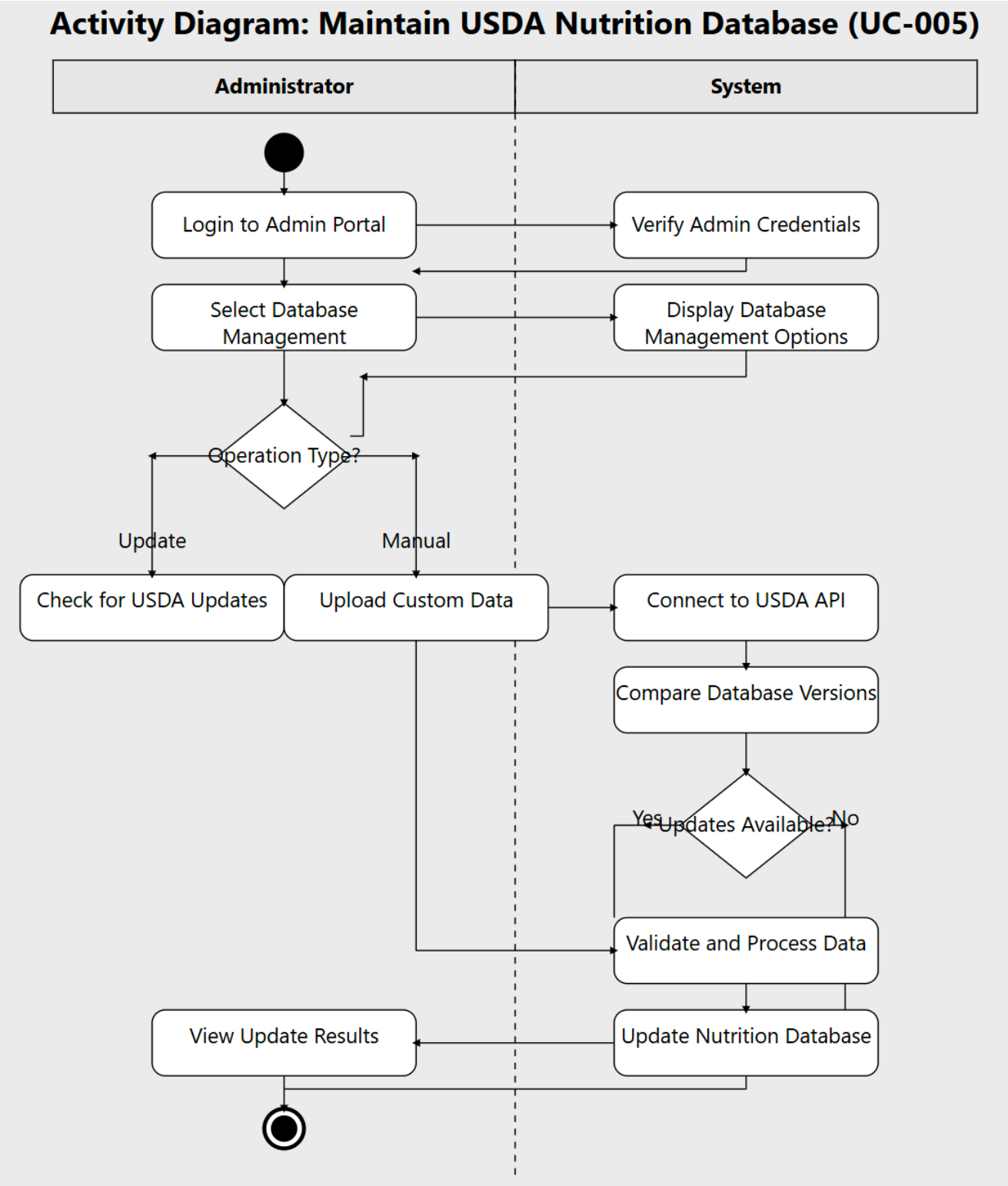
Special Requirements:

- Must maintain version history of database changes
- Must perform updates during low-usage periods
- Must support rollback to the previous version if issues are detected

*Maintain USDA Database (**Figure 11**) UML Diagram*



Maintain USDA Nutrition Database (Figure 12) Activity Diagram



UC-006: User Authentication

ID: UC-006
Priority: High
Primary Actor: User
Stakeholders: User, Admin

Goal: Enable secure user authentication, including registration, login, logout, and password management

Preconditions:

- The system is operational and accessible
- Authentication services are functioning

Success Guarantee:

- The user can securely create, access, or recover their account
- Credentials are stored securely
- Sessions are managed appropriately

UC-006.1: Register Account

Main Success Scenario:

1. The user navigates to the application landing page
2. The user selects the "Create Account" option
3. The system displays the registration form
4. The user enters the required information (email, password, profile details)
5. The user reviews and accepts the terms of service and privacy policy
6. The user submits the registration form
7. The system validates email format and password strength
8. The system checks that the email is not already registered
9. The system creates the user account with an encrypted password
10. The system sends a verification email to the user
11. The user clicks the verification link in the email
12. The system activates the user account
13. The system redirects the user to the login page with a success message

Alternative Flows:

A1. Social Media Registration

1. At step 2, the user selects "Register with [Social Platform]"
2. The system redirects to the social platform authentication page
3. The user logs in to the social platform and authorizes the application
4. Social platform returns an authentication token to the system
5. The system creates an account using social platform profile information
6. Continue at step 9 of the main flow

A2. Skip Email Verification

1. After step 9, the system offers the option to proceed to the application while verification is pending
2. The user can access limited features until the email is verified
3. The system displays a reminder banner about pending verification

Exception Flows:

E1. Email Already Registered

1. At step 8, if the email is already registered
2. The system displays the "Email already registered" message
3. The system offers a "Forgot Password" option
4. The user can choose to recover the password or use a different email

E2. Invalid Password Format

1. At step 7, if the password doesn't meet security requirements
2. The system displays specific password requirements
3. The system highlights which requirements are not met
4. User corrects password

5. Continue at step 6 of the main flow

E3. Verification Link Expired

1. If the user attempts to verify the account after the link has expired (24 hours)
2. The system displays the "Verification link expired" message
3. The system offers a "Resend Verification Email" option
4. The user requests a new verification email
5. The system sends a new verification email
6. Continue at step 11 of the main flow

UC-006.2: Log in to Account

Main Success Scenario:

1. The user navigates to the application login page
2. The user enters their email and password
3. The user submits the login form
4. The system validates credentials
5. The system creates an authenticated session
6. The system redirects the user to the application dashboard
7. The system records login timestamps for audit purposes

Alternative Flows:

A1. Remember Me Option

1. At step 2, the user checks the "Remember Me" option
2. Continue with the main flow
3. After step 5, the system sets a persistent cookie with an encrypted token

4. On subsequent visits, the user is automatically authenticated

A2. Social Media Login

1. At step 2, the user selects "Login with [Social Platform]"
2. The system redirects to social platform authentication
3. The user authenticates with a social platform
4. Social platform returns an authentication token
5. The system validates the token and identifies the user account
6. Continue at step 5 of the main flow

A3. Two-Factor Authentication

1. After step 4, if 2FA is enabled for the user
2. The system sends a verification code to the user's mobile device
3. The system displays the 2FA code entry screen
4. The user enters the received code
5. The system validates code
6. Continue at step 5 of the main flow

Exception Flows:

E1. Invalid Credentials

1. At step 4, if the credentials are invalid
2. The system increments the failed login counter for the account/IP
3. The system displays the "Invalid email or password" message
4. If failed attempts exceed the threshold (5 attempts)
5. The system implements a temporary lockout (15 minutes)
6. The system offers a "Forgot Password" option

E2. Account Locked

1. At step 4, if an account is locked due to suspicious activity

2. The system displays the "Account locked" message
3. The system provides instructions for contacting support
4. The system offers a password reset option

E3. Invalid 2FA Code

1. At step 5 of A3, if 2FA code is invalid
2. The system allows three retry attempts
3. The system displays remaining attempts
4. If all attempts fail, user is returned to the login screen
5. The system sends an account alert email to the user

UC-006.3: Log Out from Account

Main Success Scenario:

1. The user selects the "Logout" option from the application menu
2. The system terminates the user session
3. The system clears session cookies
4. The system redirects the user to an application landing page
5. The system displays a logout confirmation message

Exception Flows:

E1. Session Already Expired

1. If the session has already expired when the user selects logout
2. The system displays a "Session expired" message
3. The system redirects to the login page

UC-006.4: Reset Password

Main Success Scenario:

1. The user selects the "Forgot Password" option on the login page
2. The system displays the password reset request form
3. The user enters a registered email address
4. The user submits a reset request
5. The system verifies that the email exists in the database
6. The system generates a secure reset token with a 1-hour expiration
7. The system sends a password reset email with a secure link
8. The user clicks the reset link within the email
9. The system verifies token validity and expiration
10. The system displays the new password form
11. The user enters and confirms the new password
12. The system validates password strength
13. The system updates the stored password with the new encrypted password.
14. The system invalidates all existing sessions for the user
15. The system displays a success message and redirects to the login page

Alternative Flows:

A1. Reset via SMS

1. At step 2, the user selects the "Reset via SMS" option
2. The user enters the phone number associated with the account
3. The system sends a verification code via SMS
4. The user enters the received code

5. The system validates code
6. Continue at step 10 of the main flow

Exception Flows:

E1. Email Not Found

1. At step 5, if the email is not found
2. The system still displays the "Reset instructions sent" message
3. No email is sent (security by obscurity)

E2. Reset Token Expired

1. At step 9, if the token has expired
2. The system displays the "Reset link expired" message
3. The system offers the option to request a new reset link
4. The user requests a new link
5. Return to step 6 of the main flow

E3. Invalid New Password

1. At step 12, if the password doesn't meet requirements
2. The system displays specific password requirements
3. The system highlights which requirements are not met
4. The user corrects the password
5. Continue at step 11 of the main flow

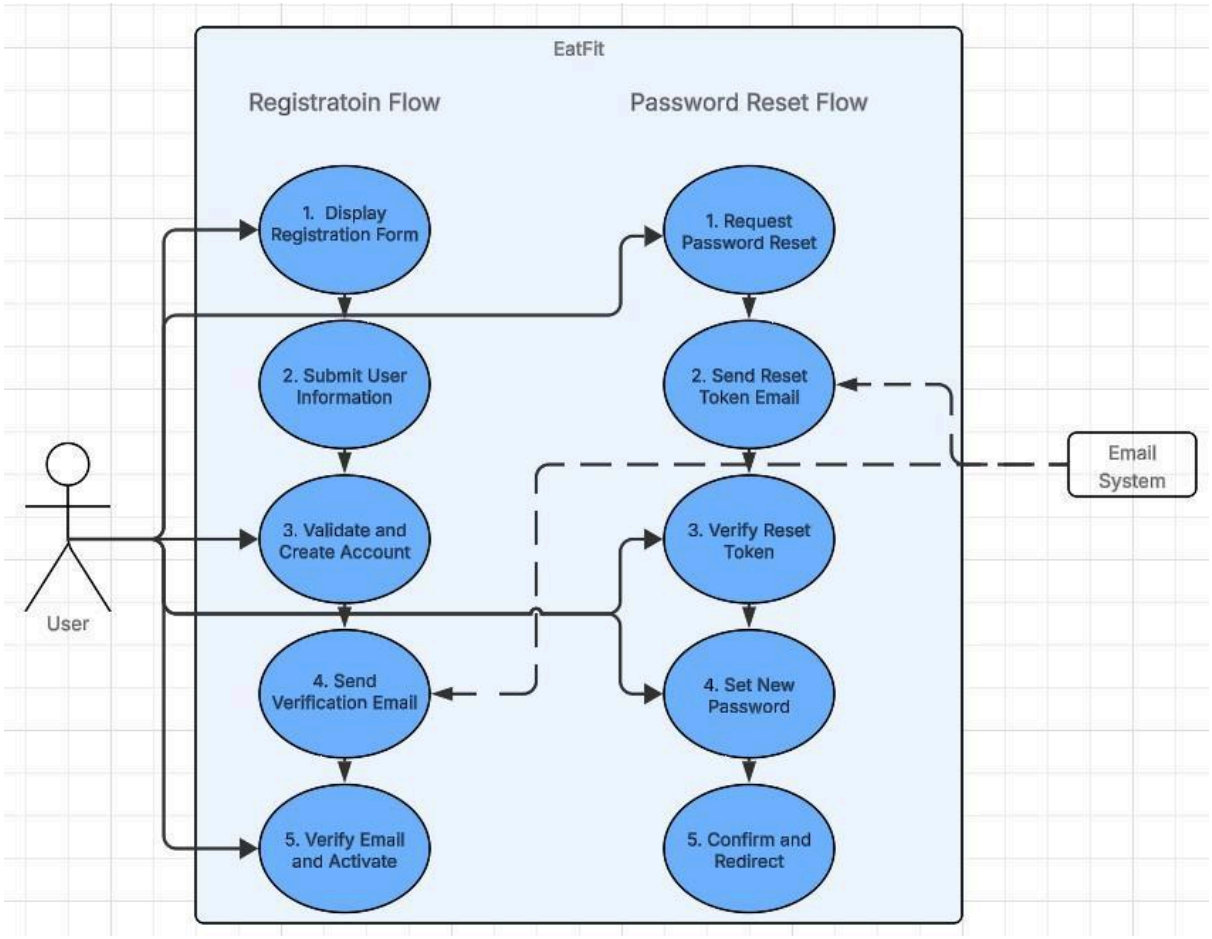
Frequency of Use:

Daily per user

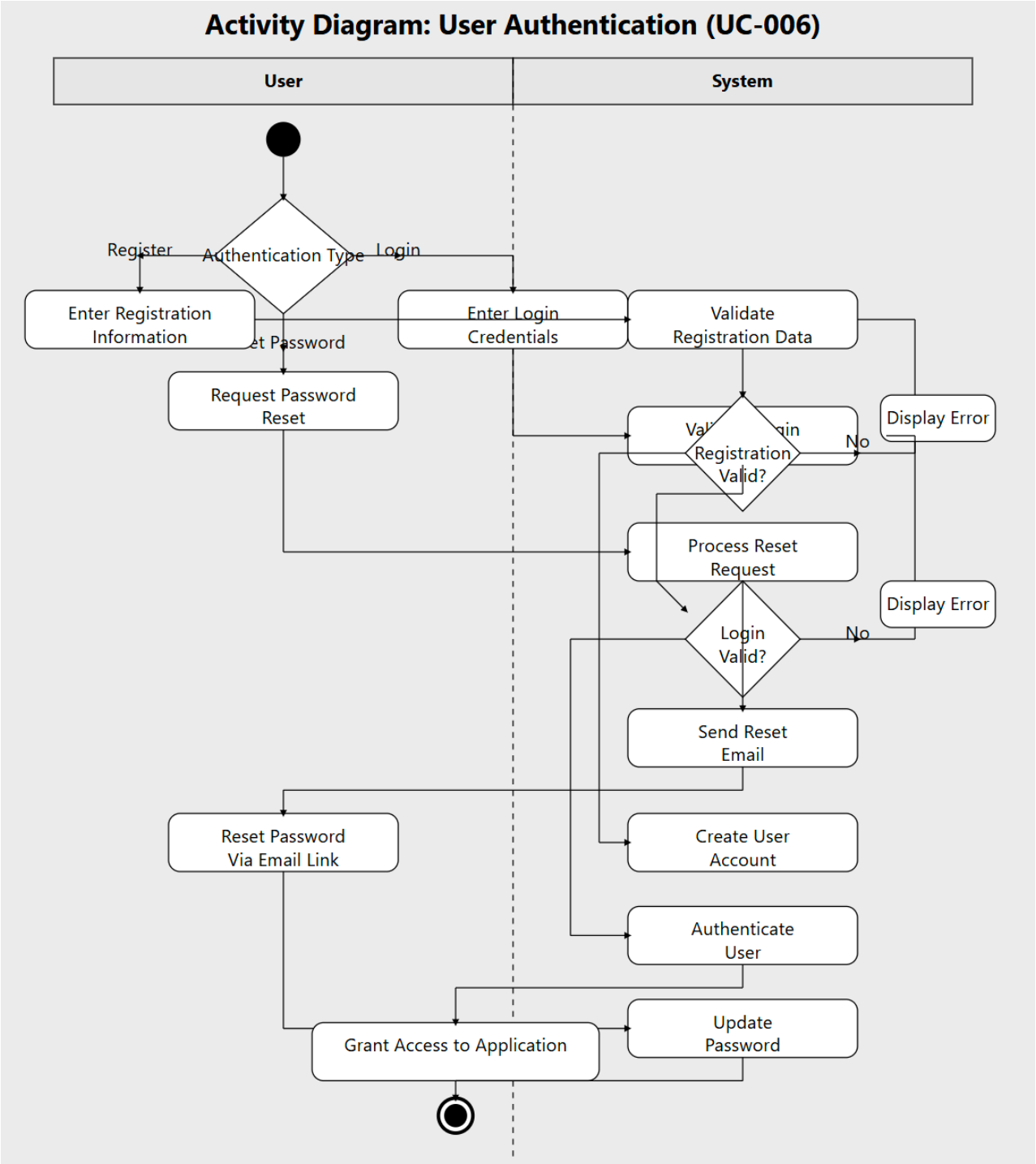
Special Requirements:

- Password requirements: minimum eight characters, mix of uppercase, lowercase, numbers, and special characters
- Reset tokens must be single-use and expire after 1 hour
- Failed login attempts must be rate-limited to prevent brute force attacks
- All authentication activities must be logged for security audit purposes

User Authentication (Figure 13) UML Diagram



User Authentication (Figure 14) Activity Diagram



UC-007: Update Profile

ID: UC-007
Priority: Medium
Primary Actor: User

Stakeholders: User, Nutritionist, Admin
Goal: Allow users to manage their profile information, preferences, and notification settings

Preconditions:

- The user is authenticated in the system
- The user has an active account

Success Guarantee:

- Profile information is updated accurately
- Preferences and settings are applied immediately
- All changes are stored securely

UC-007.1: Edit Account Information

Main Success Scenario:

1. The user navigates to the "My Profile" or "Account Settings" section
2. The system displays current profile information
3. The user selects the "Edit Profile" option
4. The system displays an editable profile form with current information
5. The user modifies desired fields (name, contact information, etc.)
6. The user submits changes
7. The system validates input data
8. The system updates the user profile in the database
9. The system displays a confirmation message
10. The system logs the profile update for audit purposes

Alternative Flows:

A1. Upload Profile Picture

1. At step 5, the user selects the "Upload Profile Picture" option
2. The system displays the file selection dialog
3. The user selects an image file
4. The system validates the file format and size
5. System processes and resizes the image
6. The system displays an image preview
7. User confirms image selection
8. Continue at step 6 of the main flow

A2. Change Email Address

1. At step 5, the user modifies the email address
2. Continue with the main flow
3. After step 8, the system sends a verification email to the new address
4. The user must verify the new email before a change is finalized
5. The old email continues to function until the new email is verified

Exception Flows:

E1. Invalid Input Data

1. At step 7, if any input data is invalid
2. The system highlights invalid fields with error messages
3. The user corrects invalid data
4. Continue at step 6 of the main flow

E2. Email Already in Use

1. At step 7, if the new email is already registered to another account
2. The system displays the "Email already in use" error message
3. The user can choose a different email or cancel the change
4. Continue at step 5 or exit the flow

E3. File Type/Size Error

1. At step 4 of A1, if the file is an invalid type or exceeds the size limit
2. The system displays a specific error message
3. The user can select a different file or cancel the upload
4. Continue at step 2 of A1 or return to the main flow

UC-007.2: Manage Communication Preferences

Main Success Scenario:

1. The user navigates to the "Notification Settings" section
2. The system displays current notification preferences
3. The user modifies preferences:
    - Email notification options (meal reminders, nutritionist messages, etc.)
    - Mobile push notification settings
    - SMS notification options
    - Newsletter and promotional communication opt-in/out
4. The user saves changes
5. The system updates preference settings in the database
6. The system applies new settings immediately

7. The system displays a confirmation message

Alternative Flows:

A1. Default Preferences Templates

1. At step 3, the user selects the "Use Preset" option
2. The system displays preset preference templates (Minimal, Standard, All Communications)
3. The user selects the desired template
4. The system populates all settings according to the template
5. The user can make additional adjustments if needed
6. Continue at step 4 of the main flow

Exception Flows:

E1. Legal Requirement Notifications

1. At step 3, if the user attempts to turn off legally required notifications
2. The system displays a message indicating that certain notifications cannot be disabled
3. The system maintains the required notification settings
4. The user continues with other preference modifications
5. Continue at step 4 of the main flow

UC-007.3: Manage Privacy Settings

Main Success Scenario:

1. The user navigates to the "Privacy Settings" section
2. The system displays current privacy settings
3. The user modifies privacy options:

    - Data sharing preferences
    - Profile visibility
    - Progress sharing with the nutritionist
    - Third-party integration permissions
4. The user saves changes
5. The system updates privacy settings in the database
6. The system applies new settings immediately
7. The system displays a confirmation message
8. The system logs privacy setting changes for compliance purposes

Alternative Flows:

A1. Data Export Request

1. At step 3, the user selects the "Export My Data" option
2. The system displays data export options (format, content selection)
3. The user selects the desired options
4. The system initiates the data export process
5. The system notifies the user when the export is ready for download
6. The user downloads a data package
7. Return to step 3 of the main flow

A2. Account Deletion Request

1. At step 3, the user selects the "Delete My Account" option
2. The system displays a confirmation dialog with implications
3. The user confirms the deletion request
4. The system requests password verification

5. The user enters a password
6. The system validates the password
7. The system initiates the account deletion process
8. The system displays confirmation and logs out the user
9. Exit flow

Exception Flows:

E1. Incomplete Information

1. At step 3, if the required privacy settings are incomplete
2. The system highlights missing information
3. The system suggests recommended settings
4. User completes the required fields
5. Continue at step 4 of the main flow

E2. Invalid Password for Deletion

1. At step 6 of A2, if the password is invalid
2. The system displays an error message
3. The system allows a retry (maximum three attempts)
4. If all attempts fail, cancel the deletion request
5. Return to step 3 of the main flow

Frequency of Use:

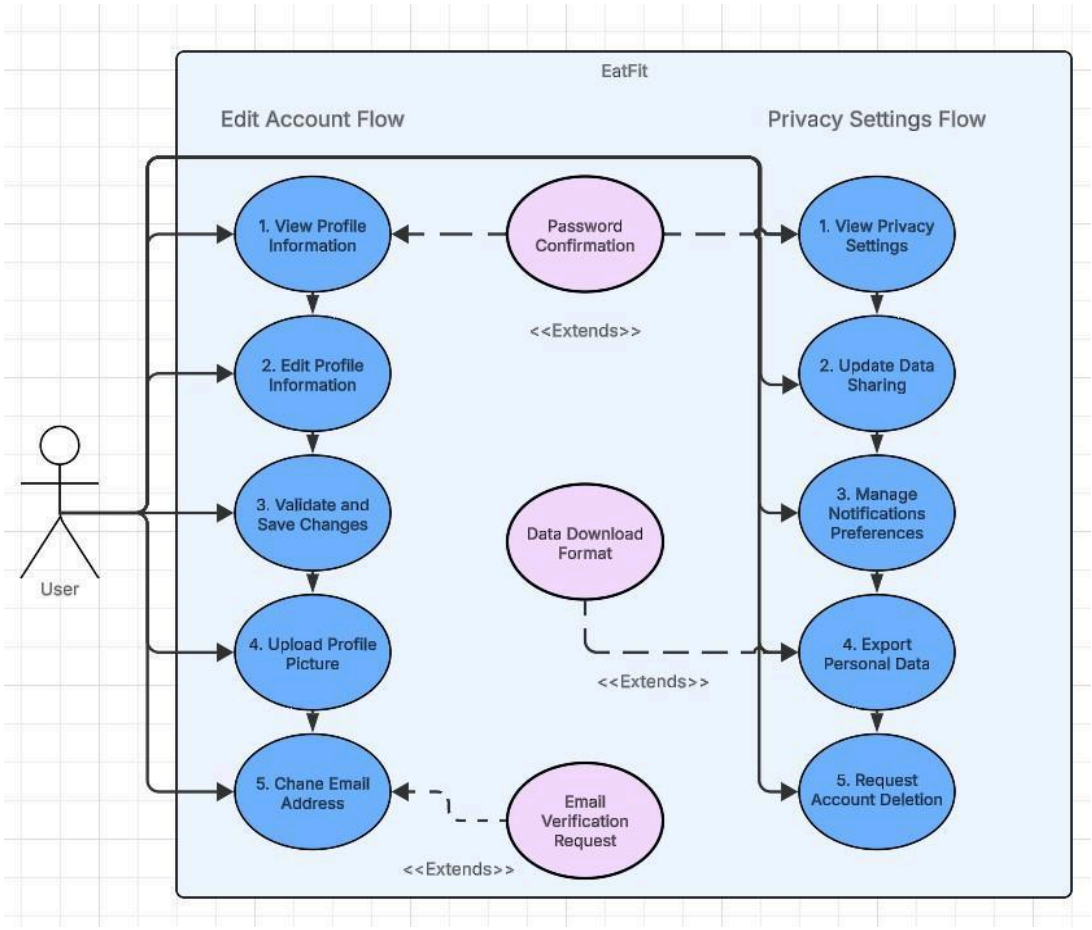
Monthly per user

Special Requirements:

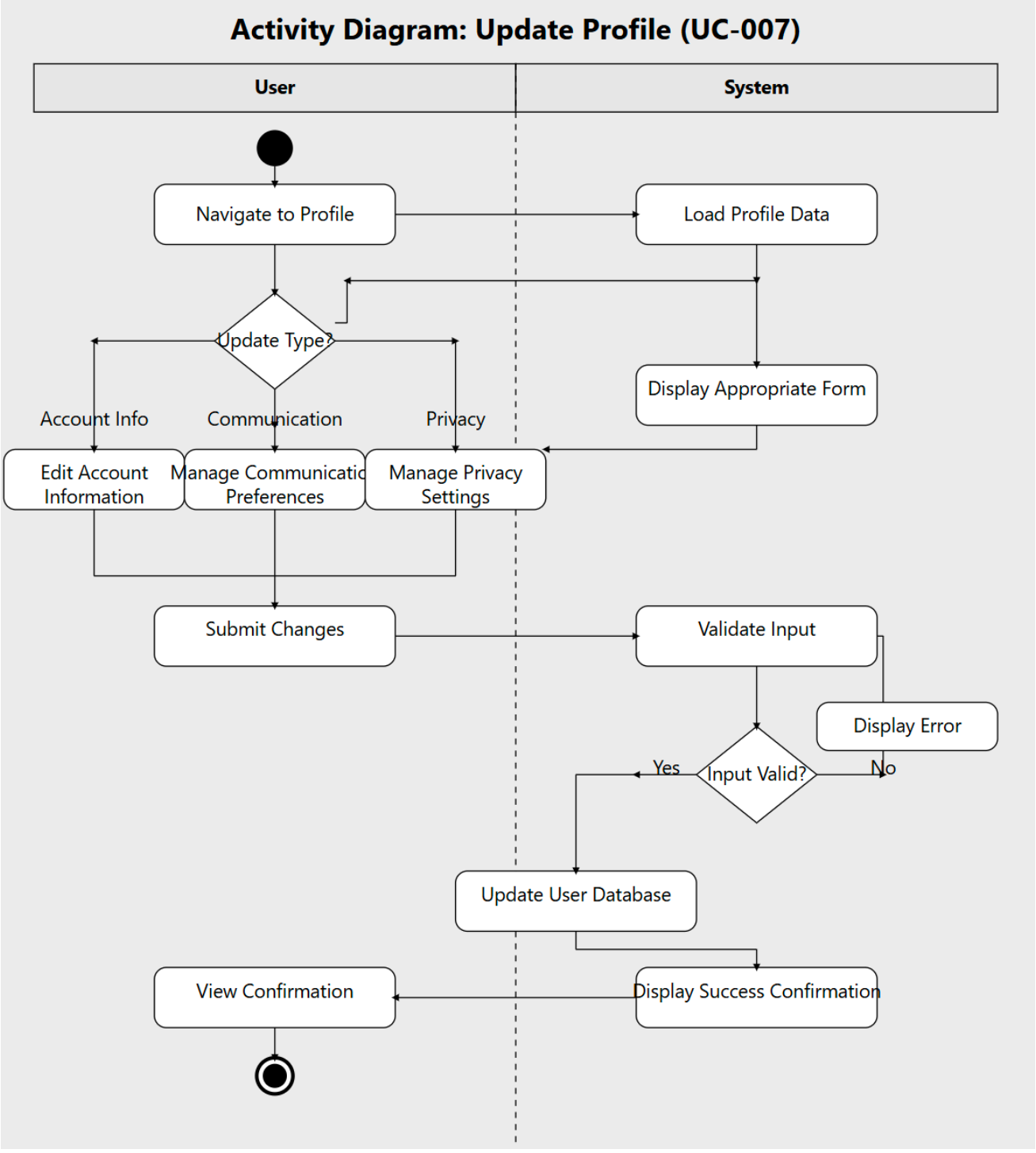
- Changes to communication preferences must be processed within 24 hours

- Privacy setting changes must be logged for GDPR compliance
- Data export must include all user-created content and personal information
- Account deletion must comply with data retention regulations (GDPR, CCPA)

*Update Profile (***Figure 15***) UML Diagram*



*Update Profile (**Figure 16**) Activity Daigram*



Account Management Relationships (Table 22)

Source Use Case ID	Relationship Type	Target Use Case ID	Description
UC-006.1	prerequisite for	UC-001 through UC-007	Registration must be completed before any other system function
UC-006.2	prerequisite for	UC-001 through UC-007	Login must be completed before any other system function
UC-006.3	terminates	UC-001 through UC-007	Logout ends all other active use cases
UC-006.4	alternative to	UC-006.2	Password reset provides an alternative way to regain account access
UC-007.1	extends	UC-006.1	Account information is initially set during registration
UC-007.2	affects	UC-004	Communication preferences affect how reminders are delivered
UC-007.3	affects	UC-002	Privacy settings affect what data is included in nutrition reports

21.User Interface Design

While detailed UI mockups are being developed as a separate deliverable, this section outlines the key UI principles and screens that will be included in the final application design.

UI Design Principles

1. Simplicity First: The interface prioritizes quick meal logging with minimal friction
2. Visual Data Representation: Nutritional data is presented using intuitive charts and color-coding
3. Progressive Disclosure: Complex features are available but not prominent in primary workflows
4. Accessibility: All screens follow WCAG 2.1 AA standards with clear contrast and readable text
5. Responsive Design: Interfaces adapt seamlessly between desktop, tablet, and mobile views

Key Screens

1. Dashboard: Personalized summary of recent meals, progress toward goals, and quick-access actions
2. Meal Logging: Streamlined input form with search, barcode scanning, and recent items
3. Nutrition Reports: Filterable, visual breakdowns of nutritional intake across timeframes
4. Goals Setting: Guided process for establishing personal nutrition targets
5. Recipe Builder: Interface for creating and storing reusable meal templates
6. Settings: Profile management, notification preferences, and privacy controls
7. Nutritionist Dashboard: Client management interface with progress monitoring and communication tools

8. Admin Console: System maintenance and user management interface

The UI design will undergo usability testing with representative users from each persona category to ensure it meets their specific needs and workflow expectations.

UI mockups will be stored in the project's Azure DevOps repository and referenced in the final documentation package.

22.Supporting Requirements

Security Requirements (Table 23)

ID	Requirement	Verification Method	Priority
SEC-001	The system must encrypt all user data at rest using AES-256 encryption	Security audit, Code review	High
SEC-002	The system must encrypt all data in transit using TLS 1.3 or higher	Security audit, Penetration testing	High
SEC-003	The system must implement role-based access control for all features	Functional testing, Security audit	High
SEC-004	User passwords must be stored using bcrypt with an appropriate salt	Code review, Security audit	High
SEC-005	The system must support two-factor authentication for all user roles	Functional testing	Medium
SEC-006	The system must automatically log out inactive sessions after 30 minutes	Functional testing	Medium
SEC-007	The system must comply with HIPAA security requirements for protected health information	Compliance audit	High
SEC-008	The system must comply with GDPR requirements for EU users	Compliance audit	High
SEC-009	The system must implement rate limiting to prevent brute force attacks	Security testing	Medium

SEC-010	API access must be controlled through tokens with appropriate expiration	Security audit	High

Availability Requirements (Table 24)

ID	Requirement	Verification Method	Priority
AVL-001	The system must achieve 99.5% uptime for backend services	Monitoring logs, SLA reporting	High
AVL-002	Planned maintenance windows must be scheduled during off-peak hours (2 AM - 4 AM local time)	Operational review	Medium
AVL-003	The system must be accessible across multiple device types (web, mobile, tablet)	Cross-platform testing	High
AVL-004	The mobile application must support offline functionality with data synchronization; timestamp-based conflict resolution with clear user notification when conflicts occur	Functional testing, Conflict simulation	Medium
AVL-005	The system must implement load balancing to handle traffic spikes	Performance testing	Medium
AVL-006	Recovery from unplanned outages must be completed within 15 minutes	Disaster recovery testing	High
AVL-007	Users must be notified of scheduled maintenance at least 48 hours in advance	Process audit	Low
AVL-008	Database replication must be implemented with automated failover	System architecture review	High

Usability Requirements (Table 25)

ID	Requirement	Verification Method	Priority
USB-001	Users must be able to log meals in three steps or fewer	Usability testing	High
USB-002	The interface must meet WCAG 2.1 AA accessibility standards	Accessibility audit	High
USB-003	The system must provide clear error messages with suggested corrections	Usability testing	Medium
USB-004	The mobile interface must be fully functional on screens from 4" to 7"	Device testing	High
USB-005	All critical functions must be accessible within two taps/clicks from the main screen	Navigation testing	Medium
USB-006	The system must support voice input for meal logging	Functional testing	Low
USB-007	Text must use a minimum 12pt font with adjustable size options	Visual inspection	Medium
USB-008	Color schemes must accommodate colorblind users (deuteranopia, protanopia, tritanopia)	Accessibility testing	Medium
USB-009	The system must provide interactive tutorials for first-time users	Usability testing	Low
USB-010	All data visualizations must include alternative text descriptions	Accessibility testing	Medium

Maintainability Requirements (Table 26)

ID	Requirement	Verification Method	Priority
MNT-001	Code must follow the MVC architecture pattern	Code review	High
MNT-002	The backend must implement RESTful APIs with documented endpoints	API documentation review	High
MNT-003	The code must maintain a minimum of 80% unit test coverage	Test coverage metrics	Medium
MNT-004	The system must use dependency injection to facilitate component testing	Code review	Medium
MNT-005	Database schema changes must be managed through migration scripts	Process review	High
MNT-006	The system must implement a modular design allowing component updates without full system changes	Architecture review	Medium
MNT-007	Front-end and back-end components must be deployable independently	Deployment testing	Medium
MNT-008	All APIs must be versioned to support backward compatibility	API testing	High
MNT-009	System configuration must be environment-based, not hardcoded	Code review	Medium
MNT-010	An automated CI/CD pipeline must be implemented for build and deployment	Process audit	Medium

Performance Requirements (Table 27)

ID	Requirement	Verification Method	Priority
PRF-001	Response time for nutrition report generation must not exceed 2 seconds	Performance testing	High
PRF-002	Database queries must be optimized to handle 1,000 concurrent users	Load testing	High
PRF-003	Page load time must not exceed 1.5 seconds for primary functions	Performance testing	Medium
PRF-004	The system must support a minimum of 100,000 registered users	Capacity planning, Database design review	Medium

ID	Requirement	Verification Method	Priority
PRF-005	API response time must not exceed 300ms under normal load	API performance testing	High
PRF-006	Mobile app startup time must not exceed 3 seconds on standard devices	Performance testing	Medium
PRF-007	The system must handle 500,000 meal logs per day	Load testing	Medium
PRF-008	Database read operations must complete in under 50ms	Database performance testing	High
PRF-009	The system must implement caching for frequently accessed nutritional data	Architecture review	Medium
PRF-010	Background processing must be used for report generation exceeding 5 seconds.	Code review, Performance testing	Medium

*Legal & Compliance (***Table 28)**

ID	Requirement	Verification Method	Priority
LGL-001	User data must be stored on servers located in the United States	Infrastructure audit	High
LGL-002	Data retention policy must be user-configurable with a default 24-month retention	Functional testing	Medium
LGL-003	The system must allow users to download all their personal data in a machine-readable format	GDPR compliance testing	High
LGL-004	The system must support complete account deletion with verification	GDPR compliance testing	High
LGL-005	The privacy policy must be explicitly accepted during registration	UI inspection	High
LGL-006	Nutritional advice must include a disclaimer about not replacing medical advice	Content review	High
LGL-007	The system must maintain audit logs for all access to personal health information	HIPAA compliance audit	High
LGL-008	The system must obtain explicit consent before sharing user data with third parties.	Process audit	High

ID	Requirement	Verification Method	Priority
LGL-009	The system must implement appropriate age verification for users under 16	Process audit	Medium
LGL-010	Cookies and tracking must comply with applicable laws (GDPR, CCPA)	Compliance audit	High

*Global Error Handling Strategy (***Table 29)**

ID	Requirement	Verification Method	Priority
ERR-001	The system must implement centralized error logging for all components	Log analysis	High
ERR-002	All user-facing errors must provide clear recovery instructions	Usability testing	High
ERR-003	API errors must return standardized error codes and messages	API testing	Medium
ERR-004	Critical system errors must trigger administrator notifications	System testing	High
ERR-005	The system must gracefully degrade functionality when dependencies are unavailable	Fault injection testing	Medium

23.Additional Requirements

User Roles & Permissions Matrix (Table 30)

Role	Action	Permission	Justification
User	Log meals	✅ Allowed	Core functionality for all users
User	View own nutrition data	✅ Allowed	Core functionality for all users
User	View other users' data	❌ Denied	Privacy protection

Role	Action	Permission	Justification
User	Edit own profile	✅ Allowed	User data management
User	Create custom recipes	✅ Allowed	Enhanced meal logging experience
Nutritionist	View the assigned clients' data	✅ Allowed	Required for providing guidance
Nutritionist	Edit client diet goals	✅ Allowed	Professional service function
Nutritionist	Access unassigned client data	❌ Denied	Privacy protection
Nutritionist	Send reminders to clients	✅Allowed	Client engagement function
Nutritionist	Access system settings	❌ Denied	Administrative separation
Admin	Manage user accounts	✅ Allowed	Administrative function
Admin	Edit the USDA database	✅ Allowed	Database maintenance function
Admin	View user health data	❌ Denied	Privacy protection unless specifically authorized
Admin	Configure system settings	✅ Allowed	System maintenance function
Admin	Run usage reports	✅ Allowed	Analytics function

Assumptions & Dependencies (Table 31)

ID	Assumption/Dependency	Impact	Validation Method
ASM-001	USDA API will remain stable and accessible	Critical for nutritional data accuracy	Regular API availability monitoring
ASM-002	Users will have an internet connection for real-time features	Medium - offline mode mitigates	Network capability testing

ID	Assumption/Dependency	Impact	Validation Method
ASM-003	Data privacy regulations will remain consistent	High - compliance requirement	Legal review
ASM-004	Mobile devices will support a minimum of iOS 12 and Android 8	Medium - affects user base	Market analysis, device testing
ASM-005	Smart device input will be implemented in version 2	Low - planned future feature	Feature roadmap review
ASM-006	Users will provide accurate personal information	Medium - affects calculation accuracy	Data validation, education
ASM-007	Cloud service providers will maintain 99.9% uptime	High - affects system availability	SLA review, monitoring
ASM-008	The user base will not exceed 500,000 in the first year	Medium - affects scaling plans	Market analysis, usage monitoring
DEP-001	Dependency on the USDA nutritional database	Critical - core data source	Alternative data source backup
DEP-002	Dependency on cloud hosting services	High - infrastructure requirement	Multi-provider strategy
DEP-003	Dependency on app stores for distribution	Medium - affects user acquisition	Web alternative availability
DEP-004	Dependency on third-party authentication providers	Medium - affects the login process	Multiple provider support
DEP-005	Dependency on payment processors	Low - for premium features	Multiple processor support

Data Requirements (Table 32)

ID	Requirement	Verification Method	Priority
DAT-001	Each meal entry must include the ingredient's name, quantity, and unit of measurement.	Data validation testing	High
DAT-002	All ingredients must be validated against USDA entries using unique identifiers	Data integrity testing	High
DAT-003	User meal data must be retained for a minimum of 12 months unless deleted by the user	Data Retention Audit	High
DAT-004	Nutritional calculations must include macronutrients and a minimum of 20 micronutrients	Functional testing	High
DAT-005	User profiles must include age, gender, height, weight, and activity level for BMI and calorie calculations	Data model validation	High
DAT-006	The system must support custom user-defined foods and recipes	Functional testing	Medium
DAT-007	Meal entries must be time stamped and categorized (breakfast, lunch, dinner, snack)	Data validation	Medium
DAT-008	Data schema must support international units of measurement with automatic conversion	Functional testing	Medium
DAT-009	The system must maintain historical weight and measurement data for trend analysis	Data model validation	Medium
DAT-010	Nutritional data precision must be maintained to 0.1g for macronutrients	Calculation accuracy testing	High

Interfaces & Integration Requirements (Table 33)

ID	Requirement	Verification Method	Priority
INT-001	The system must integrate with the USDA API for nutritional data	Integration testing	High
INT-002	The system must provide a REST API for third-party integrations	API testing	Medium
INT-003	The mobile app must support barcode scanning for packaged foods	Functional testing	Medium
INT-004	The system must implement OAuth 2.0 for third-party authentication	Security testing	Medium
INT-005	Future integration with the MyFitnessPal API is planned for version 2	Architecture review	Low
INT-006	Future integration with the Fitbit API is planned for version 2	Architecture review	Low
INT-007	The system must integrate with Azure Notification Hubs for cross-platform notifications	Integration testing	High
INT-008	The system must provide export functionality in multiple formats (CSV, JSON, PDF)	Functional testing	Medium
INT-009	The system must expose webhook endpoints for event notifications	API testing	Low
INT-010	The system must implement a GraphQL API for optimized data retrieval	Performance testing	Low

Backup & Recovery Requirements (Table 34)

ID	Requirement	Verification Method	Priority
BCK-001	Daily backups of user and nutritional data must be performed	Process audit	High
BCK-002	The system must support recovery from backup within 4 hours of failure	Disaster recovery testing	High
BCK-003	Users must be notified of service outages longer than 15 minutes	Process audit	Medium
BCK-004	Backups must be encrypted using AES-256 encryption	Security audit	High
BCK-005	The system must maintain at least 7 days of backup history with point-in-time recovery capability	Process audit	Medium
BCK-006	Database transactions must be logged to support point-in-time recovery	Technical audit	High
BCK-007	Backup restoration testing must be performed monthly	Process verification	Medium
BCK-008	Geo-redundant backup storage must be implemented	Infrastructure audit	Medium
BCK-009	The system must support automated failover to the secondary region	Disaster recovery testing	Medium
BCK-010	User-deleted data must be purged from backups within 30 days	Compliance audit	High
	and stored securely		

Logging & Audit Trail Requirements (Table 35)

ID	Requirement	Verification Method	Priority
LOG-001	All user logins, meal entries, and data exports must be time stamped and logged	Log review	High
LOG-002	Administrative changes to the database must be recorded with the user ID, timestamp, and action	Log review, Audit	High
LOG-003	Logs must be retained for at least 12 months	Compliance audit	High
LOG-004	Logs must be tamper-evident with cryptographic verification	Security audit	Medium
LOG-005	The system must log all access to protected health information	HIPAA compliance audit	High
LOG-006	Failed login attempts must be logged with IP address information	Security audit	High
LOG-007	The system must generate alerts for suspicious access patterns	Security monitoring test	Medium
LOG-008	Logs must be searchable by user ID, action type, and date range	Functional testing	Medium
LOG-009	The system must generate usage analytics reports from log data	Report verification	Low
LOG-010	Logs must be backed up daily and stored separately from application data	Process audit	Medium

24.External Interface Specifications

This section provides detailed specifications for external interfaces used by the EatFit application.

USDA FoodData Central API

- Purpose: Primary source of nutritional information for ingredients
- Interface Type: REST API

- Authentication: API Key
- Base URL: https://api.nal.usda.gov/fdc/v1/
- Primary Endpoints:
    - `/foods/search`: Search for foods by keyword
    - `/food/{fdcId}`: Get detailed information for a specific food
    - `/foods`: Retrieve multiple food items in a single request
- Request Limits: 3600 requests per hour
- Error Handling: Implement caching and exponential backoff strategy
- Response Format: JSON
- Sample Request:

```
Unset

GET
/foods/search?query=apple&dataType=Foundation,SR%20Leg
acy&pageSize=10
```

- Fallback Strategy: Local database of common foods with periodic updates

Mobile Device Camera API

- Purpose: Barcode scanning for packaged foods
- Interface Type: Native device API
- Platforms: iOS Camera API, Android Camera2 API
- Implementation: React Native Camera module
- Permissions Required: Camera access
- Supported Formats: UPC-A, UPC-E, EAN-13, EAN-8
- Error Handling: Manual entry fallback option
- Accessibility: Voice prompt alternative for visually impaired users

Push Notification Services

- Purpose: Deliver reminders and alerts to users
- Interface Types:
    - iOS: Apple Push Notification Service (APNs)
    - Android: Firebase Cloud Messaging (FCM)
    - Web: Web Push API
- Message Types:
    - Meal reminders
    - Goal achievement notifications
    - System alerts
- Delivery Priority: Normal (default), High (for time-sensitive reminders)
- Payload Limitations: 4KB maximum
- User Controls: Configurable notification categories and quiet hours

Authentication Providers

- Purpose: User authentication and identity management
- Primary Provider: Azure AD B2C
- Secondary Providers:
    - Google Sign-In
    - Apple Sign-In
- Authentication Flow: OAuth 2.0 Authorization Code Flow with PKCE
- Token Storage: Secure storage with encryption
- Session Management: Refresh tokens with 30-day expiration
- Multi-factor Authentication: Optional SMS or authenticator app verification

Future Third-Party Integrations (Version 2)

- Fitness Platforms:
- Apple Health: HealthKit API
    - Google Fit: REST API
    - Fitbit: REST API
- Smart Home Devices:
    - Amazon Alexa: Alexa Skills Kit
    - Google Assistant: Actions on Google
- Data Exchange: Standardized nutrition and activity data format
- Privacy Controls: Granular permission settings for shared data

25.Business Rules

Business Rules (Table 36)

ID	Rule	Description	Verification Method	Priority
BUS-001	Age Restriction	Users must be at least 13 years old to create an account	Registration validation	High
BUS-002	BMI Calculation	BMI = Weight(kg) / Height²(m) with classification thresholds per WHO standards	Calculation verification	High
BUS-003	Calorie Recommendation	The base metabolic rate was calculated using the Mifflin-St. Jeor equation with an activity factor multiplier	Calculation verification	High
BUS-004	Nutrient RDA	Daily recommended values based on age, gender, and activity level per USDA guidelines	Data validation	High
BUS-005	Meal Categorization	Meals categorized by time: Breakfast (4-10 am), Lunch (10 am-3 pm), Dinner (3 pm-10 pm), Snack (any time)	Functional testing	Medium

BUS-006	Weight Change Safety	The system must flag and warn if the weight change exceeds 2kg/week	Alert testing	High
BUS-007	Nutritionist Qualification	Nutritionist accounts require verification of professional credentials before activation	Process audit	High
BUS-008	Client Limit	Each nutritionist can manage a maximum of 50 active clients	Validation testing	Medium
BUS-009	Custom Food Creation	Custom foods require minimum data: name, serving size, calories, protein, carbs, and fat	Data validation	Medium
BUS-010	Goal Warnings	The system must warn users if goals deviate more than 30% from recommended values.	Alert testing	Medium

26.Deployment Requirements

*Deployment Requirements (***Table 37)**

ID	Requirement	Description	Priority
DEP-001	Environment Separation	The system must maintain separate development, testing, staging, and production environments	High
DEP-002	Zero-Downtime Deployment	System updates must be deployable without service interruption	Medium
DEP-003	Rollback Capability	All deployments must support immediate rollback capability	High

ID	Requirement	Description	Priority
DEP-004	Database Migration	Database schema changes must be managed through versioned migration scripts	High
DEP-005	Feature Flags	The system must support feature flags for controlled feature rollout	Medium

DEP-006	Mobile Release Process	Mobile app releases must follow a phased rollout approach (10%, 25%, 50%, 100%)	Medium
DEP-007	Release Documentation	Each release must include release notes and deployment documentation	Medium
DEP-008	Health Checks	The system must implement health check endpoints for monitoring service status	High
DEP-009	Deployment Automation	All deployments must use automated CI/CD pipelines	Medium
DEP-010	Environment Configuration	All environment-specific configuration must be stored in a secure parameter store.	High

27.System Quality Attributes

Quality Attributes (Table 38)

Quality Attribute	Description	Measurement	Target
Reliability	The system's ability to perform without failure	Mean time between failures	>720 hours
Scalability	The system's ability to handle a growing user base	Response time degradation under load	<20% at 2x load
Portability	The system's ability to operate across platforms	Number of supported platforms	Web + iOS + Android

Quality Attribute	Description	Measurement	Target
Interoperability	The system's ability to exchange information	Number of supported integration points	≥3 external systems
Recoverability	The system's ability to recover from failures	Recovery time objective	<15 minutes
Maintainability	Ease of making changes to the system	Time to implement typical change	<2 developer-days
Localizability	Support for multiple languages	Number of supported languages	English + Spanish + French

28.OpenUP Methodology Alignment

This document has been structured to align with the OpenUP (Open Unified Process) methodology, incorporating its key principles and practices.

29.1 OpenUP Principles Applied

OpenUp Applications (Table 39)

OpenUP Principle	Application in EatFit Project
Collaborate to align interests and share understanding	Stakeholder collaboration model with regular touchpoints (Section 7.1)
Balance competing priorities to maximize stakeholder value	MoSCoW prioritization with stakeholder input (Section 8)
Focus on the architecture early	Architecture-centric approach with early risk reduction (Appendix C and D)
Evolve to continuously obtain feedback and improve	Iterative development plan with feedback cycles (Section 14)

29.2 OpenUP Practices Implemented

1. Incremental Development: Requirements and architecture evolve across iterations (Section 14)

2. Risk-Value Lifecycle: Highest risks addressed in earliest iterations (Section 13.1)
3. Test-Driven Development: Test specifications created before implementation (Section 17.9)
4. Continuous Integration: Work items integrated continuously throughout iterations (Section 14)
5. Shared Vision: Vision statement aligned with stakeholder needs (Sections 4 and 5)
6. Architecture Evolution: Architecture evolves throughout project lifecycle (Appendix D)

29.3 OpenUP Work Products

This document combines several OpenUP work products:

- Vision document
- Use case model
- Software architecture document
- Test cases
- Risk list
- Iteration plan

By integrating these work products into a cohesive document, we provide a comprehensive foundation for the EatFit project while maintaining the lightweight, pragmatic approach advocated by OpenUP methodology.

Document Finalization Checklist

Prior to submission, the following verification steps have been completed:

Consistency Check

- [✓] Consistent formatting and labeling throughout document

- [✓] All cross-references verified for accuracy
- [✓] All diagrams have proper captions and references
- [✓] Terminology used consistently across sections

Completeness Verification

- [✓] Each use case has complete flow descriptions
- [✓] All requirements have priority, verification method, and responsible party assigned
- [✓] All terms in glossary are used in document
- [✓] All referenced documents are properly cited

OpenUP Alignment

- [✓] Document demonstrates alignment with OpenUP principles
- [✓] Incremental, collaborative approach is evident throughout
- [✓] Risk-focused development strategy is clearly articulated
- [✓] OpenUP work products appropriately integrated

Stakeholder Review

- [✓] Stakeholder review completed on April 18, 2025
- [✓] Feedback incorporated from Product Owner, Technical Lead, and User Representatives
- [✓] Outstanding concerns documented for post-submission action
- [✓] Final approval pending document submission

Completed by: William Richmond Date: April 25, 2025

Appendix A: Traceability Matrix

This comprehensive traceability matrix ensures bidirectional traceability between business needs, requirements, use cases, work items, and verification methods.

Core Use Cases to Business Needs (Table 40)

Use Case ID	Business Need	Functional Requirements	Non-Functional Requirements	Verification Method
UC-001: Log Meal Ingredients	Track nutritional intake of home-cooked meals	DAT-001, DAT-002, DAT-006, DAT-007	PRF-003, USB-001, USB-006	TST-003, TST-006, TST-010
UC-002: View Nutrition Report	Analyze dietary habits and nutritional balance	DAT-004, DAT-008, DAT-010	PRF-001, PRF-008, USB-004, USB-008	TST-003, TST-004, TST-006
UC-003: Set Diet Goals	Establish personalized nutrition targets	BUS-003, BUS-004, BUS-010, DAT-005	USB-003, USB-010	TST-003, TST-010
UC-004: Send Reminders	Improve adherence to nutrition plans	INT-007, LOG-001	AVL-004, USB-003	TST-003, TST-007, TST-010
UC-005: Maintain USDA Database	Ensure accurate nutritional information	DAT-002, DAT-004, DAT-010	PRF-009, MNT-005, LOG-002	TST-002, TST-009
UC-006: User Authentication	Secure personal health information	SEC-003, SEC-004, SEC-005, SEC-009	LGL-007, LGL-008, PRF-005	TST-002, TST-005, TST-009

UC-007: Update Profile	Personalize user experience	DAT-005, DAT-009, LOG-001	USB-003, LGL-003, LGL-004	TST-003, TST-010

Security Requirements to Compliance Needs (Table 41)

Security Requirement	Compliance Requirement	Technical Implementation	Verification Method
SEC-001: Data Encryption	LGL-001, LGL-007	Azure storage encryption, AES-256	TST-005, Security audit
SEC-002: Transport Security	LGL-001, LGL-007	TLS 1.3, HTTPS-only	TST-005, Penetration testing
SEC-003: Access Control	LGL-007, LGL-008	Azure AD role-based access	TST-005, Security audit
SEC-007: Audit Logging	LGL-007	Comprehensive activity logging	TST-005, Compliance audit
SEC-008: Data Privacy	LGL-003, LGL-004, LGL-008	Privacy controls, data export/deletion	TST-005, GDPR compliance testing

Performance Requirements to Technical Components (Table 42)

Performance Requirement	System Component	Implementation Approach	Verification Method
PRF-001: Report Generation	Backend API, Database	Optimized queries, caching	TST-004, Performance testing
PRF-003: Page Load Time	Frontend, API	Code splitting, lazy loading	TST-004, Performance testing

PRF-005: API Response Time	Backend API, Network	Performance optimization, CDN	TST-004, API performance testing
PRF-006: Mobile Startup	React Native	Optimized bundle size	TST-004, Mobile performance testing
PRF-009: Data Caching	Backend, Database	Redis cache, in-memory caching	TST-004, Performance testing

A.1 Requirements to Use Cases Traceability

Use Case Security Requirements (Table 43)

Requirement ID	UC-001	UC-002	UC-003	UC-004	UC-005	UC-006	UC-007
DAT-001	✓						
DAT-002	✓				✓		
DAT-003		✓					
DAT-004		✓			✓		
DAT-005			✓				✓
SEC-001						✓	
SEC-002						✓	
PRF-001		✓					
PRF-003	✓						

A.2 Requirements to Work Items Traceability

Work Items Traceability Requirements (Table 44)

Requirement ID	Work Items
DAT-001, DAT-002	WI-02, WI-03, WI-10
SEC-001, SEC-002	WI-11
PRF-001, PRF-003	WI-12

A.3 Bidirectional Traceability Example

Business Need: Track nutritional intake of home-cooked meals

- Traces down to: UC-001 (Log Meal Ingredients)
    - Traces down to: DAT-001, DAT-002, DAT-006, DAT-007
        - Traces down to: WI-02, WI-03, WI-10
            - Traces down to: Test cases TC-01, TC-02
    - Traces up to: Business need: Track nutritional intake

This bidirectional traceability ensures that every requirement, work item, and test case can be traced both to its origin (business need) and its implementation/verification methods, maintaining a complete chain of evidence throughout the project lifecycle.

Appendix B: Decision Log

Decision Log (Table 45)

ID	Decision	Rationale	Alternatives Considered	Date
DEC-001	Use REST API architecture	Industry standard, developer familiarity, and wide support	GraphQL, SOAP	2025-03-15
DEC-002	Cosmos Database is the primary database	NoSQL document database provides flexibility for varying nutritional data structures and scales well with Azure services	MongoDB, PostgreSQL	2025-03-18
DEC-003	React Native for mobile apps	Code reuse across platforms, faster development	Native iOS/Android, Flutter	2025-03-20
DEC-004	Microsoft Azure as the cloud platform	Comprehensive services, team familiarity, and compliance certifications	AWS, Google Cloud	2025-03-25
DEC-005	Implement offline-first design	User feedback indicating the need for offline access	Server-dependent design	2025-04-01

Appendix C: Architecture Notebook (Summary)

This appendix outlines key architectural decisions made during the EatFit project. These decisions guided the selection of tools, frameworks, and design patterns to ensure the application is scalable, secure, and compliant with HIPAA and GDPR regulations.

A.1 System Structure

- Frontend: Built with React Native to allow for a cross-platform user interface with a single codebase supporting Android and iOS.
- Backend: Powered by Microsoft Azure API Apps, offering scalability, authentication, and HIPAA-compliant hosting.
- Database: Cosmos DB was chosen for its high availability, scalability, and support for secure document-based storage.

A.2 Security Considerations

- Authentication & Authorization: Utilizes Azure AD B2C for secure login, role-based access, and optional multi-factor authentication (MFA).
- Encryption: Data is protected with SSL/TLS in transit and encrypted at rest via Azure's built-in encryption features.
- Privacy Protections: Follows HIPAA and GDPR standards, including data masking in logs and audit trail planning.

A.3 Integration Strategy

- USDA API: Primary source for verified nutritional data. Access patterns are optimized using caching and backoff strategies.
- Voice Input & Smart Devices *(Planned)*: Designed to support future voice command integration with Alexa and Google Home.

- Analytics Dashboard *(Planned)*: Future enhancements will include admin tools for monitoring trends, usage, and nutritional feedback loops.

A.4 Design Patterns & Principles

- Model-View-Controller (MVC): Ensures clear separation of logic and presentation layers in both frontend and backend services.
- API Design: Each functional unit (Modular logging, analyzing, listing, etc.) is encapsulated within its own API endpoint for testability and security.
- Future-Ready Scalability: All services are designed with horizontal scalability and container-based deployment in mind (via Azure Containers or Kubernetes, if needed).

Appendix D: Architecture Evolution Plan

Following OpenUP guidance on evolutionary architecture, this section outlines how the EatFit architecture will evolve across iterations.

D.1 Architectural Decisions Points

Iterative Decision Points(Table 46)

Iteration	Architecture Focus	Decision Point	Evaluation Criteria
Iteration 1	Initial architecture vision	Selection of overall platform approach	Alignment with business goals, scalability potential
Iteration 2	Data model foundation	Database technology selection	Data flexibility, query performance, compliance capability
Iteration 3	API layer definition	API design pattern selection	Interface clarity, extensibility, developer experience
Iteration 4	Front-end architecture	React component structure, state management	Performance, reusability, maintainability
Iteration 5	Security architecture	Authentication/authorization approach	HIPAA/GDPR compliance, usability, security strength
Iteration 6	Integration architecture	Service communication patterns	Reliability, performance, fault tolerance

D.2 Technical Debt Management

Each iteration will include explicit time for addressing technical debt:

1. Debt Identification: Architecture reviews identify suboptimal implementations
2. Debt Tracking: Technical debt items tracked in backlog with clear impact metrics
3. Remediation Planning: Each iteration allocates 15% capacity to debt reduction
4. Architectural Refactoring: Scheduled refactoring iterations after major milestones

D.3 Architecture Validation

The architecture will be validated through:

1. Architectural Spikes: Targeted experiments for high-risk components
2. Prototyping: Working prototypes of key architectural elements
3. Peer Reviews: Architecture review board assessments
4. Performance Testing: Early performance testing of critical paths

This evolutionary approach ensures the architecture remains adaptable to emerging requirements while maintaining alignment with quality attributes and stakeholder needs.