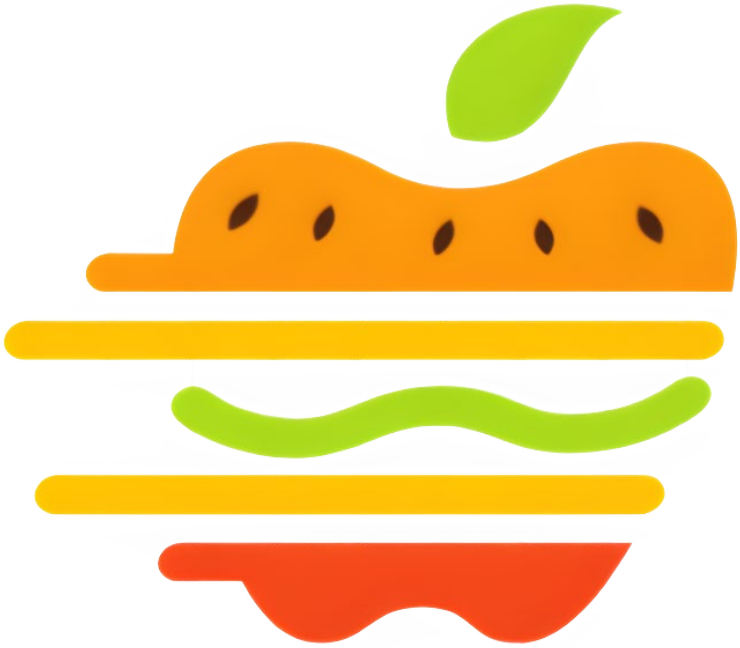

NUTRISWAP

Software Project Planning, Design, and Quality Management

Prepared By: Marella Morad

Student ID: 103076428

Tutor Name: Harsharan Kaur

Tutorial Class: Monday 2:30 Ba708

Table of Contents

Table of Figures

1 Introduction:

In this report, we embark on a journey into the realms of software project management, design, and quality assurance, all while exploring the exciting potential of a project idea that interests me, I have called it NutriSwap. The NutriSwap project is aimed at addressing a pressing concern in today's world – the need for healthier dietary choices and improved access to nutritional information. This report will delve into the project's business needs, usage scenarios, product backlog items, schedule planning, architectural considerations, and quality management strategies. It is through this comprehensive analysis and planning that we will lay the foundation for a successful software project.

2 Background:

In today's fast-paced world, the importance of making healthier dietary choices is more evident than ever. NutriSwap comes as a response to the growing need for a solution that simplifies the process of finding healthier food alternatives and gaining insights into one's nutritional choices. Many health apps lack accurate information about home-cooked food, making this an opportunity to provide a valuable solution. This project is designed to offer a convenient and user-friendly solution to empower individuals in their journey toward healthier eating.

3 Scope:

3.1 Objective:

The objective of the NutriSwap project is to develop a user-friendly, cross-platform web application that provides valuable resources and suggestions for making healthier food choices. The project aims to enable users to access accurate nutritional information for various food items and discover substitutes with lower calories and healthier ingredients. By doing so, NutriSwap intends to foster a healthier lifestyle and create a more informed food consumption culture.

3.2 In Scope:

3.2.1 Nutritional Information:

The application will offer users access to comprehensive and accurate nutritional data for a wide range of food items, including macronutrients, micronutrients, and caloric content.

3.2.2 Suggest Healthier Alternatives:

NutriSwap will provide users with personalized, health-conscious substitutes for the foods they search for in the app. These recommendations consider individual preferences and dietary restrictions outlined in the user's profile. These results also align with the user's specific objectives for using the app, such as cutting calories, reducing sugar, or adhering to specific diets like gluten-free or vegan, etc…

3.2.3 Locates Healthier Alternatives:

NutriSwap will also provide users with links to get the suggested alternatives from local stores or an e-commerce platform. The local store search will be based on the address information provided by the user (stored in the user profile).

3.2.4 Recipe Transformation:

Users will be able to input their favourite recipes, either by typing it in, or by entering the URL to an online recipe (in any language), and the application will provide alternatives to make the same dish with healthier ingredients.

3.2.5 Personalised Recommendations:

Leveraging AI and machine learning algorithms, NutriSwap will learn from user preferences and offer personalised dietary recommendations over time**.** This feature will only be enabled if the user agrees to it as it will be using the user data for this analytic purpose.

3.2.6 User-Friendly Interface:

One of NutriSwap's main goals is to simplify the process of finding healthy food for everyone. Therefore, it needs an intuitive GUI for seamless user interaction and data entry.

3.2.7 User Registration and Profile Creation:

Users need to create accounts, allowing the app to tailor recommendations based on their information, like address, dietary needs, and health goals.

3.3 Out of Scope:

3.3.1 Healthy Food Education:

NutriSwap won't have a dedicated education section about healthy eating.

3.3.2 Detailed Mapping and Navigation Integration:

The app won't provide directions to stores, just store names or online store links.

3.3.3 Payment Processing:

The application will not handle payment processing, as its focus is on providing nutritional information and food alternatives not purchasing these alternatives.

3.3.4 Third-party Integrations:

The app won't connect to external systems beyond nutrition and food suggestions.

4 Deliverables and Schedule:

4.1 Deliverables:

- **Source Code:** A complete source code of the application for potential future modifications and enhancements.
- **Running Application:** A fully functional software web application, compatible with all platforms.
- **User Manual:** A comprehensive user manual that guides users on how to effectively use the application.
- **Training Materials:** Basic training videos to help users become familiar with the application's features and functions.

4.2 Product Backlog Items & Business Values Justification:

The following is a list of product backlog items thoughtfully allocated across two consecutive sprints: each spanning two weeks or 10 working days. Our team is composed of six dedicated professionals: a Product Owner, a Scrum Master, Three Developers, and a Tester. Adhering to the given constraint of dedicating 8 hours per team member per week, we have meticulously designed these sprints to accommodate a total of 96 hours. The subsequent backlog items, each with their respective importance explained, have been planned in consideration of the available resources and the expected timeframe.

Note: In square brackets, you will find a reference to the listed In Scope features that each task corresponds to.

4.2.1 Sprint 1: Core Functionality and User Interaction

No.	Item Name	BV	Justification	Reference
1	Create User Database Table	8	To store user details and preferences for personalized recommendations	3.2.1
2	Develop User Registration Forms and Validation	8	Develop sign-up forms for new users to join securely	3.2.7
3	Create User Profile Pages	7	Enable users to input dietary needs and health goals	3.2.3
4	Implement User Authentication and Login	8	Ensure secure logins for a personalized experience	3.2.4
5	Develop User Settings	8	Let users control their recommendations	3.2.5
6	Develop a User Dashboard	8	Create a user-friendly interface to manage profiles	3.2.6
7	Plan User Registration and Profile Management Test Cases	7	Plan test cases to test the functionality of user registration and profile management	3.2.7
8	Execute User Registration and Profile Management Unit Testing	8	Verify the correctness of user profile features by internally testing the functionalities	3.2.7

9	Develop Location-Based Search	8	Improve user experience with location-based searches	3.2.2
10	Display Search Results with Links to Stores	7	Show healthier food options with store links	3.2.3
11	Create Recipe Database Table	8	Important to save the imported recipes so that the user can retrieve them from their account whenever they need them	3.2.4
12	Develop Recipe Import Page	7	Simplify recipe entry by importing the recipe from an online source (Website)	3.2.4
13	Develop the Manual Recipe Entry Page	7	Allow users to input recipes directly (if it's a personal recipe for example)	3.2.4
14	Implement Recipe Transformation Algorithm	9	Implement an algorithm to analyse recipes, identify ingredients, and propose healthier alternatives	3.2.4
15	Display Transformed Recipes	9	Display transformed recipes with detailed nutritional information for each ingredient	3.2.4

4.2.2 Sprint 2: Advanced Functionality and Quality Assurance

No.	Item Name	BV	Justification	Reference
1	Integrate a translation tool for recipes in different languages	7	Adds multilingual support to translate recipes into different languages to expand the app's reach and enable the user to have healthier authentic food	3.2.4
2	Plan Location-Based Search and Recipe Transformation Test Cases	7	Create tests for location-based search and recipe transformation	3.2.3, 3.2.4
3	Execute Location-Based Search and Recipe Transformation Unit Testing	7	Check the accuracy and effectiveness of these core features	3.2.3, 3.2.4
4	Design AI and Machine Learning Architecture for Personalized Recommendations	8	Design architecture for personalized dietary recommendations (turned off by default – for privacy reasons, the user can choose to turn it on)	3.2.5
5	Develop a Personalized Recommendation Engine	8	Build personalized dietary suggestions for user engagement	3.2.5
6	Develop a Page to display the privacy statement	7	Provide transparency on data usage and address privacy concerns	3.2.6
7	Plan Personalized Recommendations and User Privacy Settings Test Cases	7	Validate personalized recommendations and privacy settings	3.2.5, 3.2.6
8	Execute Personalized Recommendations and	8	Ensure the security and effectiveness of advanced features	3.2.5, 3.2.6

	User Privacy Settings Unit Testing			
9	Finalize Internal Application Testing and Bug Fixing	9	Thoroughly test for critical bugs to ensure reliability	3.2.1 - 3.2.7
10	Run Usability Testing	8	Enhance user experience through user-friendly design and functionality	3.2.1 - 3.2.7
11	Run User Acceptance Testing	8	Confirm the app meets user expectations and requirements	3.2.1 - 3.2.7
12	Another round of Bug Fixes	8	Address issues raised by the Usability and User Acceptance testing	3.2.1 - 3.2.7
13	Create a User Guide and Documentation	7	Provide resources for effective app usage and issue troubleshooting	3.2.1 - 3.2.7
14	Develop Video Tutorials for using NutriSwap	7	Offer visual guidance to enhance app usability	3.2.1 - 3.2.7
15	Prepare the application for release and deployment	8	Get the app ready for user access	3.2.1 - 3.2.7

5 Solution Direction and High-Level Design:

5.1 Web-Based Application

The decision to choose a web-based application is so that NutriSwap is accessible through web browsers on different devices, offering a more versatile and platform-independent solution. The web application still provides users with a user-friendly interface to access nutritional data, discover healthier food options, and manage their profiles. The server-side components handle data processing, recommendation algorithms, and data synchronization for a seamless user experience, while external APIs and databases provide accurate, up-to-date information. The AI and machine learning engine continues to deliver personalized food recommendations in alignment with the project's goals.

5.1.1 High-Level Architecture Diagram:

In this solution direction, NutriSwap will be developed as a web-based application, accessible through web browsers on various platforms. Below is the high-level architecture diagram for the web application:

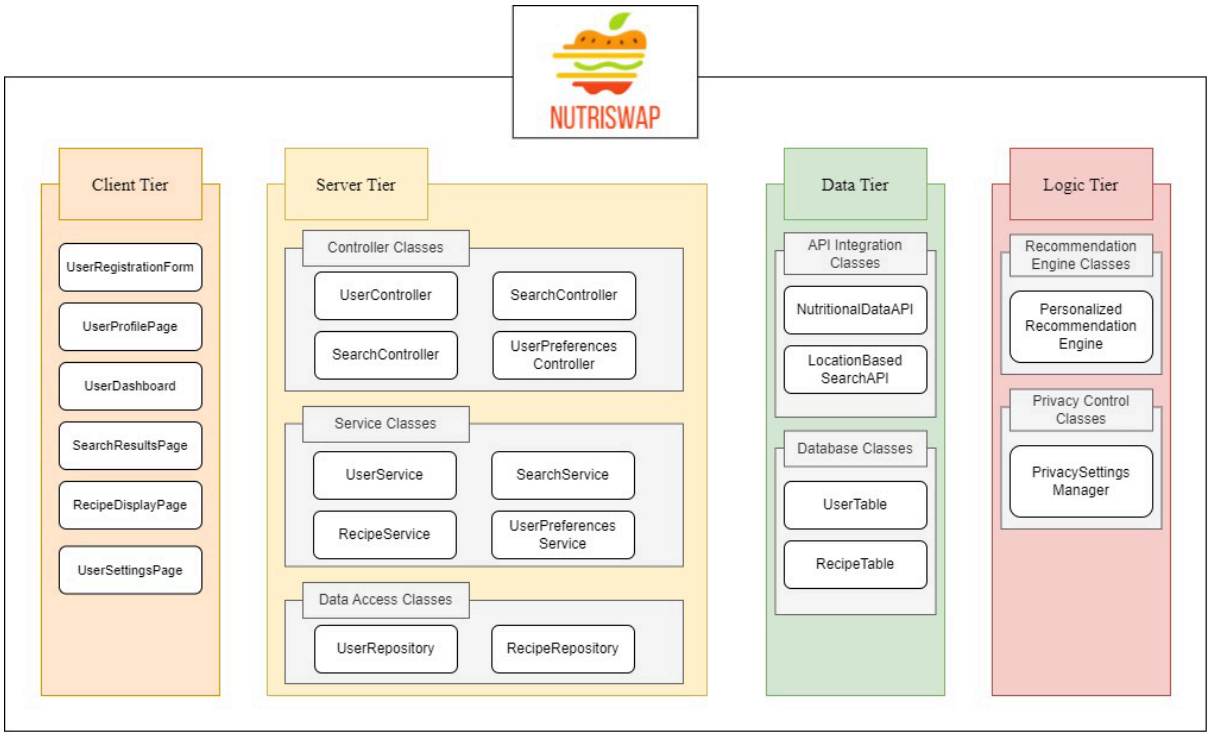

Figure 5.1 High-Level Design of NutriSwap

5.1.2 Architecture Components:

5.1.2.1 Web Application (Client Tier):

Role: The web application is the client-side interface that users access via web browsers. It's responsible for rendering the user interface, collecting user inputs, and displaying search results, recipes, and recommendations.

Responsibilities: The web app communicates with the server-side components to request nutritional data, search for healthier alternatives, and manage user profiles. It integrates features such as location-based search, recipe transformation, and multilingual support.

5.1.2.2 Web Server (Server Tier):

Role: The web server serves as the server side of the web application, handling data processing, storage, and business logic. It connects to external APIs and databases to provide accurate nutritional data and alternatives to the web app.

Responsibilities: The web server manages user data, including profiles, preferences, and health goals. It integrates with external sources to fetch nutritional information and provides real-time suggestions based on user inputs. It supports location-based search and recipe transformation algorithms.

5.1.2.3 External APIs and Databases (Data Tier):

Role: External APIs and databases are responsible for supplying nutritional data, recipes, and alternative food choices to the web server. These external sources offer real-time, accurate data for NutriSwap.

Responsibilities: External APIs provide access to comprehensive nutritional data for various food items, while databases store user profiles and their saved recipes for easy retrieval. The web server accesses this data to provide users with accurate information.

5.1.2.4 AI and Machine Learning Engine (Logic Tier):

Role: The AI and machine learning engine, part of the web server, is responsible for personalized dietary recommendations. It analyses user preferences and dietary restrictions to offer tailored food suggestions over time.

Responsibilities: The engine continuously learns from user behaviour and preferences to improve the quality of recommendations. It respects user privacy settings and only operates when explicitly enabled by the user.

5.2 Version Control and Issue Tracking

NutriSwap's development process will be managed through GitHub and Jira to facilitate efficient and organized software development.

5.2.1 GitHub:

GitHub will be used for version control, allowing the development team to collaboratively work on code, track changes, and manage different versions of the application. GitHub simplifies code merging and provides a repository for the entire project.

5.2.2 Jira Integration:

To streamline development efforts and enhance project management, NutriSwap's GitHub repository will be integrated with Jira. This integration will provide several advantages:

- **Issue Progress Synchronization:** Jira issues will be synchronized with the progress of corresponding code-related activities in GitHub. For instance, when a pull request is merged, the related issue in Jira will be moved to the "Done" status.
- **Issue Assignment:** When a task is moved to "In Progress" in Jira, a branch will be automatically created in GitHub, helping developers stay organized.
- **Automated Workflows:** The integration between Jira and GitHub will establish automated workflows, reducing manual overhead and providing a more accurate reflection of the project's status.
- **Efficient Collaboration:** Developers, testers, and other team members can easily collaborate within Jira, ensuring that everyone is aligned with the project's objectives and progress.

5.3 Design Justification using Design Principles

5.3.1 Object-Oriented Principles:

At NutriSwap, we place a strong emphasis on object-oriented principles to ensure that our codebase is not only well-structured but also easily maintainable and extensible. We've chosen object-oriented languages and frameworks such as React, and C# to help us better comply with these principles. Our adherence to these principles is reflected in the following aspects:

5.3.1.1 Strong Cohesion and Weak Coupling

Our software design is built upon the foundation of strong cohesion and weak coupling. We have meticulously organized our code into separate modules, each dedicated to specific functionalities. This division enhances code clarity, reduces complexity, and fosters maintainability. For instance, rather than creating a monolithic component for the entire application, we've wisely divided it into smaller, cohesive components, each responsible for distinct tasks. This approach enables individual components to interact with one another with minimal dependencies, achieving weak coupling. The benefit is that these components can function effectively without requiring an intricate understanding of the internal workings of other components. This modularity empowers us to make changes or introduce new features without causing ripple effects throughout the codebase.

5.3.1.2 Separation of Concerns (SoC)

We've adhered to the separation of concerns principle as well, ensuring that different aspects of the application are treated as separate concerns. This helps us maintain code clarity and makes it easier to manage the different layers and functionalities within NutriSwap. For example, user registration and profile management are distinct concerns separated from core features like search functionality and recipe transformation. This separation of concerns keeps our codebase organized and promotes maintainability.

5.3.1.3 Encapsulation

Encapsulation is another fundamental aspect of our design. Each component in NutriSwap encapsulates its behaviour and rendering logic, exposing only necessary interfaces to the outside world. This approach safeguards data integrity and ensures that changes to one part of the system do not unintentionally impact other parts.

5.3.1.4 Information Hiding

NutriSwap is designed with information hiding in mind, concealing internal details and data within modules. This minimizes complexity, enhances security, and prevents unintended data modifications.

5.3.1.5 Scalability and Performance Considerations

The architecture of NutriSwap has been carefully crafted to accommodate scalability and optimize performance. We employ techniques such as load balancing, caching, and asynchronous processing to ensure that the application responds to user interactions within an acceptable response time and can handle anticipated user loads without significant performance degradation.

5.3.2 Designing the Logo and UI

At NutriSwap, we've given careful thought to the visual identity of our web application. We recognize the significance of creating a distinct and memorable user experience, and the design of our logo and user interface (UI) plays a pivotal role in achieving this objective.

5.3.2.1 Logo Design

We with crafting a logo that embodies the essence of warmth and professionalism, promoting healthier choices and symbolising NutriSwap's name. After undergoing several design iterations as you can see below:

Figure 5.2 Initial Logo Design Ideas

However, there was still something missing from the logo. That is, the logo didn't depict the idea of "Swapping" junk food to healthy food. Which we managed to achieve in the chosen logo that shows a Hamburger being transformed into an apple.

Figure 5.3 Chosen Logo

5.3.2.2 Colour Palette

In addition to the logo, we spent considerable time choosing a colour palette for our web application. Ensuring that our colours reflect bright colours (symbolising vegetables, fruits and healthy food in general) as well as ensuring we have a range of contrasting colours to comply with accessibility standards.

Figure 5.4 NutriSwap's Colour Palette

5.3.3 Naming Conventions:

5.3.3.1 Component and Class Names:

When it comes to naming components and classes in NutriSwap, we adhere to a structured approach that emphasizes clarity and consistency. We have adopted the PascalCase naming convention, which means that names are written with the first letter of each word capitalized. For example, "UserRegistrationForm," "SearchResultsPage," and "UserService." This naming convention helps developers quickly identify and understand the purpose and functionality of each component, ensuring our codebase is well-organized and comprehensible.

5.3.3.2 Variable and Function Names:

In the context of variable and function names, we follow the camelCase naming convention, which is an established industry standard. This convention is known for its readability and

ease of use. With camelCase, names are written with an initial lowercase letter, and subsequent words within the name are capitalised.

5.4 Use Case Diagram

Figure 5.5 NutriSwap's Use Case Diagram illustrates the core functionality and interactions within the NutriSwap web application, with the "User" as the primary actor. Users initiate various use cases, reflecting the application's objectives to promote healthier eating and informed food choices. The diagram begins with the "Register" use case, where users create accounts to personalize their experience. The "Login" use case, included in the "Register" process, grants access to the application. Subsequently, the "Search for Healthier Alternatives" use case is activated, allowing users to discover and view alternative food choices with improved nutritional profiles after they have signed in to the application. This process may extend to "View Detailed Nutritional Information," offering in-depth insights.

In parallel, the "Import Recipe" use case emerges, enabling users to input their favourite recipes either by importing from a URL or manually typing in the recipe. Following this, users may opt for "Receive Recipe Transformation Recommendations," further aligning with their health goals. Additionally, users can "Translate Recipe" if they wish to make their favourite dishes healthier. Furthermore, there is an "Include" relationship between "Search for Healthier Alternatives" and "Locate Healthier Alternatives at Local Store" to facilitate finding local stores for recommended alternatives.

This Use Case Diagram encapsulates the core functions of NutriSwap, emphasizing user account management, food alternative discovery, and recipe transformation, all contributing to the project's mission of promoting healthier eating and informed food choices.

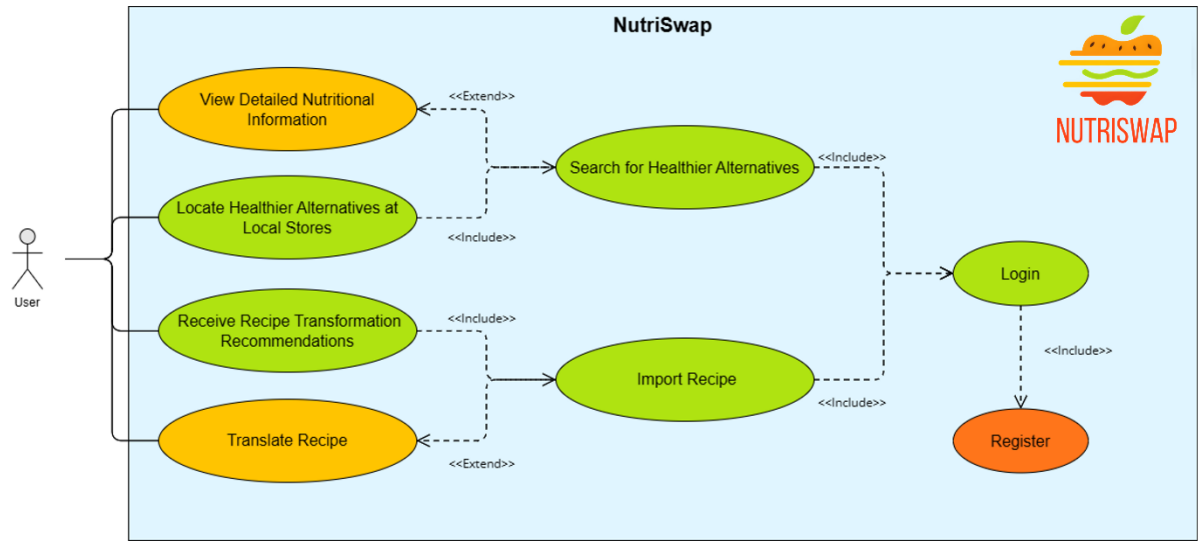

Figure 5.5 NutriSwap's Use Case Diagram

6 Definition of Done and Quality:

6.1 Definition of Done:

The “Definition of Done” (DoD) for NutriSwap, in alignment with the ISO 25010 Software Quality Model and project-specific goals, comprises a comprehensive set of criteria. It serves as a vital framework to determine the completion of features or user stories, safeguarding the software's overall quality and its ability to fulfil its intended purposes. This DoD encompasses both functional and non-functional (quality) requirements, guaranteeing the NutriSwap web application's readiness for successful delivery and release (Standards Australia/New Zealand, 2013).

6.1.1 Functional Suitability (ISO 25010 – 4.2.1)

- All specified user stories have been implemented and tested successfully.
- All core features, including user registration, profile management, search functionality, recipe transformation, and personalized recommendations, are fully functional and meet the defined acceptance criteria.

6.1.2 Performance Efficiency (ISO 25010 – 4.2.2):

- The application responds to user interactions within an acceptable response time.
- The application can handle the expected user load without significant performance degradation.

6.1.3 Compatibility (ISO 25010 – 4.2.3):

- The web application is compatible with commonly used web browsers (Chrome, Firefox, Safari, Edge).
- It provides a responsive design that adapts to various screen sizes and resolutions, including mobile devices and desktops.

6.1.4 Usability (ISO 25010 – 4.2.4):

- The user interface is intuitive and user-friendly, allowing users to easily navigate, search for food items, and view recommendations.
- User feedback is positive, indicating high usability and user satisfaction.

6.1.5 Reliability (ISO 25010 – 4.2.5):

- The application is stable and does not crash or produce errors during normal usage.
- Data integrity is maintained, ensuring that user profiles, preferences, and dietary information are accurately stored and retrieved.

6.1.6 Security (ISO 25010 – 4.2.6):

- User data is protected through secure authentication and authorization mechanisms.
- Personalized recommendations respect user privacy settings and only operate when explicitly enabled by the user.

6.1.7 Maintainability (ISO 25010 – 4.2.7):

- The codebase is well-documented, organized, and follows coding standards.

- Updates and changes can be made with relative ease to accommodate future feature enhancements and bug fixes.

6.1.8 Portability (ISO 25010 – 4.2.8):

- The web application can be deployed on various web hosting platforms.
- It is cross-browser compatible and works on different devices and operating systems (Windows, Android, IOS, etc…).

6.2 SMART Goals:

6.2.1 S: Specific

- Improve the user interface for mobile devices to enhance accessibility.

6.2.2 M: Measurable

- Achieve a user satisfaction rating of at least 4 out of 5 based on user feedback and reviews.
- Reduce the average response time for user interactions to less than 1 second.

6.2.3 A: Achievable

- Develop and implement a personalized recommendation system based on machine learning to improve user engagement.
- Ensure that the web application is compatible with the latest versions of Chrome, Firefox, Safari, and Microsoft Edge.

6.2.4 R: Relevant

- Align the application's dietary recommendations with the Australian Dietary Guidelines to promote healthier food choices.
- Enhance security by implementing two-factor authentication to protect user data.

6.2.5 T: Time-bound

- Complete the integration of a translation tool for recipes in various languages within the next three months.
- Roll out regular updates and feature enhancements every two months to ensure the application remains competitive and user-friendly.

6.3 Quality Model:

6.3.1 Functional Suitability:

6.3.1.1 Functional Completeness:

- Percentage of Completed Features
  - ≥ 95% of planned features are implemented and operational.

6.3.1.2 Functional Correctness:

- Number of Critical Bugs
  - ≤ 1 Critical Bug in the application.

6.3.1.3 Functional Appropriateness:

- Compliance with Dietary Guidelines
  - All dietary recommendations align with the Australian Dietary Guidelines (National Health and Medical Research Council 2013)

6.3.2 Performance Efficiency:

6.3.2.1 Time Behaviour:

- Average Response Time for User Interactions
  - ≤ 1 second

6.3.2.2 Resource Utilization:

- The percentage of Application Resource Usage
  - < 80% of system resources

6.3.2.3 Capacity:

- The number of concurrent users the application supports without a significant drop in performance:
  - 100 Concurrent Users

6.3.3 Compatibility:

6.3.3.1 Co-existence:

- Browsers on which NutriSwap functions without critical errors
  - The latest versions of Chrome, Firefox, Safari and Microsoft Edge

6.3.4 Usability:

6.3.4.1 Appropriateness Recognizability:

- The percentage of recognizable key features by users without guidance
  - ≥ 85%

6.3.4.2 Learnability:

- Amount of training required for non-technical and technical users to be able to learn basic functions respectively:
  - < 1 hour and < 30 minutes respectively

6.3.4.3 *Operability:*

- The success rate of users completing key tasks (such as searching for a healthier alternative)
    - ≥ 90%

6.3.4.4 *User Error Protection:*

- Error Handling
    - Users receive clear error messages for incorrect inputs, reducing user errors by ≥ 90%.

6.3.4.5 *Accessibility:*

- Compliance with WCAG (Web Content Accessibility Guidelines) Standards
    - NutriSwap complies with WCAG 2.1 (latest) accessibility standards for users with disabilities.

6.3.5 Reliability:

6.3.5.1 *Maturity:*

- The continuous period the application operates without a critical failure.
    - ≥ 30 days

6.3.5.2 *Availability:*

- Percentage of uptime the system is required to have.
    - ≥ 99% (excluding planned maintenance windows)

6.3.5.3 *Fault Tolerance:*

- Error Recovery
    - The application can recover from non-critical errors without data loss.

6.3.5.4 *Recoverability:*

- Data Recovery of Accidental Deletion
    - 24 Hours from deletion (for example, if the user accidentally deletes one of their imported recipes, the application should offer an option to retrieve it within 24 hours.

6.3.6 Security:

6.3.6.1 *Confidentiality:*

- All Personally Identifiable Information (PII) will be inaccessible until an authorized user has been authenticated.
- Sensitive User data is encrypted using AES-256 an industry-standard encryption algorithm.

6.3.6.2 *Integrity:*

- The number of vulnerabilities found in the software through security assessment.
    - 0

6.3.6.3 Non-repudiation:

- User Actions Tracking
  - The system tracks user actions to prevent the denial of user actions.

6.3.6.4 Accountability:

- User Activity Logging
  - User activities are logged for accountability and audit purposes.

6.3.6.5 Authenticity:

- User Authentication
  - All users will require authentication before being allowed access to the system using a 2-factor authentication method (e.g., Google Authenticator or Text Message Authentication).

6.3.7 Maintainability:

6.3.7.1 Reusability:

- Code Reuse
  - Code components are designed for reuse in future features or modules.

6.3.7.2 Analysability:

- Code Documentation
  - The codebase is well-documented, allowing for easy analysis and understanding.

6.3.7.3 Modifiability:

- Code Change Complexity
  - The average complexity of code changes should be below a value of 10 in terms of the Cyclomatic Complexity metric.

6.3.7.4 Testability:

- The percentage of code paths tested effectively.
  - ≥ 90%

6.3.8 Portability:

6.3.8.1 Adaptability:

- Cross-platform Compatibility
  - The web application functions on different platforms (e.g., Windows, macOS, Linux).

6.3.8.2 Replaceability:

- Third-party Integration
  - The application can integrate with third-party services for future enhancements or extensions.

7 Sprint 1 Detailed Plan:

In the development of our Sprint 1 Detailed Plan, we employed a variety of estimation techniques to ensure a well-rounded and accurate assessment of the tasks at hand. These techniques included Estimation by Analogy, Estimation by Size Comparison, and the "Experts" or "Delphi" Techniques. By utilizing different methods tailored to the specific nature of each task, we aimed to provide a comprehensive and reliable plan for Sprint 1. This approach allowed us to leverage the collective expertise of our team members, make informed comparisons to past efforts, and ensure that all aspects of task complexity were considered.

7.1.1 Estimation by Analogy

Working on the GotoGro project provided me with valuable experience, especially in the realm of estimating story points. For Task 2, I decided to allocate 8 story points based on a similar task we worked on for GotoGro. This estimation also takes into account the task's complexity, considering that it involves setting up the first web page of the entire application and configuring frameworks like React, which can be time-consuming.

Likewise, Task 3 is allocated 4 story points, primarily because the initialization of the web application was completed as part of Task 2. Using a similar technique, I assigned 4 story points to tasks 5, 10, 13, and 15, as they share commonalities. Task 6, on the other hand, received a higher estimate due to its broader scope, encompassing the viewing and editing of profile information.

No.	Item	Estimation
2	Develop User Registration Forms and Validation	8
3	Create User Profile Page, including input fields for dietary needs and health goals	4
5	Develop User Settings Page	4
6	Develop a Basic User Dashboard to view and edit profile information	5
10	Display Search Results with Links to Stores	4
13	Develop a Manual Recipe Entry Page	4
15	Display Transformed Recipes with Nutritional Information	4

In addition to my development experience, I've also invested time in testing GotoGro and gained professional testing experience at work. This exposure has given me insights into the time required for test planning, which is why I estimated 3 story points for Task 7. Likewise, I've allocated 3 story points for executing unit tests and usability tests, leveraging my understanding of the testing process and its associated effort.

No.	Item	Estimation
7	Plan and Design Test Cases for User Registration and Profile Management	3

8	Execute Unit Testing and Validate User Profile Functionality	3

7.1.2 Estimation by Size Comparison

Task 1, which involves creating the user database, is relatively small in size and complexity compared to other tasks. However, it includes setting up the database as an internal step. Drawing from my experience working on GotoGro, where we used Supabase for our backend, I found that setting up and creating tables was straightforward. Nonetheless, considering the expected user base of NutriSwap, it's clear that Supabase might not be the most suitable solution. Therefore, we've chosen to adopt a more traditional approach by creating a backend with APIs that communicate with both the frontend and the database.

Once the initial database schema is completed, Task 11, which involves creating the recipe database table, is a relatively straightforward task, and I've allocated it 1 story point to reflect its simplicity and low complexity.

No.	Item	Estimation
1	Create User Database Table	5
11	Create Recipe Database Table	1

Task 9 is estimated to be of higher complexity compared to Task 7 primarily because it involves all three layers of the application working in concert to achieve its objectives. In Task 9, the frontend initiates a search request, the backend retrieves the user's address details from the database, and then conducts the search based on the user's address. This interplay between the frontend, backend, and database adds to the complexity, and I've allocated 6 story points to reflect this.

Similarly, Task 12 is estimated to be of similar complexity, receiving 6 story points. This task involves the installation and integration of third-party libraries to enable the import functionality. This process can be intricate and, therefore, justifies a higher point estimate.

Finally, Task 14 is also given 6 story points. While it partially relies on the search algorithm from Task 9, the difference lies in the search being conducted on all recipe ingredients. This expanded scope increases the complexity, resulting in a higher point estimate.

No.	Item	Estimation
9	Develop Location-Based Search Functionality	8
12	Develop Recipe Import Page	6
14	Implement Recipe Transformation Algorithm	4

7.1.3 Estimation by "Experts" / "Delphi" Techniques:

I decided to use the "Experts" / "Delphi" technique for tasks that I have limited exposure to or tasks that are complex and need expert advice. Task 4 is a prime example of such a task, which is relatively new to me. However, upon research, it became evident that Task 4 is too broad (Sanwar Ranwa, 2020). As a result, it was recommended to break this task down into

smaller, more manageable sub-tasks, each with its respective estimation. This approach allows for a more accurate estimation and better planning of the work involved.

No.	Item	Estimation
4	Implement User Authentication and Login Functionality • Develop server-side API for registration, login, and logout. • Write C# server-side code for database interaction and authentication. • Create React-based client-side login form for API communication. • Securely store user credentials, including sensitive data encryption.	10 3 3 2 2

7.2 Overall Sprint Tasks and Estimations:

No.	Item	Dep	Estimation
1	Create User Database Table		5
2	Develop User Registration Forms and Validation	1	8
3	Create User Profile Page, including input fields for dietary needs and health goals	2	4
4	Implement User Authentication and Login Functionality • Develop server-side API for registration, login, and logout. • Write C# server-side code for database interaction and authentication. • Create React-based client-side login form for API communication. • Securely store user credentials, including sensitive data encryption.	2	10 3 3 2 2
5	Develop User Settings Page	3	4
6	Develop a Basic User Dashboard to view and edit profile information	4, 5	5
7	Plan and Design Test Cases for User Registration and Profile Management		3
8	Execute Unit Testing and Validate User Profile Functionality	7	3
9	Develop Location-Based Search Functionality		8
10	Display Search Results with Links to Stores	9	4
11	Create Recipe Database Table		1
12	Develop Recipe Import Page	11	6
13	Develop a Manual Recipe Entry Page	11	4
14	Implement Recipe Transformation Algorithm	12, 13	4
15	Display Transformed Recipes with Nutritional Information	14	4

8 References:

National Health and Medical Research Council 2013, *Australian Dietary Guidelines*, viewed <https://www.eatforhealth.gov.au/sites/default/files/2022-09/n55_australian_dietary_guidelines.pdf>.

Sanwar Ranwa 2020, 'Create User Registration and Login Using Web API and ReactJS', *dzone.com*, DZone, viewed 17 October 2023, <https://dzone.com/articles/create-user-registration-and-login-using-web-api-a>.

Standards Australia/New Zealand, 2013, Systems and software engineering - Systems and software quality requirements and evaluation (SQuaRE) - System and software quality models, AS/NZS ISO/IEC 25010:2013, Sydney/Wellington: Standards Australia/New Zealand, viewed 18 August 2023.