Software Engineering
Software Requirements Specification (SRS) Document

NutriLog: Smart Nutrition

February 12, 2024

0.1.0

By: Carlos Villarreal, Michael DeHaan, and Salvador Macias

In Adherence to the UNCG Academic Integrity Policy

Table of Contents

1. Introduction

1.1. Purpose

[The goal of your project and the objectives it wishes to accomplish]

The goal of NutriLog: Smart Nutrition is to create a smart-food tracking application that allows users to see information about the foods they consume to provide guidance on nutritional education. The app will advise users on the Nutritional values of the foods they consume so that users can make informed decisions on the foods they eat, ultimately leading to better health outcomes.

1.2. Document Conventions

[Full description of the main objectives of this document in the context of your project.
Here's how you should begin this section:
"The purpose of this Software Requirements Document (SRD) is to..."
"In it, we will . . ., . . ., and . . .."]

The purpose of this Software Requirements Document (SDR) is to describe the client-view and developer view requirements for the Nutrilog: Smart Nutrition web application. In it, we will describe the system to a user and to a software developer. Client-oriented requirements will detail the system from the perspective of a regular user. The regular users of the system will be the administrator, the nutrition specialist, and the common user (i.e. adults or children). The requirements also include a description of the users of the system, such as adults or children, nutrition specialists, and the administrator. Developer-oriented requirements describe the system from the perspective of a software developer. The software requirements include detailed information about functional, data, performance, and other necessary requirements.

1.3. Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.
For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.

API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build NutriLog.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
CSS	Cascading Style Sheets. This is the code that will be used to style the content of the web application.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
IntelliJ IDEA	An integrated development environment (IDE) for Java. This is where our system will be created.
API	Application Programming Interface.
FDC API	An API that provides access from the USDA Global Branded Foods Database. This is where the core backend functionality will come from.

1.4. Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

The roles of stakeholders, developers, and project managers are all shared by the developers who created NutriLog: Carlos Villarreal, Michael DeHaan, and Salvador Macias. The user's role is that of anyone who wishes to use the service.

- Parts 1 and 2: Offer high-level overviews of the product that all roles may benefit from viewing.
- Part 3: Offers more detailed views for the developers and project managers.
- Parts 4 and 5: Offers views created based on user/customer specifications but within the SRS are meant for developers and project managers to implement these user-defined high-level requirements.

1.5. Project Scope

[Specify how the software goals align with the overall business goals and outline the benefits of the project to business.]

The goal of the software is to provide an easy-to-use interface for all customers, employees, and managers of a restaurant, as well as provide customers with flexibility to meet their needs. This aligns with the overall business goals of a restaurant as a restaurant requires fast and efficient service in order to fulfill the needs of its customers.

The benefits of the project to business include:

- Relieving stress and pressure from employees and managers as customers are given the opportunity to request services when needed.
- Increasing pleasure to customers as they are given more power when they want to order rather than having to wait for an employee to ask for their order.
- Reducing the amount of time that a customer needs to wait; therefore, increasing the amount of customers that are able to be served in the restaurant within a day.

The goal of NutriLog is to offer individuals a comprehensive and easy-to-use platform to take control of their daily nutrition. This aligns with the goal of promoting health and well-being among users by enabling them to make informed food choices based on their personal goals.

The benefits of the project to individuals include:

- Enhancing food safety by flagging recipes that contain foods a user may be allergic to.
- Promoting overall well-being by ensuring users have a more nutritionally complete diet.
- Improving nutritional education by providing specialist services to users who need additional assistance.

1.6. Technology Challenges

[Any technological constraints that the project will be under. Any new technologies you may need to use]

The challenges from this project will be from frontend implementation and API integration. Every developer team member is familiar with Java and backend logic but not UX and UI development. The new technologies will be SpringBoot and integrating the Spoonacular API with it.

1.7. References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

Academic integrity policy. Student Affairs. (2023, November 3). https://sa.uncg.edu/division-of-student-affairs/students/academic-resources/student-policy-handbook/academic-integrity-policy/

2. General Description

2.1. Product Features

[A high-level summary of the functions the software would perform and the features to be included.]

The product features include the ability for individual climbers and climbing gyms to create accounts and the ability for administrators to manipulate those accounts. Climbers can also add climbing routes to their profiles, where they can track their climbing progress, with different tracking options based on climbing

style. For gyms, the functionality also includes the ability to create climbing routes. They can also create events with a title and information. For administrators, the functionality also includes the possibility to view and delete accounts.

The product features allow users to find recipes and save them and flag recipes they search for any potential ingredients they may be allergic to. The users are also able to request for specialist assistance allowing them to obtain help in creating a more healthy and balanced diet. The specialist can then create a meal plan based on the specific nutritional needs of the user. The admin can create new accounts and delete ones, generate reports, and view metadata.

2.2. User Class and Characteristics

[A categorization and profiling of the users the software is intended for and their classification into different user classes]

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser, or knowledge of astronomy. Our website application has removed the need for them to have astronomy, math, or science knowledge and allows the user to focus on exploring the night sky.

Our application only expects our users to have prior knowledge of computers from using a web browser. NutriLog is intended for users with any background in nutrition, ranging from no prior nutritional knowledge to expert knowledge. A basic understanding of dietary components such as micro and macronutrients may be beneficial but is optional.

2.3. Operating Environment

[Specification of the environment the software is being designed to operate in.]
The application is designed to operate on the web across many different devices.

This application is designed to operate on any modern web browser regardless of the device.

2.4. Constraints

[Any limiting factors that would pose a challenge to the development of the software. These include both design as well as implementation constraints.]

Due to the use of a 3d engine, we had to limit the web browsers supported. To limit user error when entering the user's address, we implemented a drop-down AJAX country, state, and city selection.

Due to safety considerations and project scope, contact between specialists and users will not be instant.

2.5. Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans. The application will also use the World Time API (http://worldtimeapi.org/) that will display the current date and time on the home dashboard for everyone to see.

The software will be dependent on Spring Boot, HTML, CSS, and Java. The software will also use the FDC API for searching and displaying recipes, and food items. The software will be developed within IntelliJ IDEA Ultimate.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

3.1. Primary

[All the requirements within the system or sub-system in order to determine the output that the software is expected to give in relation to the given input. These consist of the design requirements, graphics requirements, operating system requirements and constraints if any.]

- FR0: The system will allow the user to lookup of vehicle owner information based on license plate number. This information will contain owner's permit number, assigned lot, and previous violations including tow history.
- FR1: The system will allow the user to enter a new vehicle into the vehicle violation database.
- FR2: The system will allow the user to issue a ticket. The ticket information will be issued in electronic and paper form.

- FR0: The system will allow the user to enter food and food ingredients that they are allergic to. This information will then be used to create a list of allergy information for that particular user that will then be used to make recommendations for recipes and foods to try out.
- FR1: The system will allow the user to request specialist assistance. This will alert specialists that a user is requesting assistance.
- FR2: The system will allow a specialist to create a nutrition plan for a user based on what information a user gives for their specific dietary needs.
- FR3: The system will allow an administrator to generate reports.

3.2. Secondary

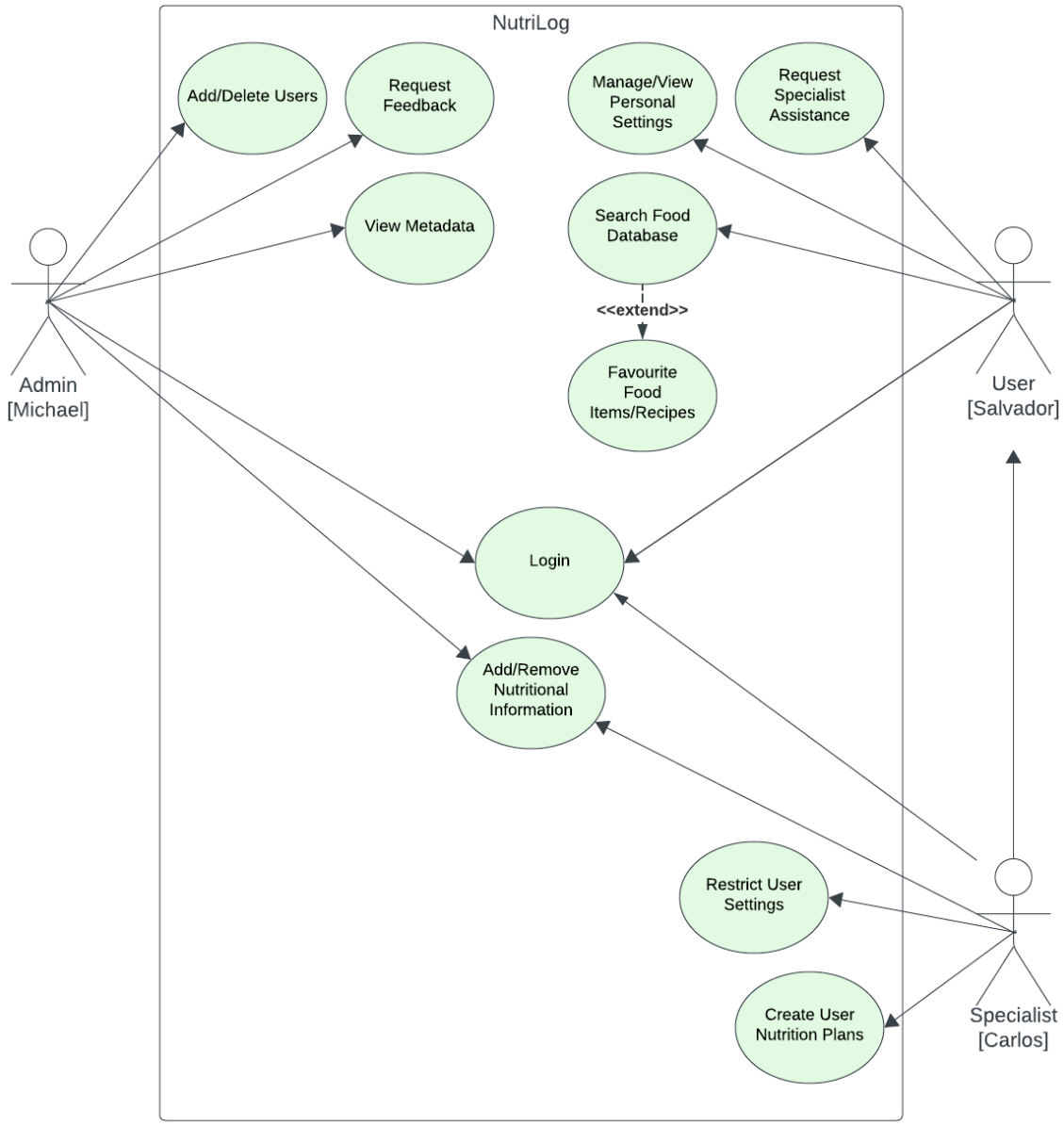
[Some functions that are used to support the primary requirements.]

- Password protection for information only accessible to employees, managers, and each individual table.
- Authorization scheme so that customers can only alter and see their orders and no other customers' orders.

- All 3 different users will only be able to view their own information, and information suitable to the type of user they are.
- All users will have a username and password they can use to login to the system.

3.3. Use-Case Model

3.3.1. Use-Case Model Diagram



3.3.2. Use-Case Model Descriptions

3.3.2.1. Actor: Manager (Alice)

- **Create Sprint**: The manager can create a new Sprint, by filling in a new sprint form and inputting dates, title, and details of the sprint.
- **Assign Task**: The manager can assign any existing to any of the employees in the team.

3.3.2.2. Actor: Admin (Michael DeHaan)

- **Delete User**: The manager can delete an existing user by using their user login ID.
- **Add New User**: The manager can create a new user and designate the type of user.

- **View Metadata**: The manager can view analytic data about users to see how they are using the app.

3.3.2.3. Actor: Specialist (Carlos Villarrea)

- **Create User Nutrition Plans**: The specialist will be able to create a Nutrition plan for a user who requests a nutrition plan.
- **Restrict User Settings**: The specialist can restrict the settings of a user based on certain criteria, such as their age. Children users will have less features than adult users.
- **Search/Save Recipe and Food Items:** The specialist will have the ability to search and save recipes that they find so that they can later make recommendations to users. The same ability will be granted to food items.
- **Add/Remove Nutritional Information:** The specialist will be able to add or remove nutritional information on the system. Nutritional information can be articles and posts talking about nutrition.

3.3.2.4. Actor: User (Salvador Macias)

- **Manage personal settings**: The user will be able to enter his name and allergy information along with any other relevant data to create a food plan.
- **Request special assistance**: The user will be able to ask for assistance from a specialist.
- **Search/Save Recipe and Food Items**: The user will be able to search for and save recipes that they would like to try based on the relevant information they gave such as allergies.

3.3.3. Use-Case Model Scenarios

3.3.3.1. Actor: Manager (Alice)

- **Use-Case Name**: Create Sprint
  - **Initial Assumption**: The manager has access to the webapp and has the manager role.
  - **Normal**: The manager will enter sprint details including start and end date, title, and details.
  - **What Can Go Wrong:** The manager selects a sprint date that is in the past. The system should be able to send an alert to notify the manager about this error.
  - **Other Activities**: Details can be left blank and added later.
  - **System State on Completion**: The new sprint is created and shows up on the sprint tab in the list of sprints.
- **Use-Case Name**: Assign Task
  - **Initial Assumption**: The manager has access to the webapp and has the manager role.
  - **Normal**: The manager selects the task to be assigned and selects the drop down of names in the assign-to field, then selects the intended employee to assign the task to.
  - **What Can Go Wrong**: n/a

- **Other Activities**: A task can be re-assigned to a different employee.
- **System State on Completion**: The task has an employee assigned to it. The task also shows on the employee's dashboard as assigned to them.

3.3.3.2. Actor: Admin (Michael DeHaan)

- **Use-Case Name**: Delete User
  - **Initial Assumption**: The admin has access to the app and the admin role.
  - **Normal**: The admin will enter the user's ID to a delete function.
  - **What Can Go Wrong**: The admin enters the wrong ID.
  - **Other Activities**: n/a
  - **System State on Completion**: The users will no longer be registered and the ID will be set as inactive. This will be reflected in the app metadata.
- **Use-Case Name**: Add New User
  - **Initial Assumption**: The admin has access to the app and the admin role.
  - **Normal**: The admin will enter the user's ID to a create function.
  - **What Can Go Wrong**: The admin enters an already existing ID.
  - **Other Activities**: n/a
  - **System State on Completion**: A new user will be established. The user ID will be added to the active ID list.
- **Use-Case Name**: View Metadata
  - **Initial Assumption**: The admin has access to the app and the admin role.
  - **Normal**: The admin will select a view to see various metadata about the app and its users.
  - **What Can Go Wrong**: n/a
  - **Other Activities**: The admin can create custom reports based on the metadata.
  - **System State on Completion**: No state change, view only.
- **Use-Case Name**: Add/Remove Nutritional Information
  - **Initial Assumption**: The admin has access to the web application and has the admin role.
  - **Normal**: The admin has the ability to search for and save articles and posts that talk about nutrition so that they appear on the system accessible to users.
  - **What Can Go Wrong**: n/a
  - **Other Activities**: (Placeholder, details can be added later)
  - **System State on Completion**: The saved articles and posts items are saved somewhere in the system so that they are accessible to users.
-

3.3.3.3. Actor: Specialist (Carlos Villarreal)

- **Use-Case Name**: Create User Nutrition Plans
  - **Initial Assumption**: The specialist has access to the web application and has the specialist role.
  - **Normal**: The specialist will be able to create custom nutrition plans for users based on the feedback users give for what their dietary needs are.

        - **What Can Go Wrong**: User's dietary needs are not given to the specialist because they did not fill out a form of some type to give to the specialist. The Specialist should have an "other" option.
          The user is unwilling to give health and dietary information necessary to create a custom nutrition plan. The specialist will then inform the user somehow that they are limited with the options they can give in a nutrition plan or that they are unable to create a suitable nutrition plan.
        - **Other Activities**: (Placeholder, details can be added later)
        - **System State on Completion**: The new nutrition plan is created and shown to the user somehow.
    - **Use-Case Name**: Restrict User Settings
        - **Initial Assumption**: The specialist has access to the web application and has the specialist role.
        - **Normal**: The specialist is able to restrict the settings that a user can have based on certain user information, such as their age.
        - **What Can Go Wrong**: The specialist accidentally restricts the setting of an adult user. The specialist should have the ability to remove restrictions.
        - **Other Activities**: (Placeholder, details can be added later)
        - **System State on Completion**: The child user no longer has access to certain settings.
    - **Use-Case Name**: Add/Remove Nutritional Information
        - **Initial Assumption**: The specialist has access to the web application and has the specialist role.
        - **Normal**: The specialist has the ability to search for and save articles and posts that talk about nutrition so that they appear on the system accessible to users.
        - **What Can Go Wrong**: n/a
        - **Other Activities**: (Placeholder, details can be added later)
        - **System State on Completion**: The saved articles and posts items are saved somewhere in the system so that they are accessible to users.

3.3.3.4. Actor: User (Salvador Macias)

- **Use-Case Name**: Manage personal settings
    - **Initial Assumption**: The user has access to the web browser and is logged in.
    - **Normal**: The user accesses and can modify his personal settings and information
    - **What Can Go Wrong**: The user accidentally enters incorrect information they will be able to edit and change their personal information.
    - **Other Activities**: (Placeholder, details can be added later)
    - **System State on Completion**: The user's information and details are saved in the system.
- **Use-Case Name**: Request special assistance
    - **Initial Assumption**: The user has access to the application and is logged in.
    - **Normal**: The user is able to request assistance from a specialist.

    - **What Can Go Wrong**: N/A
    - **Other Activities**: (Placeholder, details can be added later)
    - **System State on Completion**: The specialist is notified that they have been requested for specialist assistance.
- **Use-Case Name**: Search food database.
    - **Initial Assumption**: The user has access to the application and is logged in.
    - **Normal**: The user is able to search the database using the API.
    - **What Can Go Wrong**: No results for what the user searches up it would display no results found.
    - **Other Activities**: (Placeholder, details can be added later)
    - **System State on Completion**: Results from the API are displayed to the user.

4. Technical Requirements

4.1. Interface Requirements

4.1.1. User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

There will be three different interfaces based on the three user types: Admin, Specialist, and User. The app aims to be user-friendly and easily accessible. Upon logging in, each user will be directed to their homepage. Each page will have a navigation bar at the top of the viewport that can direct them to the subpages of their user type by using links and buttons. Input will be specified by using various HTML tags such as select, checkbox, etc. Styling will be done primarily through CSS and CSS templates.

4.1.2. Hardware Interfaces

[All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.]
The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

The app will run locally on any modern web browser that is connected to the internet and has the ability to interact with web page components. This includes but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.1.3. Communications Interfaces

[Determination of all the communication standards to be utilized by the software as a part of the project]
It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the World Time API and return the current date and time.

It must be connected to the internet to connect to the FDC API. Local communication between specialists and users does not require an internet connection, but they must be on the same local network or machine. Communication between users, the FDC API, and specialists will all be done via HTTP.

4.1.4. Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to the used, the database management system and libraries describing the need and the purpose behind each of them.]

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

We will use Spring Boot and ThymeLeaf to build the front end and Microsoft Excel for the backend database functionality. We will also use Spring Boot with Java to connect the front end to the back end.

5. Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

5.1. Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

- NFR0(R): The local copy of the vehicle violation database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the vehicle violation database) will consume less than 50MB of memory
- NFR2(R): The novice user will be able to create and print a ticket in less than 5 minutes.
- NFR3(R): The expert user will be able to create and print a ticket in less than 1 minute.

- The system should be able to work on all of the popular internet browsers, i.e. Chrome, Firefox, Edge, etc. (Memory requirements unknown as of now).
- A novice user may take anywhere between 5 and 10 minutes to manage dietary preferences and to save it to the system for the first time.
- An expert user may take 2-3 minutes to change their dietary preferences and save it to the system.

5.2. Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

Disclaimer: Nutrilog is not intended to be a service for people that may suffer from eating disorders to get guidance on their diet. Those that suffer from eating disorders should always consult a doctor or other trained medical professional.

5.3. Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

- NFR4(R): The system will only be usable by authorized users.

- The system will be usable by all intended users: administrators, specialists, and users with accounts.
- Users without accounts can still access the system, but they will not have the same number of functionalities that users with accounts have.
- The system will allow only the administrator to have access to personal user information, such as name, age, address, etc.
- The system will allow specialists to have access to certain user information, such as contact info.

5.4. Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

5.4.1. Availability

Available on all of the popular internet browsers. Requires internet connection to use.

5.4.2. Correctness

The software should accomplish its purpose and produce correct results under all expected conditions. Users should be able to look up a food database to find healthier foods and request assistance from a specialist whenever they need it.

5.4.3. Maintainability

The codebase should be well-organized, documented, and modular to facilitate ease of maintenance. This involves clear coding standards, comments, and adherence to best practices to allow for efficient updates and bug fixes.

5.4.4. Reusability

Components of the software should be designed in a modular fashion to promote reusability across different parts of the application or in future projects.

5.4.5. Portability

We intend for our software to be available on all web browsers, so anyone with a laptop or tablet that has access to the internet should be able to access the system.

5.5. Process Requirements

5.5.1. Development Process Used

Nutrilog: Smart Nutrition's development follows the Iterative Model. The process so far has been initial planning (Project Proposal) and then listing the requirements of the software in a document. Throughout the process, we expect to come back to certain aspects, whether it be requirements, analysis and design, etc, and to keep continuously improving/changing aspects of the software as we go along.

5.5.2. Time Constraints

All 3 software developers have other obligations besides this class that might hamper our time to work on the project. We have until April 30, 2024 to complete development of Nutrilog: Smart Nutrition.

5.5.3. Cost and Delivery Date

The cost of development is the cost of our tuition for taking CSC-340, so the price of 3 credit hours at UNCG is the cost of development. $955.13 for in-state, and $2,923.37 for out-of-state tuition for 3 credit hours.

Delivery date is April 30, 2024.

5.6. Other Requirements

(Placeholder, it's too early to tell whether the system will have other requirements).

6. Design Documents

6.1. Software Architecture

6.2. High-Level Database Schema

6.3. Software Design

6.3.1. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.2. State Machine Diagram: Actor Name (Responsible Team Member)

6.3.3. State Machine Diagram: Actor Name (Responsible Team Member)

6.4. UML Class Diagram

7. Scenario

7.1. Brief Written Scenario with Screenshots