Software Engineering
Software Requirements Specification (SRS) Document

ScanSafe

Sep 21 2023

Version 1.0

By: Caleb Youngdahl, Robby Martin, Jackson Astraikis

WE HAVE ABIDED BY THE UNCG *Academic Integrity Policy* ON THIS ASSIGNMENT.

Student's Signatures Caleb Youngdahl, Robby Martin, Jackson Astraikis Date September 26, 2023

Table of Contents

1. Introduction

1.1. Purpose

The goal of the ScanSafe application is to allow consumers to easily check consumable products for potentially harmful additives, chemicals, or allergens by simply scanning a barcode. This will allow them to make informed decisions on what to eat all while allowing them to shop at stores that fit their budget. Scansafe also aims to educate and share knowledge by allowing specialized users to create content for other users and allow them to suggest ingredients that should be added to the watch list.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the developer-view requirements for the ScanSafe (SS) application. In it, we will introduce and describe the features of ScanSafe along with the goals we wish to achieve for the software's intended audience. Furthermore, the SRD will describe the functional, non-functional, and technical requirements with the intent of collaboration between project members and to set the standards to which to hold each other accountable.

1.3. Definitions, Acronyms, and Abbreviations

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

API	Application programming interface. We will be using Open Food Facts API to obtain and display ingredients.
Java	An object oriented programming language. We will be using this language to build ScanSafe.
MongoDB	Open-source non-relational document database management system.
React	A JavaScript library for building user interfaces. This will be used for building the frontend.
SpringBoot	This will be used to create and run ScanSafe.

1.4. Intended Audience

Part 1: Introduction - intended for all users, developers, and project managers.
Part 2: General Description – intended for all users, developers, and project managers.
Part 3: Functional Requirements - intended for all users, developers, and project managers.
Part 4: Technical requirements – intended for developers and project managers.
Part 5: Non-functional requirements – intended for developers and project managers.
Part 6: Design Documents – intended for developers and project managers.
Part 7: Scenario – intended for project managers.

1.5. Project Scope

The goal of the software is to provide experience in the software development cycle for the members of this project group. We will gain knowledge about the processes and models of software development which aligns with our collective collegiate goals

The benefits of this project include:

- Experience collaborating in a team setting.
- Gain knowledge about software development models.
- Practice problem solving.
- Become proficient in developing software from start to finish.

1.6. Technology Challenges

1.7. References

2. General Description

2.1. Product Perspective

The idea for ScanSafe came from learning that Whole Foods bans a number of different ingredients in their stores which allows their shoppers to feel confident that any product they buy will meet a baseline standard. Because many other grocery stores do not operate this way, ScanSafe is a way to give shoppers the same confidence in the products they buy regardless of what stores they have access to.

2.2. Product Features

ScanSafe will allow a user to scan the barcode for a food product and receive a list of the ingredients with potentially harmful ingredients flagged. Clicking on these flagged ingredients will display links to reputable health sites explaining the potential risk. Users can also add ingredients to a personal list that will be checked alongside the main list of potentially harmful ingredients. Another set of users, called influencers, will be able to create curated lists of food products that other users can browse. Influencers will also be able to propose new ingredients to be added to the list of potentially harmful ingredients being checked. Admins will be able to approve or deny these ingredient proposals as well as delete curated product lists. They will also be able to delete or update users.

2.3. User Class and Characteristics

Base users will only need a basic understanding of using a computer and a web browser. Influencer accounts will only be available to people deemed knowledgeable (at the discretion of site admins) about health and wellness.

2.4. Operating Environment

ScanSafe will be a web application capable of operating on any device with a web browser and a camera.

2.5. Constraints

We may run into challenges with how to find user-specified ingredients in ingredient lists if they are not properly formatted.

2.6. Assumptions and Dependencies

This application will depend on Spring Boot for the REST API, and React for the frontend. It will also rely on the `react-qr-barcode-scanner` library for scanning the barcodes. We will be using the Open Food Facts API for getting the ingredient lists.

3. Functional Requirements

3.1. Primary

- FR0: A base user can scan a barcode and receive a list of the ingredients with potentially harmful ingredients flagged.
- FR1: Flagged ingredients will have links to reputable health sites explaining their risks.
- FR2: A base user can add an ingredient to their personal list that will be checked along with the main list created by the admins.
- FR3: Influencers can create curated lists of products that can be viewed by other users.
- FR4: Influencers can propose new ingredients to be added to the main list of checked ingredients.
- FR5: Admins can approve or deny ingredient proposals.
- FR6: Admins can delete curated product lists.
- FR7: Admins can delete or update base user and influencer accounts.

3.2. Secondary

- Base users can create password protected accounts.
- Influencers and admins can create password protected accounts only if they have an influencer or admin code, respectively.
- Access to certain API endpoints, like ones used to delete users, will only be available to users of the correct type.

4. Technical Requirements

4.1. Operating System and Compatibility

This application will be compatible with any operating system able to interact with web pages. Scanning barcodes will require access to a webcam.

4.2. Interface Requirements

4.2.1. User Interfaces

4.2.2. Hardware Interfaces

This application will be compatible with any hardware device with a webcam that can access the internet.

4.2.3. Communications Interfaces

The frontend and backend will communicate via HTTP. We plan to use Mongoose to interact with the MongoDB database.

4.2.4. Software Interfaces

We will use React to create the frontend and Spring Boot to build the backend API. Our database will be hosted on MongoDB and Mongoose will be used to communicate with this database.

5. Non-Functional Requirements

5.1. Performance Requirements

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.]

5.3. Security Requirements

- NFR1(R): The system will only be usable by authorized users.

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

[Details]

5.4.2. Correctness

[Details]

5.4.3. Maintainability

[Details]

5.4.4. Reusability

[Details]

5.4.5. Portability

[Details]

5.5. Process Requirements

5.5.1. Development Process Used

Agile Incremental Development Model

5.5.2. Time Constraints

- Prototype due Oct 3, 2023
- Design document due Nov 2, 2023
- Final completed project due Dec 5, 2023
- Final report due Dec 6, 2023

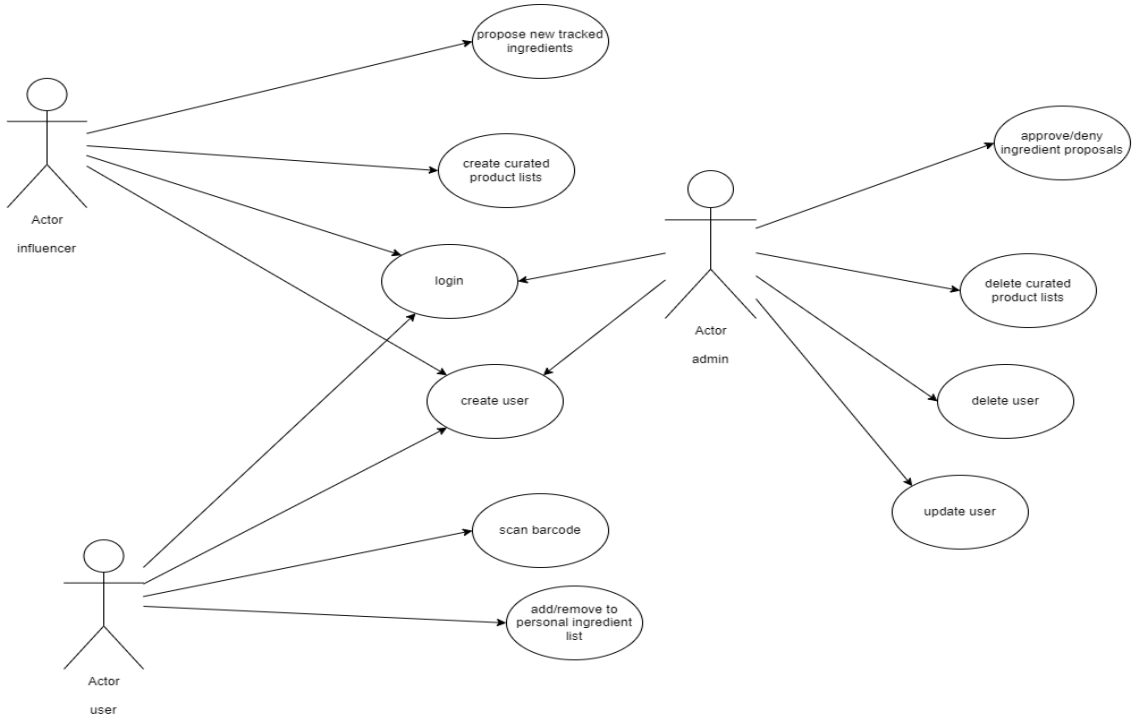
5.5.3. Cost and Delivery Date

- Delivery date: Dec 5, 2023

5.6. Other Requirements

TBD

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Base User (Jackson Astraikis)

- **Scan barcode**: Base users can scan a barcode and receive a list of the ingredients with potentially harmful ingredients flagged.
- **Add/remove to personal ingredient list**: Base users can add or remove ingredients to a personal list that is checked alongside the main list.

5.8.2. Actor: Influencer (Robby Martin)

- **Create curated product lists**: The influencer will be able to create lists with products from the database and post for other users to view. This list should allow for a title, brief description, and comments. After posting, they should be able to edit the list and delete if necessary.

- **Propose new tracked ingredients**: The influencer will be able to suggest ingredients for the application to track. They will be able to name the ingredient, give a description of the ingredient, as well as state why it should be tracked. They will also have the option to submit hyperlinks of supporting documentation to further their claim.

5.8.3. Actor: Admin (Caleb Youngdahl)

- **Approve/Deny Ingredient Proposals**: Admins will be able to view/edit lists of harmful ingredients and make changes to what is considered a harmful ingredient. Ingredients marked as potentially harmful will be included in the list of flagged ingredients when the base user scans
- **Delete Curated Product Lists**: Admins will be able to remove curated lists of ingredients that the influencer user has created, if for any reason it is determined that a certain list is harmful or should be removed for any reason.
- **Update User:** Admins can change and edit the access of Influencer and Base user accounts, base users can be changed to influencer access level and vice versa.
- **Delete User:** Admins are able to ban users from the app and delete the accounts of the other two users should a rules violation or harmful activity occur.

5.9. Use-Case Model Scenarios

5.9.1. Actor: Base User (Jackson Astraikis)

- **Use-Case Name**: Scan barcode
    - **Initial Assumption**: The user has a webcam.
    - **Normal**: The user scans the barcode and receives a list of the ingredients with potentially harmful ingredients flagged.
    - **What Can Go Wrong:** The product may not be found in the APIs database, the barcode scanner may fail to read the UPC, the user may not have allowed access to their webcam, the API may return the wrong product.
    - **Other Activities**:
    - **System State on Completion**: This action should not impact system state.
- **Use-Case Name**: Add/remove to personal ingredient list
    - **Initial Assumption**: The user inputs a valid ingredient.
    - **Normal**: The ingredient is added to the personal ingredient list successfully and in a format consistent with how the Open Food Facts API formats their ingredients.
    - **What Can Go Wrong**: Ingredient is not formatted correctly, saving ingredient to database fails, removing ingredient from database fails.
    - **Other Activities**:
    - **System State on Completion**: Ingredient is saved to the database.

5.9.2. Actor: Influencer (Robby Martin)

- **Use-Case Name**: Create curated product lists.
    - **Initial Assumption**: User is registered as an influencer.
    - **Normal**: The influencer can create a list from products in the database. The influencer can title the list, add/remove items, write a description, and publish the list.
    - **What Can Go Wrong**: Saving the list or publishing the list may fail. The products desired to add to the list may not be in the database.
    - **Other Activities**: The influencer can save draft of the list for later publishing.
    - **System State on Completion**: The list is saved to the database.

- **Use-Case Name**: Propose new tracked ingredients.
    - **Initial Assumption**: User is registered as an influencer.
    - **Normal**: The influencer completes a form that proposes a new ingredient be added to the watch list of harmful ingredients. The form is then submitted to the admin for approval.
    - **What Can Go Wrong**: The form may fail to submit.
    - **Other Activities**:
    - **System State on Completion**: This action should not impact the system state.

5.9.3. Actor: Actor Name (Caleb Youngdahl)

- **Use-Case Name**: Approve/Deny Ingredient proposals
    - **Initial Assumption**: Proposal is valid and received from Influencer.
    - **Normal**: A form is received from an influencer user, and the admin is able to either allow or disallow the ingredient to be added to the list of harmful ingredients.
    - **What Can Go Wrong**: When approved, the ingredient is not added to the list. The form may not be received. Form is submitted from a base user.
    - **Other Activities**: Automatic demotion button for erroneous submissions.
    - **System State on Completion**: Ingredient is added to database if approved, no change if denied.
- **Use-Case Name**: Delete Curated Product lists
    - **Initial Assumption**: A product list has been created from an Influencer user.
    - **Normal**: A curated product list can be viewed, reviewed and removed if it is determined to be harmful
    - **What Can Go Wrong**: Product list is not removed.
    - **Other Activities**:
    - **System State on Completion**: Curated product list is removed from the database.
- **Use-Case Name**: Update User
    - **Initial Assumption**: User(to be updated) is registered as either a Influencer or Base
    - **Normal**: A user's access is changed from Base to Influencer or vice versa
    - **What Can Go Wrong**: User's access is not updated
    - **Other Activities**:
    - **System State on Completion**: List of influencers/base users updated.
- **Use-Case Name**: Delete User
    - **Initial Assumption**: User(to be deleted) is registered as an Influencer or Base user
    - **Normal**: User to be deleted has Influencer or Base access, and is removed from the accounts database.
    - **What Can Go Wrong**: User is not deleted.
    - **Other Activities**: Set duration of removal.
    - **System State on Completion**: Database of users is updated.

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