Software Requirements Specification

[GrumpyChef]

[Version 1.0]

[Joseph Sluterbeck, Trevor Hollack, Braden Stromdahl]

[April 27, 2025]

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Revision 1	Apr 27	Final Design	1.0

1. Introduction

Purpose

*<This SRS describes the final design and functionality of **GrumpyChef**, a multi-user full-stack web application that allows users to inventory the ingredients in their home, create and share recipes, and generate shopping carts. The application enables users to sign up, log in, manage pantry items, create/edit/delete recipes and shopping carts. All data is user-specific and persists across sessions. This document covers **GrumpyChef** version 1.0.>*

Document Conventions

<Requirements in this document are organized by functional areas (user authentication, pantry management, recipe management, shopping cart management). Bold text is used to highlight key terms such as page names (profile page, shopping cart page, recipe page, etc.). Each requirement has been given equal priority unless otherwise specified.>

References

<Flask Framework documentation: https://flask.palletsprojects.com/>
<SQLAlchemy Documentation: https://docs.sqlalchemy.org/>
<HTML Standard: https://html.spec.whatwg.org/>
<CSS Standard: https://www.w3.org/Style/CSS/>
<JavaScript Reference: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

2. Overall Description

Product Perspective

*<**GrumpyChef** is a self-contained web application designed to help users manage their kitchen inventory, create recipes, and create shopping carts for groceries. This product is a new development, and not a replacement for any existing systems. It combines functionalities that assist users in organizing their pantry and meal planning. It is designed with simplicity and user-friendliness in mind while providing a full-stack web application with secure data persistence across multiple user sessions.>*

Product Features

*<**User Authentication:** Users can sign up, log in, and manage their profile with their personal information.*
***Pantry Management:** Users can view, add, edit, and delete pantry items, keeping track of the ingredients they have at home.*
***Recipe Management:** Users can view, create, edit, and delete recipes to simplify meal planning.*
***Shopping Cart Management:** Users can view, create, edit, and delete shopping carts to plan out the ingredients they need to purchase.*
***User Profile Page:** Displays the user's name, email, recipes, and shopping carts.*
***Persistent Data:** All data is stored in a database and persists between user logins.*
***Multi-User Platform:** Multiple users can create accounts, manage their own data, and interact with the application simultaneously.>*

User Classes and Characteristics

*<**General Users:***
***Frequency of Use:** Regular users who frequently manage their pantry, plan recipes, and create shopping carts.*
***Functionality:** Have access to all core features, including creating/editing recipes, pantry items, and shopping carts.*
***Technical Expertise:** Basic to intermediate technical understanding, as the system is designed to be user-friendly with minimal learning curve.*
New Users:
***Frequency of Use:** Infrequent users who may need assistance during initial setup.*
***Functionality:** Limited to account creation and the ability to explore available features once logged in.*
***Technical Expertise:** Advanced, requiring knowledge of the application's backend.>*

Operating Environment

*<**Hardware:** The web application is hosted locally on a computer but will eventually be accessible on all platforms.*
***Operating System:** The software is expected to run on devices such as Windows, macOS, and Linux operating systems.*
***Software:** The web application is developed using the following technologies:*
***Frontend:** HTML5, CSS3, JavaScript*
***Backend:** Flask (Python)*
***Database:** SQLAlchemy>*

Design and Implementation Constraints

*<**Technology Stack:** The application must be developed using the selected full-stack technologies: HTML, CSS, JavaScript, Flask, and SQLAlchemy.*
***Database:** The application is limited to the selected relational database for strong user-specific data. It may not utilize non-relational databases or third-party databases.*
***Security:** User authentication must be secure, ensuring that sensitive data like passwords are encrypted and stored safely using appropriate hashing algorithms.*
***Performance:** The application should load within an acceptable time for key pages.>*

Assumptions and Dependencies

*<**User Account Management:** It is assumed that all users will have access to the internet and a modern web browser. We also assume that users will not require an advanced knowledge of web applications.*
***Flask:** The application relies on Flask and its associated libraries. Any future updates to Flask or its dependencies may require corresponding updates to the system.*
***SQLAlchemy:** The application uses SQLAlchemy for database functionality. The system assumes that SQLAlchemy's object-relational mapping works as expected and does not require extensive custom configuration.*
***Security:** It is assumed that the application will be deployed in an environment with appropriate security practices.>*

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product.>

User Authentication (Sign Up, Login, Logout)

- ***3.1.1 Description and Priority***
- ***Description: The User Authentication feature allows users to create an account, log in to their profile, and securely log out of the application. User data such as name and email are stored in a database to maintain their session across logins.***
- ***Priority: High***
    - ***Benefit: 9***
    - ***Penalty: 9 (Critical, as users cannot access the app without authentication)***
    - ***Cost: 6 (Moderate development cost for implementing secure authentication)***
    - ***Risk: 8 (Risk of security vulnerabilities if not implemented correctly)***
- ***3.1.2 Stimulus/Response Sequences***
- ***Stimulus 1****: User clicks "Sign Up" on the landing page.*
  Response*: The system displays the sign-up form.*
- ***Stimulus 2****: User enters their email, name, and password, and clicks "Create Account."*
  Response*: The system validates the inputs and stores the data in the database.*
- ***Stimulus 4****: User enters credentials and clicks "Log In."*
  Response*: The system authenticates the user, retrieves user data from the database, and redirects them to their profile page.*
- ***Stimulus 5****: User clicks "Log Out" on the profile page.*
  Response*: The system clears the session and redirects the user to the login page.*
- ***3.1.3 Functional Requirements***
- ***REQ-1****: The system must allow users to create an account by entering a valid email, password, and name.*
- ***REQ-2****: The system must securely store user passwords using hashing.*
- ***REQ-3****: The system must allow users to log in with their email and password.*
- ***REQ-4****: The system must validate user credentials before granting access.*
- ***REQ-5****: The system must allow users to log out and clear their session.*
- ***REQ-6****: The system should display an error message for invalid login attempts (e.g., incorrect email or password).*

Pantry Management

- ***3.2.1 Description and Priority***

- ***Description****: The Pantry Management feature allows users to view, add, edit, and delete pantry items. Users can track the ingredients available in their kitchen and update quantities as needed.*
- ***Priority****: High*
    - ***Benefit****: 10*
    - ***Penalty****: 9*
    - ***Cost****: 5 (Low cost for implementation)*
    - ***Risk****: 6 (Moderate risk of incorrect data handling)*
- ***3.2.2 Stimulus/Response Sequences***
- ***Stimulus 1****: User clicks on the "Pantry" link in the navigation menu.*
  Response*: The system displays the list of pantry items with their quantities.*
- ***Stimulus 2****: User clicks "Add Item" and enters item name and quantity.*
  Response*: The system adds the item to the database and updates the pantry list.*
- ***Stimulus 3****: User clicks "Edit" next to a pantry item.*
  Response*: The system displays an editable form where the user can update the item name and quantity.*
- ***Stimulus 4****: User clicks "Delete" next to a pantry item.*
  Response*: The system removes the item from the pantry list.*
- ***3.2.3 Functional Requirements***
- ***REQ-7****: The system must allow users to view a list of their pantry items with corresponding quantities.*
- ***REQ-8****: The system must allow users to add new pantry items with a name and quantity.*
- ***REQ-9****: The system must allow users to edit existing pantry items (name and quantity).*
- ***REQ-10****: The system must allow users to delete pantry items with confirmation.*
- ***REQ-11****: The system must ensure that all pantry item data is stored and persists between user sessions.*
- ***REQ-12****: The system must handle invalid input.*

Recipe Management

- ***3.3.1 Description and Priority***
- ***Description****: The Recipe Management feature enables users to create, edit, and delete recipes. Users can also view their own recipe collection.*
- ***Priority****: High*
    - ***Benefit****: 10*
    - ***Penalty****: 9*
    - ***Cost****: 6*
    - ***Risk****: 5*
- ***3.3.2 Stimulus/Response Sequences***
- ***Stimulus 1****: User clicks on "Recipes" in the navigation menu.*
  Response*: The system displays the list of the user's recipes.*

- ***Stimulus 2****: User clicks "Create Recipe" and fills out the recipe form with ingredients, steps, and other details.*
  Response*: The system saves the recipe and displays it in the recipe list.*
- ***Stimulus 3****: User clicks "Edit" next to a recipe.*
  Response*: The system displays an editable form with the current recipe details.*
- ***Stimulus 4****: User clicks "Delete" next to a recipe.*
  Response*: The system deletes the recipe.*
- ***3.3.3 Functional Requirements***
- ***REQ-13****: The system must allow users to create, edit, and delete recipes.*
- ***REQ-14****: The system must allow users to view a list of their recipes.*
- ***REQ-15****: The system must store recipes with user-specific data, ensuring they persist between sessions.*
- ***REQ-16****: The system must ensure that recipes have a name, list of ingredients, and preparation steps.*
- ***REQ-17****: The system must display error messages if the recipe form is submitted with missing or invalid information.*

Shopping Cart Management

- ***3.4.1 Description and Priority***
- ***Description****: The Shopping Cart Management feature enables users to create, view, edit, and delete shopping carts. A shopping cart contains a list of ingredients that a user needs to purchase based on the recipes they wish to make.*
- ***Priority****: High*
  - ***Benefit****: 10*
  - ***Penalty****: 9*
  - ***Cost****: 6*
  - ***Risk****: 6*
- ***3.4.2 Stimulus/Response Sequences***
- ***Stimulus 1****: User clicks on "Carts" in the navigation menu.*
  Response*: The system displays the user's shopping carts.*
- ***Stimulus 2****: User clicks "Create Shopping Cart" and adds ingredients.*
  Response*: The system saves the cart and updates the list.*
- ***Stimulus 3****: User clicks "Edit" next to a shopping cart.*
  Response*: The system allows the user to modify the cart's contents.*
- ***Stimulus 4****: User clicks "Delete" next to a shopping cart.*
  Response*: The system deletes the cart.*
- ***3.4.3 Functional Requirements***
- ***REQ-18****: The system must allow users to create, edit, and delete shopping carts.*
- ***REQ-19****: The system must allow users to view their shopping carts and the items in each cart.*
- ***REQ-20****: The system must store shopping carts with user-specific data, ensuring persistence between sessions.*

- ***REQ-21****: The system must allow users to add ingredients to their shopping cart.*
- ***REQ-22****: The system must ensure that shopping cart data is accurate.*

4. External Interface Requirements

User Interfaces

*< **General Interface Characteristics**: The web application will provide a user-friendly interface to navigate between pages for sign up, login, profile, pantry, recipes, and shopping cart. All pages should be responsive and designed to work on desktop. Standard buttons such as “Sign Up,” “Log In,” “Log Out,” “Add Item,” and “Delete” will be consistent across all pages. Error messages should be displayed in a clear and user-friendly manner, such as "Invalid email or password". Every page should have a navigation bar with links to the profile page, pantry, recipe list, and shopping cart.*
***GUI Standards**: The interface will follow basic web accessibility guidelines, ensuring a simple color scheme, high contrast, and readable fonts.*
***Screen Layout Constraints**: The layout should avoid cluttering the page with excessive information. Each screen should have a focused purpose.*
***Standard Buttons and Functions**:*
***Login Page**: "Log In" button, “Sign Up” link.*
***Profile Page**: "Log Out" button.*
***Pantry Page**: "Add Item" button, "Edit" button, "Delete" button.*
***Recipe Page**: "Create Recipe" button, "Edit" button, "Delete" button.*
***Shopping Cart Page**: "Create Shopping Cart" button, "Edit" button, "Delete" button.*
>

Hardware Interfaces

*< **Supported Devices**: The system is designed for use on desktop. The web application does not have specific hardware dependencies beyond a device capable of running a modern web browser.*
***Device Types**: Desktop (Windows, macOS, Linux)*
***Data and Control Interactions**: The system does not interact directly with specific hardware components; the focus is on the software application accessed via web browsers.*
***Communication Protocols**: The web application communicates over HTTP/HTTPS protocols using Flask routes hosted locally via localhost:5000.*
>

Software Interfaces

*< **Databases**: **SQLAlchemy** will be used to interact with the database. The software uses SQLAlchemy models to manage data for users, pantry items, recipes, and shopping carts.*
***Libraries**:*
***Flask** (Python) - for routing and server-side logic.*
***SQLAlchemy** - for ORM-based database interaction.*
***External Software Components**: The system currently does not integrate with external APIs or services but may expand in the future to include third-party recipe APIs, or user authentication services.*
***Data Flow**: User inputs (e.g., login credentials, pantry item names) are sent from the front-end (browser) to the server (Flask) via HTTP POST requests. Server-side responses (e.g., success, error messages, updated data) are sent back to the front-end, and the page is re-rendered with updated content.*
>

Communications Interfaces

<***Communication Protocols**:*
***HTTP/HTTPS** - All data will be transmitted via standard HTTP/HTTPS protocols.*
*The application uses **Flask routes** to handle user requests (e.g., GET and POST for login, user data submission).*
***Web Browser**: The system is designed to run in modern web browsers. It will use Flask to serve the pages and handle the requests.*
***Message Formatting**:*
***JSON** format will be used for any potential future interactions between the front-end and server, especially when dealing with form submissions or API integrations.*
>

5. Other Nonfunctional Requirements

Performance Requirements

*<**Response Time**: The system should provide feedback to users within 2 seconds for standard interactions, such as navigating between pages or submitting forms. For operations involving database queries (e.g., retrieving user-specific recipes or pantry items), the response time should be no more than 5 seconds considering a typical user load.*
***Scalability**: As the number of users increases, the system should scale effectively. Although the current deployment uses a local server, future deployments may require hosting on cloud platforms which will allow for automatic scaling based on demand. The database should be optimized for handling growing data, particularly when users store many items in their pantry or recipes.*
***Concurrency**: The system should support multiple users concurrently accessing and modifying their accounts, recipes, pantry items, and shopping carts without significant degradation in performance. The server should be capable of handling at least 50 simultaneous users without performance issues.*
>

Safety Requirements

*<**Data Loss Prevention**: The system must ensure data persistence between sessions, such that user-generated data (e.g., recipes, pantry items, shopping carts) is stored securely in the database and not lost during server restarts. Regular database backups should be performed in production environments to prevent data loss.*
***Error Handling and Recovery**: The system must display clear error messages if unexpected failures occur, especially during critical actions like saving recipes or updating the shopping cart. Users should be informed if their changes fail, and the system should attempt to recover from the failure without losing data.*
***User Data Integrity**: Any critical user data, such as login credentials or shopping cart details, should be handled with integrity checks to prevent corruption or unintended modifications.*
>

Security Requirements

*<**User Authentication**: All users must authenticate before accessing any personalized features, such as creating or viewing recipes, pantry items, or shopping carts. Authentication will be achieved through a secure login process using a unique username/email and password combination. The password storage must be encrypted using a strong hashing algorithm to prevent unauthorized access.*
***Data Privacy and Protection**: User data (e.g., names, emails, recipes, pantry items) should be stored securely in the database with appropriate encryption to prevent unauthorized access. All user passwords should be hashed before storage in the database to ensure they are not retrievable in plaintext.*
>

6. Key Resource Requirements

<Identify the skill or expertise required for each major project activity. Identify any appropriate internal staff resources available. Indicate whether a resource only can commit part-time in the comments. If no internal resources are available, indicate the outside sources needed for the required skills. You may need more than one outside resource.>

Major Project Activities	Skill/Expertise Required	Internal Resource	External Resource
Frontend Development (HTML, CSS, JS)	Knowledge of web development technologies, responsive design	Team	None
Backend Development (Flask, Python)	Experience with Flask (Python web framework), SQLAlchemy database management	Team	None
Database design and management	Knowledge of relational databases, SQL, database design (SQLAlchemy)	Team	None
User authentication	Understanding of user authentication, secure password storage, session management	Team	None
UI design	Experience in developing user-friendly and visually appealing interfaces, knowledge of design principles	Team	None
Testing and quality assurance	Knowledge of manual testing and bug tracking	Team	None

Deployment	Knowledge of web application deployment (Flask)	Team	None
Project management	Project planning, task tracking, team coordination	None	None
Documentation	Technical writing, ability to document the architecture, features, and usage	Team	None