Software Requirements Specification

MasterChef Application

v1.0

Subiksha Vaidhyanathan
Kyle Cox
Emmanuel Gutierrez Rivera

April 30th, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Original	4/30/2025	Initial document creation.	v1.0

1. Introduction

Purpose

This document details the software requirements specification for the ***MasterChef v1.0.0*** application & website project.

Many people have a pantry with groceries and utilize recipes to cook their favorite meals. The ***MasterChef*** application is designed to simplify this process, by enhancing pantry tracking, and providing a platform for researching and finding recipes, as well as creating and sharing their own. Users have access to these features just by creating an account, allowing saving of their pantry and recipes to their user data.

The purpose of this project is to provide a platform for users to track their pantry, and create and share recipes, in a much more convenient way than what is currently available.

Document Conventions

The ***MasterChef*** application development process was undergoing during the creation of this document, so the requirements stated within this document will be already satisfied. Mentions of the ***MasterChef*** website name is bold and italicized for easier viewing.

Requirements in this document, as detailed in Section 3, will inherit the priority of their primary system feature. This is because these individual requirements are needed for the intended system feature to function correctly.

References

This SRS does not refer to any documentation or resources.

2. Overall Description

Product Perspective

This Software Requirements Specification (SRS) describes the functions and requirements of the ***MasterChef*** application. This product is a brand-new concept that does not build from any existing product family. This product utilizes a combination of Angular and Firebase to create the application and the database that contains all the user, pantry, and cart data.

Product Features

The ***MasterChef*** application serves as a resource for users to enhance their grocery shopping and recipe-making experiences. Users can create accounts to save their data. The application utilizes hashing algorithms through the Firebase database system to ensure that passwords are safe.

After creating an account, users can create recipes and share them with other users. These recipes can be browsed through by users and saved. Every user also has a "pantry" that contains all the ingredients that they currently have, which integrates with a recipe that they choose to identify what ingredients the user already has. Users can upload pictures of items that they put in their pantry, so they are more easily able to locate them. Any ingredients that the user does not have for a specific recipe can be put into a "cart" so the user knows what they need to purchase at a store to fulfill the recipe they want to create.

User Classes and Characteristics

This project is meant to be utilized by anyone who fits any one of these descriptions:

- General users who can use the software to cover their needs:
    - People of all age groups without much UI experience; the user interface is designed to be user friendly with simple functionalities set in place.
    - Users of any cooking skill can utilize the app for convenience. Those with less cooking skill can browse recipes. Those with more skill can create their own recipes and share them if they desire.
- Consistent users who use the software very often:
    - There is an extremely high limit on the number of recipes that can be created and downloaded; this allows long-term users to continue using the application without being limited.
    - The pantry on the software will change consistently, just like how a pantry would change in real life – it will see consistent use, if the user keeps it properly updated. Picture uploading will ensure that the user is able to physically locate the items that are in their virtual pantry.

Operating Environment

The ***MasterChef*** application utilizes Angular (version 19) for the purpose of website/application development and design. For the backend database, Angular will integrate with Firebase (version 11) to store information about users, ingredients in the pantry or cart, and recipes safely.

MasterChef is intended to run on any device with an application that can connect to a website on the Internet. It has been tested on:

- Microsoft Edge
- Google Chrome
- Firefox Browser

Design and Implementation Constraints

The ***MasterChef*** application front-end is written using Angular, which utilizes TypeScript, CSS, and HTML. This integrates with the Firebase database system that is used to operate most of the back end through functions that are written in TypeScript to save and load data, among other actions. Developers will need to be proficient with these tools to effectively update this application.

The website currently operates using localhost. If the website eventually runs on a server for anyone to connect to, security considerations will need to be made to ensure safety of data, including the Firebase's database keys.

Assumptions and Dependencies

As the application has been tested and ran utilizing applications on a computer, the use of a mobile device may result in an inaccurate or incorrect view due to the differing screen resolutions.

3. System Features

The ***MasterChef*** application contains many different features to enhance the user experience and allows for creativity and convenience. All features are contained within the application through different tabs, and any data changed by the user for their specific account, including their recipes, cart, and pantry, will be saved and be able to be reobtained upon logging back in.

3.1. Account Creation

3.1.1 Description and Priority

Priority: Very High
Volatility: Low

Users of the application can create their own accounts on a sign-in page. These accounts require their first and last name, a username, and a password upon creation. The password must follow certain requirements to ensure that a secure password is created by the user. Each user should only need to go through the account creation process once to ensure the best application experience.

3.1.2 Stimulus/Response Sequences

- User enters in valid account creation credentials; the account is created, stored in the Firebase database with an encrypted password, and is now able to be logged into by the user in the future. The account is ready to have new data that is tailored to it specifically.
- User enters an invalid password; the application tells them how to improve the password.
- User enters in an existing username; the application tells them the username is taken and chooses another one.
- User enters in blank account creation credentials; these are not accepted.

3.1.3 Functional Requirements

- Requirement 1.1: The application will have a sign-up page that allows users to create new accounts.
- Requirement 1.2: The application will only allow valid account information to be submitted.
- Requirement 1.3: Account information will consist of a first name, last name, username, and password.
- Requirement 1.4: The application will notify the user when they enter invalid account information.
- Requirement 1.5: The application will require users to have a password with, at minimum, one capital letter and one number.
- Requirement 1.6: The Firebase database will encrypt passwords that are entered in by users for new accounts.
- Requirement 1.7: The Firebase database will save new account credentials when a valid new account is created, allowing them to log into the account on the sign-in page.

3.2. User Login

3.2.1 Description and Priority

Priority: Very High
Volatility: Low

Users of the application will be able to log into their accounts through a log-in page, if they have already created an account in the process established in the previous feature overview. The user must enter their username and password, and once done so successfully, they will automatically access their account with all the data that is saved on it. This data includes their pantry, their cart, and their recipes, all of which will be defined in later feature overviews.

3.2.2 Stimulus/Response Sequences

- User enters in valid account credentials for an existing account; they are logged into their account and redirected to the user home page. They are now officially “logged in” to their account.
- User enters in an invalid username; the user is told that the username does not exist.
- User enters in a valid username but invalid password; the user is told that the password is incorrect.

3.2.3 Functional Requirements

- Requirement 2.1: The application will have a sign-in page that allows users to log into their already existing accounts.
- Requirement 2.2: The application will only accept valid account information to log a user into an account.
- Requirement 2.3: The application will inform users if they enter invalid information.
- Requirement 2.4: If a valid username is found, the Firebase database will encrypt the password and compare it with the already-encrypted password that is tied to the username in the database.
- Requirement 2.5: The application will redirect the user to the User Home Page on a successful log-in attempt.
- Requirement 2.6: The application will fetch all of the data for the specific account when logging in, including the information shown on the pantry, cart, and recipes pages.
- Requirement 2.7: The user will be redirected back to the login page if they are logged out while on an account-only page.

3.3. Pantry

3.3.1 Description and Priority

Priority: Very High
Volatility: Medium

Users of the application with an account will be able to add and remove items from their virtual “pantry.” This pantry, which can be accessed on a certain part of the website, represents the user’s actual pantry at home. Users can manually input data that describes an ingredient, including the amount, the unit type, and a picture if desired. This virtual pantry integrates with the other systems defined in later feature overviews and will be automatically removed from.

3.3.2 Stimulus/Response Sequences

- User logs in and goes to the Pantry page; their pantry items that are already there will show.

- User enters in a valid ingredient; it is added to their pantry, both on the database, and on the pantry list that they can see. This data is persistent. Pictures are optional.
- User enters in an ingredient with no name/no unit/no amount; the ingredient will not be added.
- User enters in an ingredient with a name and unit that is already in the pantry; the ingredient will be added correctly to the existing amount
- User deletes an ingredient in their pantry; the ingredient will be removed from the pantry, both on the database, and on the pantry list that they can see.

3.3.3 Functional Requirements

- Requirement 3.1: The application will have a pantry page where users can view and manage their pantry.
- Requirement 3.2: The application will allow users to add items to their pantry.
- Requirement 3.3: The application will allow users to remove items from their pantry.
- Requirement 3.4: The Firebase database should add or remove items from the pantry as these actions are done.
- Requirement 3.5: The application will allow users to specify item name, count, and unit.
- Requirement 3.6: The application will optionally allow users to include a picture with the item.
- Requirement 3.7: The units that users can specify will be selected in a dropdown menu.
- Requirement 3.8: The application and the Firebase database will be able to manage when a quantity of an item is added that already exists in the user's pantry, by merging them together.
- Requirement 3.9: The pantry will be able to interact with other parts of the application, particularly the Recipes page.

3.4. Cart

3.4.1 Description and Priority

Priority: High
Volatility: Medium

Users of the application with an account will be able to add and remove items from their virtual "cart." This cart operates very similarly to the pantry, but functions differently with the rest of the site's features. The cart will contain items that the user does not physically have – they need to instead go "buy" these items for the purpose of any recipes they are making. Therefore, the cart is automatically added to by other functionalities of the website – but the ability to manually add or remove items still exists if the user wants to make changes.

3.4.2 Stimulus/Response Sequences

The cart has a similar response sequence to the pantry. This is because they are quite similar. The primary difference is that the cart does not allow pictures.

- User logs in and goes to the Cart page; their cart items that are already there will show.
- User enters in a valid ingredient; it is added to their cart, both on the database, and on the cart list that they can see. This data is persistent.
- User enters in an ingredient with no name/no unit/no amount; the ingredient will not be added.
- User enters in an ingredient with a name and unit that is already in the cart; the ingredient will be added correctly to the existing amount
- User deletes an ingredient in their cart; the ingredient will be removed from the cart, both on the database, and on the cart list that they can see.

3.4.3 Functional Requirements

- Requirement 4.1: The application will have a cart page where users can view and manage their cart.

- Requirement 4.2: The application will allow users to add items to their cart.
- Requirement 4.3: The application will allow users to remove items from their cart.
- Requirement 4.4: The Firebase database should add or remove items from the cart as these actions are done.
- Requirement 4.5: The application will allow users to specify item name, count, and unit.
- Requirement 4.6: The units that users are able to specify will be selected in a dropdown menu.
- Requirement 4.7: The application and the Firebase database will be able to manage when a quantity of an item is added that already exists in the user's cart, by merging them together.
- Requirement 4.8: The cart will be able to interact with other parts of the application, particularly the Recipes page.

3.5. Recipe Creation

3.5.1 Description and Priority

Priority: High
Volatility: High

Users of the application with an account will be able to create their own recipes. These recipes contain a description of how to make the recipe, images of the recipe, as well as an ingredient list, with ingredients following the same format as what would be found in both the pantry and the cart. When a recipe is created by a user, it can take ingredients from the user's pantry and state that the user "has" these items already. Any ingredients that the user does not have can be automatically added to the user's cart.

3.5.2 Stimulus/Response Sequences

- User creates a valid new recipe in "Created" mode; saves the recipe to the user's account and deducts the ingredients from their pantry/cart.
- User creates a valid new recipe in "Saving" mode; saves the recipe to the user's account.
- User creates an invalid recipe (no ingredients, no images, no instructions); denies the creation of the recipe.

3.5.3 Functional Requirements

- Requirement 5.1: The application will have a recipes page where users will be able to view and create/delete their recipes.
- Requirement 5.2: The application will allow users to create their own recipes, consisting of instructions, ingredients, and pictures.
- Requirement 5.3: Recipes will be required to have one picture, but more can be added.
- Requirement 5.4: Recipes will be required to have one ingredient, but more can be added.
- Requirement 5.5: Recipes will be required to have instructions.
- Requirement 5.6: The application will have two methods of creating recipes: one mode where the ingredients are removed from the pantry and added to the cart, and another where they are not.
- Requirement 5.7: The Firebase database will save the recipe to the user's account when it is created.
- Requirement 5.8: The Recipes page will allow users to "make" their saved recipes.
- Requirement 5.9: A user making a recipe will deduct any items required for the recipe from the pantry.
- Requirement 5.10: A user making a recipe will add any items required for the recipe to the user's cart if it is not found in the pantry.
- Requirement 5.11: Recipes that are created are able to have a timer function associated with them, to allow whoever is making the recipe to time themselves.

3.6. Recipe Sharing

3.6.1 Description and Priority

Priority: Medium-Low
Volatility: High

Users of the application with an account and created recipes will be able to "share" those recipes with other users of the application. This connection can be established through a friend system. This will allow these users to make the recipes themselves, checking their pantry and adding them to their cart if necessary. Users who do not have created recipes can still utilize shared recipes for themselves. This allows users who are not as experienced with cooking to discover new recipes and make them.

3.6.2 Stimulus/Response Sequences

- User sends a friend request to another user; the other user can "accept" or "deny" this friend request.
- User chooses the "Share" option for their recipe; the recipe is now able to be shared and "saved" by other users, specifically their friends.
- User chooses a recipe to "Save;" the recipe is now saved to their recipes list.
- User chooses a recipe to "Create;" the recipe begins the creation process, deducting the user's pantry and adding to the cart.

3.6.3 Functional Requirements

- Requirement 6.1: The application will have an interface on the header that will allow users to send friend requests to other users.
- Requirement 6.2: The application will have an interface on the header that will allow users to accept or deny friend requests from other users.
- Requirement 6.3: The Firebase database will store users' friends information.
- Requirement 6.4: Establishing a friend connection will allow users to share recipes between each other if they choose to.
- Requirement 6.5: Users will be able to view the profiles of their friends.
- Requirement 6.6: Users will be able to share their recipes with their friends.
- Requirement 6.7: On the user home page, users will be able to accept or deny recipes that are shared with them.
- Requirement 6.8: Accepted shared recipes will be added to the user's own recipes page, allowing them to make the recipe.

4. External Interface Requirements

User Interfaces

The interfaces/GUIs on the website have a set color standard. Buttons and header text are dark orange colored. The background colors are a lighter orange color. Text boxes and tables are all black and white, as this is where data goes.

While a user is logged into an account, the header of the website has different symbols that lead to various parts of the website. Clicking the “Master Chef” text will send the user back to the user home page. The box symbol, book symbol, cart symbol, and profile picture symbol lead to the pantry, recipes, cart, and account pages or dropdowns, respectively.

This user interface is constructed through Angular and only requires the user to be able to run a website through some type of application (like Microsoft Edge or Google Chrome). Some types of user interface viewer, like a screen, are required to properly operate this software.

Hardware Interfaces

Any hardware that can run a website on a computer screen with a keyboard and mouse is able to reliably run the ***MasterChef*** application. Screen size is a limiting factor, as a smaller screen is not easily supported by some of the pages due to the positioning certain functionalities of the website.

Software Interfaces

The website runs from a local-hosted Angular program using the “ng serve” command on Windows. For future development, we hope to make the switch to supporting a HTTP-ran website. Information is communicated between Angular and a Firebase database through certain functions written in Typescript that communicates with Firebase.

Data transferred between Angular and Firebase makes up user account data, which includes their username, password (encrypted), first and last name, as well as the user’s pantry, cart, and recipes. This data is both sent to the Firebase database and is retrieved from Angular depending on the actions that are done on the website by a user. No user will be able to access another account’s data, except for recipes that are shared.

Communications Interfaces

A web browser and an Internet connection will be required to access the ***MasterChef*** application. This is so the user can connect to the website itself and allow it to interact with the Firebase database to log in and save/change data. The communication standard used to connect to the application will ideally be through the HTTP standard but is currently run locally for demonstration purposes.

5. Other Nonfunctional Requirements

Performance Requirements

The ***MasterChef*** application has no strict performance requirements on its' own but instead shares the performance requirements of the software the website is logged into from. The only true requirement is for a device to be able to properly run software that can load a website, such as Internet Explorer or Google Chrome.

Safety Requirements

There are not any requirements that would specifically lead to any losses or damage, mainly because the ***MasterChef*** application is a software product. However, it is important to ensure that no one can directly attack the Firebase database by entering in harmful data. Therefore, data entry points on the ***MasterChef*** website are checked to ensure that the data is valid – for example, entries that require a numerical response only accept those types of answers.

Security Requirements

Accounts consist of information containing a first and last name, username, and password (Requirement 1.3). Accounts also include information for the user's pantry, cart, and recipes page (Requirement 2.6). This will include ingredients and recipes but will also include images. These images may depict pictures of ingredients in the user's home, which could be location-defining depending on how the user takes these pictures (Requirements 3.6 & 5.3). Therefore, account security and privacy are especially important for users.
To authenticate into their account, users must provide the username and password. The password itself is encrypted in the database, to ensure that the account is secure on the backend of the application itself (Requirement 1.6).

6. Key Resource Requirements

Major Project Activities	Skill/Expertise Required	Internal Resource	External Resource
Account Creation	Angular & Firebase	Emmanuel Gutierrez Rivera	Documentation
User Login	Angular & Firebase	Emmanuel Gutierrez Rivera	Documentation
Pantry	Angular & Firebase	Subiksha Vaidhyanathan	Documentation
Cart	Angular & Firebase	Kyle Cox	Documentation
Recipe Creation	Angular & Firebase	Subiksha Vaidhyanathan	Documentation
Recipe Sharing	Angular & Firebase	Emmanuel Gutierrez Rivera	Documentation
Website	Angular	Kyle Cox, Subiksha Vaidhyanathan, Emmanuel Gutierrez Rivera	Documentation
Database Management	Firebase	Emmanuel Gutierrez Rivera	Documentation