SWE 3313

Software Requirements Specification Document

Mobile Cooking Companion

Team 6
Jeremy Ray, Thomas Roberts, Vitor Santos, Joshua Schoen, and Katlin Scott

Version: (4) **Date: (10/24/2021)**

1. Introduction

1.1 Purpose

This SRS clarifies the goals, software requirements, and constraints for the Mobile Cooking Companion to unify the work effort. Since this team does not have a Marketing or Sales Department, this SRS will be used by Developers, Design, and Management as a reference for the application.

Specifically, this SRS will define what the application can and cannot do to clarify what features the developer team needs to produce. Testers can use those clarifications regarding the features to test the application for faults. Also, the SRS will provide definitions and objectives to minimize confusion and reduce errors as the app is produced.

1.2 Scope

The Mobile Cooking Companion is an Android application that compares the user's current pantry to recipes in its database and returns the recipes that the user already has ingredients for. To narrow down results, the user has the option to use only a selection of ingredients in the pantry. The user is responsible for updating the pantry and entering recipes in the application.

The goal of the Mobile Cooking Companion is to cut down on food waste and help the user find recipes they can use with on-hand ingredients. As the application reduces food waste, it saves the user grocery money which benefits the user overall.

1.3 Definitions, Acronyms, and Abbreviations

MCC: Acronym for "Mobile Cooking Companion".
Pantry: A list of cooking ingredients that the user has on hand.
Recipes: A list of cooking recipes that the user has input.

1.4 References

Documents referenced within this SRS document:

- Project Plan, submitted October 3rd, 2021,
    - PDF found at: https://tinyurl.com/MCC-Project-Docs

2. Specific Requirements

2.1 External Interfaces

Inputs: Pantry items: (Food, drink, and spice ingredients, i.e., ground beef, black beans, almond milk, chicken broth, etc.)
Outputs: Recipes from Cooking Mobile Companion database.

2.2 Functional Requirements

The system shall...

- Break down recipes into accurate measurements of ingredients for precision, i.e., ¼ cup of black beans to a hot pot.
- Display a list of recipes from a list of ingredients that are inputted by the user.
- Memorize recipes that are used the most by the user and prioritize recipes that have been frequently used before.

2.3 Non-Functional Requirements

2.3.1 Performance

The time it takes to search for a recipe that matches the ingredients the user has logged in their pantry shall be processed in less than 5 seconds when they are accessing their saved recipes.

2.3.2 Reliability

The factors that establish reliability are accurate input features that can repeatedly store the correct information with the same inputs. If the user constantly inputs the same pantry items, it should be expected that the cooking mobile companion app shows the same list of recipes consistently. The input of the user to create a new recipe will be checked and sanitized by the system to make sure that it is able to display them properly when using the recipe.

2.3.3 Availability

The factors of this app would be on-demand as it is an Android app and not something that runs 24/7 on the back end of a phone. If it crashes during cooking, the app will autosave the last recipe accessed. When the user re-opens the app, the app will have features like “recover last recipe” like Microsoft products remember the last changes you made to your paper.

2.3.4 Security

So far, in the development of the Mobile Cooking Companion application, the data will be saved to a private database and most likely some kind of local file in which the program accesses, and each function of the app will only have access to what it needs to, such as the recipe database to the user’s pantry.

No other user will have access to a single user's pantry or recipe database unless the said user has given authorized access to said, other individuals. However, because this feature will likely not be implemented due to time constraints (as mentioned in our Project Plan), there will be no need for worry over remote access from a separate, unauthorized device.

2.3.5 Maintainability
The application will have two separate databases for the Pantry and Recipes and will allow for changes in one database's functionality with minimal impact on another as the database will call only one function from the other to retrieve what it needs. The application will also be built in an object-oriented approach, allowing for new functionality to be built upon the returns of functions of specific objects without having to change the underlying function.

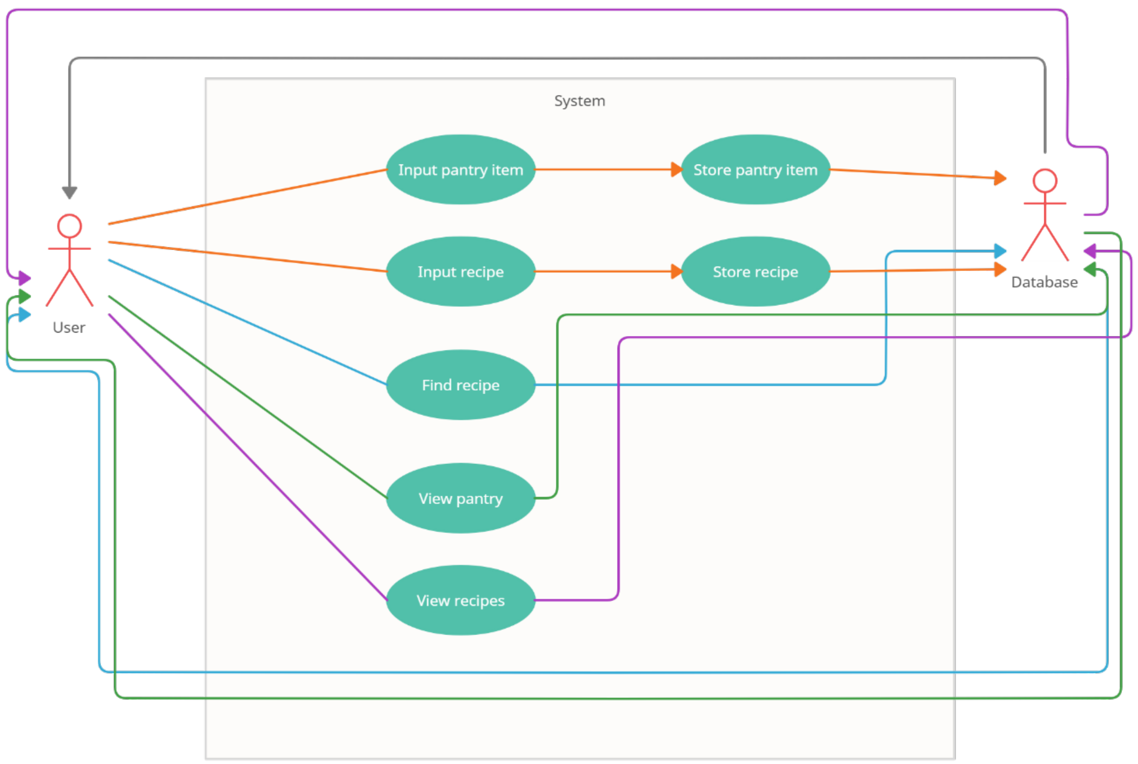
2.3.6 Portability
The application will be created to run on Android and as such will be coded in Java (or Kotlin if Java doesn't work). Due to the time limitations of the class, there will be no online functionality in the application, and everything will run locally and be dependent on the user's device.

2.4 Database Requirements
Every ingredient added to the pantry will have an amount associated with it. The recipes will involve the required ingredients and the multiple steps to make the dish. Each of the steps will involve whatever ingredient or appliance is required.

2.5 Design Constraints
Some design constraints that can be imposed are some hardware limitations on some Android devices. Newer Android devices could run the Mobile Cooking Companion without an issue, but older Android devices could have some trouble doing so. Because of the limited time to complete the project, there will more than likely not be too many extra features to the mobile application.

3. Use-Case Models

3.1 Use-Case Diagrams



3.2 Use-Case Descriptions

Requirements: The use-case must allow the user to input pantry items and recipe items as well as find said inputted recipes if requested.
Pre-conditions: The app must be open and fully launched before any of the following functions can be used or triggered. None of the use-cases take part without the application being actively opened.
Post-conditions: The app must be in a state to trigger another use-case if necessary or to be able to remain idle on the last screen.
Scenarios: (Example) A user wants to keep track of what they have in their pantry. The user inputs all items they have in their pantry (maybe one at a time or maybe as a sequence of items separated by commas) into the application for storage. They also want to input some family recipes into the application, which also stores for later use. Later the same day, say the user wants to see what they can easily make with what ingredients they already have, so they use the find recipes function of the application.

1. **Pantry Input**: User → Input pantry item → Store pantry item
2. **Recipe Input**: User → Input recipe → Store pantry
3. **Find Recipe**: User → Find Recipe → Stored pantry items → Stored recipes → User
    a. User → Find Recipe → Database → User
4. **View Pantry**: User → View Pantry → Database → User
5. **View Recipes**: User → View Recipes → Database → User