Software Requirements Specification

for

Diabetic Meal Planner

Version 2

December 11, 2024

Prepared by

Francisco Cruz-Urbanc

Krisi Hristova

Carson Ford

Jared Jackson

Thomas Capro

CI 491 – Senior Project

Filippos Vokolos

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for the Diabetic Meal Planner application, a mobile solution designed to assist individuals with diabetes in managing their nutritional intake. This document provides a detailed overview of the system's requirements, including functional and non-functional specifications, to guide the development team in creating an application that meets the needs of its target users and aligns with project objectives.

1.2 Intended Audience

This document is intended for use by developers, project managers, testers, and stakeholders involved in the Diabetic Meal Planner project. Developers will use it as a blueprint to build the app, testers will refer to it to validate that all requirements are met, and stakeholders will use it to ensure that the app aligns with their vision and requirements.

1.3 Overview

The Diabetic Meal Planner application will provide users with a simple, accessible way to browse a database of diabetes-friendly recipes, create meal plans, and generate grocery lists with cost estimates based on local stores. The application's primary goal is to empower users, particularly those from low-income, Spanish-speaking, and non-tech-savvy backgrounds, to make informed meal choices that support diabetes management. This document will cover the functional requirements, user characteristics, and constraints to ensure that the app's design and features align with user needs.

The rest of this document will follow the following outline:

1. **Introduction:** Provide an overview of the application and the structure of this document.
2. **Overall Description:** Provide constraints and factors that affect the application and its requirements.
3. **Specific Requirements:** Outline specific functional, non-functional, and external interface requirements.
4. **Appendix:** A single Appendix A consisting of a glossary of important terms used throughout the document.

1.4 Project Scope

The Diabetic Meal Planner application will offer a repository of diabetes-friendly recipes, with nutritional information for each recipe, allowing users to track carbohydrate intake. Users can create personalized meal plans and generate a weekly grocery list, including estimated costs at nearby stores. The app is designed to be user-friendly, with features specifically tailored to accommodate a non-tech-savvy, English and Spanish-speaking audiences. Future versions may expand to additional languages or incorporate further customizations based on user feedback.

2 Overall Description

2.1 Product Perspective

The Diabetic Meal Planner is a stand-alone mobile iOS application targeted to aid individuals with diabetes in effectively managing their dietary needs. The application integrates with several third-party APIs and cloud services to provide comprehensive features. The rest of this section will describe how the system operates inside various constraints.

1. System Interfaces

The application integrates with the following external interfaces for functionality:

1. Edamam Recipe API to supply data for diabetes friendly recipes including nutrient information in English and Spanish.
2. Map APIs to locate grocery stores based on nearby or given locations and to provide estimated price rankings.
3. Amazon EC2 to host a backend cloud server.
4. Amazon RDS (PostgresSQL) to host a PostgresSQL database to store user profile information, saved meal plans and preferences.
5. Amazon Cognito to manage user authentication and secure access.

2. User Interfaces

The application will have a simple and user-friendly interface that allows for language preference selection between English and Spanish.

3. Hardware Interfaces

The software intends to support Apple devices of iOS 18 or later such as iPhone and iPad. A Mac computer running on macOS 14.5 or later is required for development with Xcode 16.1 and Swift 6.0.2.

4. Communications Interfaces

The application uses HTTP protocols for secure transactions between client-side and server-side software. There shall also be real-time synchronization with cloud services when storing and retrieving user data.

5. Operations

a. The application utilizes user-initiated requests for the following:
    1. Create and manage accounts
    2. Set language preferences
    3. Search and favorite recipes
    4. Add recipes to weekly meal plans
    5. Edit auto-generated grocery lists from selected recipes
    6. Select location-based grocery stores

b. The application utilizes automatic processing for the following:
    1. Display filtered recipe searches,
    2. Find and display diabetes friendly recipes
    3. Generate recommended recipes from previously saved recipes
    4. Generate ingredient lists from recipes added to weekly plans.
    5. Synchronize new user data with the cloud and authenticate users when they try to log in.

6. Site Adaptation Requirements

The application requires that the database is preconfigured for user data storage of profile information, weekly meal plans, saved recipes, and ingredient lists.

2.2 Product Features

The key features of the Diabetic Meal Planner mobile application include the creation of individual user accounts, the display of diabetic friendly recipes, the saving of favorited recipes, the creation of weekly meal plans, the tracking of the total number of important macro and micronutrients of each meal and the search for location and price-range based grocery stores to purchase meal ingredients from. Smaller features include toggling language of the displayed text of the user interface and recipes between English and Spanish, filtering searches for specific food types, and choosing what ingredients users want to purchase based on their preferences.

2.3 User Characteristics

The intended primary users of the Diabetic Meal Planner application are diabetic patients seeking a way to make informed decisions about their diets to stay diabetes friendly and healthy. In addition, the application is tailored to support low-income and Spanish speaking individuals in the United States. There are three main characteristics that define the primary user: Spanish speaking, low-income, minimal technical experience.

1. **Low Income and Spanish Speaking**
    a. According to data from the U.S. Census Bureau seen in [1], Spanish was the top language spoken by low-income immigrants in 2019 making the requirements for enabling Spanish translations within the application a desired component.
2. **Low Technical Experience**
    a. Potential users with low technical experience will benefit from easy and clear navigation so design needs to be simple and effective.

2.4 Constraints

The software will be created using the Swift programming language for iOS development and requires a Mac computer running macOS 14.5 or later, which supports the latest version of Xcode. Currently the latest version is Xcode 16.1 which requires Swift 6.0.2. The application shall be developed for iOS 18 or later, which is supported by the aforementioned environment versions.

It is also recommended that an iOS device such as an iPhone or iPad are available for application testing. If this is not possible, simulators for testing included with Swift can be utilized but note that functionality may be slower when compared to testing on real devices.

The software will be constricted to using the following third-party APIs:

1. Edamam Recipe API for recipe and nutritional data.
2. Map API, with options including Google Maps Platform, Mapbox, or Amazon Location Services.

Hosting will utilize an Amazon EC2 instance on the us-east-1 region, with Amazon RDS (PostgresSQL) for database storage (maximum capacity: 200GB). Amazon Cognito will handle user authentication and authorization services.
Finally, a valid Apple Developer account, costing $99 annually, will be required to deploy the application on the Apple App Store.

2.5 Assumptions and Dependencies

The successful development of the Diabetic Meal Planner application depends on the following:

1. **Development Environment**
    a. Availability of compatible macOS, Xcode, and Swift versions.
2. **Third-Party API Accounts**
    a. The system assumes proper registration, payment, and access to API keys for the Edamam Recipe API and selected map API.
3. **IOS Device Compatibility**
    a. The application assumes compatibility with devices of iOS 18 or later for seamless operation.

3 Specific Requirements

The following subsections provide detailed requirements of the Diabetic Meal Planner application to specify what developers need to design and develop. The order of these requirements is:

- Functional requirements
- Non-functional requirements
- External Interfaces

3.1 Functional Requirements

The functional requirements are denoted the following priorities:

1. High
    a. Critical to the core functionality and goals of the application to perform and be successful.
2. Medium
    a. Important features of the system related to user experience and performance, but the system can run without them.
3. Low
    a. Features that would be nice to have for release but are not essential to the application’s goals.

3.1.1 User can create an account (REQ1)

Description: The user shall be able to create an account for the application so that their information is saved and they can come back to it later.
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. User has access to the internet.
2. User is using a device that supports the application.
3. The app server is online.

Action Sequence:

1. User enters application.
2. Application establishes a connection to server.
3. Application loads all necessary resources from the server.
4. Application establishes a connection to login database.

Postconditions:

1. Application is ready and able to read user's login.
2. Application validates login by reading from database.
3. Application saves user information.
4. User can logout, then log back in, and information will still be saved.

Dependency: None

3.1.2 User can receive information from APIs (REQ2)

Description: The user needs to be able to use certain features that access information from APIs (such as meal recipe recommendations).
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. The user's health information has been recorded in server.
2. User is on meal recipe/store location page.

Action Sequence

1. Server requests information from corresponding API.
2. API returns promise with the requested information.

Postconditions:

1. The application displays the information correctly to the user.

Dependency: REQ1

3.1.3 User can toggle Spanish mode (REQ3)

Description: The user shall be able to flip a toggle in settings to change language to Spanish and back to English.
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. The user is able to access the settings page.

Action Sequence:

1. User visits settings page.
2. Toggle is displayed to accurately represent the current language setting (English default).
3. User can tap toggle to flip setting between Spanish and English.

Postconditions:

1. All text within the application is translated to the chosen language.
2. This setting is saved to the user's account.

Dependency: REQ1

3.1.4 User can save meal recipes to account (REQ4)

Description: The user shall be able to save meal recipes recommended to them by the API to their account so they can come back to them later.
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. User visits suggested recipes page.

Action Sequence:

1. Button appears on recipe that saves the recipe to the user's account.
2. User presses button.

Postconditions:

1. User shall see the recipe in their saved recipes page.
2. User shall be able to logout, log back in and still see the saved recipe.

Dependency: REQ1, REQ2

3.1.5 User can plan meals for the week (REQ5)

Description: Users shall be able to plan their saved or suggested meal recipes on specific days of the week. This will keep meals organized and allow the user to plan their meals in advance.
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. User shall have some saved recipes and suggested recipes available to plan.
2. User visits weekly calendar page.

Action Sequence:

1. User selects a day and views saved recipes as well as suggested recipes.
2. User can select up to three of those recipes to plan for that day.

Postconditions:

1. Planned recipes shall be saved and be able to be revisited.
2. Planned recipe shall appear on main page for that day.

Dependency: REQ4

3.1.6 User can track daily nutrient count (REQ6)

Description: The user shall be able to track their daily carb, sugar, and fiber counts as diabetics use this information to manage their blood sugar levels. There shall be an active counter of that nutrient digested that day and an input field for the user to enter the amount of that nutrient they have taken.

Priority: **Medium**
Stimulus/Response Sequences:

Preconditions:

1. User is on main page.

Action Sequence:

1. User enters nutrient amount (in grams) into counter input field.
2. Counter increases and keeps a counter of what has been entered so far that day.
3. Counter keeps track of the date and resets the next day.

Postconditions:

1. Counter shall connect to server and request the date.
2. Counter shall use date information to reset accordingly.

Dependency: REQ1

3.1.7 User can set price range for suggested meal recipes (REQ7)

Description: The user shall be able to set a preferred price range for the meal recipes that are suggested to them.
Priority: **Medium**
Stimulus/Response Sequences:

Preconditions:

1. User is on suggested meal recipes page.

Action Sequence:

1. User inputs their preferred price range.

Postconditions:

1. Price range shall be saved to user's account.
2. Recipes that were already saved or suggested shall be untouched.
3. Currently suggested recipes shall refresh and suggest recipes only within the preferred price range.

Dependency: REQ4

3.1.8 User can find grocery stores on interactable map (REQ8)

Description: Our map API shall know the user's current location if they opt to share their location or a location can be entered manually. API shall suggest nearby grocery stores where users can buy the ingredients for their recipes. This map shall be interactable and should show each store's price range.
Priority: **High**
Stimulus/Response Sequences:

Preconditions:

1. Server is connected to map API.
2. User is on map page.

Action Sequence:

1. User's location is shown.
2. Pins are seen on nearby store locations.
3. User taps a pin to view more information on store.

Postconditions:

1. Store's information such as name, address, website, and price range is shown.

2. User can exit the store information and interact with the map.

Dependency: REQ1

3.1.9 User can input allergens that will be excluded in meal recipes (REQ9)

Description: The user shall be able to enter allergens into their account information so that foods containing them will not be included in recommended meal recipes searched for in the Edamam API.

Priority: **Medium**

Stimulus/Response Sequences:

Preconditions:

1. User has created account and is connected to server.
2. User is on account page.

Action Sequence:

1. User's location is shown.
2. Pins are seen on nearby store locations.
3. User taps a pin to view more information on store.

Postconditions:

1. Allergen is saved to account information.
2. When user goes to recommend meal recipes, recipes with account allergens are not included.
3. Saved recipes with this allergen are also not shown.

Dependency: REQ1, REQ2, REQ4

3.1.10 User can opt into getting notifications for meal recipes (REQ10)

Description: The user shall be able to toggle notifications on their device from the app that tell them what meals they planned for the day as well as random meal recommendations.

Priority: **Low**

Stimulus/Response Sequences:

Preconditions:

1. User is on settings page.

Action Sequence:

1. User can toggle notifications.
2. When turned on, notifications will pop up on the device at some time in the morning (local time) that tell the user what meals they have planned for that day.
3. Other notifications can also appear at random times of the day that suggest new meal recipes.

Postconditions:

1. Notifications toggle setting is saved to account.
2. Notifications appear as expected.

Dependency: REQ1, REQ2, REQ4, REQ5

References

[1] J. G. Rodriguez Valerie Lacarte, Joshua, “A Profile of Low-Income Immigrants in the United States,” *migrationpolicy.org*, Nov. 15, 2022. https://www.migrationpolicy.org/research/low-income-immigrants

3.2 Non-Functional Requirements

3.2.1 Performance

P1 The system will aim for crash-free rates by the industry standard, meaning that greater than 99% of all users and greater than 99.95% of all sessions will be crash-free.

P2 Regarding app launch time expectations, the system will target a cold launch time of 1.5-2 seconds, a warm launch time of less than 1 second, and a hot launch time of less than 0.5 seconds.

P3 Network response times encompass the full journey of a user request through the app, the app’s request to the relevant API, and the API’s response and relevant data being returned and displayed through the app to the user. This overall network response time will be less than 2 seconds.

P4 The execution of all modular logic of the app (e.g., fetch recipes based on search, save a recipe to favorites, etc.) should be completed in under 2 seconds on average.

P5 The UI response time will aim for an average of 250ms, to avoid any type of UI hang that would deliver a seemingly choppy performance to the user.

3.2.2 Security

S1 The system shall retain audit logs for all successful back-end operations.

S2 All API keys shall be stored securely and only accessible by the relevant developers.

S3 User authentication data, such as usernames and passwords, shall be encrypted and stored in a secure database that has restricted access to the relevant developers.

S4 Personal user data, such as saved recipes and meal plans, shall be stored in a secure database that has restricted access to the relevant developers.

S5 Modifications made to the main source code of the system shall require explicit approval from at least one other developer.

3.2.3 Robustness

R1 An error tracking system shall be in place.

R2 Previous versions of the system shall be tracked and commented in a version control system.

R3 Modifications made to the main source code of the system shall require a code-review and must include unit tests for the newly added updates and/or must pass all previously created unit tests.

R4 Database transactions shall guarantee data consistency in the case of failure.

3.2.4 Operating Environment

OE1 The system shall operate on at least as early as version 18 of iOS.

OE2 The system shall operate on a server running on AWS.

3.2.5 Scalability

Sca1 The system should be able to cope (without crashing) with a wide range of stress situations.
Sca2 The system should perform reasonably under important workloads (queries and data discovery).

3.2.6 Design and Implementation

DI1 The system shall be written in Swift.
DI2 The system should be built in a modular way, which ensures that component functionalities can be adjusted, added, improved, removed, or exchanged without impacting the application as a whole.
DI3 Design principles should be followed that will enable future growth both in terms of new functionality or through updating existing functionality.

3.2.7 Supportability

Sup1 A clear and standard practice for the maintenance and documentation of code should be set in place, agreed, and communicated as a best practice.

3.3 External Interface Requirements

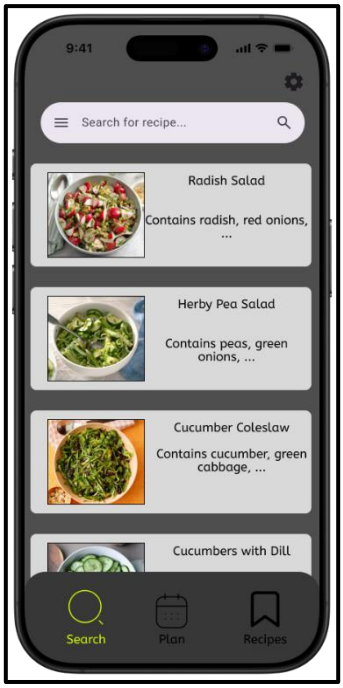
3.3.1 User Interfaces

1. **Home Screen**

The app will have a home screen where a user will be able to navigate to different screens to utilize the app's features (settings, recipe search, and more). If the user is not logged in, the app will display login text fields. If the user is logged in, the app will display the day's planned recipes by default when first launching the app.

2. **Search Screen**

The app will have a search screen where a user will be able to search for recipes. They will be able to apply different search filters to help them find the types of recipes they want to make (recipe search, ingredient search). When a search term or search filter is applied, the screen will populate with recipes that match the user's search criteria. The recipes will populate as a scrollable list.

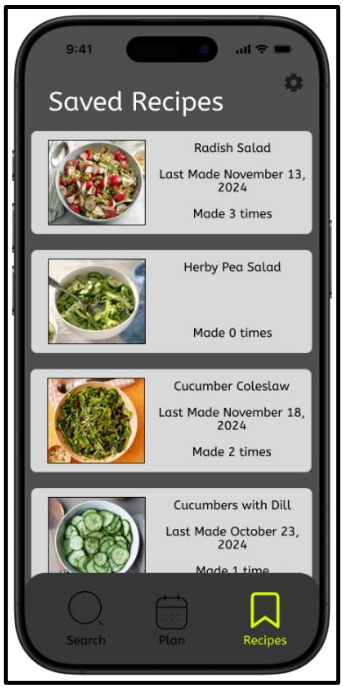


3. **Detailed Recipe Information Screen**

When the user finds a recipe on the search screen that they are interested in, they will be able to tap on it to view more information. Tapping on a recipe on the search screen will navigate the user to the detailed recipe information screen. On this screen, the user will be able to view the recipe's name, images, ingredients, nutrient quantities, and instructions on how to make the recipe.

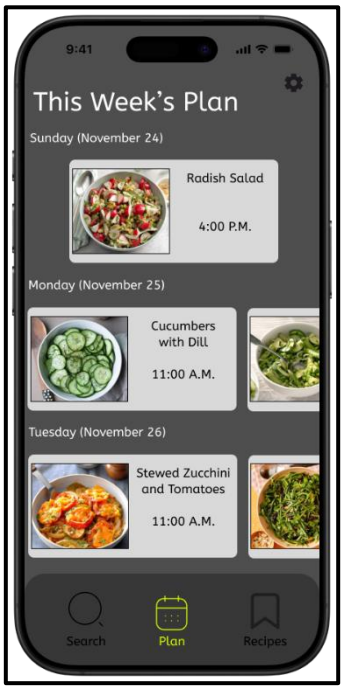
4. **Saved Recipes Screen**

The user will be able to save recipes they find on the search screen. This screen will display all of their saved recipes in the form of a scrollable list.



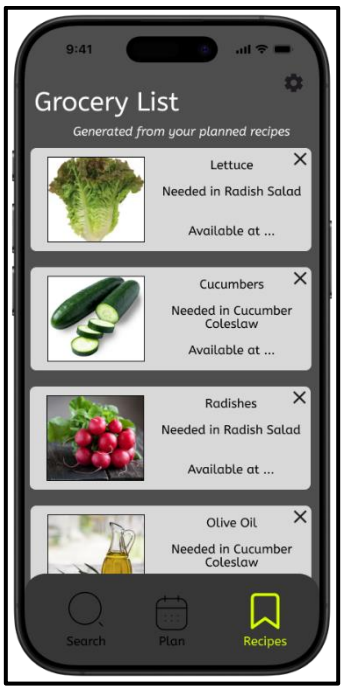
5. **Weekly Meal Planner Screen**

The user will be able to schedule the meals they would like to make in a week. The screen will consist of the current week's dates (from Sunday to Saturday). Each weekday will have multiple meal slots so a user can add a recipe and time to a slot. *Previous meal planner schedules will be archived and accessible by the user for 1-2 months after they were created.*



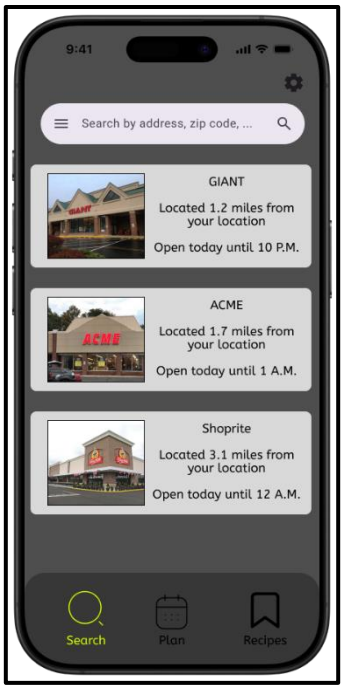
6. **Grocery List Generator Screen**

The app will be able to generate a grocery shopping list based on the user's scheduled meals for the week. On the screen, users can choose to add or remove specific food items based on if they already have specific items or want to use different items in the recipe.



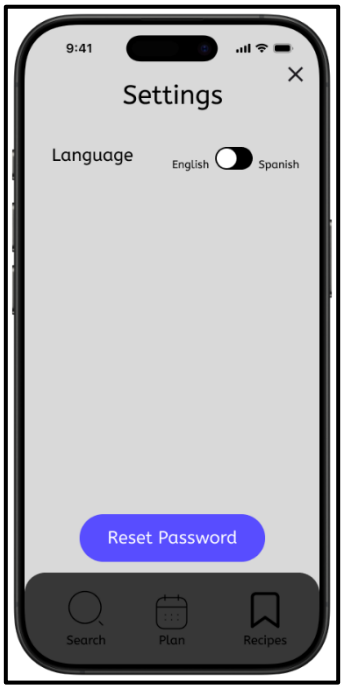
7. Grocery Store Map Screen

The user will be able to search for grocery stores near their current location or a specified address. They will initially be ranked from closest to farthest. To refine their search, users will be able to change the mile radius of their search to see more or less results. Also, users will be able to filter and rank the grocery stores in their area based on food item prices and availability.



8. Settings Screen

The user will be able to change app and account settings on this screen. One setting will be Language, which the user can choose either English or Spanish.



3.3.2 Hardware Interfaces

A user of our app needs to own a functional iOS device. Additionally, their device must be able to run the most recent version of iOS which is currently iOS 18.

3.3.3 Software Interfaces

The app will use Apple's built-in software to handle touch gestures, graphics, and network functionality.

The app will use a few different APIs for functionality.

1. Edamam Recipe API: for recipe data (image, ingredients, title), meal type and cuisine filters, diet/health/allergy filters, measure and quantity for ingredients, recipe caching (recipe id and name), shopping aisle for ingredients, and full nutritional details.
2. A Map API (Google Maps, Mapbox, or Amazon Location Services): for a list of nearby grocery stores, navigation directions, and food item prices. One or more of these APIs will be used in order for the app to access the information it needs.

The app will use a database system (either PostgresSQL or SQLite) to store references to saved recipes, shopping/ingredients lists, weekly meal plans, account information, and total daily carb counts.

Appendix A: Glossary

Here is a list of important terminology that is used throughout the document.

- **Amazon Cognito:** An Amazon AWS service for authentication and authorization of users using a created user directory and consumer identity providers like Google and Facebook.
- **Amazon EC2 – Amazon Elastic Compute Cloud:** An Amazon owned scalable computing capacity that allows for the launch and running of virtual servers.
- **Amazon RDS – Amazon Relational Database Service:** An Amazon service providing fully managed relational database that operates with open-source databases(PostgresSQL, MySQL, etc.) in the cloud.
- **Cold App Launch:** When an app is starting up from scratch; the app's process has not been created by the system until this point. This occurs when the app is opened for the first time since the system's booting, or if the app was killed and then reopened.
- **Hot App Launch:** When an app's process is fully running in the background; the system just has to bring the app's activity to the foreground.
- **Warm App Launch:** When an app is launched, and its main process is still running, but it needs to recreate some minor processes and retrieve data that was evicted from memory. This launch is somewhere in between a cold launch and a hot launch.