10/10/2018

Gavin Hanna

Higher diploma in science in web technologies

Requirements Specification

Recipe Book

# Requirements Specification (RS)

## Document Control

### Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Scope of Activity** | **Prepared** | **Reviewed** | **Approved** |
| 10/10/2018 | 1 | Create | AB |  |  |
| 21/10/2018 | 2 | Update | CD |  |  |

### Distribution List

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Version** |
| Liam McCabe | Lecturer |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Related Documents

|  |  |
| --- | --- |
| **Title** | **Comments** |
| Title of Use Case Model |  |
| Title of Use Case Description |  |

Table of Contents

[Requirements Specification (RS) 1](#_Toc527301228)

[Document Control 1](#_Toc527301229)

[Revision History 1](#_Toc527301230)

[Distribution List 1](#_Toc527301231)

[Related Documents 1](#_Toc527301232)

[1 Introduction 4](#_Toc527301233)

[1.1 Purpose 4](#_Toc527301234)

[1.2 Project Scope 4](#_Toc527301235)

[1.2.1 Motivation 4](#_Toc527301236)

[1.2.2 Project Objectives 5](#_Toc527301237)

[1.2.3 Project Expectations 5](#_Toc527301238)

[1.3 Definitions, Acronyms and Abbreviations 5](#_Toc527301239)

[2 User Requirements Definition 5](#_Toc527301240)

[3 Requirements Specification 6](#_Toc527301241)

[3.1 Functional Requirements 6](#_Toc527301242)

[3.1.1 Use Case Diagram 8](#_Toc527301243)

[3.1.2 Requirement 1: User Registration 9](#_Toc527301244)

[3.1.3 Requirement 2: User Login 10](#_Toc527301245)

[3.1.4 Requirement 3: Create Recipe 11](#_Toc527301246)

[3.1.5 Requirement 4: Update Recipe 12](#_Toc527301247)

[3.2 Non-Functional Requirements 14](#_Toc527301248)

[3.2.1 Performance/Response time requirement 14](#_Toc527301249)

[3.2.2 Availability requirement 14](#_Toc527301250)

[3.2.3 Reliability requirement 14](#_Toc527301251)

[3.2.3 Maintainability requirement 14](#_Toc527301252)

[4 GUI 15](#_Toc527301253)

[4.1 Landing Page 15](#_Toc527301254)

[4.2 Registration Page 16](#_Toc527301255)

[4.3 Login Page 16](#_Toc527301256)

[4.4 Profile Page 17](#_Toc527301257)

[4.5 Recipe Book Page 17](#_Toc527301258)

[4.6 Following Page 18](#_Toc527301259)

[4.7 New Recipe Page 19](#_Toc527301260)

[4.8 Recipe Page 20](#_Toc527301261)

[4.9 Add Comment – Detail 21](#_Toc527301262)

[5 System Architecture 22](#_Toc527301263)

[6 System Evolution 23](#_Toc527301264)

[7 References 24](#_Toc527301265)

# 1 Introduction

## Purpose

The purpose of this document is to document the requirements for the development of a recipe based social network web application, specifically designed to offer a space on the web for publishing self-made recipes to share and also to follow your favourite recipe creators and discuss the recipes themselves.

The application is intended to be used by people with an interest in recipes, either simply by browsing some interesting user created content or by sharing and discussing their own and other’s content.

Each user has a specific “Recipe Book” attached to that user, whereby they can save a collection of recipes that they enjoy. Users can add or remove recipes from this book at any time. Recipe Books can then be shared with and viewed by other users.

## Project Scope

The scope of this project is to design and develop a responsive progressive web application that allows users view user created recipe content and search by key words to view specific types of recipes. Users can also choose to register and create an account on the application which will allow them to create their own recipes, comment on their and others’ recipes and follow their favourite content creators.

Users will be able to directly link to a personal public profile which will display their personal recipes and link to the profiles of the creators they follow.

An unregistered user will be restricted from commenting on recipes, from following creators content and from saving any favourite recipes. However they will be able to view any recipe by searching or direct linking.

The application will be a PWA, a progressive web app, meaning that it will be able to be saved to the home screen of a mobile device and act in a similar fashion to a native application. It will have some level of offline capability, either by serving content from the in-browser database indexedDB or by rendering an “offline” page to let the user know there is no available connection.

The application will also be an SPA, a single page application, having the front end built with React, which will enable very responsive application navigation via the router. Data will be sent and received to a node/express back end application and updated in the client’s view via a Redux store utilized by React.

### Motivation

The main motivation for developing this application is to attempt to provide a social hub for people that enjoy sharing recipes, both their own and sampling other people’s recipes and meal ideas. A secondary motivation is to get a deeper understanding of the development of a full stack application with separate but communication front and back end applications, including adding the functionality associated with a PWA.

### Project Objectives

To create a functional and easy to use social network application for users to store and share their favourite meals, try other users meals and add their opinions. When the application has been completed it will be uploaded to Heroku to be used by the public. A feature could be added by which users could suggest improvements or submit bugs that may have slipped through the testing phase.

### Project Expectations

The project could be monetized should the application become popular enough to warrant upgrades to the hosted database plan.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Acronym | Definition |
| PWA | Progressive Web App (Kapoor, 2018), web application that can mimic the behaviours of native applications (offline capability, start app extremely fast from cached data). |
| SPA | Single Page Application is a website that re-renders its content in response to navigation actions (e.g. clicking a link) without making a request to the server to fetch new HTML (Sherman, 2018) |
| JS | The JavaScript programming language |
| CSS | Cascading Style Sheets |
| HTML | Hyper-Text Mark-up Language |
| URL | Uniform Resource Locater, essentially a web address |

# 2 User Requirements Definition

The system must allow unregistered users to browse recent recipes, search recipes by specific keywords and view recipe creator profile pages.

The system must offer an ever present link to the Register and Login pages in the navigation bar and on the landing page of the application.

Once a user has register they will be offered the chance to submit their own recipes, to be able to view their own personal profile page. Users will be able to find and follow other users and to comment on and like other users recipes. Each user can then add recipes that they enjoy to their very own “Recipe Book”, which can be shared with and viewed by other users.

Users should be able to edit their previously submitted comments or previously published recipes.

A mobile site visitor should be offered the option to add the application to their home screen, regardless of their devices operating system, which will act as if it was a native application on that device. If a user opens the application without an internet connection, they should be notified and offered a chance to reload in an attempt to reconnect.

The application will have a responsive navigation bar at the top of the screen to permit easy navigation of the application.

# 3 Requirements Specification

A simple, coherent and friendly user interface is the highest priority for the application. The applications interface should be self-explanatory and do exactly what the user would expect it to do, having used other social networks previously, or never having used them at all. Buttons and navigation should be clear and their destinations obvious.

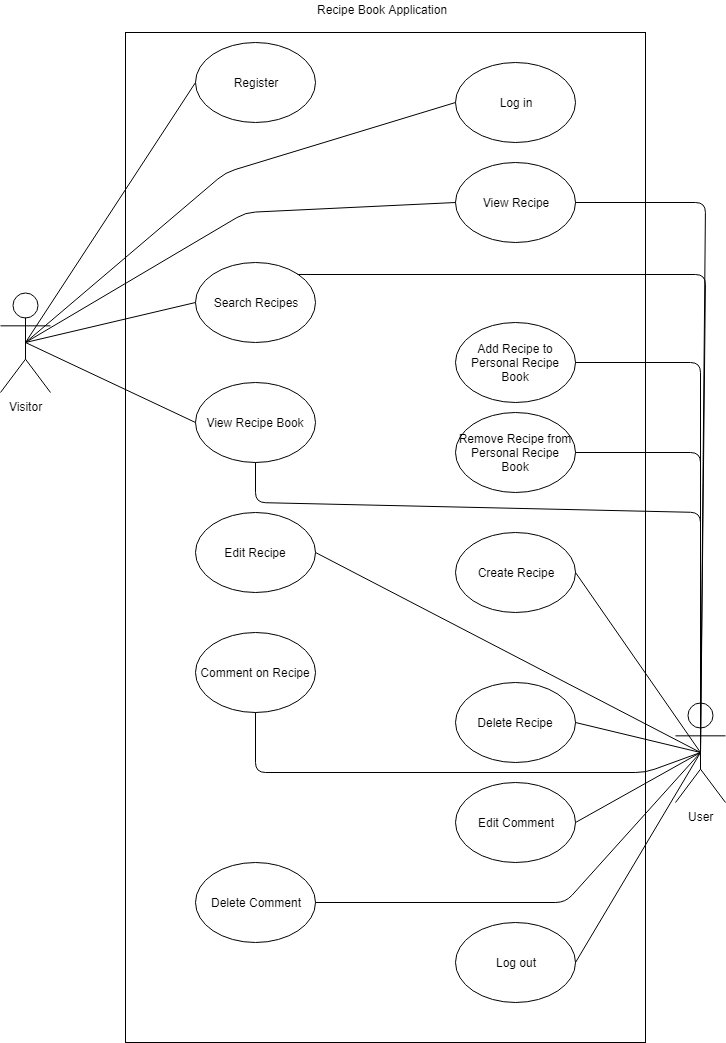
When a user sends or received data from the server side of the application, they should be updated as to the changes made in an understandable and non-invasive way.

## 3.1 Functional Requirements

The following functional requirements represent the minimum viable functionality by which the application could be considered usable.

* Upon reaching the home page, the content of the page should render correctly. Displaying information about the application and links to Register and Login functionality.
* Register and Login forms should be fully validated.
* Users should be able to view a selection of recipes without having to be logged in or registered.
* Users should be able to register for an account using a valid email address and password. The username should be anything they choose between 3 and 30 characters long.
* Logged in users should be able to view their personal “user” page, containing any previously submitted recipes.
* Logged in users should be able to comment on their and other users recipes.
* Logged in users should be able to create a new recipe, and edit any previously created recipe.
* Creating or updating recipes should have fully functional validation.
* Logged in users should be able to “follow” other users and see their recipes on the initial users homepage, or “feed”.
* Logged in users can add any viewable recipe to their own “Recipe Book”.
* Logged in users can remove recipes from their own “Recipe Book”.
* Logged in users can view other users Recipe Books.
* Once a user logs out, any information relating to the previously logged in user should be removed from the specific client side rendering of the application.

### 3.1.1 Use Case Diagram

The following Use Case Diagram represents functionality offered to both a non-logged in user (represented as Visitor) and a logged-in user (represented here as User).

### 3.1.2 Requirement 1: User Registration

#### 3.1.2.1 Description & Priority

A user can remain a “visitor” on the website, meaning they can peruse and search for recipes, but cannot create a profile, follow creators or comment on recipes until they register for an account. To unlock full functionality, a visitor must register and become a user.

#### 3.1.2.2 Use Case: Register

**Scope**

The scope of this use case is to allow a user to register an account with the application.

**Description**

This use case describes the steps to be taken in order to successfully create an account on the application.

**Flow Description**

**Precondition**

The system is idle

**Activation**

This use case starts when a user clicks on the “Register” link.

**Main Flow**

1. The user navigates to the “Register” page on the web application.
2. The user enters a username of between 3 and 30 characters.
3. The user enters a valid email address.
4. The user enters a valid password of between 6 and 30 characters.
5. The user confirms the original password.
6. The users clicks on the “Submit” button.
7. The system sends information to the backend server.
8. The server-side application validates the inputs. <E1>
9. The server-side application stores the users details in the database.
10. The server-side application responds with the registered users details (minus the password).
11. The system allows the user to log in to the website.

**Exceptional Flow**

E1: Inputs are invalid

1. The user inputs which are invalid are displayed in red, with a message indicating the reason for the error under each invalid input.
2. The user inputs correct and valid inputs in the form.
3. Return to step 7 of Main Flow.

**Termination**

The application reports a successful registration.

**Post Condition**

The system returns to a wait state.

### 3.1.3 Requirement 2: User Login

#### 3.1.3.1 Description & Priority

In order for a user to access their account and use it’s full functionality, its necessary to that user to login to their account using details used during registration.

#### 3.1.3.2 Use Case: Sign In

**Scope**

The scope of this use case is to allow a user to enter into their personal account on the application.

**Description**

This use case describes the steps to be taken in order to successfully log in to their personal account on the application.

**Flow Description**

#### **Precondition**

The system is idle

**Activation**

This use case starts when a user clicks on the “Login” link.

**Main Flow**

1. The user navigates to the “Login” page on the web application.
2. The user enters their username of between 3 and 30 characters.
3. The user enters their specific email address.
4. The user enters their specific password of between 6 and 30 characters.
5. The users clicks on the “Submit” button.
6. The system sends information to the backend server.
7. The server-side application validates the inputs. <E1>
8. The server-side application responds with the registered users details (minus the password).
9. The system logs the user in to the web application.
10. The system displays the users profile page.

**Exceptional Flow**

E1: Inputs are invalid

1. The user inputs which are invalid are displayed in red, with a message indicating the reason for the error under each invalid input.
2. The user inputs correct and valid inputs in the form.
3. Return to step 7 of Main Flow.

**Termination**

The application reports a successful login and shows the user profile page.

**Post Condition**

The system returns to a wait state.

### 3.1.4 Requirement 3: Create Recipe

#### 3.1.4.1 Description & Priority

In order for a user to publish her/his own recipe, they must first go through the process of submitting validated information via the Create Recipe functionality of the web application.

#### 3.1.4.2 Use Case: Create Recipe

**Scope**

The scope of this use case is to allow a user to submit a new recipe in the form of form data in the web application.

**Description**

This use case describes the steps to be taken in order to submit a new recipe to the web application.

**Flow Description**

**Precondition**

The system is idle

**Activation**

This use case starts when the user clicks on the “Create Recipe” link.

**Main Flow**

1. The user navigates to the “Create Recipe” page on the web application.
2. The user enters the specific recipe information into the input form.
3. The user clicks on the “Submit” button.
4. The application sends the form data to the server-side application.
5. The server-side application validates the inputs. <E1>
6. The server-side application saves the new recipe to the database.
7. The server-side application responds with the newly saved recipe data.
8. The new recipe data is added to the front end application state.
9. The user is taken to the new recipe’s URL.

**Exceptional Flow**

E1: Inputs are invalid

1. The user inputs which are invalid are displayed in red, with a message indicating the reason for the error under each invalid input.
2. The user inputs correct and valid inputs in the form.
3. Return to step 6 of Main Flow.

**Termination**

The application reports the successful saving of the new recipe and shows the user the specific recipe page.

**Post Condition**

The system returns to a wait state.

### 3.1.5 Requirement 4: Update Recipe

#### 3.1.5.1 Description & Priority

**Scope**

The scope of this use case is to allow a user to edit a previously submitted recipe in the form of form data in the web application.

**Description**

This use case describes the steps to be taken in order to update a previously submitted recipe in the web application.

**Flow Description**

**Precondition**

The system is idle

**Activation**

This use case starts when the user clicks on the “Edit Recipe” link on a specific user created recipe’s page.

**Main Flow**

1. The user selects “Edit Recipe” on a specific recipes page.
2. The user is shown a similar form with inputs as in the “Create Recipe” page, however the inputs have been pre-populated with the current recipe’s data.
3. The user edits or changes the pre-populated input fields with new and updated data.
4. The user clicks on the “Submit” button.
5. The application sends the updated form data to the server-side application.
6. The server-side application validates the inputs. <E1>
7. The server-side application updates the recipe in database.
8. The server-side application responds with the newly updated recipe data.
9. The updated recipe data is added to the front end application state.
10. The user is taken to the updated recipe’s URL.

**Exceptional Flow**

E1: Inputs are invalid

1. The user inputs which are invalid are displayed in red, with a message indicating the reason for the error under each invalid input.
2. The user inputs correct and valid inputs in the form.
3. Return to step 7 of Main Flow.

**Termination**

The application reports the successful saving of the updated recipe and shows the user the specific recipe page.

**Post Condition**

The system returns to a wait state.

## 3.2 Non-Functional Requirements

The following requirements add extra interactivity and functionality to the application and are to be added to the application after the minimum viable product has been achieved. These requirements apply only to logged-in users.

* A user can “like” another user’s recipe. Clicking the like button again to un-like.
* A user can add the ingredients of the currently viewed recipe to a “shopping list”, which can then be viewed via the navigation bar.
* A user can cross out already purchased items from the shopping list.
* A user can comment on any recipe.
* A user can offer a “rating” for a recipe.
* A user can view a dashboard with information about their own recipes, their recipes ratings, comment count, like count etc.

### 3.2.1 Performance/Response time requirement

The applications response time will depend on the end users internet connectivity. However, regardless of connection, some information should be displayed to the user very quickly. In the event of zero connection, an offline screen should appear and suggest that the user attempt to reconnect. In the event of a slow connection, the basic structure of the application should quickly load and a “loading spinner” should inform the user that information is being requested through the network.

### 3.2.2 Availability requirement

The application will be hosted online via the Heroku platform and will be available to users via the URL. On mobile devices (Android, iOS and Windows Mobile), users will be offered the choice to add the application to their phones home screen for easy access. This PWA approach will offer a near native experience to mobile users.

### 3.2.3 Reliability requirement

The application will function reliably with or without an internet connection, and with or without strong connectivity. While simply being available via the website URL, the application can also be made available on the Android Play Store, the iOS App Store and the Windows App Store.

### 3.2.3 Maintainability requirement

The application will be developed using a modular design, allowing for easy code maintainability and upgradability. The front-end application uses React, which is a component based UI library based on reusable class based components which can be easily reused and modified. The server-side application will modularized and organized into specific folders and associated files for easy readability and updatability.

# 4 GUI

The application will implement the GUI using the Twitter Bootstrap CSS library. The library handles mobile responsiveness and some common functionality like alerts and information boxes, dropdown menus, invalid input warning messages etc. Through appropriate use of this library we can guarantee that our application will work comfortably on any sized screen, on any device with a relatively modern browser.

The wireframes for this project were produced directly in Twitter Bootstrap. The main benefit of producing the initial mockup in Bootstrap, is that snippets of this code will be taken directly from the HTML of these files and placed in the React component files. Therefore by producing the mockup in Bootstrap, the developer has already completed part of the work of producing the React client-side application.

Note that the Add Recipe Page at fig 4.7 will be essentially identical to the Edit Recipe page, and is therefore not shown here.

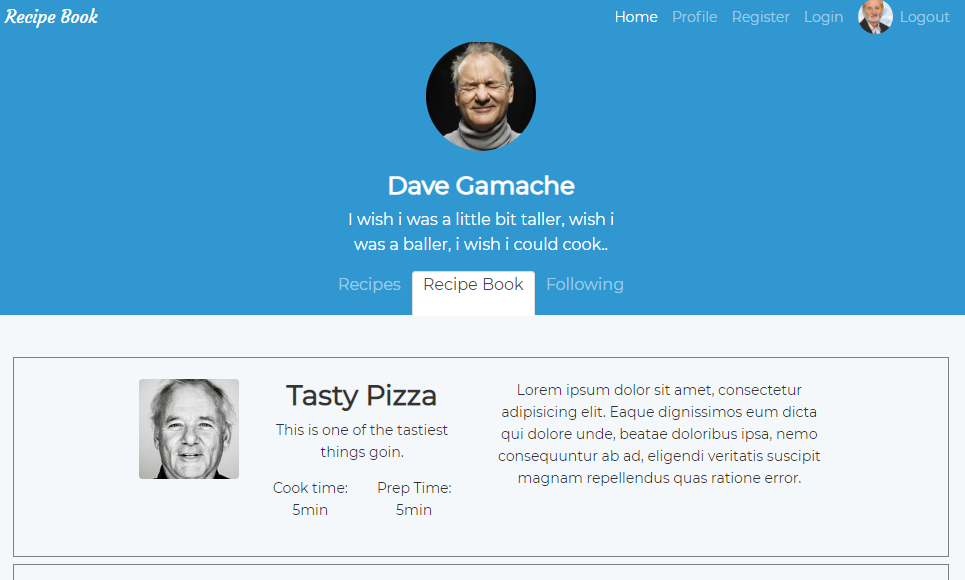
## 4.1 Landing Page

## 4.2 Registration Page

## 4.3 Login Page

## 4.4 Profile Page

## 4.5 Recipe Book Page



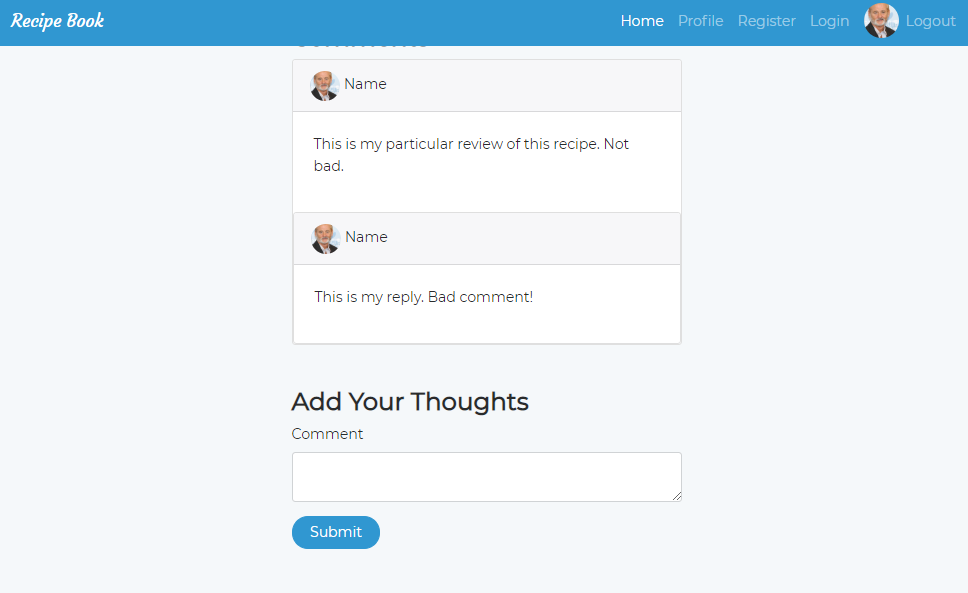
## 4.6 Following Page

## 

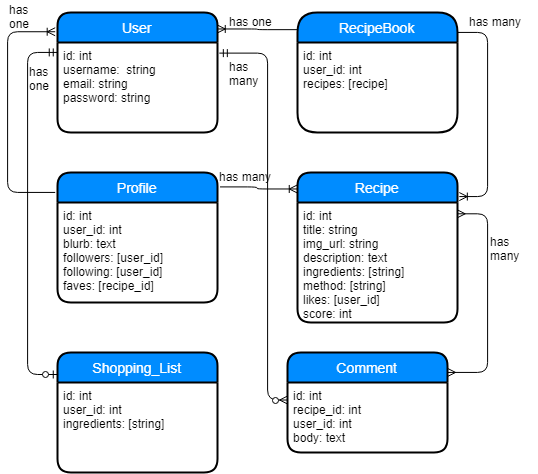
## 4.7 New Recipe Page

## 4.8 Recipe Page

## 4.9 Add Comment – Detail



# 5 System Architecture



# 6 System Evolution

Because of the modular nature of the structure of the application, adding new or editing or expanding current functionality would be relatively easy to do.

Thanks to the full-stack approach of constructing the application with a separate server-side and client-side application, native Android or iOS applications could be built and use the same server-side application as it stands now.

If the user count rose to a significant number, a “premium” account could be developed whereby a premium member would have access to extra features that were unavailable to regular users. Examples of this could be having more than one associated Recipe Book, an information dashboard which could graphically display data about the users recipes and comments, and perhaps simply to remove advertisements that could be implemented on the regular web application.

# 7 References

Kapoor, S., 2018. *freecodecamp.org.* [Online]   
Available at: https://medium.freecodecamp.org/progressive-web-apps-101-the-what-why-and-how-4aa5e9065ac2  
[Accessed 10 October 2018].

Sherman, P., 2018. *Medium.* [Online]   
Available at: https://medium.com/@pshrmn/demystifying-single-page-applications-3068d0555d46  
[Accessed 10 October 2018].