Recipe King

Group #15

Shayne Smither, Jiayu Xia, Kyle Stevenson, Zhiguang Liu,
Abdulah Sibalo

CMP_SC 4320

September 27, 2018

Contents

Report 1

Part 1.1: Customer Statement of Requirements

Problem Statement

Preparing a meal does not come naturally to most. For many people, the process of preparing a meal is quite involved, and there is a steep learning curve to cooking and preparing satisfying meals. The biggest issue faced by most is the issue of looking for and finding a meal worth preparing that fits within their constraints. We are constrained by time, budget, knowledge, materials, etc. Time and materials available are an especially relevant constraint when we are considering what to prepare in the moment. There are hundreds of thousands of meals available, so a great deal of time can be wasted searching for something. Additionally, when you find something that interests you, the cruel reality may be that you simply don't have nearly enough ingredients to prepare the meal as it is specified.

All these problems compound when you factor in health. A trip to any of the myriad of fast-food places is tempting when you don't have time to prepare a meal for yourself, sometimes due to your limited awareness of the meals that can be prepared with the ingredients you have on-hand. Yet eating at any restaurant, fast-food or not, is also considerably more expensive than preparing your own food in terms of real monetary expenses and physical health costs. In short, it is difficult to do your own cooking due to the constraints of individual creativity, time, and effort required, yet it is desirable to do so since the alternatives are often unhealthy and considerably more expensive.

These are all problems that cost people a lot of time, effort, and money. We are busy and we value convenience, this much is clear. In order to bypass the hassle and time-cost of cooking, we turn to convenient alternatives like fast-food, as previously mentioned. Similarly, we settle for frozen meals and resort to over-relying on our microwaves. These alternatives, however, are unhealthy and more expensive in the long run.

Also indicative that there is a real problem is the recent rise of meal-kit delivery companies, which promise to deliver fresh ingredients for a pre-planned meal to one's door. The main value proposition of such companies is that you don't have to spend hours looking for meals and then going to the grocery store to buy the requisite materials. Instead, the ingredients are nicely packaged for you along with an easy-to-follow recipe guide. The issue with this solution is that the meal-kits often cost significantly more than if you were to buy the ingredients yourself, and do not always arrive in perfect condition, if at all, due to the sensitive and vulnerable supply chain of such companies.

Current treatments are inadequate; cookbooks and online platforms do not deliver a customizable service to the client. Indeed, these solutions typically just list meals that are available in the database, their ingredients, and how to prepare the meal. A user may have to spend significant time searching for something worth devoting their time to preparing. The typical process goes something along the lines of: look for a meal out of the countless many offered, read through the ingredients only to realize you don't have the materials necessary, and continue to iterate through this process until you find something suitable. There is significant time lost and it is a pain to find a suitable meal.

We need a solution that emphasizes convenience and is capable of catering to the masses who want to cook but are constrained by various factors, and this is what

we are proposing to build. Our goal is to build a convenient and easy-to-use web application that can offer a simple and convenient solution to the problems discussed above.

Part 1.2: System Requirements

Enumerated Functional Requirements

Below are the functional requirements of the web application. These were created with the customer in mind, and we believe these are the components necessary to meet our proposed solution criteria and create a satisfactory experience. We follow the convention of using the phrasing “the system shall” for high-priority and necessary requirements and “the system should” for lower-priority items. The priority weights range from 1 to 5, with 5 being highest priority and 1 being lowest.

Identifier	Priority	Description
REQ-1	5	The system shall take a list of ingredients as input and give a list of recipes sourced from an API as output.
REQ-2	4	The system shall function on multiple devices such as computers and mobile devices.
REQ-3	3	The system should allow users to create a profile..
REQ-4	3	The system should be able to store user’s recipes.
REQ-5	2	The system should allow users to comment on recipes.
REQ-6	2	The system should allow users to give recipes a rating.
REQ-7	4	The system shall allow users

		to sort recipes based on various criteria.
REQ-8	1	The system should allow users to upload images.

Enumerated Non-functional Requirements

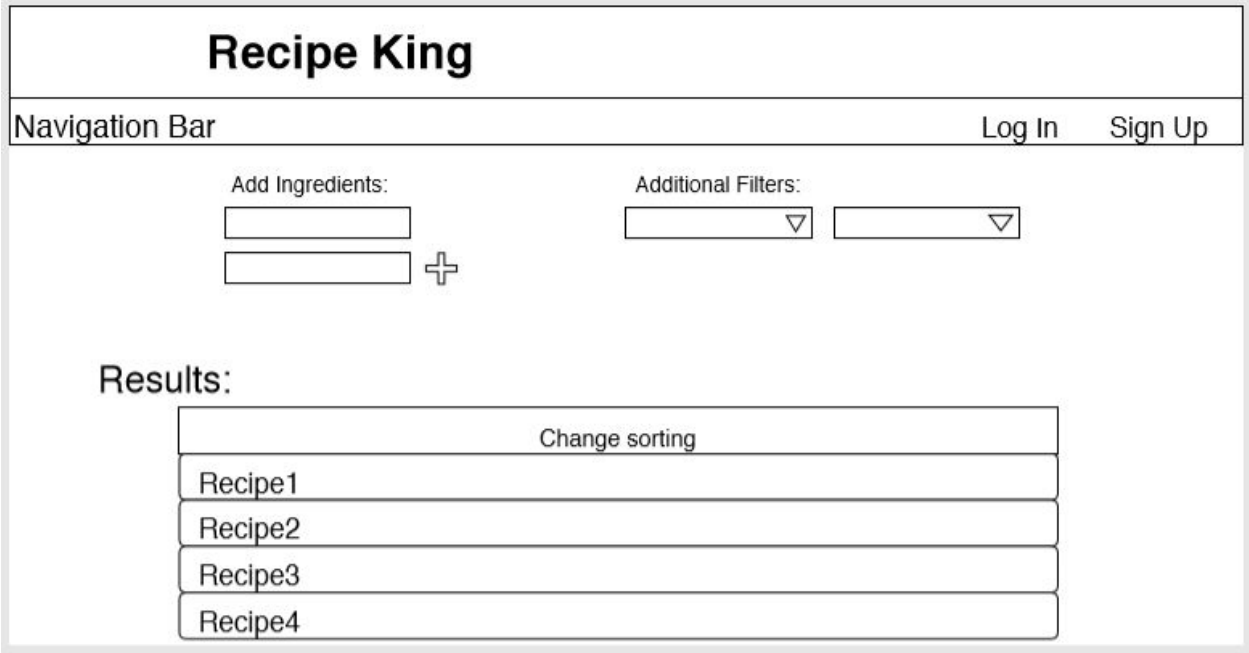
The following are the non-functional requirements. We follow the same conventions as with the functional requirements.

Identifier	Priority	Description
REQ-9	5	The system shall be easily usable and not require too much time to process input or display results client-side.
REQ-10	3	The system should be maintainable and not rely exclusively on the API for providing the user with recipe data.
REQ-11	2	The system should be scalable to accommodate a growing user base.
REQ-12	1	The system should be able to handle thousands of requests at once.

On-Screen Appearance Requirements

This section shows a prototype design of the user interface. The first image shows the main page where a user inputs the ingredients they have on-hand and receives a list of possible recipes for available meals. There is also a section to add filters with which to

sort the results. The second image shows a recipe's information after it is clicked on. This information includes the recipe itself and associated information (cooking time, nutrition info, etc.), a photo of the completed dish, comments by other users who have made the dish, and a simple rating.



Part 2.1: Functional Requirements Specification

Stakeholders

Stakeholders are anyone and everyone who has interest in a product or system that they can be benefited by being involved with or using. The amount of stakeholders attached to any given system largely depends on the versatility and nature of the system itself. In the case of Recipe King, stakeholders include anyone who has the desire or responsibility to put together food for themselves and/or others. The vast

majority of people on Earth are at least semi-responsible for feeding themselves or others and are therefore is stakeholders in Recipe King. However, specific examples of stakeholders include:

1. College students who do not live on campus and do not have a dining plan with access to cafeterias must grocery shop and cook meals for themselves
2. Working adults who have full time jobs spending most of their daylight hours away from home and their kitchen. They likely have little time to grocery shop let alone formulate meal ideas and execute them.
3. Parents, guardians and any form of caretakers who must provide meals for children or others that can not provide for themselves.
4. High school students who require or desire more freedom and responsibility for what they consume.

Actors and Goals

Actor	Actor's Goal
User (initiator)	Receive a list of possible recipes
User (initiator)	Choose a certain recipe and see the details of how to create it, what is required, and an image of the completed recipe
User (initiator)	Create a profile with which to save, comment on, and rate recipes

Use Cases

Casual Description

Use Case	Action	Description
UC-1	inputList	The user can enter a list of their currently owned ingredients
UC-2	fetchRecipes	The system will pulls a list of possible recipes from a connected database
UC-3	mobileDisplay	The system displays all required features properly on mobile devices
UC-4	profileManagement	The user creates a profile in the system to store data
UC-5	recipeSort	The system sorts the recipes based off of user criteria

Traceability Matrix

	Priority Weight	UC-1	UC-2	UC-3	UC-4	UC-5
REQ-1	5	X	X			
REQ-2	4			X		
REQ-3	3				X	
REQ-4	3				X	
REQ-5	2				X	
REQ-6	2				X	
REQ-7	4					X
REQ-8	1				X	
Total Weight		5	5	4	11	4

Fully-Dressed Description of Use Cases

Use Case UC-1: inputList
Related Requirements: REQ-1 **Initiating Actor:** User Interface **Participating Actor:** Recipe Fetcher **Actor's Goal:** Input a list of ingredients that will be used to generate a corresponding list of recipes. **Preconditions:** ● The web page has loaded correctly with all UI elements displaying properly **Postconditions:** ● The system returns a list of recipes corresponding to the ingredients input by the user **Flow of Events:** ● User inputs the ingredients which are fed into inputList(); ● inputList() is passed to fetchRecipes(), which makes a request to the API to receive the recipes based on the ingredients provided.

Use Case UC-2: fetchRecipes
Related Requirements: REQ-1 **Initiating Actor:** Recipe Fetcher **Participating Actor:** User Interface, API **Actor's Goal:** Generate a list of recipes from a database based off of a provided list of ingredients. **Preconditions:** ● A list of ingredients is provided **Postconditions:** ● The system returns a list of recipes corresponding to the ingredients input by the user **Flow of Events:** ● A list of recipes is returned to the web browser based on what was contained in inputList(). ● The browser display the available recipes along with photos of the prepared meals.

Use Case UC-4: profileManagement
Related Requirements: REQ-3, REQ-4, REQ-5, REQ-6, REQ-8 **Initiating Actor:** User **Participating Actors:** User Interface, User Database **Actor's Goal:** Create and display data based on the logged in user **Preconditions:** ● A user is logged in. **Postconditions:** ● The system displays data that the user has previously created such as comments, ratings, and saved recipes **Flow of Events for Main Success Scenario (User is logged in):** ● The user comments on the recipe ● The user's comment is saved in the database OR ● The user favorites/saves the recipe ● The data is saved in the database OR ● The user rates the recipe ● The rating information is added to the total in the database and the average user rating is updated and displayed **Flow of Events for Alternate Success Scenario (User is not logged in):** ● The user selects the recipe he/she wants. ● No data is saved/stored, and the user cannot comment on, favorite, or rate a recipe.

Part 3.1: Domain Analysis

Domain Model

Concept Definitions

Responsibility	Type	Concept Name
Coordinates all interactions between UI and inputs/requests, controls API access and data retrieval	D	Controller
Displaying UI in browser	D	Viewer
Container for user's input ingredient list	K	IngredientStorage
Container for recipes available and their ingredients and associated data	K	RecipeStorage
Checking ingredients against those in database	D	IngredientChecker
Determining appropriate recipes for user	D	RecipeMatcher
Retrieving recipes from API based on input ingredients	D	RecipeRetriever
Updating UI to show recipes and associated data	D	Viewer
Allowing user to sort and filter based on preferences	D	ViewCustomizer
Allowing user to create an account and log in/out	D	AccountCreator
Container for user's profile data	K	AccountStorage
Logging activity of logged-in users (comments, favorites, and recipes ratings)	D	Logger

Association Definitions

Concept Pair	Association Description	Association Name
Controller ↔ Viewer	Controller passes requests to the Viewer and receives updated webpage	Conveys requests
RecipeRetriever → Viewer	Data from RecipeRetriever passed to View to display	Displays recipes
ViewCustomizer → Viewer	Filters and/or sorting request sent to Viewer to update the displayer recipes	Update displayed recipes

Attribute Definitions

Concept	Attributes	Attribute Description
IngredientChecker	IsMajority	Check for recipes with at least some threshold (TBD) of user-specified ingredients.
ViewCustomizer	Ascending, Descending, CookTime, PrepTime	Sorts or filters the View based on user-specified requirements.

Part 3.2: Plan of Work

Our team will be divided into two teams. One team will work on the front-end of the website. The second team will work on the back-end.The front-end team will be Kyle Stevenson and Jiayu Xia. The back-end team will be Abdulah Sibalo, Shayne Smither, and Zhiguang Liu. We will use the VOIP program Discord to communicate as well as in-person meetings on an as-needed basis.

The front-end team will work on the following tasks:

- Building the input form for the recipe search.
- Display of recipe results to the user.
- Allow user filtering and sorting of results.
- Develop a consistent user experience across all devices.
- Add additional search options.
- Develop front-end functionality of low-priority features:
    - User login and sign up page
    - Rating, Commenting, and Saving recipes
    - User uploaded images to recipes

The back-end team will work on the following tasks:

- Server side functionality to call api service, and return results to the front-end.
- Develop back-end functionality of low-priority features:
    - Create a database and schema to track users.
    - Save user data related to recipes in database: Saved recipes, Comments, Ratings.
    - Allow for images to be saved for related recipes.