Let's Cook!

Amanda Hoelting, Ellison Largent, Lily Logan, Will Marceau

Software Requirements Specification

1. The Concept of Operations (ConOps)

Let's Cook is a web application that allows users to swipe through recipe cards in a given recipe deck. Each recipe will have a prep time, rating, ingredient list, and recipe instructions. The decisions are simple: like, dislike, save for later. The liked recipes will be added to the user's meal plan while saved recipes can be found in the user's saved tab. By navigating to the meal plan, the user can see all the recipes they've liked for that period. Clicking on the saved tab will bring the user to all of their saved recipes which can then be added to their meal plan if they wish. Lastly, the recipe creator feature allows users to import their own recipes and add them to their meal plan, save recipes, or even to the public recipe deck!

- LL 10/10/2024

1.1. Current System or Situation

Meal planning and recipe selection are fundamental daily/weekly activities that impact individuals and families everywhere. Currently, most people rely on solutions including traditional cookbooks, google searches, social media platforms, and word-of-mouth recommendations to discover and plan their meals. (Treadway, 2021) These scattered resources, while valuable, often lead to decision fatigue and an inefficient use of time and money. Users will typically work with multiple platforms and systems, searching for recipes on one platform, while maintaining grocery lists on another, and attempting to track successful meals through another different program. While not always, this process usually takes more time than necessary and can add to decision fatigue when doing so on an empty stomach.

The process of deciding what to eat remains a significant daily challenge for many people, despite (and sometimes because of) the abundance of available options. Recipe websites and cookbooks, while comprehensive, can overwhelm users with choices and fail to account for individual preferences or dietary restrictions. Additionally, the disconnect between recipe discovery and grocery planning often results in multiple store visits, food waste, and poor resource management. Most existing solutions address only one or two aspects of the meal planning process, forcing users to manually coordinate the rest.

Another challenge in meal planning is time management. Between searching for recipes, planning meals, creating grocery lists, actually cooking, and managing household preferences, individuals spend considerable time coordinating these separate but interrelated tasks. This inefficiency often leads people to either spend more money on takeout to save time or default to a limited rotation of familiar recipes, potentially sacrificing variety and nutrition for convenience. The lack of an integrated system that combines recipe discovery, preference tracking, meal planning, and grocery list management creates unnecessary complications in what should be a straightforward daily task.

- EL 10/10/2024

1.2. Justification for a New System

The current situation demonstrates significant inefficiencies, impacting both daily decision-making capacity and household finances. According to the USDA (2023), the average American household wastes almost $1,500 in food as a result of poor meal planning and spends an average of 37 minutes a day on meal-related decisions. Although existing solutions such as recipe websites, google searches, and meal planning apps exist, their overwhelming recipe options and presentation can overwhelm users with the paradox of choice. These solutions also generally require the use of multiple platforms to perform all steps of meal planning, grocery shopping, and cooking, leading to decision fatigue and greater pressure to just opt for takeout or else limit their dietary variation for the sake of time.

Our proposed web application *Let's Cook* addresses these challenges through a more binary decision-making interface that integrates meal discovery, meal planning, and grocery list management all into a single site.

- EL 10/10/2024

1.3. Operational Features of the Proposed System

The key operational features could include a swiping feature that allows users with an account to swipe left and right on meal recipes to either discard or save them, view their saved recipes, and add recipes to their weekly meal planner, adding the ingredients to their grocery checklist. We will be showing recipes on the meal swiping page from a public database, so we do not need to worry about intellectual property at this point.

- AH 10/10/2024

1.4. User Classes

The *Let's Cook!* application will support two distinct user classes: Recipe Finders and an Administrator.

Recipe Finders class consists of individuals seeking new recipes. They will interact with the web application by swiping through a collection of recipes, and selecting which they want to include in their weekly meal plans. Additionally, Recipe Finders will have access to personalized features such as saved recipes, a dislike list (to filter unwanted ingredients or recipes), and an automatically generated grocery list based on their meal plan selections.

The Administrator will be responsible for overseeing the system's user management. This database stores detailed information about each Recipe Finder. The database will contain user profiles, preferences (such as dislikes), saved recipes, meal plans, and grocery lists.

- LL 10/10/2024

1.5. Modes of Operation

Let's Cook! will have two different modes of operation, admin and user. The user mode will allow the user to login and create an account. Additionally, when successfully logged in, the user then can interact with all parts of the software as intended. The administrator mode will have a lot more freedom with the software. This mode will allow them to monitor the app, and database to ensure everything is running smoothly. Additionally, this mode will have the ability to make changes to the app or database when necessary.

- WM 10/11/2024

1.6. Operational Scenarios (aka "Use Cases")

Use Case: Save new recipes to the user's library.
Brief description: This use case describes how a user would use the platform to find a recipe they might want to cook later.
Actors: A user.
Preconditions:

1. The user has internet access.
2. The user has created an account on *Let's Cook!*.

Steps to Complete the Task:

1. The user signs into their account on the login page.
2. The user is then redirected to the home page with swipeable recipes, where they can swipe left to discard a recipe and swipe right to save the recipe.
    a. There are arrows on both sides indicating which side to swipe to save the recipe and which side to discard it.
    b. Each recipe card will have the recipe name, list its ingredients, and list the first few steps.
3. The user swipes until they find the recipe(s) they want to save.

...

Postconditions:

The new recipes the user just swiped to save will be in their saved recipes library in the saved recipes tab. They will be able to search for the recipe by name or by an ingredient. When they click on the recipe, they will be able to see a full list of steps and ingredients.

Use Case: Form a weekly recipe plan to create a grocery list.
Brief description: This use case describes how a user would use the platform to add saved recipes to their weekly meal plan that will then automatically create a grocery list.
Actors: A user.
Preconditions:

1. The user has internet access.
2. The user has created an account on *Let's Cook!*.
3. The user has one or more saved recipes.

Steps to Complete the Task:

1. The user signs into their account on the login page.
2. The user is then redirected to the home page with swipeable recipes, where they can follow the steps to add more recipes to their saved recipes tab if they choose.
3. The user can then navigate to the grocery list tab.
4. On this page, the user will be able to add recipes from their existing saved recipes list to their weekly meal plan panel.
    a. They will not have to assign any meal to a day. They can add as many or as few recipes to their weekly meal plan.
5. The grocery checklist will be below the weekly meal plan panel and will automatically populate with the ingredients needed for the recipe(s) they added to the meal plan.

...

Postconditions:
The user has a list of the meals they are planning to make that week and a checklist of all the ingredients they will need to purchase at the store.

2. Specific Requirements

(**NOTE:** For this section:
(M): Stands for a must have requirement
(S): Stands for a requirement the software should have
(C): Stands for a requirement the software could have
(W): Stands for a requirement the software won't have

2.1. External Interfaces (Inputs and Outputs)

This section should describe inputs into and outputs from the software system. (ISO/IEC/IEEE 29148:2011)

Each interface description should include the following:

1. Name of item.
2. Description of purpose.
3. Source of input or destination of output.
4. Valid ranges of inputs and outputs.
5. Units of measure.
6. Data formats.

Functional

1. User Accounts
   1.1. Accept Login Credentials **(M)**
      1.1.1 Purpose: The software must accept login credentials from the user
      1.1.2 Source of Input: The source of the input will be the login forum
      1.1.3 Valid Ranges:
         -The Username must be 5 - 50 characters long (0-9, A-Z, -, _, .)
         -The Password must be at least 8 characters long
      1.1.4 Units of Measure: N/A
      1.1.5 Data formats: The data will be received in string format
   1.2. Accept Account Creation Credentials **(M)**
      1.2.1 Purpose: The software must accept login credentials to use for new accounts
      1.2.1 Source of Input: The source of the input will be the account creation page
      1.2.2 Valid Ranges:
         -The Username must be 5 - 50 characters long (0-9, A-Z, -, _, .)
         -The Password must be at least 9 characters long
      1.2.4 Units of Measure: N/A
      1.2.5 Data formats: The data will be received in string format

2. Recipe Tinder
   2.1. Recipe Prefrencest **(M)**
      2.1.1 Purpose: The software must account for liked, disliked and maybe recipes indicated by the user
      2.1.2 Source of Input: The source of the input will be the recipe tinder section of the application, and whether the user swipes right, left, or says maybe to the recipe
      2.1.3 Valid Ranges: The input must be either -1, 0, 1
      2.1.4 Units of Measure: N/A
      2.1.5 Data formats: The data will be received in signed integer format
   2.2. Preferences Output **(M)**
      2.2.1 Purpose: The software must keep track of the users liked recipes
      2.2.2 Destination of Output: the location of this output will be in the liked recipes section of the app
      2.2.3 Valid Ranges: The output can range from no liked recipes to all recipes in the database liked by the user
      2.2.4 Units of Measure: N/A
      2.2.5 Data formats: The data will in string format with the inclusion of a picture

3. Meal Planner
   3.1. Meal Choices **(S)**
      3.1.1 Purpose: The software must take in the users choice of meals for the week
      3.1.2 Source of Input: The source of the input will be the meal planner section of the app
      3.1.3 Valid Ranges: The input must be a list recipe names with a length ranging between 0 to 25
      3.1.4 Units of Measure: N/A
      3.1.5 Data formats: The data will be received as a string
   3.2. Grocery List **(C)**

3.2.1 Purpose: The app will take the meals that the user wants to eat this week and generate a grocery list consisting of all the ingredients they need to cook those meals

3.2.2 Destination of Output: the location of this output will be in the Grocery List section of the app

3.2.3 Valid Ranges: The output must be a list of ingredients with a length ranging between 0 to 100

3.2.4 Units of Measure: The units of measure will be ingredient specific based on the needs of the recipes selected

3.2.5 Data formats: The data will be outputted as a string

- WM 10/11/2024

2.2. Functions

Define the actions that must take place in the software to accept and process inputs and generate outputs (ISO/IEC/IEEE 29148:2011). These definitions must include:

1. Validity checks on the inputs.
2. Sequence of operations in processing inputs.
3. Responses to abnormal situations, including error handling and recovery.
4. Relationship of outputs to inputs, including
   (a) input/output sequences
   (b) formulas for input-output conversion

Functional

1. User Accounts
   1.1. Account Login **(M)**

1.1.1 Validity check: The software must check that the Username and Password given in the account login are a valid combination present in the accounts table of the database

1.1.2 Sequence of operations:

1. The software gets the input from the login page
2. The software checks the validity of the entered credentials as described above.
3. On a successful login, the software must get and display the account landing page for the user to interact with.
4. On an unsuccessful login the software must stay at the login page

1.1.3 Error handling and response: On an unsuccessful login attempt, the software must keep the user on the login screen, give them a warning on what went wrong like "incorrect username and password", and allow them to try again.

1.1.4 Relationship of outputs and inputs: The input of the login credentials leads to the output of either the account landing page or an error message depending on the validity check

1.2. Account Creation **(M)**

1.2.1 Validity check: The software must check if the Username has been taken and if the Username and Password meet the specified requirements

1.2.2 Sequence of operations:

1. The software gets the input from the sign up page
2. The software check the validity of the credentials as described above
3. On a successful validity check, the software must add their username and password to the account table in the database
4. The software redirects to the login screen where they can log in using their new credentials

1.2.3 Error handling and response: On an unsuccessful attempt to create an account, the software must display a warning on what went wrong like “Username Taken”, and must allow them to try again

1.2.4 Relationship of outputs and inputs: The input of the signup credentials leads to the output of either the account being created and then the software redirecting to the login page or an error message depending on the validity check

2. Recipe Tinder

2.1. Recipe Preferences **(M)**

2.1.1 Validity check: The software must check if the preference (1, 0, -1) was assigned correctly

2.1.2 Sequence of operations:

1. The software gets the input from the user
2. The software assigns the preference to the recipe
3. If the preference was liked, then it adds the recipe to the liked recipe list
4. The software must then give the user another recipe to look at prioritizing recipes that have either not been seen or set to maybe

2.1.3 Error handling and response: On an unsuccessful attempt to assign the users preference, the software must display a message like “error occurred, try again” and let the user input their preference again.

2.1.4 Relationship of outputs and inputs: The input of the users preference leads to the output of expanding the recipe list and generating the next recipe that the user might like depending on the preference given

3. Meal Planner

3.1. Grocery List **(C)**

3.1.1 Validity Check: The software must check if the meals given are valid and if the Grocery List is properly updated based on the users selection of recipes when either input new meals or removing existing ones

3.1.2 Sequence of operations:

1. The software gets the input from the user in the meal planner section
2. It then updates the grocery list with the ingredients required to make all the rules

2.1.3 Error handling and response: If an error occurs when updating the grocery list or meal planner, the software must display a message like “error occurred, try again” and let the user try and put their desired recipe in the meal planner again

2.1.4 Relationship of outputs and inputs: The input of the meals the user wants to cook for the week leads to the output of expanding their grocery list based on what new ingredients the user now needs, if the user removes a recipe from their meal planner, the grocery list must remove only those ingredients from the grocery list

- WM 10/11/2024

2.3. Usability Requirements

Define usability requirements and objectives for the software system, including measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use. (ISO/IEC/IEEE 29148:2011)

Non-Functional

1. Usability
    1.1. Mobile Usage **(M)**
        1.1.1 The app must be easy read on mobile devices
        1.1.2 The app must have an intuitive UI design that is conducive to usage on a mobile touchscreen device
    1.2 Computer Usage **(M)**
        1.2.2 The app must be easy to read on laptops and desktops
        1.2.2 The app must have an intuitive UI design that is conducive to usage on a laptop or desktop computer
    1.3 Error Handling **(S)**
        1.3.1 When there is a user error, the user must get clear feedback about what went wrong within 5 seconds
    1.4 Ease of Use **(C)**
        1.4.1 A first time user must be able to competently interact all portions of the website within the first 10 minutes of use
        1.4.2 An experienced user must be able to navigate to their desired section of the website within 5 clicks
2. Satisfaction
    2.1. User Satisfaction **(S)**
        2.1.1 The app must reach a satisfaction rate of at least 85% amongst users

- WM 10/10/2024

2.4. Performance Requirements

Specify the static and dynamic numerical requirements placed on the software or human interaction with the software. For example: (a) Static numerical requirements may include the amount and type of information the system processes. (b) Dynamic numerical requirements may include the amount of data processed within specific periods.

State performance requirements in measurable terms. For example, “95% of the transactions shall be processed in less than 1 second” rather than “An operator shall not have to wait for the transaction to complete” (ISO/IEC/IEEE 29148:2011).

Non-Functional

1. Efficiency

1.1. Login Efficiency **(M)**

1.1.1 An experienced user must be able to login to the website in under 10 seconds of entering their credentials, if they are correct.

1.2. Recipe Tinder Efficiency **(S)**

1.2.1 When a user swipes a recipe left or right, a new recipe must replace the old one within 5 seconds

1.2.2 When a user adds a recipe to their linked list, it must be added before they view their list

1.3 Recipe List Efficiency **(S)**

1.3.1 When a user wants to view their recipe list, it must take no longer than 5 seconds

1.4 General Transitions Efficiency **(S)**

1.4.1 Transitions between pages of the website must take no longer than 10 seconds to load on a stable internet connection

- WM 10/10/2024

2.5. Software System Attributes

Specify the required attributes of the software product, such as reliability, security, privacy, maintainability, or portability. (ISO/IEC/IEEE 29148:2011) Review a comprehensive list of software attributes or software qualities, such as are provided in van Vliet (2008) Chapter 6. Decide on a relatively small number of the most important attributes for this system. Explain why each attribute is important and what steps you will take to achieve those attributes. The attributes include constraints on attributes of the system's static construction, such as testability, changeability, maintainability, and reusability. (Faulk, 2013)

1. Security

1.1. Password Encryption **(M)**

1.1.1 All passwords in the database must be encrypted for user protection, this is important because while our program will not contain very important data, many people reuse their passwords for many different accounts. Thus if someone got into our database and we did not encrypt the passwords, we could be exposing our users to potentially devastating attacks. Additionally, despite there being no sensitive information on our users in the app, we still want to ensure that their accounts and data are safe from harm.

2. Portability

2.1. Cross-Platform usability **(M)**

2.1.1 The software must be usable on both mobile and computer. Due to the bigger screen and ease of use, many people like to browse for recipes and keep their recipe list on their computer, so it is important that we do not alienate that audience from our app. Conversely, users typically do not rely on computers for managing their grocery lists, as they cannot take them to the store. Therefore, it is imperative that our app's design is intuitive and easy to use on mobile devices, and computers so that we can accommodate both of these audiences.

- WM 10/10/2024

3. References

Treadway, Tony. (2021) Creative Energy Agency. Where Do Consumers Go for Recipes? https://creativeenergy.agency/food/where-do-consumers-go-for-recipes-we-have-the-answer/

USDA Economic Research Service. (2023). "Food Waste and Consumer Behavior Report." https://www.ers.usda.gov/amber-waves/2024/october/u-s-consumers-increased-spending-on-food-away-from-home-in-2023-driving-overall-food-spending-growth/

This section lists the sources cited in the creation of this template document. An SRS should reference all of the sources that it draws from. This section may not be necessary if sufficient citations are provided "inline" (at the point of reference) in the document.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oracle. (2007). White Paper on "Getting Started With Use Case Modeling". Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

Sethi, Ravi. (2023). *Software Engineering. Basic Principles and Best Practices*. Cambridge Press.

Work Breakdown Structures. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Work_breakdown_structure.

N2 Charts. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/N2_chart.

Functional Flow Block Diagrams. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Functional_flow_block_diagram.

Structure Chart. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Structure_chart.

Data-flow Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Data-flow_diagram.

Object Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Object_diagram.

System Context Diagram. In *Wikipedia*, n.d.
https://en.wikipedia.org/wiki/System_context_diagram.

Storyboard. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Storyboard.

Entity Relationship Model. In *Wikipedia*, n.d.
https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model.

4. Acknowledgments

List here all sources you used to create the document and support you received from anyone not on your team.