

Recipe Buddy

Software Design Document

By gOatmeal

By **Noah Lampron, Andy Mahoney, Austin Dykeman, Tyler Neese, Michael Shae**

SECTION 1: INTRODUCTION

The purpose of this software design document is to provide a low-level description of the RecipeBuddy system. Topics covered include the following:

- Class hierarchies and interactions
- Data flow and design
- Processing narratives
- Algorithmic models
- Design constraints and restrictions
- User interface design
- Test cases and expected results

This document will give the readers a solid understanding of the inner workings of the RecipeBuddy system.

1.1 Goals and Objectives

The purpose of RecipeBuddy is to provide an easy and intuitive method of finding and obtaining recipes, step-by-step instructions on how to cook said recipes, and the addition, edition, deletion of recipes. The primary objective is to help users cook.

As such, the final product must be easy to use, and user friendly. The user interface shall not have a learning curve. The application will provide the following functionalities:

- Add, edit, delete recipes
- Search for recipes
- Shopping cart
- Infinite scrolling of recipes
- Step-by-step instructions
- Timer
- Sort by tags
- Uploading pictures

1.2 Project Overview and Scope

The RecipeBuddy system is composed of three main components: a database that stores the data, a client-side application that receives the user input and shows the graphical user interface, and a server-side application that provides data validation on user inputs and communicates with the database. The system can be broken up into two groups: core features, which are critical to the system, and additional features, which add to the system.

CORE FEATURES

1. Server Connection
 - a. Client-side application must connect with the server.
 - b. Database must connect to the server.
2. Show Recipe
 - a. Show detailed information on the recipe, including estimated cooking time, and calories.
 - b. Can add, edit, or delete the recipe by clicking on the respective buttons.
3. Add Recipes
 - a. User can create a new recipe and add it to the database through the server.
 - b. Recipes may be duplicated.
4. Edit Recipes
 - a. Allows the user to make changes to an already existing recipe, and save the changes to the database through the server.
 - b. The server must handle the input validation.
5. Delete Recipes
 - a. Allows the user to delete the recipe, and saves the change to the database through the server.
 - b. User must confirm the deletion before the change is saved.
6. Search For Recipes
 - a. Allows the user to look for recipes on the home page.
 - b. User is allowed to search using key-words, and tags.
7. Cook Recipes
 - a. Provides step-by-step instructions to the user.
 - b. Provides an optional timer for the user to use.
8. Shopping Cart
 - a. Stores a list of ingredients and amounts for the user. User may add ingredients individually.
 - b. Ingredients are automatically added when a recipe is cooked.
9. Infinite Scrolling
 - a. Allows the user to scroll through the home page of recipes, and will not end until all recipes in the database have been shown.
 - b. The order in which recipes will be shown this way is random. Infinite scrolling also works during a search.

ADDITIONAL FEATURES

1. Add/Remover Ingredients
 - a. Allows the user to add/delete ingredients to a recipe.
 - b. Available in add, edit, and deletion of recipes.
2. Add/Remove Steps
 - a. Allows the user to add/delete instruction steps to a recipe.
 - b. Available in add, edit, and deletion of recipes.
3. Add/Remove Tags
 - a. Allows the user to add/delete tags to a recipe.
 - b. Available in add, edit, and deletion of recipes.
4. Finish Step
 - a. Allows the user to move onto the next step during cooking mode.
 - b. This gives the user freedom to view portions of the recipe at their own pace.
5. Timer
 - a. Available for all steps of the cook mode. Provides a timer for the user.
 - b. The user can set the timer to any reasonable time they desire. It is optional.
6. Complete Cook
 - a. The finish step for the final instruction.
 - b. Allows the user to go back to the home page. Also, signifies that the cooking of the recipe is done.
7. Upload Pictures
 - a. Allow the user to upload and add pictures to recipes via add, edit features.
 - b. Pictures are displayed on both the show recipe page, and on the home page, while scrolling through the recipes.

1.3 SOFTWARE CONTEXT

RecipeBuddy will be offered on FireFox free of charge. Future development and maintenance costs are nonexistent. The system is being developed as a learning experience, and as such the team is moving forward with the goal of learning.

1.4 MAJOR CONSTRAINTS

The main constraints are experience and time. We are using MongoDB for our database and no one on the team has used MongoDB before, so we have to learn it as we go. We are working on a time crunch due to the fact that RecipeBuddy has a deadline of December 16. However, we will endeavor to complete at least the core features.

1.5 INTENDED AUDIENCE AND READING SUGGESTIONS

While the software requirement specification document is written for a more general audience, this document is intended for individuals directly involved in the development of RecipeBuddy. This document does not need to be read sequentially. Below is a brief overview of each part of the document:

- Part 1 (Introduction)
 - This section offers a summary of RecipeBuddy, as well as a guide of what sections contain what information.
- Part 2 (Data Design)
 - This section shows how RecipeBuddy handles data, along with the data structures, and flow patterns used by the system.
- Part 3 (Architectural and Component-Level Design)
 - This section describes all classes in detail, along with interface, hierarchies, design constraints, process details, and algorithmic details.
- Part 4 (User Interface Design)
 - This section shows and covers all related graphical components of RecipeBuddy. This will show how a tentative final product will look.
- Part 5 (Restrictions, Limitations, and Constraints)
 - This section discusses the general constraints imposed upon the project.
- Part 6 (Testing Issues)
 - Readers interested in the testing process of the software should read this section. It offers test cases, expected results, and other relevant information.
- Part 7 (Appendices)
 - This section includes any additional information that may be helpful to readers.

SECTION 2: DATA DESIGN

2.1 INTERNAL SOFTWARE DATA STRUCTURE

Recipe Buddy's internal structure is divided into two parts: server-side and client-side.

On the client side, the user has the ability to view all the recipes, add new ones, edit ones that already exist and delete ones they no longer need. In addition, the user can "cook" recipes which means they are walked through every step individually and can click through as well as a timer for each step, if necessary. The client side is a website written in HTML, with JavaScript and the layout uses CSS. The user accesses this via a web browser, particularly FireFox. Due to being a website, the user can use any operating system to access Recipe Buddy.

The data structure on the server will have the classes defined later in the document to perform the functions that the user wants to do. The recipes will be stored in a MongoDB database. The server will access the database and be able to add, edit and delete data. The server will retrieve information from the database based on the ID number and pass it back to the client as needed.

The client and server will communicate with each other using JavaScript and the recipes have their own class with a unique ID with all the information. This object is passed from the server to the client.

2.2 GLOBAL DATA STRUCTURE

The global data structure of this application is best characterized by the database. The database structure shows the data involved in the application in its purest sense. The local web page will never access this database directly; it will instead issue requests to the server.

2.3 TEMPORARY DATA STRUCTURE

Temporary data structures, as they relate to Recipe Buddy, refer to the data objects that are created on the local client's computer, and also to the JavaScript objects that interchanged between the server and the client. The data objects created on the local device will only exist for the duration of time that the application is running, and will subsequently be destroyed.

The JavaScript objects will only exist for the duration of the transaction between the client and server. The server will destroy the objects after sending them, and the client will destroy the JavaScript objects once they have been parsed.

2.4 DATABASE DESCRIPTION

```
-- Table structure for table `recipes`  
  
--  
CREATE TABLE `recipes` (  
  `ID` int NOT NULL auto_increment,  
  `name` varchar(30) NOT NULL,  
  `ingredients` [(varchar(50),varchar(50),int)],  
  `steps` [(varchar(255), float)],  
  `tags` [varchar(50)],
```

```
`description` varchar(255)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

SECTION 3: ARCHITECTURAL & COMPONENT-LEVEL DESIGN

3.1 SYSTEM STRUCTURE

The RecipeBuddy system is composed of three main components: a MongoDB database, a client-side application, and a server-side application.

The client-side FireFox application that receives the user input and shows the graphical user interface. This component is written with JavaScript to communicate with the rest of the system and with HTML for the functional aspects of the website and with CSS for the graphical aspects. The functional parts include the obtaining the user input, and telling the website what to display. The graphical parts is simply the graphical user interface. All aspects work with each other.

The server-side application that provides data validation on user inputs and communicates with the database. The communication with the rest of the system is one using JavaScript, data validation and most functionality are done using HTML. This will form the core functionality of the system.

3.2 HOME CLASS

The home class is a class that is meant to represent the home page of the recipe buddy website. It stores recipes from the recipe class as well as their IDs. This is in order to display the recipes on the home screen.

3.2.1 PROCESSING NARRATIVE

When the user loads up the recipe buddy website, the server looks at the home class in order to display some of the information that is included on the homepage mainly the recipes that are stored on the database. The information that it stores specifically is as follows:

- List of recipes on the database
- List of IDs associated with those recipes

3.2.2 INTERFACE DESCRIPTION

Add()
Edit(Int ID)
Delete(Int ID)
Search(Int ID)
Cook(Int ID)
AddToCart()

3.2.3 PROCESSING DETAIL

There are no algorithms that are associated with the home class but the functions that it contains include muttors and functions to start other processes.

3.2.3.1 DESIGN CLASS HIERARCHY

The home class has no parent or children classes but it does interact with them as well as with the server and database.

3.2.3.2 RESTRICTIONS/LIMITATIONS

The home class currently has no restrictions or limitations.

3.2.3.3 PERFORMANCE ISSUES

There should be no performance issues with the home class because it doesn't have to do a lot of work compared to some other classes and mainly calls other functions or finds information to display.

3.2.3.4 DESIGN CONSTRAINTS

A design constraint is that there is only one home class for each user so there is a problem in the code, there is no chance for it to be edited and it will always show that error.

3.2.3.5 PROCESSING DETAIL FOR EACH OPERATION

- Add()
 - The add function allows the user to add a new recipe to the database and occurs when the user presses add on the home screen.
- Edit(Int ID)
 - The edit function take an ID of a recipe and allows the user to edit that recipe. This function will be activated once someone hits the edit button on a recipe from the home screen
- Delete(Int ID)

- Delete takes in an ID of a recipe and removes it from the database. It will also give a conformation to make sure the user wants to remove it before it does so. This occurs when the user presses delete on a recipe from the home screen.
- Search(Int ID)
 - Search takes in a recipe ID and allows the user to search through the database for a specific recipe.
- Cook(Int ID)
 - The cook function occurs when someone presses the cook button on a recipe and it takes them to the cook page with information about that recipe. It takes in a recipe ID so that it knows which recipe to retrieve the cook information for.
- AddToCart()
 - The add to cart function allows the user to add a ingredient to a shopping cart that they have clicked on.

3.3 RECIPE CLASS

The recipe class is a class that is meant to store the different recipes that the users input as well as all the information about these recipes. It includes the name of the recipe, a ID to identify recipes of the same name, ingredients, cooking steps, tags to categorize the recipe, and a description. There is a new recipe class created for each new recipe and a previously existing recipe class can be edited to store new information.

3.3.1 PROCESSING NARRATIVE

When a user selects the option to make a new recipe, a new recipe object is created that is used to store all the information that the user entered .The object is responsible for holding the following information:

- Recipe Name
- Recipe ID (Automatically generated)
- Ingredients
- Cooking Steps
- Tags
- Description

Whenever information about a recipe is needed, the website will access the appropriate recipe object and retrieve the information in order to display it to the user.

3.3.2 INTERFACE DESCRIPTION

Cook(Int ID)

AddIngredient (Int ID , [(String,String,Int)] Ingredients)

AddStep(Int ID, [(String,Float)] CookingSteps)

RemoveIngredients (Int ID , [(String,String,Int)] Ingredients)

RemoveStep (Int ID, [(String,Float)] CookingSteps)
EditTagList (Int ID, [String] Tags)

3.3.3 PROCESSING DETAIL

The recipe class does not use any algorithms and the only functions in the class are mutators in order to edit or store the information in the recipe class.

3.3.3.1 DESIGN CLASS HIERARCHY

The recipe class does not have any parent or child classes but it does interact with the other classes and servers by sending and receiving information about the current recipe object.

3.3.3.2 RESTRICTIONS/LIMITATIONS

Because of data storage there might be a limit to how large someone can make a recipe while maintaining functionality so that will be limited after testing.

3.3.3.3 PERFORMANCE ISSUES

Since the recipe class is pretty small it by itself doesn't have any performance issues however since there will be many recipes, the large amount of recipe objects could cause performance issues for other classes.

3.3.3.4 DESIGN CONSTRAINTS

Since many parts of recipe are used by other parts of the website and since the format of all recipes are the same, having it work correctly is a large part of making sure the recipe buddy website is functional. If one recipe's format is broken then so are all other recipe formats.

3.3.3.5 PROCESSING DETAIL FOR EACH OPERATION

- Cook(Int ID)
 - This function takes in the ID value of a recipe and then starts the cooking process. It begins by bringing the user to another page then bringing up the first step in that particular recipe.
- AddIngredient (Int ID , [(String,String,Int)] Ingredients)
 - This function is used when a user wants to add an ingredient to the recipe that is specified by the ID value. It will then wait for an input then verify it and after doing so add it to the recipe object under the ingredients variable.
- AddStep(Int ID, [(String,Float)] CookingSteps)
 - This function is used when a user wants to add a step to their recipe. The function locates the correct recipe by the ID number then accepts an input from

the user which is the new step then adds that to the recipes CookingSteps variable.

- RemoveIngredients (Int ID , [(String,String,Int)] Ingredients)
 - If the user wants to remove an ingredient from a recipe then this function is used and it takes a input from the user which is the ingredient that they want to be removed then removes it from the correct recipe specified by the ID value.
- RemoveStep (Int ID, [(String,Float)] CookingSteps)
 - If the user wants to remove a step from the recipe then this function is called and the user specifies which step they want to remove, then the function removes that step from the correct recipe specified by the ID value.
- EditTagList (Int ID, [String] Tags)
 - If the user wants to edit the tags that are being used for the current recipe then this function is called. It takes in a new list of tags that are then applied to the Tags variable for the correct recipe specified by the ID variable.

3.4 COOK CLASS

The cook object is used for when the user selects a recipe to cook. This object is used to display the steps, ingredients, and timers while the user is using the website to make their recipe. It uses the information from the recipe classes in order to display the steps and ingredients to the user while making their meal.

3.4.1 PROCESSING NARRATIVE

When a user selects that they want to cook a recipe, it changes pages and uses the cook class to complete the displaying of the recipe. This object holds the following information:

- Name of the recipe
- ID of the recipe
- Ingredients
- Cooking Steps

This object is used only to show the user the cooking steps and ingredients and although it does store some information that recipe stores, this is a temporary storage so that we don't overload the database by constantly asking for information from a recipe.

3.4.2 INTERFACE DESCRIPTION

Cook

FinishStep()

StartTimer()

StopTimer()

Complete()

3.4.3 PROCESSING DETAIL

There are no algorithms that are associated with this class and the functions also don't alter any data. The class only communicates with the server in order to get the initial recipe data, then it uses its own client side information to complete the cook process.

3.4.3.1 DESIGN CLASS HIERARCHY

The cook class is not a parent or child of any other class so it doesn't derive anything from the other classes but it does interact and get information from other classes. It has to communicate with the server and database in order to get information from recipes.

3.4.3.2 RESTRICTIONS/LIMITATIONS

There are currently no restrictions or limitations for the cook class.

3.4.3.3 PERFORMANCE ISSUES

A performance issue with the cook class is that it displays a lot of information to the user so it is important that it works correctly. It needs to perform smoothly and can't be slow or inefficient at grabbing information for the user.

3.4.3.4 DESIGN CONSTRAINTS

There are currently no design constraints for the cook class.

3.4.3.5 PROCESSING DETAIL FOR EACH OPERATION

- FinishStep()
 - When a user is finished with the step that they are currently on, they select the next step button which activates this function. This is used to tell when the class should switch to the next step.
- StartTimer()
 - When the user wants to use the built in timer function in the cook interface, this command is run in order to start the timer as well as alert the user when the timer is finished.
- StopTimer()
 - The StopTimer function is used when the user either manually stops the timer by pressing the stop button or when the full amount of time on the timer expires. This should stop the timer from counting down anymore.
- Complete()

- Once the user has completed their recipe and reached the end or has stopped midway and closed the cook section of the website, this function is used in order to delete the temporarily stored data from recipe as well as bring them back to the main page of the website.

SECTION 4: USER INTERFACE DESIGN

The user interface consists of three main pages through which the user can interact with Recipe Buddy and its database. The pages are the “Home” page, the “Recipe” page, and the “Cook” page. The Home page will be a list of recipes containing brief descriptions and quick information about the Recipe. The Recipe page will contain the information related to that recipe and when editing will have text boxes appropriately places for their respective pieces of information. The Cook page will have a similar layout as the Recipe page but also display a timer and have next step and return buttons. The user will interact with these pages via a browser and our website.

4.1 DESCRIPTION OF THE USER INTERFACE

Each page will consist of various GUI components, such as buttons, labels, text fields, and list objects. These components will be arranged in such a way that the user will be able to quickly grasp the purpose of each page and perform whatever task it is designed for efficiently. A detailed description of these pages and their interactions with each other will be described in section 4.1.2.

4.1.1 OBJECTS AND ACTIONS

The next several pages describe and illustrate the following Recipe Buddy pages:

For all users:

- Home
- Recipe
- Cook
- Shopping Cart

For All Users


Home

GOATMEAL



Search

ADD






#tag1 #tag2 #tag3

gOatmeal

A brief description of gOatmeal.

Time
Cal.



- When the user comes to the site they will be greeted with the Home page. The Home Page displays in the center a list of recipes, each recipe contains a picture, a description, some related tags, as well as a calorie count and total cook time. Each recipe will also have buttons for cook (which will bring the user to the cook page), edit (which will bring the user to the Recipe page with the edit function), and delete (which will remove the recipe from the database and page). At the top of the home page there will be a button for adding a new recipe and a search bar for finding recipes by tags.

Recipe



Total Cook Time:
15 Minutes

Total Calories:
461430 Calories.

#tag1 #tag2 #tag3
#gOatmeal #goat



gOatmeal



Ingrediants

Oatmeal	4 Cups	150 Cal.
Goat	3 Units	461280 Cal.

Add Ingredient

Cooking Steps

Make the Oatmeal	7 Minutes
Add three whole goats (live)	5 Minutes
Mix thoroughly	3 Minutes

Add New Step

- Whether the user has clicked on a recipe or selected the edit button, or the add button, all will lead to a version of this page. This page contains a section for ingredients with information, such as name of ingredients, how much of it for the recipe, and the number of calories for the ingredient. It will also have a section for cooking steps with information, such as, step to perform and for how long. Along the left there will be an image of the recipe and below is a brief description followed by tags. Along the top will be buttons to return home, edit the recipe, and go to the Cook page for the recipe. Edit will have input boxes for each piece of information as well as buttons to add new ingredients and cooking steps. All the input boxes will be pre-populated with the information the recipe already had, there will also be a save button. The Add button will bring you to a completely blank edit page.

Cook



gOatmeal



Ingrediants

Oatmeal	4 Cups	150 Cal.
Goat	3 Units	461280 Cal.

4:29

Cooking Steps



Make the Oatmeal

7 Minutes



Add three whole goats (live)

5 Minutes



Mix thoroughly

3 Minutes

Complete

- The Cook page is designed in a way to make users feel familiar. On the left column is a picture of the recipe and below it is the timer with buttons to pause and resume it. The Cook page has the same top buttons as the Recipe page, as well as the same ingredient location and format. The cook page also has the same format for its cooking steps as the Recipe page but has the added button to end the step which removes it from the users view. The Cook page will also have a return to Recipe page button in the bottom right.

Shopping Cart

GOATMEAL

Shopping Cart

Tomato	2 whole	
Cheddar	2lbs bag	

- The shopping cart will display the list of ingredients the user has saved from all the recipes the user clicked “cook” on. The ingredients can be removed by clicking a button next to their name. A Button for returning to the Home page will be displayed in the top left.

4.2 INTERFACE AND DESIGN RULES

The interface design rules for Recipe Buddy are to be simple, modern, easy to use, and intuitive.

1. Simple and Intuitive Design
 - a. Every page should use the same theme and a similar layout. The Recipe page should be used for multiple functions only changing slightly to keep the design consistent and the user feeling they're in a familiar place when using new functions. Buttons should be large with matching colors or images so that any user can look and intuitively know what the buttons function is.
2. Offer Simple Error Handling
 - a. As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.
 - b. If fields are not properly filled out when a user presses the submit button for a page, they will be notified, and suggestions will be made to help them fix the mistake.

4.3 COMPONENTS AVAILABLE

HTML/CSS/JavaScript provides a plethora of useful GUI components. The components that Recipe will be using include the following:

- Flexbox
 - FlexBox is a CSS tool that allows CSS boxes to be made in a much more dynamic and intuitive motion, it will be used to generate the list of recipes on the Home page as well as the ingredients and cooking steps lists where those are used .
- Button
 - The Button component allows users to interact by pushing it. When a button is pushed it creates an event which can be handled by an EventHandler outside of the Button class.
 - Recipe Buddy will handle most of its human-computer interaction through buttons. Each page will have its own unique EventHandlers to handle the user's inputs.

SECTION 5: RESTRICTIONS, LIMITATIONS, AND CONSTRAINTS

A restriction placed upon us by the customer, is that RecipeBuddy will be used on FireFox 57+, and all versions of FireFox that succeeded FireFox 57+.

A limitation is time, due to time it is currently unknown if all core and additional features will be completed on time. We shall endeavor to complete all the core features, at least.

A limitation is the experience of the team. As we are not very experienced it shall be known that the implementations created by gOatmeal, may be lacking in certain areas.

A constraint is that RecipeBuddy requires an internet connection to use. This is due to the fact that RecipeBuddy is used on the FireFox Browser. This is important because if the connection goes down then the communication with the server and database will not occur, destroying the usability of RecipeBuddy in such a scenario.

SECTION 6: TESTING

Everything will be tested individually as the function is being implemented. Once everything has been completed it will all be put together to ensure that everything works with each other.

6.1 TESTING CASES AND EXPECTED RESULTS

We will be conducting the following tests.

6.1.1 White Box Testing

As each function is being implemented, it will be tested by the developer to ensure that it fits within the testing parameters defined below in feature testing. If the developer needs assistance, we have given each section of the project two team members, so they may converse with one another while minimizing our overhead.

6.1.2 Black Box Testing

This will happen once all of our white box testing has been completed. This will be a vigorous testing of all components put together. We will run through every feasible path the user may take when using our program and ensure that everything follows our intended output.

6.1.3 Feature Testing

The following subsections provide a brief overview of the testing process for each feature.

6.1.3.1 Server Connection

- Description: Connection has been successfully established with the server
 - Input: Device successfully connects to the server
 - Output: Device gets information about recipes from server
- Description: Device is unable to connect to the server
 - Input: Device is unable to connect to the server
 - Output: Popup alerting the user that they're offline and will be unable to fetch new recipes or alter current recipes

6.1.3.2 Add Recipe

- Description: Correct data input
 - Input: Valid recipe name
 - Output: Recipe is created on server. User sent back to the homepage
- Description: correct data input
 - Input: Duplicate recipe name that's already found in database
 - Output: Recipe is created and put into the database with a different ID
- Description: Incorrect data input
 - Input: Blank or none alphanumeric string
 - Output: Recipe not created, user asked for a valid recipe name

6.1.3.3 Edit Recipe

- Description: User clicks edit recipe
 - Input: Changes to the current fields
 - Output: Recipe gets updated in the database
- Description: Incorrect data input
 - Input: Invalid field
 - Output: Recipe not edited, user asked to make field valid

6.1.3.4 Delete Recipe

- Description: User clicks delete recipe
 - Input: Delete button is clicked
 - Output: Recipe is found on the database and deleted

6.1.3.5 Search

- Description: Valid search parameters
 - Input: A string that matches what they are looking for
 - Output: All recipes in the database whose name contains the string
- Description: Empty search parameter
 - Input: Empty string
 - Output: Every recipe

6.1.3.6 Cooking

- Description: Cook recipe button clicked
 - Input: Cook recipe button is clicked
 - Output: Screen transition to the cook recipe window with all appropriate information listed.

6.1.3.7 Add To Cart

- Description: Called by cook
 - Input: Called by the cook method
 - Output: Adds the ingredients of the recipe to the shopping cart

6.1.3.8 Add Ingredient

- Description: Add Ingredient button is clicked
 - Input: Add Ingredient button is clicked
 - Output: A popup window to add ingredient information
- Description: Valid input
 - Input: Valid (String, String, Int) tuple
 - Output: Ingredient is added to the recipe ingredient list
- Description: Invalid input
 - Input: Invalid (String, String, Int) tuple
 - Output: User is prompted to enter valid input

6.1.3.9 Add Step

- Description: Add step button clicked
 - Input: User clicks add step button
 - Output: User prompted to add the step
- Description: Valid (String, Float) pair entered
 - Input: Valid (String, Float) pair
 - Output: Step is added to the recipe
- Description: Invalid (String, Float) pair entered
 - Input: Invalid (String, Float) pair
 - Output: User is prompted to enter valid input

6.1.3.10 Remove Ingredient

- Description: Remove ingredient button is clicked
 - Input: Remove ingredient button is clicked
 - Output: Ingredient is removed from the list, database is updated

6.1.3.11 Remove Step

- Description: Remove step button is clicked
 - Input: Remove step button is clicked
 - Output: Step is removed from the list, database is updated

6.1.3.12 Edit Tags

- Description: User clicks edit tag button
 - Input: Edit tag button clicked
 - Output: Editable list of tags
- Description: Valid Input
 - Input: User inputs alphabetical string
 - Output: String added as a tag for the recipe
- Description: Invalid Input
 - Input: User inputs non-alphabetical string
 - Output: User prompted to enter valid alphabetical string

6.1.3.13 Finish Step

- Description: User clicks finished step
 - Input: User clicks the finish step button
 - Output: Current step deleted from list of steps to complete. Next step highlighted.

6.1.3.14 Start Timer

- Description: User clicks start timer button
 - Input: Start timer button clicked
 - Output: Timer created and displayed that shows current step duration

6.1.3.15 Stop Timer

- Description: User clicks stop timer button
 - Input: Stop timer button clicked
 - Output: Timer is deleted

6.1.3.16 Complete

- Description: User clicks complete button
 - Input: Complete button clicked
 - Output: User brought back to the recipe page

6.2 PERFORMANCE BOUNDS

Performance isn't of utmost importance to the Recipe Buddy application. We would like to limit how many hits to our database we create with our functions though. We don't have a

budget for databases, so our current one will have very limited performance that isn't designed for high volume usage.

If Recipe Buddy grows, we will need to upgrade the bandwidth of the server. Though that is unlikely to be a worry.

6.3 CRITICAL SYSTEMS

The main critical system of this website are the database. This will be what stores all of our recipe information and is absolutely critical to the seamless usage of the site. As such we will have to perform rigorous testing on the database using functions. If for some reason one of them fails to work at any point, it could cause immense damage.

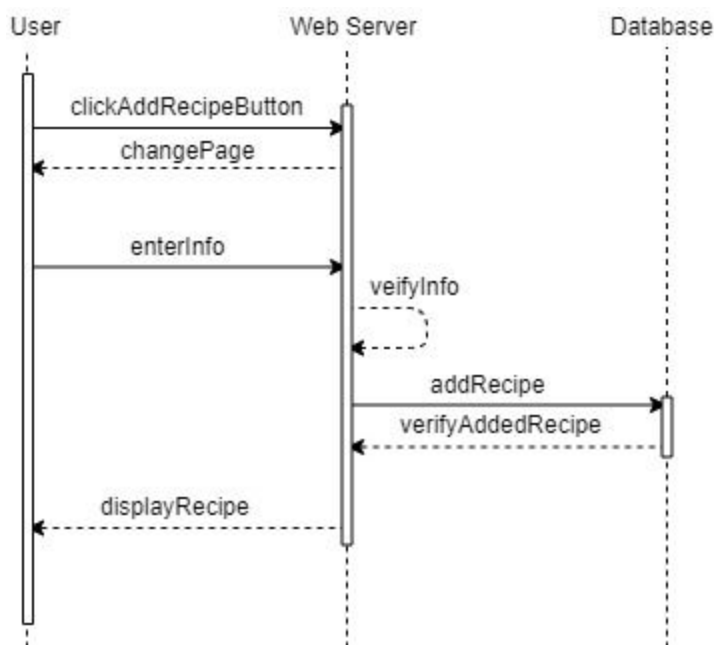
Section 7: Appendices

7.1 PACKAGING AND INSTALLATION ISSUES

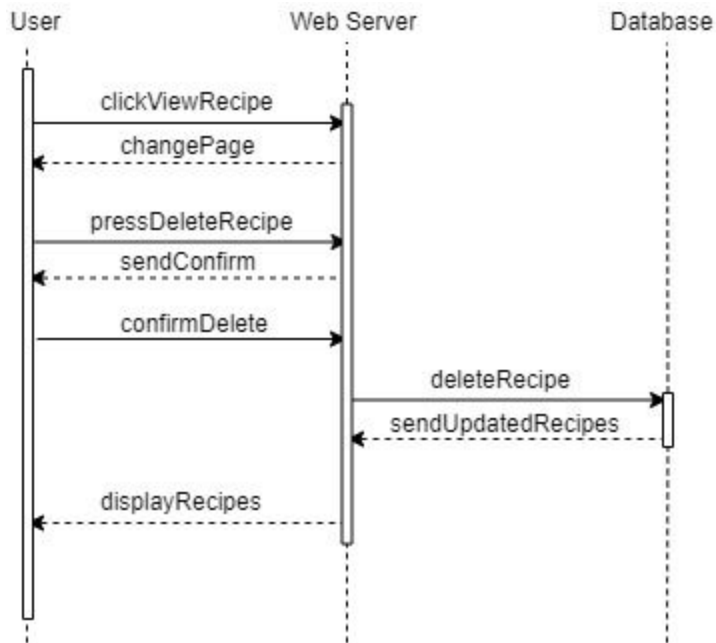
7.2 DESIGN METRICS TO BE USED

7.3 SEQUENCE DIAGRAMS

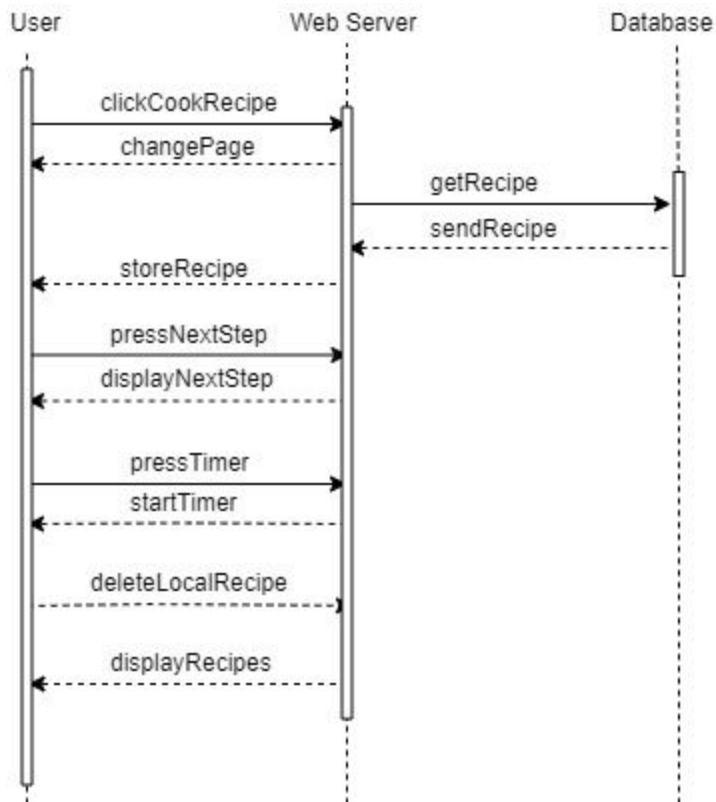
Add Recipe Sequence Diagram:



Delete Recipe Sequence Diagram:



Cook Recipe Sequence Diagram:



7.4 UML DIAGRAM