ChefItUp

Software Requirements Specifications.

CSE 308

Authors: Jay Verma
Kishore Thamilvanan
Rohit Luthra
Rohit Singh

Based on IEEE Std 830TM-1998 (R2009) document format

1 Introduction

The economy of modern-day America sees consistently and steadily rising costs of everyday utilities, food and rations, leaving the human race facing the uniquely negative predicament of extensive demand and crippling supply. It is in these times that one must look to reduce the expenditure of the limited resources that are available while ensuring safety in the kitchen. One such project aiming to achieve that purpose is ChefItUp.

ChefItUp is a purely web based platform where instructors/users can create original recipes, upload family and fan favorites, and design a virtual environment to simulate cooking in a purely virtual environment with an emphasis on fire safety and healthy food practices.

1.1 Purpose

The purpose of this document is to describe our blueprint of ChefItUp. This outline will help us to start developing and expanding upon the platform, and with regular reviewal of the specifications, we will be able to change and implement new technologies over time. The intended audience for this document is all the members of the development team- from the User Interface designers to the Software Engineers and even Salespeople, who will ultimately interface with potential customers and advertisers. This document serves as an agreement among all parties and a reference for how the application should ultimately be constructed. Upon completing the reading of this document, one should clearly be able to visualize how the platform will operate as well as what it ultimately will produce.

1.2 Scope

ChefItUp's goal is to allow consumers/students to educate themselves with countless cooking recipes all around the world, spanning different continents and cultures, without excessive utilization of limited and expensive resources and with proper regard for safety.

1.3 Definitions, acronyms, and abbreviations

Framework – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

GUI – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

IEEE – Institute of Electrical and Electronics Engineers, the "world's largest professional association for the advancement of technology".

UML – Unified Modeling Language, a standard set of document formats for designing software graphically.

UML Use Case Diagram – A UML document format that specifies how a user will interact with a system. Note that these diagrams do not include technical details. Instead, they are fed as input into the design stage (stage after this one) where the appropriate software designs are constructed based in part on the Use Cases specified in the SRS.

UML Class Diagram – A UML document format that specifies the structural design of an object oriented software system.

UML Sequence Diagram – A UML document format that specifies the dynamic design of a software system.

1.4 References

IEEE Std 830TM-1998 (R2009) – IEEE Recommended Practice for Software Requirements Specification

Wikipedia – Provides adequate definitions for UML Diagrams.

1.5 Overview

This Software Requirements Specifications document will define how ChefItUp will be presented to potential consumers and its inner workings. It is to be noted that this is not a software design description (SDD), which would decide how to construct the software using UML, but rather a means of describing its construction. This document does not specify how to build the appropriate technologies; it is simply an agreement that states how the aforementioned software will be built. Section 2 of this document will provide the context for the project and specify all the conceptual design. Section 3 will present what user interface controls are needed and all program functionality. Section 4 provides a Table of Contents, Index, and References.

2 Overall description

Are you a chef tired of trying to save your students from kitchen injuries because they don't listen to you, or are you someone who wants to be saved from those injuries and prepare yourself prior to having an hands-on experience to learn cooking? This website is a cooking simulator which has custom made recipes designed by trained chefs which enables students who are curious in learning how to cook to have a close to real experience.

2.1 Product perspective

This website provides a platform for chefs to create and store their recipes online for their students. For the students it will provide a real life like experience for them to learn how to cook. This website offers students an interactive learning atmosphere, where they can get the benefits of cooking without being in the vicinity of harmful equipment.

2.1.1 System Interfaces

Our website will have two different users: the instructors who design, edit, and upload new recipes and organize the necessary materials, and the students who learn from the recipes. The instructors will have a user interface where they specify the ingredients and steps for the recipes and the students will be provided with a UI that contains the instructions provided by the instructors.

2.1.2 User Interfaces

Use case number	Screen	Use Cases
1	Sign-up	New user signs-up.
2	Student Login	Student logs in.
3	Teacher Login	Instructor logs in.
4	Teacher Class Dashboard	Instructor creates a new class.
5	New Recipe Dashboard	Instructor creates a new recipe.
6	Edit Recipe Dashboard	Instructor edits an existing recipe.
7	Teacher Class Dashboard	Instructor searches for an existing class.
8	Teacher Class Dashboard	Instructor deletes an existing class.
9	Teacher Recipe Dashboard	Instructor searches for an existing recipe.
10	Teacher Recipe Dashboard	Instructor deletes an existing recipe.
11	Teacher New Recipe workspace.	Instructor publishes a new recipe.
12	Student Class Dashboard	Student searches for an existing class.
13	Student Recipe Dashboard	Student searches for an existing recipe.
14	Global Class Dashboard	Student subscribe to a class.
15	Student Class Dashboard	Student unsubscribe a class.
16	Student Recipe Dashboard	Cook a recipe.
17	Any screen	Student logs out.
18	Any screen	Instructor logs out.

Figure 2.2: Overview of Use-Case Diagrams.

Use Case 2.1: New user signs-up.

Use-Case:	New user sign-up
Primary actor:	New user (Can be a student or teacher)
Goal in context:	Lets the user create a new account
Preconditions:	None.
Scenario:	User wants to create an account.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #1
Frequency of use:	Every time a new user wants to access the website
Open issues.	None

Use Case 2.2: Student Login.

Use-Case:	Student Login.
Primary actor:	Student.
Goal in context:	Student is able to sign into their account.
Preconditions:	Student must have an account.
Scenario:	When the student wishes to access their account.
Exceptions:	If they have the wrong credentials.
Priority:	Essential, must be implemented.
When available:	Build #1
Frequency of use:	Everytime the student wishes to access the website.
Open issues.	None.

Use Case 2.3: Teacher Login.

Use-Case:	Teacher Login.
Primary actor:	Teacher.
Goal in context:	Teacher is able to sign into their account.
Preconditions:	Teacher must have an account.
Scenario:	When the student wishes to access their account.
Exceptions:	If they have the wrong credentials.
Priority:	Essential, must be implemented.
When available:	Build #1
Frequency of use:	Everytime the teacher wishes to access the website.
Open issues.	None.

Use Case 2.4: Teacher creates a new class.

Use-Case:	Teacher creates a new class.
Primary actor:	Instructor
Goal in context:	Instructor creates a class to add new recipes in.
Preconditions:	Must be logged in as an instructor
Scenario:	When the instructor wants to have a new class.
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #1
Frequency of use:	Every time the instructor wants to create a new recipe.
Open issues.	The limit of the number of classes displayed on one page.

Use Case 2.5: Teacher creates a new recipe.

Use-Case:	Instructor creates a new recipe.
Primary actor:	Instructor
Goal in context:	Instructor can add a new recipe to their class
Preconditions:	Must be logged in as an instructor
Scenario:	When the instructor wants to add a new recipe to their class
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #1
Frequency of use:	Every time the instructor wants to add a new recipe
Open issues.	Instructor must provide correct information with supporting images.

Use Case 2.6: Instructor edits a recipe.

Use-Case:	Instructor edits an existing recipe.
Primary actor:	Instructor.
Goal in context:	Instructor can edit an existing recipe in their class.
Preconditions:	Must be logged in as an instructor.
Scenario:	When the instructor wants to edit a particular recipe from their class.
Exceptions:	None.
Priority:	Essential, must be implemented
When available:	Build #2.
Frequency of use:	Every time the instructor wants to edit a recipe
Open issues.	Updating the new images in the database by deleting the previous ones.

Use Case 2.7: Teacher searches for an existing class.

Use-Case:	Instructor searches for an existing class.
Primary actor:	Instructor
Goal in context:	Instructor can search an existing class in their dashboard.
Preconditions:	Must be logged in as an instructor.
Scenario:	When the instructor wants to search for a particular class.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #2.
Frequency of use:	Every time the instructor wants to check or add or delete an ingredient
Open issues.	Place of the button still not decided

Use Case 2.8: Teacher deletes an existing class.

Use-Case:	Instructor deletes an existing class.
Primary actor:	Instructor
Goal in context:	Instructor is able to delete a class that they don't want any more.
Preconditions:	● Must be a Teacher ● Class must exist.
Scenario:	When the teacher wishes to delete a class.
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #3.
Frequency of use:	Everytime a teacher wants to discard a class.
Open issues.	Dynamically update the list of classes displayed in the dashboard and in the database.

Use Case 2.9: Instructor searches for an existing recipe.

Use-Case:	Instructor searches for an existing recipe.
Primary actor:	Instructor
Goal in context:	Teacher can search an existing class in their dashboard.
Preconditions:	Must be logged in as an instructor.
Scenario:	When the instructor wants to search for a particular class.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #2.
Frequency of use:	Every time the instructor wants to check or add or delete an ingredient
Open issues.	Place of the button still not decided

Use Case 2.10: Instructor deletes an existing recipe.

Use-Case:	Instructor deletes an existing recipe.
Primary actor:	Instructor
Goal in context:	Instructor can delete an existing class in their dashboard.
Preconditions:	● Must be logged in as an instructor. ● Class must exist.
Scenario:	When the instructor wants to delete a particular class.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #3.
Frequency of use:	Every time the instructor wants to check or add or delete an ingredient
Open issues.	Dynamically update the list of classes displayed in the dashboard and in the database.

Use Case 2.11: Instructor publishes a new recipe.

Use-Case:	Instructor publishes a new recipe.
Primary actor:	Instructor
Goal in context:	To publish a recipe that the instructor has worked on.
Preconditions:	Must be logged in as an instructor
Scenario:	When the instructor is done creating a recipe.
Exceptions:	Recipe can be without any images.
Priority:	Essential, must be implemented
When available:	Build #3
Frequency of use:	Every time the instructor is done with creating a recipe and wants to publish it.
Open issues.	Place of the button still not decided

Use Case 2.12: Student searches for an existing recipe

Use-Case:	Student searches for an existing recipe.
Primary actor:	Student
Goal in context:	Students can search for an existing recipe in their dashboard.
Preconditions:	Must be logged in as a Student .
Scenario:	When the instructor wants to search for a particular recipe.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #2.
Frequency of use:	Every time the Student wants to search for a particular recipe.
Open issues.	The search bar can be anywhere on top of the header of the website.

Use Case 2.13: Student searches for an existing class.

Use-Case:	Student searches for an existing class.
Primary actor:	Student
Goal in context:	Students can search for an existing class in their dashboard.
Preconditions:	Must be logged in as a Student .
Scenario:	When the instructor wants to search for a particular class.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #2.
Frequency of use:	Every time the Student wants to search for a particular class
Open issues.	The search bar can be anywhere on top of the header of the website.

Use Case 2.14: Student subscribe to a class.

Use-Case:	Student subscribe to a class
Primary actor:	Student
Goal in context:	Student is able to subscribe to a class from the global list of classes
Preconditions:	He can only select from pre existing classes.
Scenario:	● When the student has a new account and wishes to learn any recipe. ● When the student wishes to learn a recipe from a new class.
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #3
Frequency of use:	Everytime the student is done with a class or wishes to learn from another class.
Open issues.	Place of the button still not decided

Use Case 2.15: Student unsubscribe a class.

Use-Case:	Student unsubscribe a class
Primary actor:	Student
Goal in context:	Student is able to unsubscribe from a class from the list of classes from their dashboard
Preconditions:	The class should exist in the dashboard to be unsubscribed.
Scenario:	● When the student is wishes to expand their tastes ● Whenever they are bored or want a new class to learn from
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #3
Frequency of use:	Everytime the student is done with a class or wishes to learn from another class.
Open issues.	Place of the button still not decided

Use Case 2.16: Student cooks a recipe.

Use-Case:	Student Cooks a recipe.
Primary actor:	Student
Goal in context:	The Student cooks a recipe to its completion
Preconditions:	● The recipe should exist ● The student should have subscribed to the class containing it.
Scenario:	When the instructor wants to delete an item
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #3
Frequency of use:	Everytime a student wishes to practice a recipe.
Open issues.	The UI should be very easy flowing and Student friendly.

Use Case 2.17: Student logs out.

Use-Case:	Student logs out.
Primary actor:	Student.
Goal in context:	Student logs out of his account.
Preconditions:	● Must be a student. ● Must be logged in. ● Must have an account.
Scenario:	When the student wants to log out of his account.
Exceptions:	None.
Priority:	Essential, must be implemented.
When available:	Build #3.
Frequency of use:	Every time a student wants to log out of his account.
Open issues.	Should retain local window storage and session storage.

Use Case 2.18: Instructor logs out.

Use-Case:	Instructor logs out.
Primary actor:	Instructor
Goal in context:	Instructor logs out of his account.
Preconditions:	● Must be an Instructor. ● Must be logged in. ● Must have an account.
Scenario:	When the student wants to log out of his account.
Exceptions:	None
Priority:	Essential, must be implemented
When available:	Build #3
Frequency of use:	Every time a student wants to log out of his account.
Open issues.	Should retain local window storage and session storage.

2.1.3 Hardware Interfaces

The target platform is the PC. This application requires a lot of textual input, which requires the use of devices with keyboards and mouses.

2.1.4 Software Interfaces

This application should be runnable on any platform that has a web browser which is connected to the Internet. **ChefItUp** will be developed using Java and JavaScript. Since it is a traditional workspace-type application, it may be best to use the Desktop Java Framework. It is important to ensure the site exported by this application can be deployed to any Web server and work via any Web browser, notably Chrome and Firefox.

2.1.5 Communications Interfaces

In this product version, the application will generate the cooking interface, deploy it to a Web server, and store data in a cloud-based database. Therefore, these two connections must be monitored vigilantly and any bugs resolved immediately.

2.1.6 Memory Constraints

This application uses a manageable amount of user-provided data so this should not be a concern.

2.1.7 Operations

The data exported to JSON files must use the proper format according to current Web page requirements per the sample files given. Data will also be updated dynamically using Ajax.

2.1.8 Site Adaptation Requirements

N/A

2.2 Product functions

N/A

2.3 User characteristics

The primary users of this platform are the cooking instructors and the students. The instructors are expected to have adequate knowledge about cooking, while the students are assumed to have varying skills sets (ranging from Beginner-Intermediate).

2.4 Constraints

This platform prevents instructors from accessing student data and monitoring account activity in order to protect the privacy of subscribed users. On the other hand, the students cannot create, delete or edit recipes and are limited to practising their chosen recipes. This is not always possible, but should enforce rules when appropriate, like in the selection of variables, method arguments, and method return types. Limiting choice in these controls through combo boxes will help employ foolproof design principles.

2.5 Assumptions and dependencies

It is assumed that the pre-made recipes cannot be completed by the user unless they are prepared in the correct order with the proper ingredients, but clues/hints will be provided in the form of dialogue boxes in order to nudge the users in the right direction.

2.6 Apportioning of the Requirements

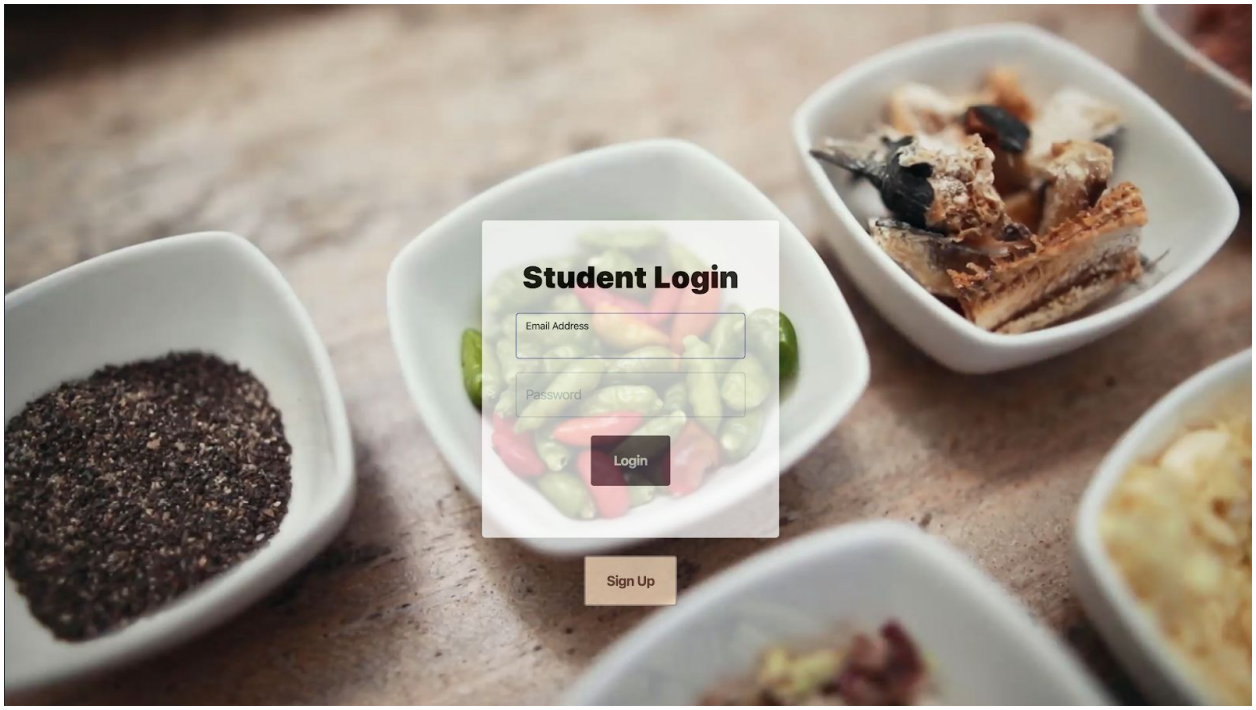
N/A

3 Specific requirements

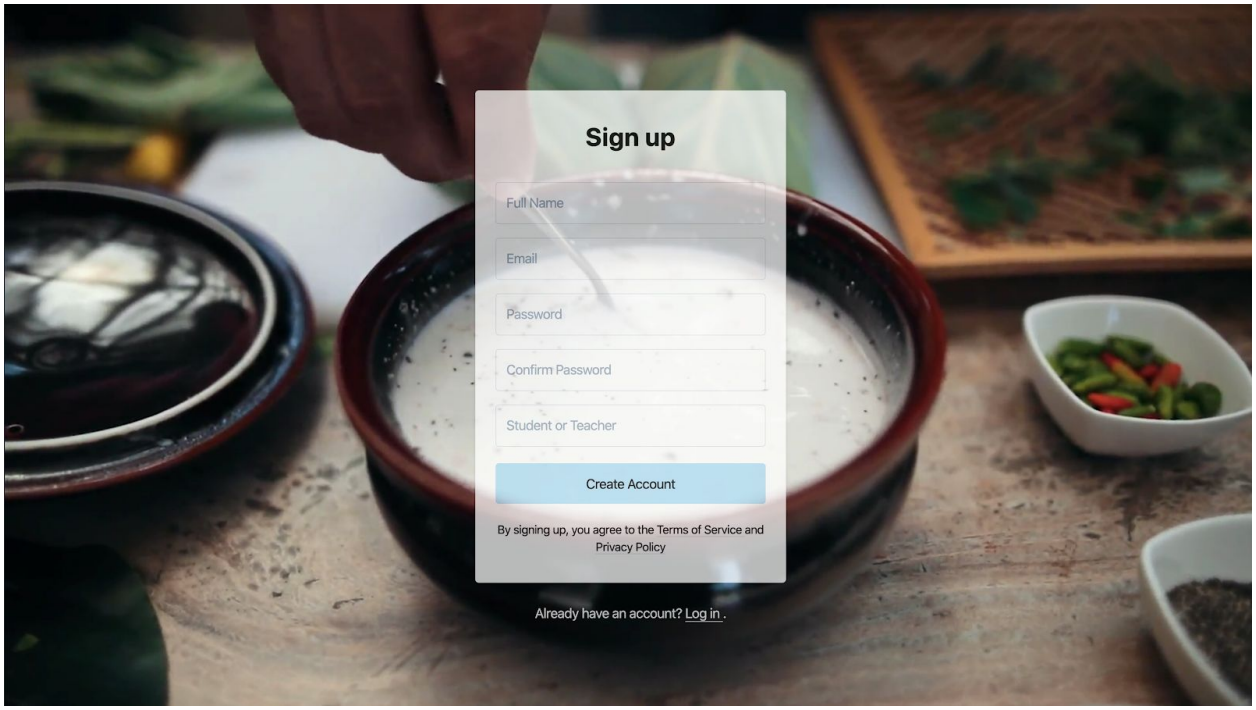
ChefItUp will be a web application that primarily involves the clicking and dragging of utensils in addition to occasional measuring and rationing of ingredients needed for preparing the recipe. This platform requires the user interface to be simple and easy to follow. Since ChefItUp is aimed to simulate the cooking environment as much as possible, the programmers should always be mindful of precise measurements and realistic visual appearance. Thereby, GUI and its controls should always be neatly drawn and outlined, precise, and carefully positioned.

3.1 External interfaces

Fig 3.1 welcome page



3.2 Student Login page



3.3 Student Sign up page

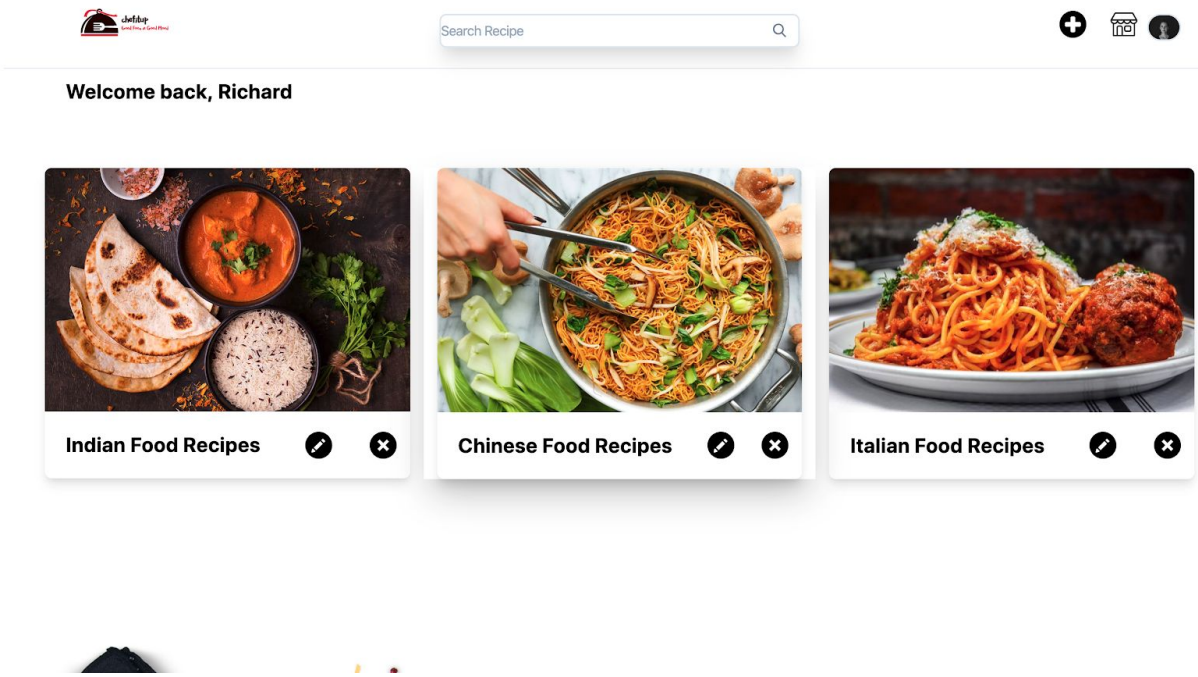


Fig: 3.1 Instructor's Homepage.

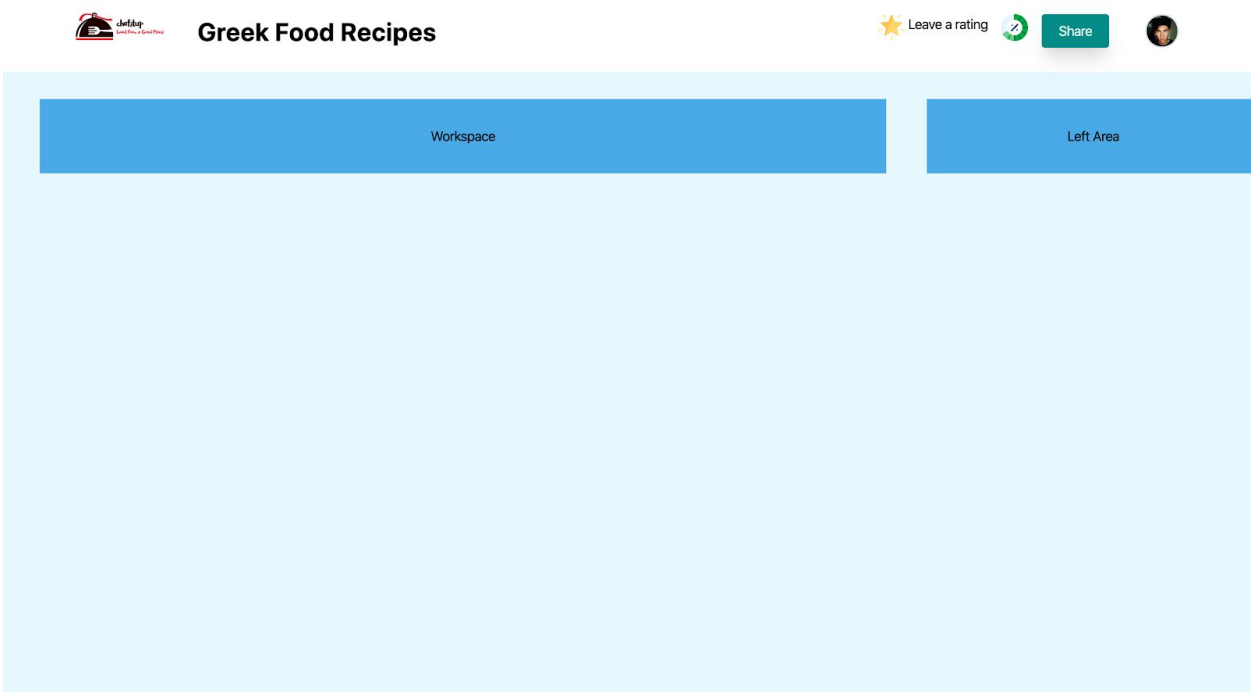


Fig: 3.2 Instructor's workspace while creating/editing a recipe.

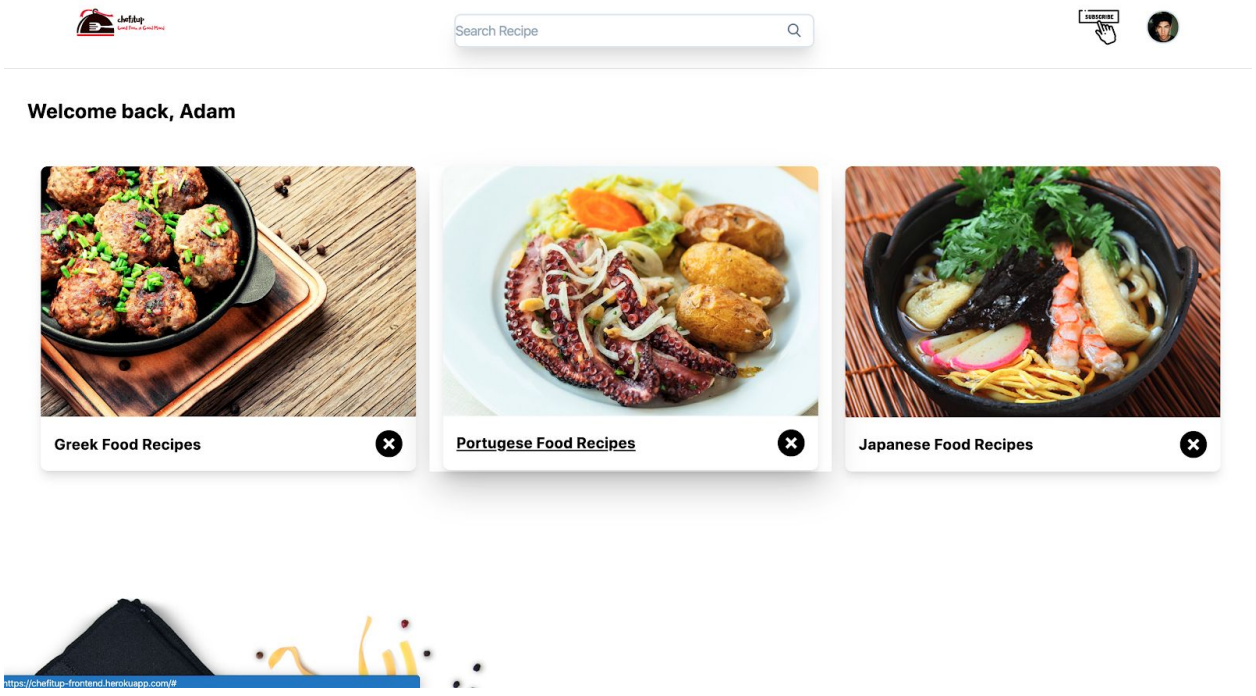


Fig: 3.3 Student's Homepage.

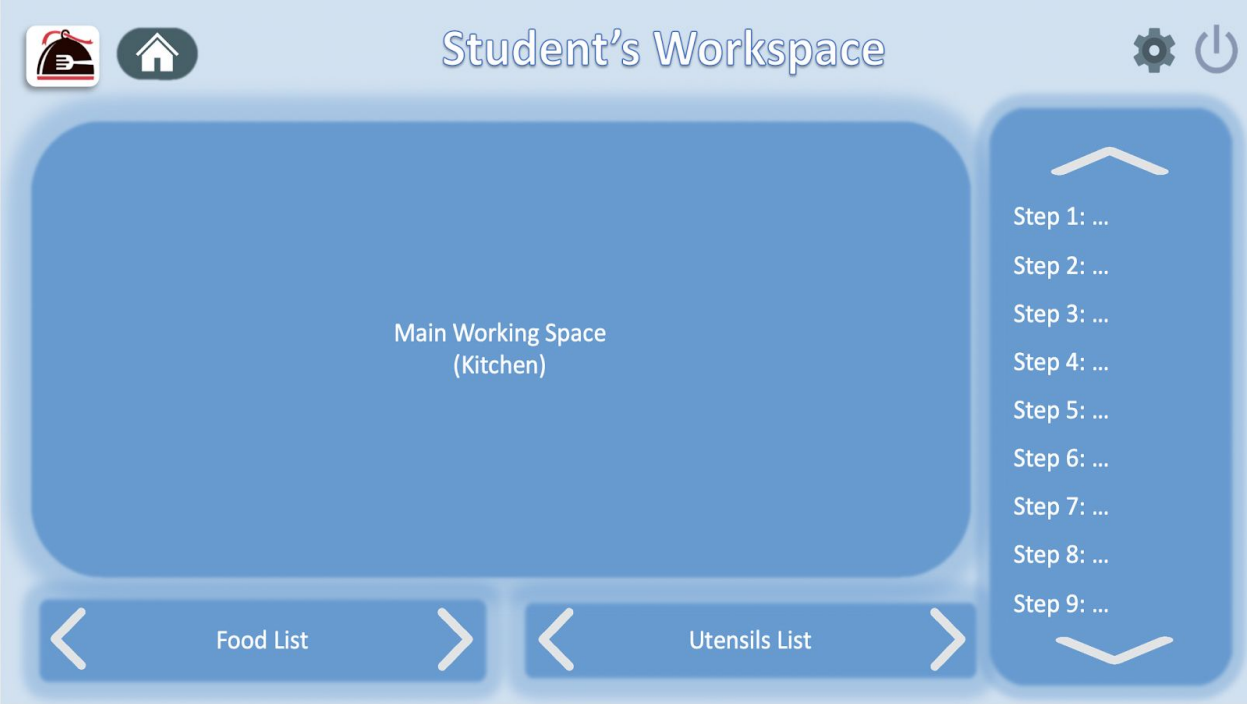


Fig: 3.4 Student's workspace.

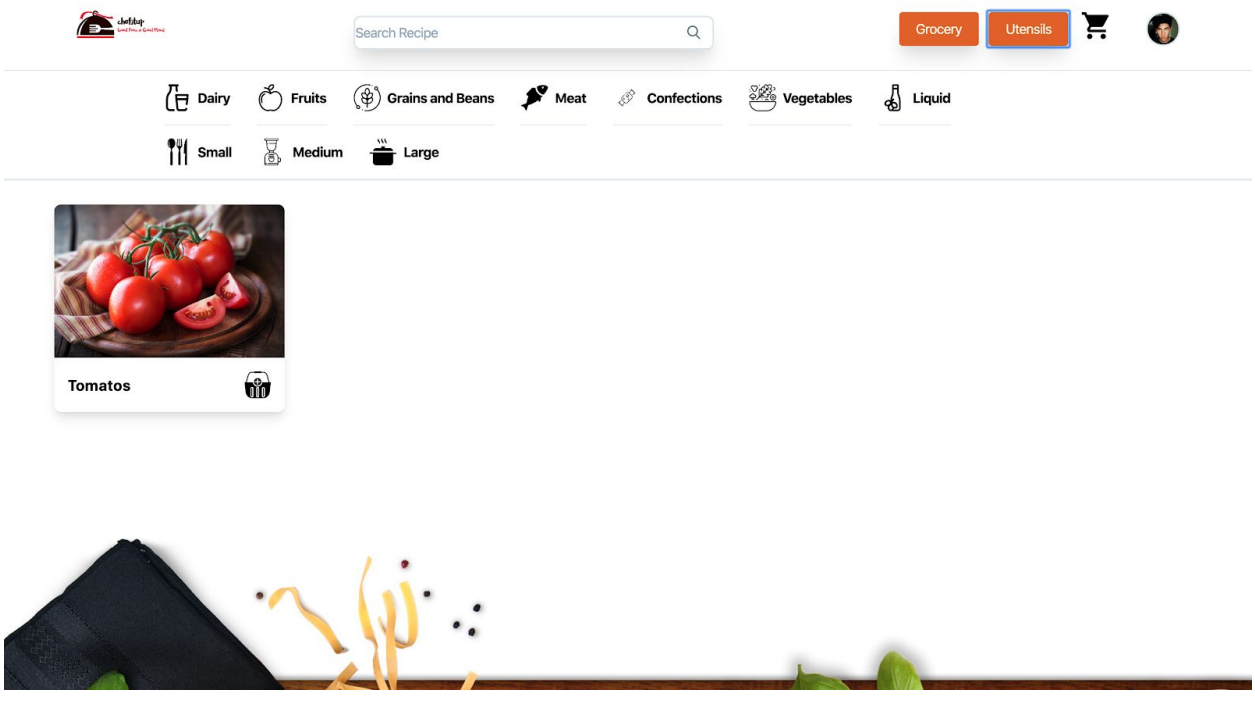


Fig: 3.4 Grocery store with ingredients.

3.1.2 Page Flow Diagram

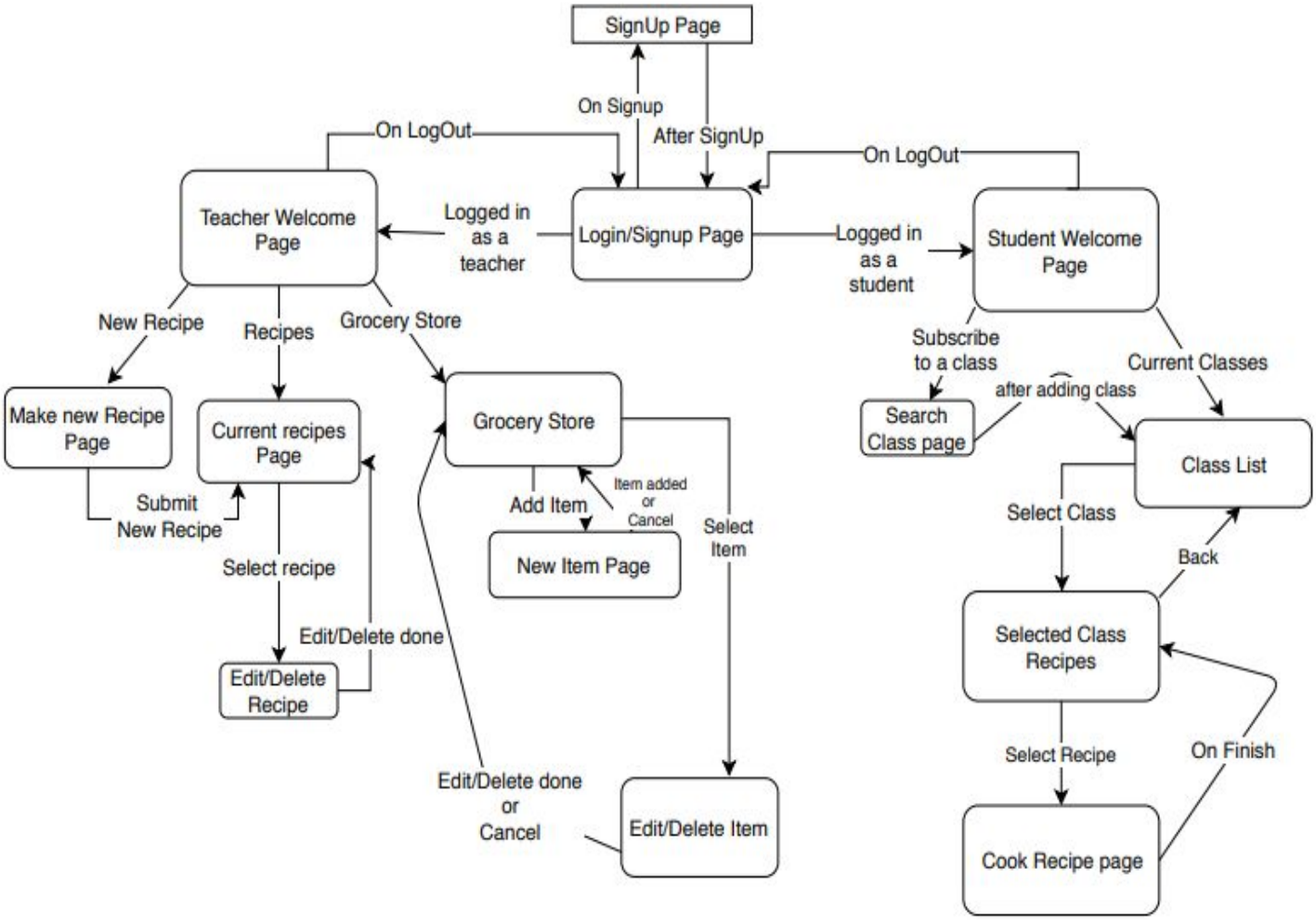


Fig. 3.1.1 Page Flow Diagram representing various modules in conjunction with each other.

3.2 Functions

One of the important features of this platform is that the user is constantly cued as to whether they are on the right track or not. Here are some cues that should be implemented:

- **Dialogue Boxes -** In order for them to comprehend their mistakes, if any, and work on it, the programmers should implement alert boxes or dialogue boxes which will precisely inform them of their error or if they are not following the instructions in the appropriate order.. The user is to be provided with a realistic outlook of potential negative consequences if these instructions are not carried out in a certain order.

- **Color Changes -** For the students to monitor their cooking process, this program should implement color change to represent the food being cooked.

- **Sound -** In the aim of providing a realistic environment for the student, the program should induce sound cues such as sizzling of the pan, the whistle of the kettle, timer of the microwave, etc.

3.3 Performance requirements

ChefItUp is not expected to crash, should handle exceptions dynamically, and be responsive throughout the completion of the course.

3.4 Logical database requirements

N/A

3.5 Design constraints

N/A

3.6 Software system attributes

As professionals, all members of this project must take this project seriously. The programmers are dedicated to developing robust software that exceeds the expectations of the customers. In order to achieve this level of quality, the product should be built with the following properties in mind:

3.6.1 Reliability – The program should be carefully planned, constructed and tested such that it behaves flawlessly for the end user. Bugs, including rendering problems, are unacceptable. In order to minimize these problems, all software will be carefully designed using UML diagrams and a Design to Test approach should be used for the Implementation Stage.

3.6.2 Availability – Customers may download and install the application for free.

3.6.3 Security – All security mechanisms will be addressed by future revisions

3.6.4 Extensibility – It is important that more features can be added to the application, so file formats for should be carefully considered such that the game can be easily extended.

1.6.5 Portability – To start with, the app will target desktop Java applications. Future ports may be as a Web app.

3.6.6 Maintainability – Update mechanisms will be addressed by future revisions.

3.7 Organizing the specific requirements

Note that the application is simple enough that an alternative arrangement of the content of this document is not necessary. The specific requirements for this application already fit neatly into the sections listed in the IEEE's recommended SRS format.

3.8 Additional comments

It is important to keep in mind that the UI designers, map creators, and sound designers should make updates to the game themes and content as needed to make something that looks appealing. It is up to their discretion to design all the interface controls in an effective, interactive style.

4 Supporting Information

Note that this SRS document should serve as a reference guide for the designers and programmers in the future stages of the development process. In order to help find important sections, a table of contents has been provided.

4.1 Table of contents

4.2 Appendixes

N/A