Tipsy

Software Requirements Specification

**Authors:** Immanuel Almosara, Ju-Hsin Chen, Rahul Sondhi, Bryan Valarezo

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

Copyright © 2019 Maroon

*No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

# **Introduction**

Have you ever wanted to learn the art of bartending? What are the various tools needed to create a good drink you could serve at a bar? Given the various forms of alcohol out there, we merge our creativity alongside it. The possibilities are endless. ***Tipsy*** is designed to teach you these skills! Users will be allowed to make their own bars, where they can then invite their friends to so they can attempt to recreate their own custom made recipes. This web application will serve as a simulator for creating various drinks of your own, or it can serve as a training tool for teaching aspiring bartenders how to create their own drinks. It teaches the ins and outs of creating your favorite alcoholic drinks right in the convenience of your own web browser.

## **Purpose**

The purpose of this document is to specify the detailed description of the *Tipsy* Web Application. The document will include the features including interface design and application functionality. The intended audience for this document is for the development team of the application and will be proposed to the instructor of CSE 308 -- Richard McKenna for his approval. This document serves as a design and an agreement for the developers of the software product. Upon completing the reading of this document, one should be able to clearly visualize how the application will look and operate as well as understand the way *Tipsy* is conducted.

## **Scope**

*Tipsy* will be a Web Application for users who want to learn the art of bartending. The user will be invited to a bar, and within the bars, there will be an example recipe of actual mixed drinks the user may learn from and recipes made by other users who were in the same bar. The users may be creative and make their own drinks, or further more, create their own bar. The web application should be able to provide the service of teaching the instructions on how to make common drinks we may see in real life.

## **Definitions, acronyms, and abbreviations**

**Alcohol** - a colorless volatile flammable liquid that is produced by the natural fermentation of sugars and is the intoxicating constituent of wine, beer, spirits, and other drinks, and is also used as an industrial solvent and as fuel.

**Author** - The creator of a recipe.

**Bar** - a counter across which alcoholic drinks or refreshments are served.  
**Bar Spoon** - A long-handled spoon used in bartending for mixing and layering of both alcoholic and non-alcoholic mixed drinks.

**Bottle opener** - A bottle opener is a device that enables the removal of metal bottle caps from glass bottles.

**Citrus Juicer** - A kitchen tool that extracts juice from citrus fruits and vegetables by shredding the flesh of the food item.

**Cocktail Strainer** - A cocktail strainer is a metal bar accessory used to remove ice from a mixed drink as it is poured into the serving glass.

**CSS** - Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

**Database -** a structured set of data held in a computer, especially one that is accessible in various ways.

**Foundation** - Foundation is a responsive front-end framework. Foundation provides a responsive grid and HTML and CSS UI components, templates, and code snippets, including typography, forms, buttons, navigation and other interface elements, as well as optional functionality provided by JavaScript extensions.

**Grater** - a device having a surface covered with holes edged by slightly raised cutting edges, used for grating cheese and other foods.

**HTML** - Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages.

**Java** - a general-purpose computer programming language designed to produce programs that will run on any computer system.

**Javascript** - an object-oriented computer programming language commonly used to create interactive effects within web browsers.

**Jigger** - a measure or small glass of spirits or wine.

**Knife** - a cutting instrument consisting of a sharp blade fastened to a handle.

**Manager** - A staff member of a bar who is able to add or remove workers.

**Mixing Glass** - a glass or metal container used to quickly chill cocktail drinks, primarily by stirring with ice using a spoon and straining with a strainer.

**Mixology -** the skill of mixing cocktails and other drinks.

**MongoDB** - an open source database management system (DBMS) that uses a document-oriented database model which supports various forms of data.

**NoSQL -** a class of database management systems (DBMS) that do not follow all of the rules of a relational DBMS and cannot use traditional SQL to query data.

**Owner -** The initial creator of a bar with full privileges of a bar.

**Pour spout** - A type of device and product created and designed in various colors, materials, mechanisms, shapes, sizes and styles used to attached to the side of a bowl, jar, pot, pan or similar shape utensil or dish.

**Sessions -** a server-side storage of information that is desired to persist throughout the user's interaction with the web site or web application.

**Spring -** A Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application**.**

**UML** - Unified Modeling Language (UML) is a standardized modeling language that enables developers to specify, visualize, construct and document the artifacts of a software system.

**Worker** - A user who is initially added by a bar owner or manager who can add their own recipes to a bar.

## **References**

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

## **Overview**

This SRS will clearly define how the *Tipsy* should look and operate. In Section 2 of this document, information of *Tipsy’s* content will be provided along with the specifications of all conceptual design. It will establish the technical requirements for this project. In Section 3, there will be the general design of the user interface layout and more specific details of the performance and database requirements. The section is written primarily for the developers for further instructions on how the web application should be put together. In Section 4, the table of contents, index and references will be provided.

# **Overall description**

## **Product perspective**

Given that this game is designed to teach new users how to create their own alcoholic drinks, it should be very easy to pick up and learn how to play. The raw gameplay of this web application will be revolved around the use of the mouse. The first step towards ensuring that the user gets the best learning experience possible is by simplifying the medium in which they learn from. At most the user would have to point and click, along with drag various items across the screen. The user will be taken to a bar which store multiple recipes in them. They will be given the task of making a drink, whether that be based off of a premade drink designed by another user or their own recipe.

## **Product functions**

**Use Case 2.1: Create Recipe**

|  |  |
| --- | --- |
| Use Case: | Create new Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wishes to make a new recipe and is logged in |
| Preconditions: | The application has been started |
| Scenario: | User clicks on the new Recipe Button |
| Exceptions: | Note that this should be a modal dialog so no other buttons or controls should be selectable until either “OK” or “Cancel” are selected |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | Size location and style of dialog should be finalized by the UI designer |

**Use Case 2.2: Edit Recipe**

|  |  |
| --- | --- |
| Use Case: | Edit Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wishes to edit a preexisting recipe |
| Preconditions: | The user is the author of the recipe |
| Scenario: | N/A |
| Exceptions: | Note that this should be a modal dialog so no other buttons or controls should be selectable until either “OK” or “Cancel” are selected |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | Size location and style of dialog should be finalized by the UI designer |

**Use Case 2.3: Clone/Copy/Extend Recipes**

|  |  |
| --- | --- |
| Use Case: | Clone/Copy/Extend Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wishes to duplicate an existing recipe |
| Preconditions: | The recipe in question should already exist and should be public |
| Scenario: | * User is on the Menu page on the recipe tab * The user clicks on the copy button |
| Exceptions: | Note that if the duplicated recipe’s author is the user, then the recipe should have a different name. |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.4: Search for Recipe**

|  |  |
| --- | --- |
| Use Case: | Search for Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to filter the recipes in a given bar by keyword |
| Preconditions: | The user is in the menu page on the recipe tab |
| Scenario: | * The user selects a bar to look in * The user types in the name of the recipe into the search bar * The user clicks search |
| Exceptions: | The app should keep track of the term typed in the search bar even after pressing the search button |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.5: Publish Recipe**

|  |  |
| --- | --- |
| Use Case: | Publish Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to publish a recipe to a bar |
| Preconditions: | The modal for creating a new recipe is popped up and the recipe should be not already be public |
| Scenario: | * The user has finished filling in all boxes for creating a new recipe * The user clicks on the publish button |
| Exceptions: | The user hasn’t filled out all the boxes before pressing on the publish button |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.6: Delete Recipe**

|  |  |
| --- | --- |
| Use Case: | Delete recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to delete a recipe from a bar |
| Preconditions: | The recipe that the user wants to delete exists and the user is the Author of the recipe |
| Scenario: | * The user clicks on a recipe * The user clicks on the delete button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.7: View Recipe**

|  |  |
| --- | --- |
| Use Case: | View Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to view a recipe before attempting to perform the drink |
| Preconditions: | The recipe in question should already exist and should be public unless they made the recipe |
| Scenario: | * The user clicks on the recipe that they want to view * A modal pops up and the users gets to see a preview of what the recipe will entail |
| Exceptions: | When this modal pops up no other function on the screen should work unless the user clicks on “Do Recipe” or “Cancel” |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.8: Perform Recipe**

|  |  |
| --- | --- |
| Use Case: | Perform Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to attempt to perform the recipe |
| Preconditions: | The recipe needs to be published. |
| Scenario: | * The user is on the view recipe modal * The user clicks on the “Perform Recipe” button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.9: Exit Recipe**

|  |  |
| --- | --- |
| Use Case: | Exit Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to quit the recipe |
| Preconditions: | The user is on the Bar page and should have the recipe open |
| Scenario: | * The user is on the Bar page * The user clicks on the Quit button |
| Exceptions: | If the user has not saved progress, a modal will pop up asking if the user wants to save changes before quitting |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.10: Debug Recipe**

|  |  |
| --- | --- |
| Use Case: | Debug Recipe |
| Primary Actor: | User |
| Goal in Context: | The user wants to debug the recipe |
| Preconditions: | The user is on the Recipe creation page for the recipe they want to debug. The user is also the author of the recipe. |
| Scenario: | * The user is on the Recipe creation page * The user clicks on the Debug recipe button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.11: Create Bar**

|  |  |
| --- | --- |
| Use Case: | Create Bar |
| Primary Actor: | User |
| Goal in Context: | The user wants to create a new bar |
| Preconditions: | The user has successfully made an account and is signed in |
| Scenario: | * The user is on the bar menu * The user clicks on the new bar button * A modal pops up and the user fills in the required fields for the bar |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.12: Add Worker to Bar**

|  |  |
| --- | --- |
| Use Case: | Add Worker to Bar |
| Primary Actor: | User |
| Goal in Context: | A user wants to add another user to a bar |
| Preconditions: | The user that’s adding another user to the bar is the owner or a manager of the bar and the user being added is not already in the bar |
| Scenario: | * The user clicks on a bar that they own or manage * The user types in the email of the user they want to add and clicks on the “Add worker to bar” |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.13: Remove Worker from Bar**

|  |  |
| --- | --- |
| Use Case: | Remove Worker from Bar |
| Primary Actor: | User |
| Goal in Context: | A user wants to delete a worker from a bar |
| Preconditions: | The user that’s removing another user from the bar is the owner or a manager of the bar and the user being added is a worker of the bar |
| Scenario: | * The user clicks on a bar that they own or manage * The user goes to the worker entry that they want to remove and clicks on the red X button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.14: Add/Assign Recipe to Bar**

|  |  |
| --- | --- |
| Use Case: | Add/Assign Recipe to bar |
| Primary Actor: | User |
| Goal in Context: | The user wants to add a recipe to a bar |
| Preconditions: | * The bar exists, and the user is the owner, a manager, or a worker of the bar * Recipe needs to be published |
| Scenario: | * The user is on the menu and clicks on a bar * The user clicks on the plus sign on the recipes section to add a new recipe * The user clicks on the recipe they want to add and clicks on the “OK” button to confirm their decision |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.15: Remove Recipe from a Bar**

|  |  |
| --- | --- |
| Use Case: | Remove Recipe from a Bar |
| Primary Actor: | User |
| Goal in Context: | The user wants to add a recipe to a bar |
| Preconditions: | * The bar exists, and the user is the owner, a manager, or a worker of the bar * Recipe needs to be published and added to the bar |
| Scenario: | * The user is on the menu and clicks on a bar * The user clicks on the red x button next to the recipe they want to remove from the bar |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.16: Save/Verification of Recipe**

|  |  |
| --- | --- |
| Use Case: | Save/Verification of Recipe |
| Primary Actor: | User |
| Goal in Context: | The author wants to save a recipe after editing it |
| Preconditions: | The user has made changes to the recipe and the user is the author of the recipe |
| Scenario: | * The user is on the edit recipe menu and has made changes * The user clicks on the save recipe button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.17: View everything**

|  |  |
| --- | --- |
| Use Case: | View everything |
| Primary Actor: | Admin |
| Goal in Context: | The admin wants to view all the users, bars, and recipes in the database |
| Preconditions: | The user is an admin |
| Scenario: | * The admin is in the menu page * The admin clicks on the Admin tab |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Third benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.18: Log in**

|  |  |
| --- | --- |
| Use Case: | Log in |
| Primary Actor: | User |
| Goal in Context: | The user wants to log into their account |
| Preconditions: | The account is registered on the database |
| Scenario: | * The user is on the Login Page * The user types in their username and password into their respective fields * The user clicks on the “login” button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | Used every time the user wants to log in to their account |
| Open Issues: | N/A |

**Use Case 2.19: Log out**

|  |  |
| --- | --- |
| Use Case: | Log out |
| Primary Actor: | User |
| Goal in Context: | The user wants to log out of their account |
| Preconditions: | * The user is already logged in on an account * The user is not on the lab or login page |
| Scenario: | * The user is logged into an account * The user clicks on the Log out button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | Once per session |
| Open Issues: | N/A |

**Use Case 2.20: Register for account**

|  |  |
| --- | --- |
| Use Case: | Register for account |
| Primary Actor: | User |
| Goal in Context: | The user wants to register for an account |
| Preconditions: | The user is on the Register page |
| Scenario: | * The user is on the Login page * The user clicks on the Register button * The user fills out the required boxes and clicks on the register button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | Once per session |
| Open Issues: | N/A |

**Use Case 2.21: Forgot password**

|  |  |
| --- | --- |
| Use Case: | Forgot password |
| Primary Actor: | User |
| Goal in Context: | The user forgot their password and wants to recover it through their email |
| Preconditions: | * The user is on the Login page * There must be an account on the database associated with the user’s email address |
| Scenario: | * The user is on the Login page * The user clicks on the Forgot Password button * The user is taken to the forgot password page and fills in their email address * The user receives a recovery email and clicks on the link to reset their password |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | First benchmark |
| Frequency of use: | Once per session |
| Open Issues: | N/A |

**Use Case 2.22: Save progress**

|  |  |
| --- | --- |
| Use Case: | Save progress |
| Primary Actor: | User |
| Goal in Context: | The user wants to save their progress on the recipe they’re currently working on |
| Preconditions: | The user has made some progress in their recipe |
| Scenario: | * The user is currently working on a recipe and has made progress |
| Exceptions: | The user will not be able to save if no progress has been made (glass is empty, ingredients haven’t been prepared, etc.) |
| Priority: | Essential, must be implemented |
| When available: | Third benchmark |
| Frequency of use: | Many times per session |
| Open Issues: | N/A |

**Use Case 2.23: Upload equipment**

|  |  |
| --- | --- |
| Use Case: | Upload equipment |
| Primary Actor: | User |
| Goal in Context: | The user wants to upload their own equipment to the game |
| Preconditions: | * The user has a png to upload * The user is on the recipe information detail page |
| Scenario: | * The user is on the recipe information detail page * The user clicks on the plus sign on the equipment section * The user selects the picture from their file manager and fills in all the required boxes and clicks on the upload button |
| Exceptions: | The user will not be able to upload their own equipment if they don’t have a png available to upload |
| Priority: | Essential, must be implemented |
| When available: | Third benchmark |
| Frequency of use: | A few times per session |
| Open Issues: | N/A |

**Use Case 2.24: Add steps**

|  |  |
| --- | --- |
| Use Case: | Add steps |
| Primary Actor: | User |
| Goal in Context: | The user wants to add steps to a recipe they own |
| Preconditions: | * The user is on the recipe creation page |
| Scenario: | * The user is on the recipe creation page * The user clicks on the add steps button * The user fills in the required fields to create a step * The user clicks on the “add step” button |
| Exceptions: | N/A |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Multiple times per session |
| Open Issues: | N/A |

**Use Case 2.25: Edit step**

|  |  |
| --- | --- |
| Use Case: | Edit step |
| Primary Actor: | User |
| Goal in Context: | The user wants to edit a step in a recipe they own |
| Preconditions: | * The user is on the recipe creation page * The step must exist in the recipe |
| Scenario: | * The user is on the recipe creation page * The user clicks on the edit steps button * The user selects the step they want to edit * The user edits the fields they want to edit in a step and clicks on the save button |
| Exceptions: | The user will not be able to edit a step if the step has not been modified from what it was previously |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Multiple times per session |
| Open Issues: | N/A |

**Use Case 2.26: Delete step**

|  |  |
| --- | --- |
| Use Case: | Delete step |
| Primary Actor: | User |
| Goal in Context: | The user wants to delete a step in a recipe they own |
| Preconditions: | * The user is on the recipe creation page * The step must exist in the recipe |
| Scenario: | * The user is on the recipe creation page * The user clicks on the delete steps button * The user clicks on the red x button next to the step they want to delete |
| Exceptions: | The user will not be able to edit a step if the step has not been modified from what it was previously |
| Priority: | Essential, must be implemented |
| When available: | Second benchmark |
| Frequency of use: | Multiple times per session |
| Open Issues: | N/A |

## **User characteristics**

The user interface should aim to be as user-friendly as possible, using the principles of fool-proof design as well as sound UI/UX design principles. The instructions of the tool usages should be clear and the ingredients should be easily recognizable as its real-life object.

## **Constraints**

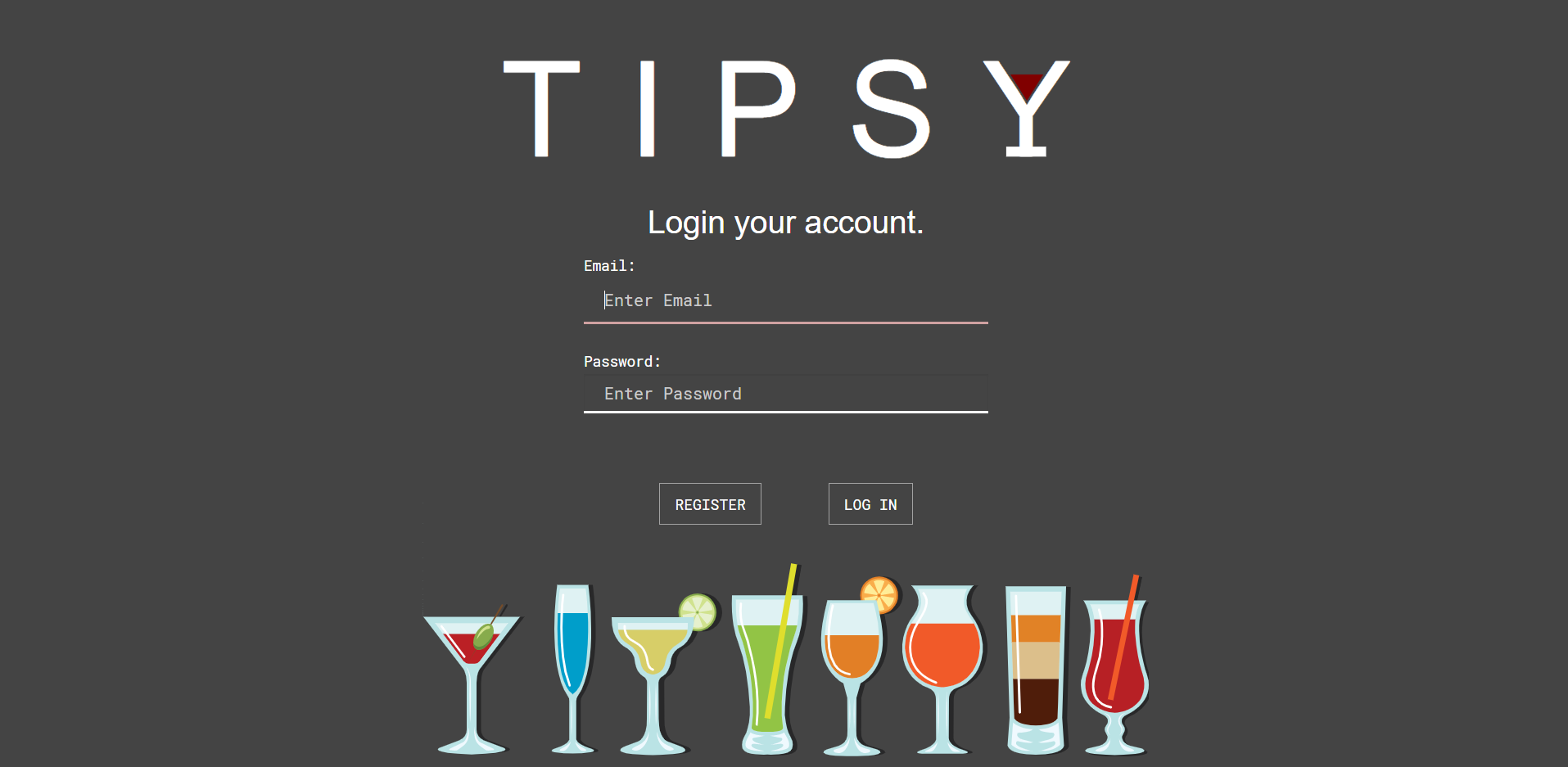
Note that this simulator will only teach you how to make a few drinks, after learning those examples you will have the choice to make drinks based on your own design. It’s also important that when learning these examples, the tools provided in the application prevents the user from taking the wrong actions to ensure that they are learning how to make the drink properly.

## **Assumptions and dependencies**

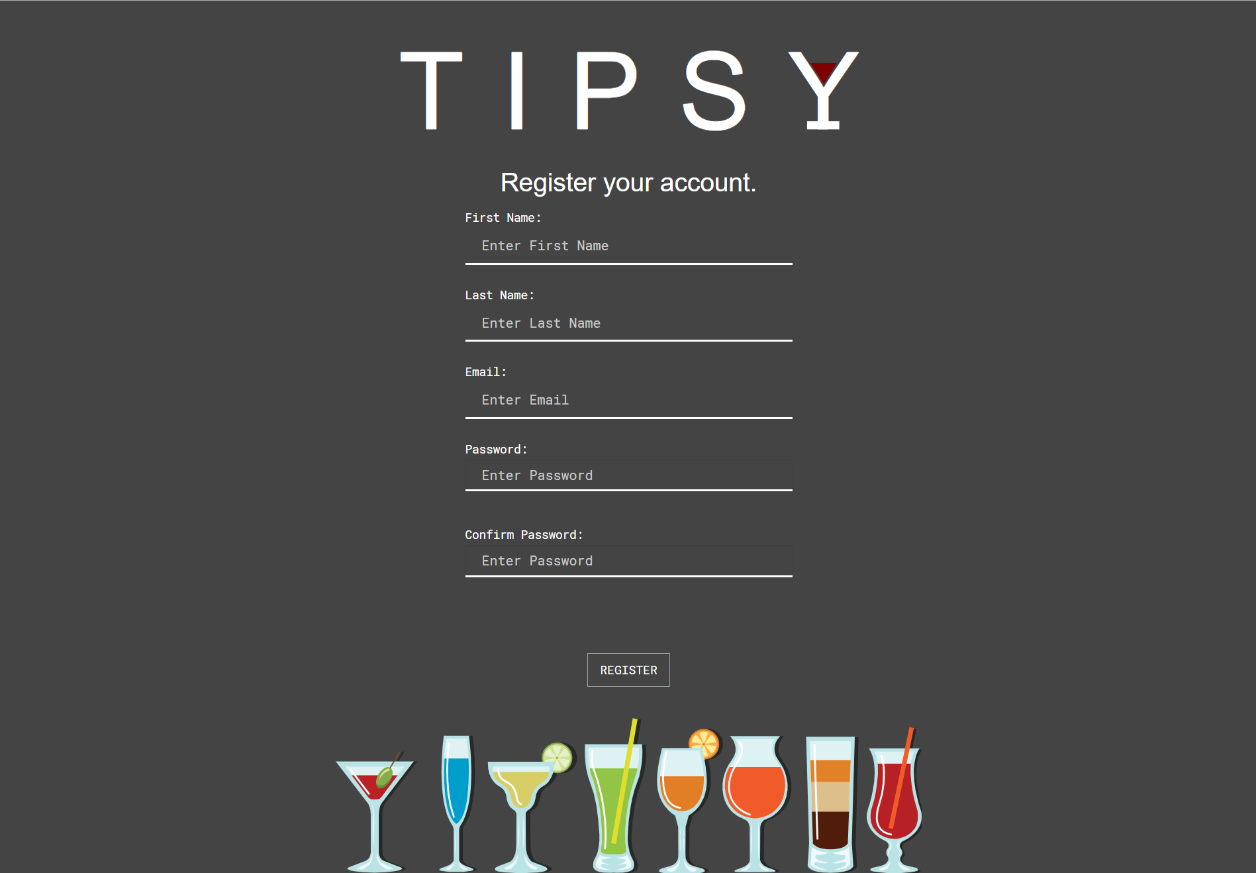
It is assumed that the user has access to the internet and can access this web application through a web browser.

# **Specific requirements**

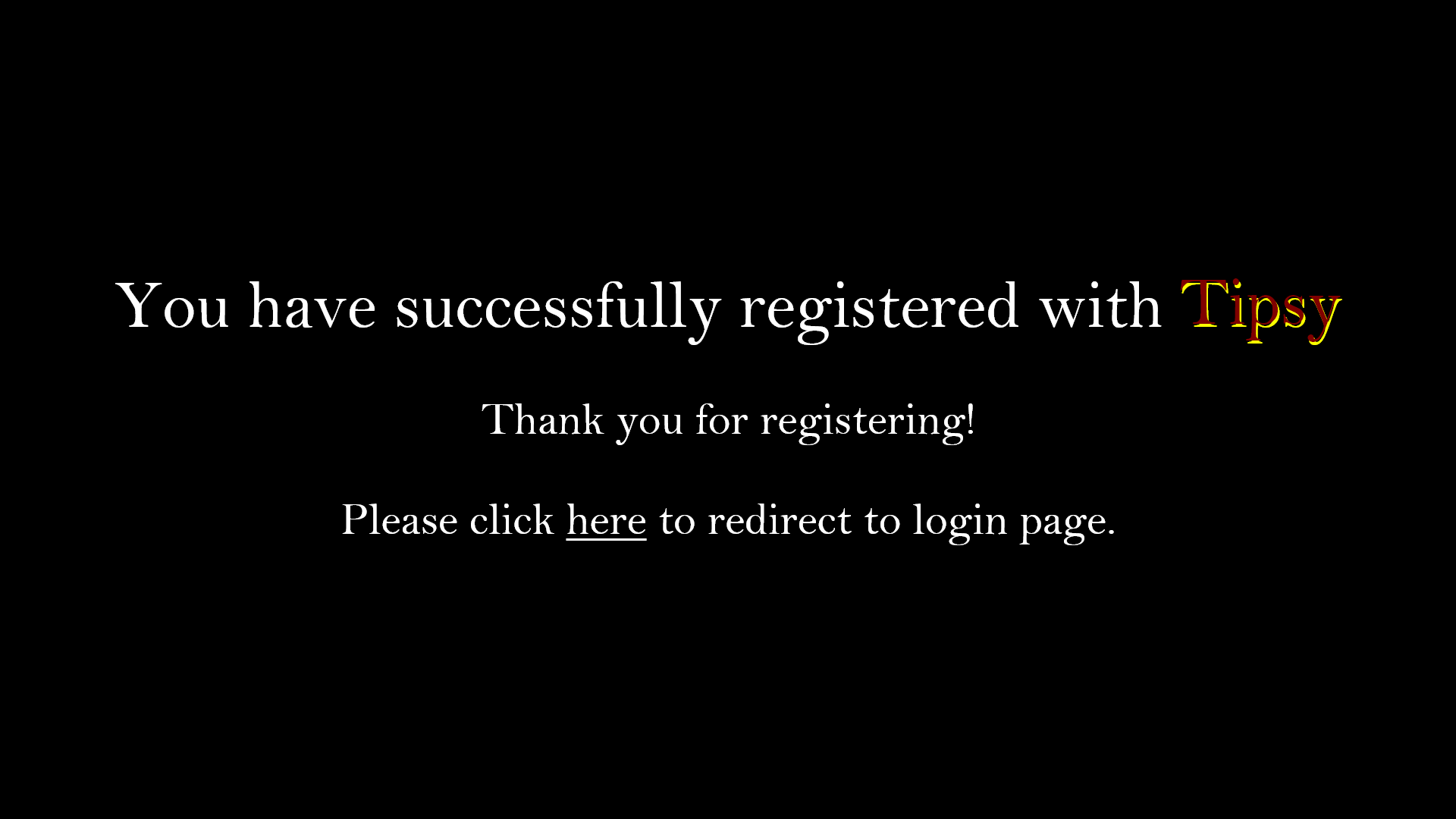
## **External interfaces**

****

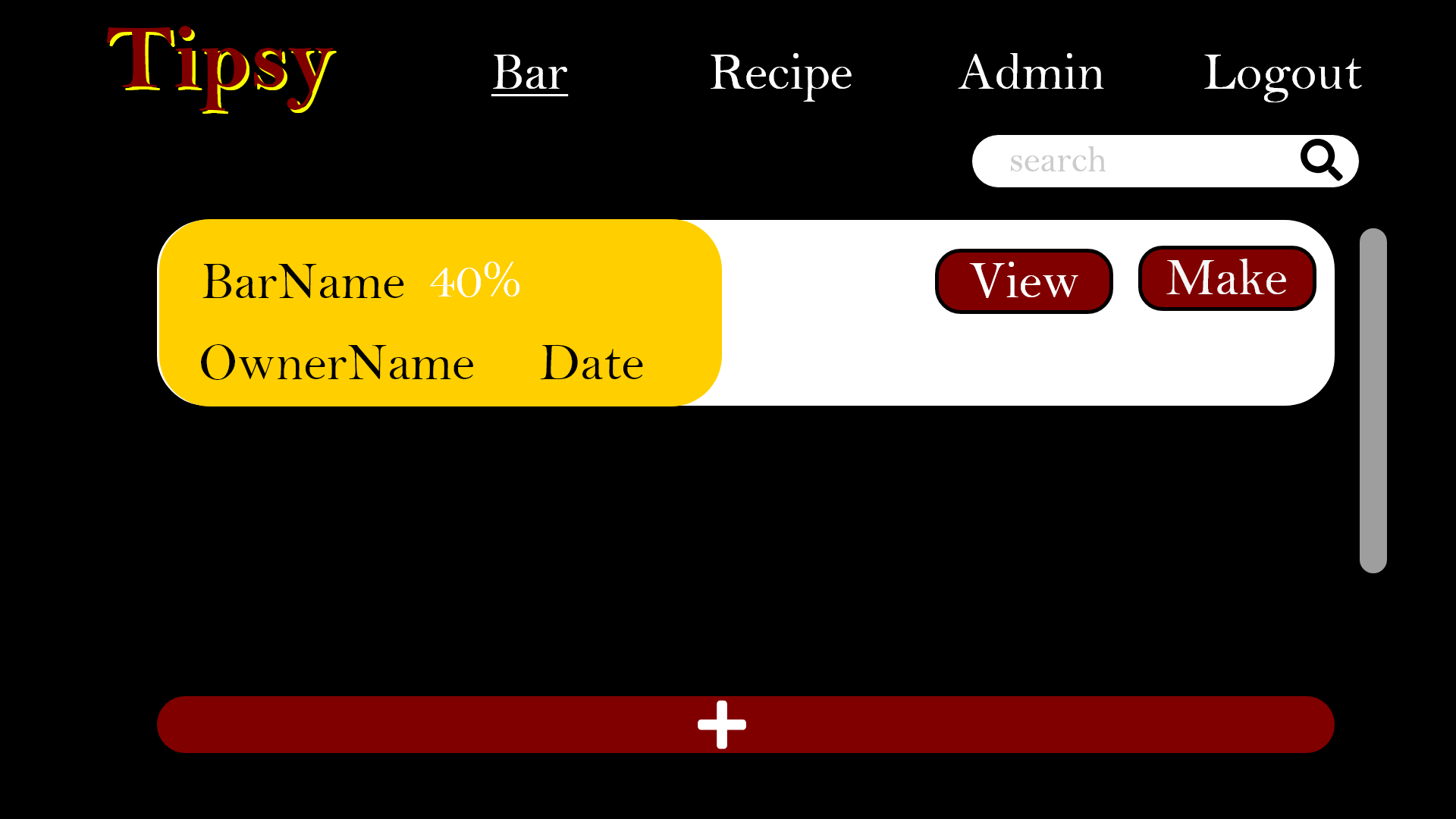
**Figure 3.1: Login Page**

****

**Figure 3.3: Register page**

****

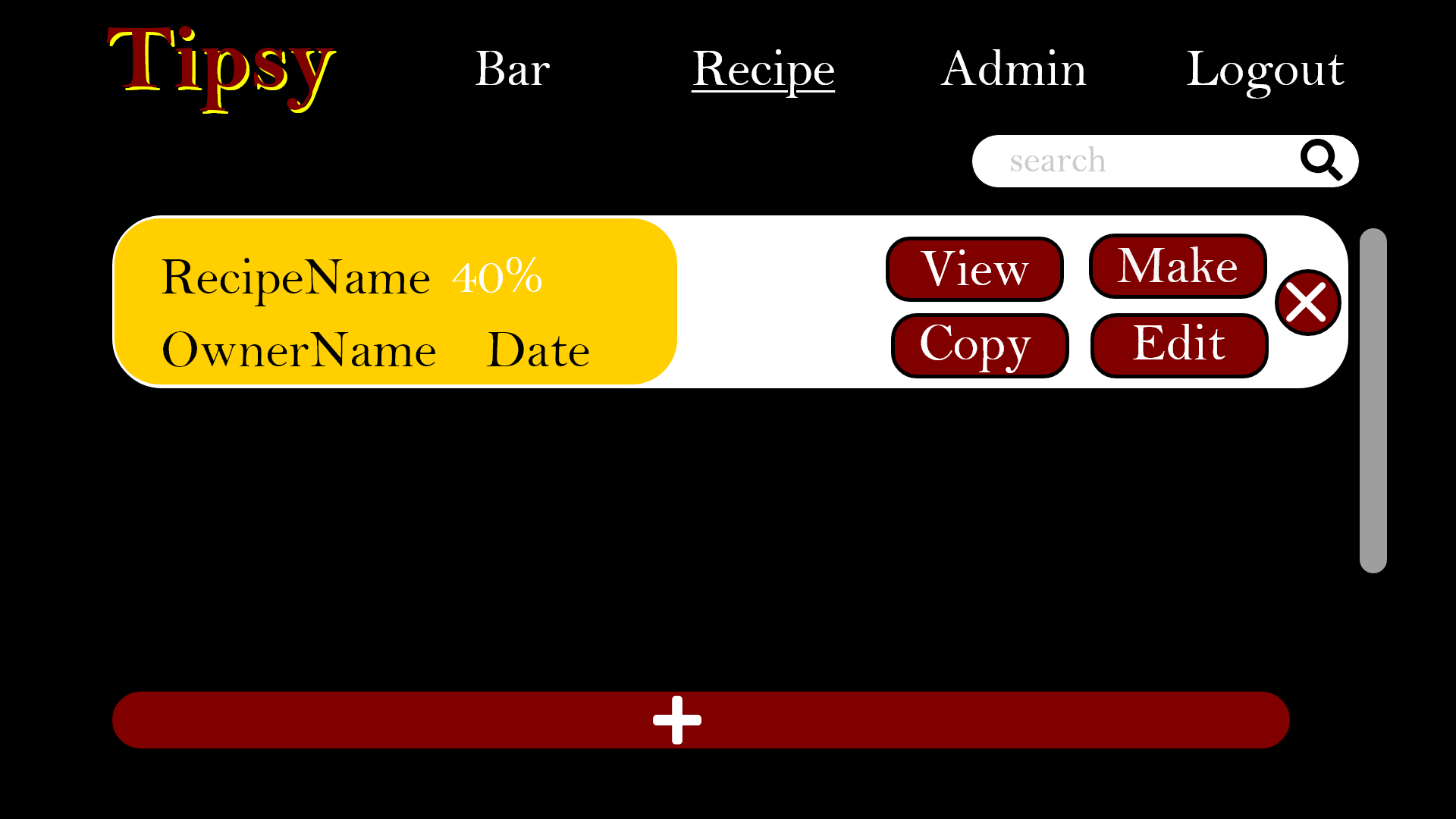
**Figure 3.4: Confirmation page**

****

**Figure 3.5: Menu Page, Bar tab**

****

**Figure 3.6: Bar information detail page**

****

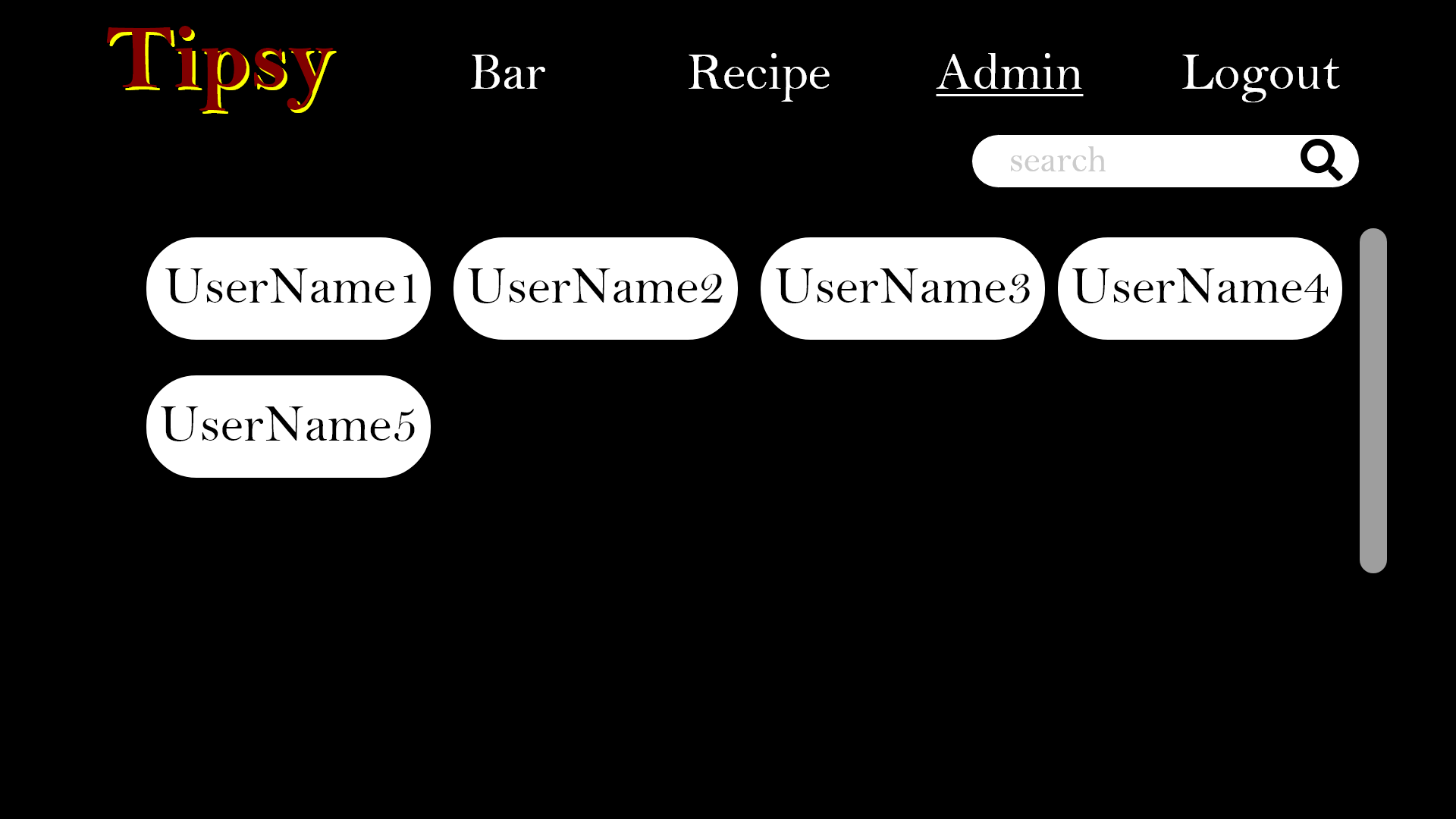
**Figure 3.7: Menu Page, Recipe tab**

****

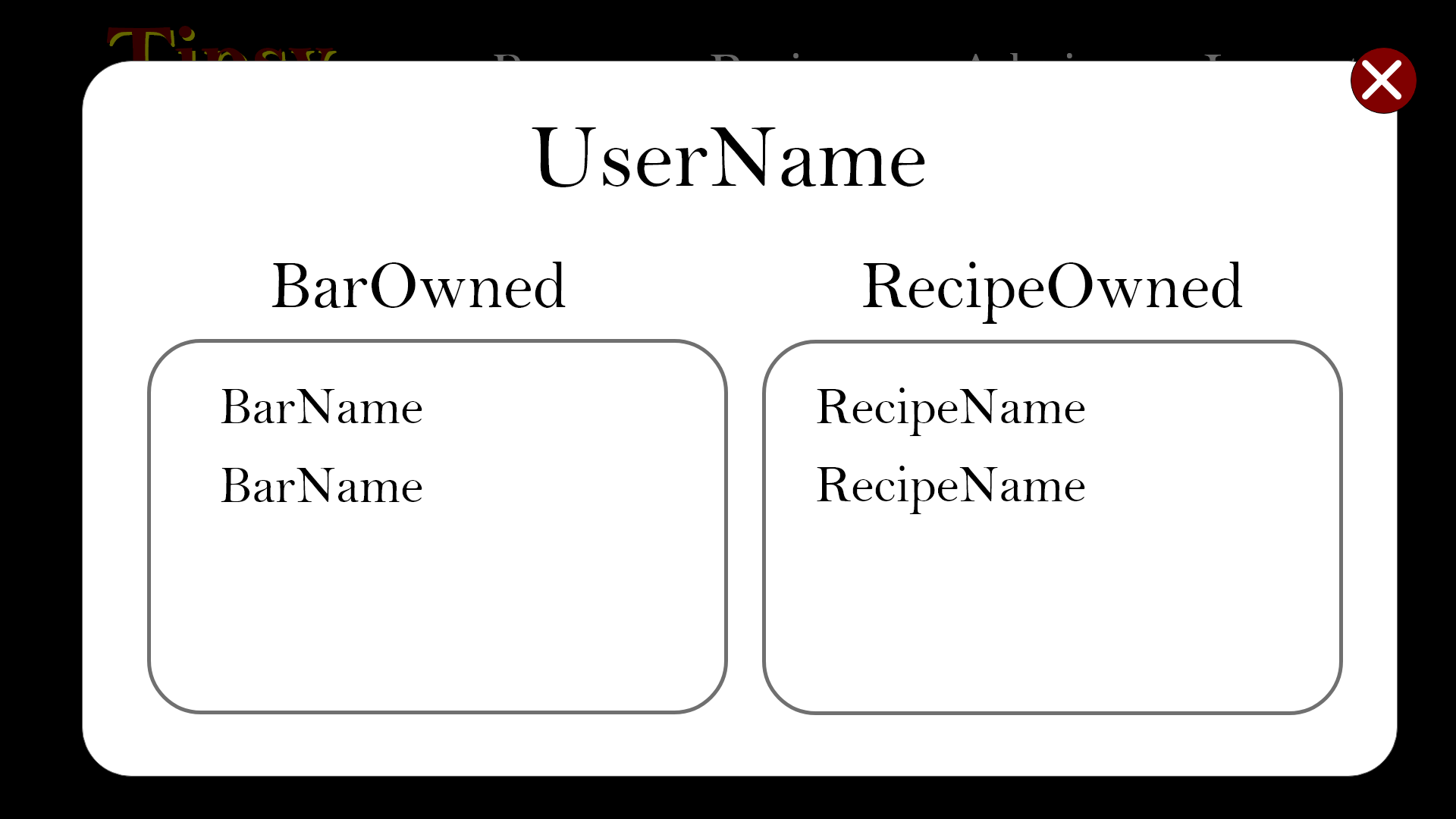
**Figure 3.8: Recipe information detail page**

****

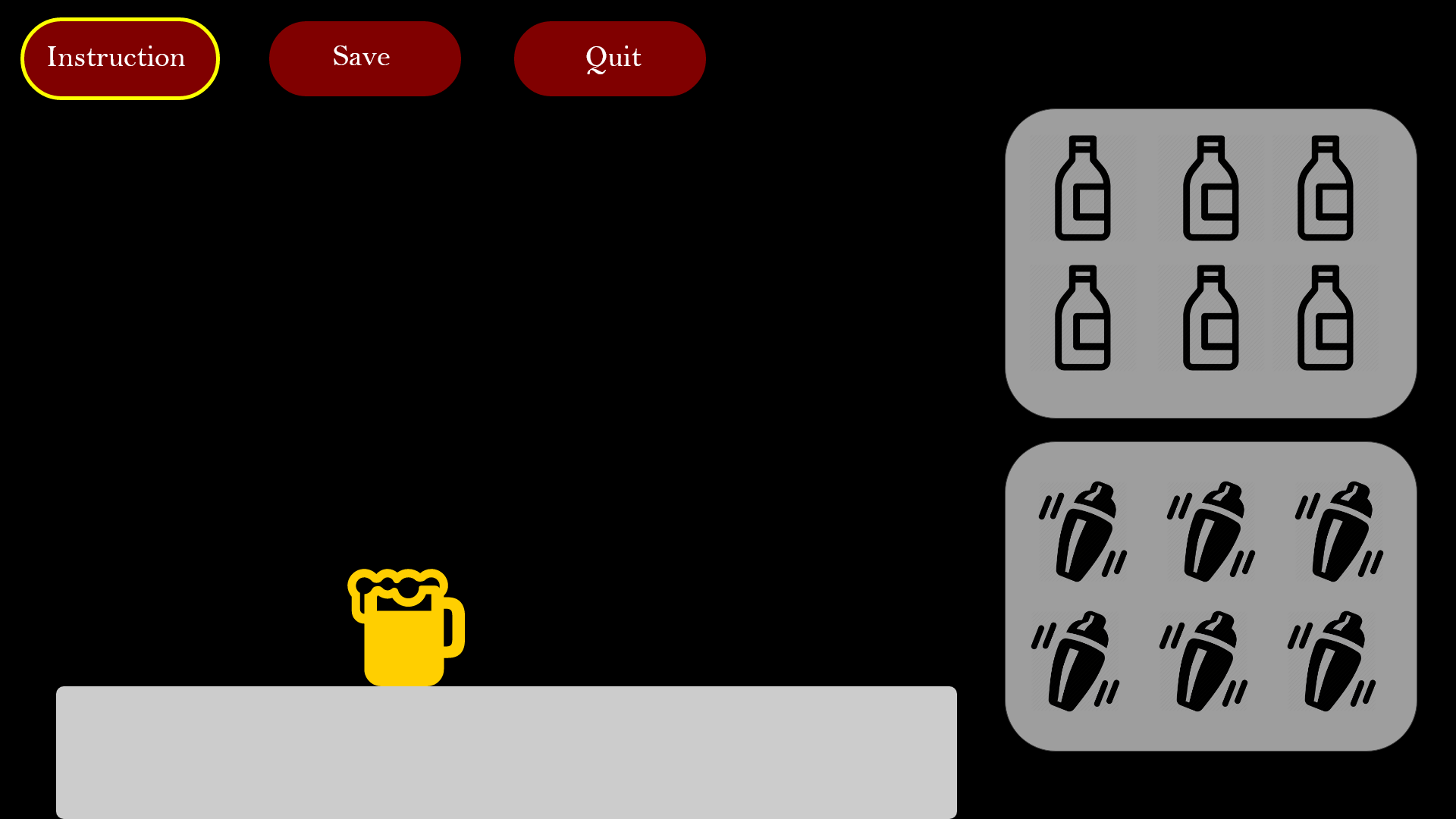
**Figure 3.8: Create recipe page**

****

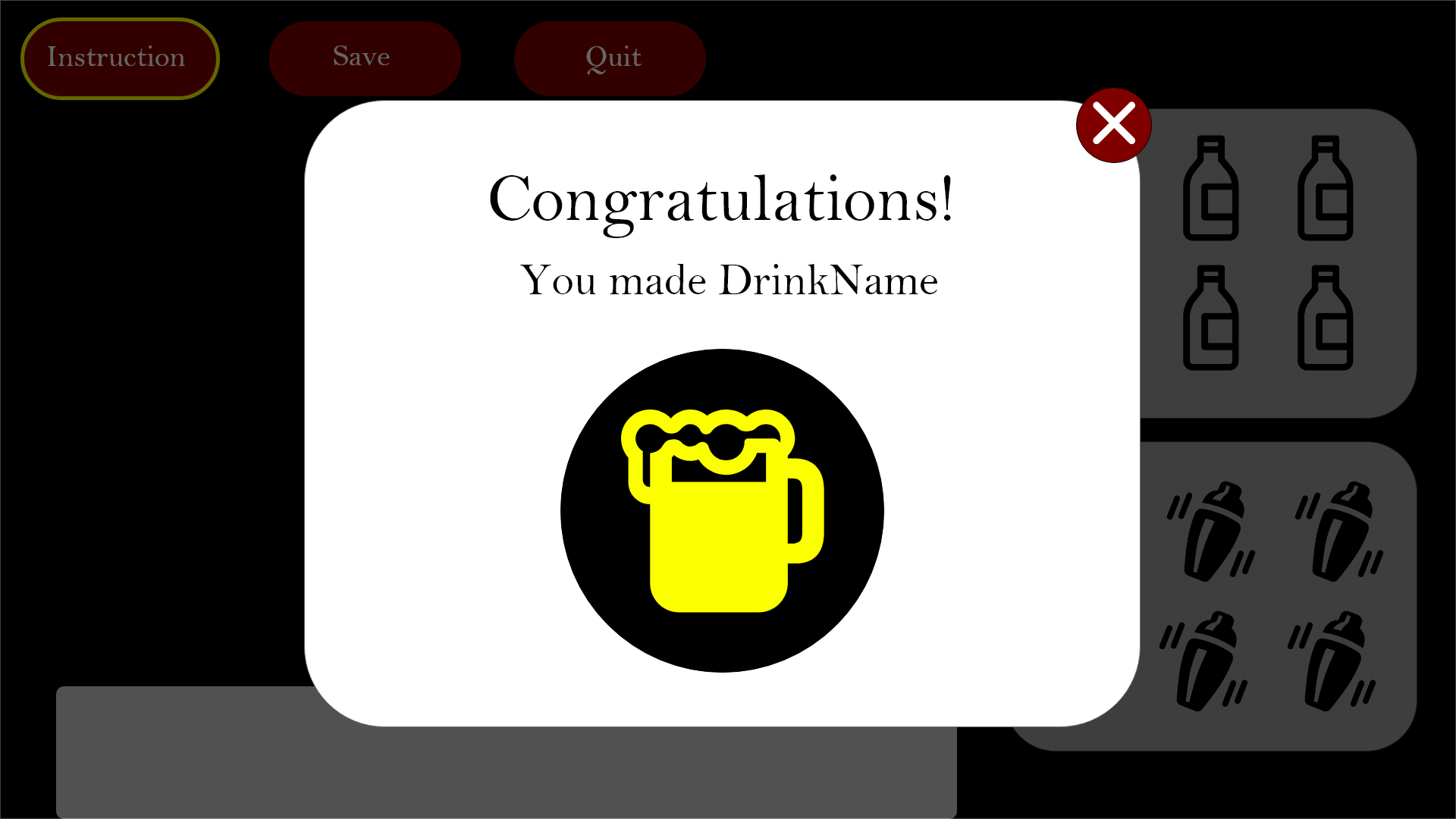
**Figure 3.9: MenuPage, Admin tab (only show up for admin permission)**

****

**Figure 3.10: MenuPage, Admin tab, user pop-up modal**

****

**Figure 3.11: Bar Page**

****

**Figure 3.12: After save recipe pop-up**

## **Functions**

It is important to provide appropriate feedback to our users when they are enrolled in a recipe learning session. The application needs to alert the user if a mistake was made in the procedure of a recipe. The application needs to provide on-screen messages giving feedback explaining why the user is receiving a message. These messages can be achieved by using visual cues and dynamically generated messages to the screen.

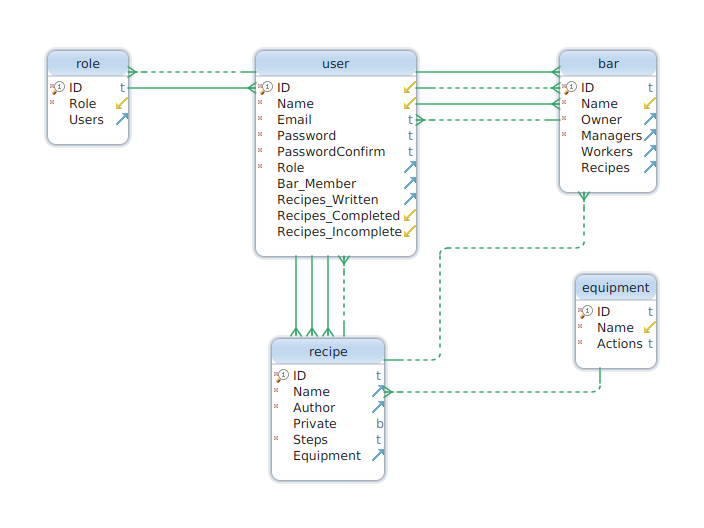
## **Performance requirements**

The web application should be responsive and handle multiple requests from multiple users. The capacity should support at least 1000 simultaneous users and can be extended in the future if needed. The web application should load to the clients machine in less than 3 seconds(Average time it takes to load a webpage). Users should be able to search for bars and recipes immediately and should be able to view each respective page immediately. The web application should also handle text data made from user input that follows best practices from other well-made CRUD and Restful web applications.

## **Logical database requirements**

The web application should store users and user data into a database. This database shall include a table to store users, a table to store recipes, a table to store bars, a table to store equipment, and a table to store user roles. The table for user roles should not be confused for the roles associated within a bar(owners, managers and workers). The user roles table is only to assign a role for a user/student and an admin role for the system administrators to maintain the web application.

Below there is an image about data entities and their relationships and integrity constraints.



## **Design constraints**

There is no specific design constraints in this project.

## **Software system attributes**

As professionals, all members of this project must take this application seriously. We are dedicated to producing a robust application that will satisfy the user’s experience. In order to achieve this quality, we should build the application with the following properties in mind:

* + 1. Reliability - The application should be planned, developed. and tested such that the user experience is flawless. The application should never fail on the client side, server side, and database side. Should the application crash, the information should be backed up and have minimal data loss
    2. Availability - The application should have a 99% uptime that is accessible to everyone with an internet connection through a web browser.
    3. Security - The web application should be protected from common exploits in web security and database security. The application should be protected from any malicious cross-site scripting, injections, and denial of service attacks. All sensitive data should be encrypted between the client and the server. All other security mechanisms will be addressed in future revisions.
    4. Extensibility - The web application should be scalable and allow for assets to be added to the game. This includes bars, recipes, and equipment not available at the launch of the web application.
    5. Portability - At launch, this application will be target web applications. This may port as a Mobile application in the future.
    6. Maintainability - All design and implementation elements should be well documented. This application involves multiple technologies and programming languages that should be updated regularly. All parts of the code should be easy to read and understand.

## **Organizing the specific requirements**

The specific requirements for this application already align with IEEE’s recommended SRS format. No additional/alternative arrangement of the content in this document is required.

## **Additional comments**

This application’s goal is to provide users an effective teaching tool to create recipes in a detailed mannered. This application will teach users new recipes in an interactive, enjoyable, and educational experience that is applicable in the real world. This application should also serve has a tool for users to educate others in productive learning environment. That being said, it is up to the designers and developers to create an application that can achieve these goals in an outstanding manner.

# **Supporting Information**

## **Table of contents**

[**Introduction**](#_7lhgpvhqxdqd) **4**

[Purpose](#_ts2sw27sl3qo) 4

[Scope](#_ts2sw27sl3qo) 4

[Definitions, acronyms, and abbreviations](#_ts2sw27sl3qo) 4

[References](#_ts2sw27sl3qo) 5

[Overview](#_ts2sw27sl3qo) 5

[**Overall description**](#_l1gw6ozaa208) **6**

[Product perspective](#_y7c5vtlli5fq) 6

[Product functions](#_y7c5vtlli5fq) 6

[User characteristics](#_y7c5vtlli5fq) 13

[Constraints](#_y7c5vtlli5fq) 13

[Assumptions and dependencies](#_y7c5vtlli5fq) 13

[**Specific requirements**](#_yg5fkwvmbno5) **14**

[External interfaces](#_oi0t7ca6zyme) 14

[Functions](#_3i91g8mqhgjj) 15

[Performance requirements](#_9e4qw9ghe83v) 15

[Logical database requirements](#_oi0t7ca6zyme) 16

[Design constraints](#_jbjd8495avgz) 16

[Software system attributes](#_oi0t7ca6zyme) 16

[Organizing the specific requirements](#_oi0t7ca6zyme) 17

[Additional comments](#_oi0t7ca6zyme) 17

[**Supporting Information**](#_wfjwqwouffsh) **17**

[Table of contents](#_v4smmwjxld2k) 17

[Index](#_v4smmwjxld2k) 18

[Appendixes](#_v4smmwjxld2k) 18

## 

## **Appendixes**

N/A