

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.

1. 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.

1.1. 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.

1.2. Scope

Tipsy is 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.

1.3. 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.

1.4. References

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

1.5. 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.

2. Overall description

2.1. 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.

2.2. Product functions

Use Case 2.1: Log in

Use Case:	Log in
Primary Actor:	User
Goal in Context:	The user wants to log into their account
Preconditions:	<ul style="list-style-type: none">• The User account is registered on the database• The User is not currently logged in
Scenario:	<ol style="list-style-type: none">1. User is on the login page2. User types in the correct email and password into their respective fields3. The user clicks on the Login 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 2.2: Log out

Use Case:	Log out
Primary Actor:	User
Goal in Context:	The user wants to log out of their account
Preconditions:	<ul style="list-style-type: none">• User currently logged in
Scenario:	<ol style="list-style-type: none">1. User hovers over the Account tab2. User clicks on the Logout 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 2.3: Register for account

Use Case:	Register for account
Primary Actor:	User
Goal in Context:	The user wants to register for an account
Preconditions:	<ul style="list-style-type: none">• User is not logged in• The account the User wants to create isn't already made
Scenario:	<ol style="list-style-type: none">1. The user is on the Login page2. The user clicks on the Register button3. The user fills out the required fields and clicks on the register button
Exceptions:	The email or nickname must not already be registered on the database
Priority:	Essential, must be implemented
When available:	First benchmark
Frequency of use:	Once per session
Open Issues:	N/A

Use Case 2.4: 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:	<ul style="list-style-type: none">• The user is on the Login page• The account that the User is trying to recover is already registered on the database
Scenario:	<ol style="list-style-type: none">1. User is on the Login page2. User clicks on the Forgot Password button3. User is taken to the Forgot Password page and fills in their email address

	4. 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 2.5: Update User Settings

Use Case:	Update User Settings
Primary Actor:	User
Goal in Context:	The User wants to update their User settings
Preconditions:	<ul style="list-style-type: none"> • User is currently logged in
Scenario:	<ol style="list-style-type: none"> 1. User hovers over the account tab 2. User clicks on the Settings button 3. User makes the changes they want to make in their respective fields 4. User clicks on the Update Settings button
Exceptions:	N/
Priority:	Essential, must be implemented
When available:	Third benchmark
Frequency of use:	Multiple times per session
Open Issues:	N/A

Use Case 2.6: Change Password

Use Case:	Change password
Primary Actor:	User

Goal in Context:	The User wants to change their password
Preconditions:	<ul style="list-style-type: none"> User is currently logged in
Scenario:	<ol style="list-style-type: none"> User hovers over the account tab User clicks on the Settings button User clicks on the Change password button User fills in both boxes with their new password User clicks on the Change password button
Exceptions:	The user will not be able to change their password if the password entered on the password and confirm password fields are not the same
Priority:	Essential, must be implemented
When available:	Third benchmark
Frequency of use:	Multiple times per session
Open Issues:	N/A

Use Case 2.7: Create Bar

Use Case:	Create Bar
Primary Actor:	User
Goal in Context:	The user wants to create a new bar
Preconditions:	<ul style="list-style-type: none"> The User is logged in
Scenario:	<ol style="list-style-type: none"> User is on the My Bar tab User clicks on the Create a Bar button User fills in all the required fields User clicks on the Create a Bar 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 2.8: Edit Bar

Use Case:	Edit Bar
Primary Actor:	User
Goal in Context:	The user wants to create a new bar
Preconditions:	<ul style="list-style-type: none"> • User is logged in • User owns the Bar they want to edit • The Bar is saved
Scenario:	<ol style="list-style-type: none"> 1. The user is on the bar menu 2. The user clicks on the new bar button 3. 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:	Second benchmark
Frequency of use:	A few times per session
Open Issues:	N/A

Use Case 2.9: 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:	<ul style="list-style-type: none"> • User is logged in • User that's adding another user to the

	<p>Bar is the owner or a manager of the Bar</p> <ul style="list-style-type: none"> • The User being added is a valid user • The User being added is not already in the Bar
Scenario:	<ol style="list-style-type: none"> 1. User is on the My Bars page 2. User clicks on the Bar that they own or manage 3. User clicks on the gear on the top right 4. User clicks on the Workers tab 5. User types in the nickname of the User they want to add 6. User selects the user they want to add as a worker and presses enter 7. User clicks on the Save 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 2.10: Add Manager to Bar

Use Case:	Add Manager to Bar
Primary Actor:	User
Goal in Context:	A user wants to add another user as a manager to a bar
Preconditions:	<ul style="list-style-type: none"> • User is logged in • The user that's adding another user to the bar is the Owner of the bar • The user being added is not already in the bar
Scenario:	<ol style="list-style-type: none"> 1. The user clicks on a bar that they own 2. The user types in the nickname 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:	Second benchmark
Frequency of use:	A few times per session
Open Issues:	N/A

Use Case 2.11: 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:	<ul style="list-style-type: none"> • The User is logged in • User that's removing another user from the Bar is either the owner or a manager of the Bar • The User being added is a valid user • The User being added is a worker of the bar
Scenario:	<ol style="list-style-type: none"> 1. The user clicks on a bar that they own or manage 2. 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 2.12: Remove Manager from Bar

Use Case:	Remove Manager from Bar
Primary Actor:	User

Goal in Context:	A User wants to delete a manager from a bar
Preconditions:	<ul style="list-style-type: none"> • User is logged in • The User that's removing a manager from the Bar is the owner of the bar • the User being removed is a manager of the Bar
Scenario:	<ol style="list-style-type: none"> 1. User is on the My Bars page 2. The user clicks on a Bar that they own 3. User clicks on the Gear button on the top right 4. User goes on the Managers tab 5. User clicks on the red X button on the manager they want to remove
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 2.13: Add Recipe to Bar

Use Case:	Add Recipe to Bar
Primary Actor:	User
Goal in Context:	The user wants to add a recipe to a bar
Preconditions:	<ul style="list-style-type: none"> • The Bar is Saved • The user is the owner, a manager, or a worker of the Bar • Recipe is published or the Recipe is saved and owned by the User
Scenario:	<ol style="list-style-type: none"> 1. User is on the My Bars page 2. User clicks on a Bar they want to add a Recipe to 3. User clicks on the gear on the top right 4. User clicks on the Recipes tab

	<ol style="list-style-type: none"> 5. User types in the name of the recipe they want to add and presses enter 6. User clicks on the Save 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 2.14: 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:	<ul style="list-style-type: none"> • 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:	<ol style="list-style-type: none"> 1. User is on the My Bars page 2. User clicks on a Bar they want to add a Recipe to 3. User clicks on the gear on the top right 4. User clicks on the Recipes tab 5. User clicks on the red X button next to the Recipe they want to remove 6. User clicks on the Save 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 2.15: 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:	<ul style="list-style-type: none">• The User is logged in
Scenario:	<ol style="list-style-type: none">1. User clicks on My Recipes tab on the navigation bar2. User clicks on the Add a Recipe button3. The User fills in a name along with an optional description and recipe photo4. The User clicks on the Create 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:	Third 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 2.16: Edit Recipe

Use Case:	Edit Recipe
Primary Actor:	User
Goal in Context:	The user wishes to edit a preexisting recipe
Preconditions:	<ul style="list-style-type: none">• User is logged in• User is the author of the recipe
Scenario:	<ol style="list-style-type: none">1. User clicks on the My Recipes tab on the navigation bar2. User clicks on a Recipe they created3. User makes the changes they want on the Recipe

	4. User clicks on Save Recipe
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 2.17: 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:	<ul style="list-style-type: none"> • The Recipe in question should already exist • The Recipe is owned by the User
Scenario:	<ol style="list-style-type: none"> 1. User clicks on the My Recipes button on the Navigation Bar 2. The user clicks on the self inker button on the top right 3. The user makes the changes they want to the cloned recipe 4. The user clicks on the clone 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 2.18: Save Recipe

Use Case:	Save Recipe
Primary Actor:	User
Goal in Context:	The author wants to save a Recipe
Preconditions:	<ul style="list-style-type: none">• User is logged in• User is on the Edit Recipe page• User is the author of the Recipe
Scenario:	<ol style="list-style-type: none">1. User is on the edit Recipe menu2. User clicks on the Save recipe button
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third benchmark
Frequency of use:	Many times per session
Open Issues:	N/A

Use Case 2.19: Publish Recipe

Use Case:	Publish Recipe
Primary Actor:	User
Goal in Context:	The user wants to publish a Recipe
Preconditions:	<ul style="list-style-type: none">• The User is logged in• User should own the recipe• The recipe should not already be published
Scenario:	<ol style="list-style-type: none">1. User is on the edit Recipe page2. User has filled in all the required fields3. User clicks on the publish tab4. The user clicks on the publish button
Exceptions:	<ul style="list-style-type: none">• The user hasn't filled out all the required fields before pressing on the publish button

Priority:	Essential, must be implemented
When available:	Third benchmark
Frequency of use:	A few times per session
Open Issues:	N/A

Use Case 2.20: Delete Recipe

Use Case:	Delete Recipe
Primary Actor:	User
Goal in Context:	The user wants to delete a Recipe
Preconditions:	<ul style="list-style-type: none"> • User is logged in • The recipe that the user wants to delete exists • The User is the Author of the recipe • The User is logged in
Scenario:	<ol style="list-style-type: none"> 1. User is on the 2. The user clicks on the delete button
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 2.21: View Recipe

Use Case:	View Recipe
Primary Actor:	User
Goal in Context:	The User wants to view a Recipe
Preconditions:	<ul style="list-style-type: none"> • User is logged in • The Recipe should already exist • User should be public unless they made the recipe

Scenario:	<ol style="list-style-type: none"> 1. User either clicks on the My Recipes page or searches for a published Recipe on the search page 2. User clicks on the Recipe they want to view
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third benchmark
Frequency of use:	Many times per session
Open Issues:	N/A

Use Case 2.22: 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:	<ul style="list-style-type: none"> • 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 2.23: Debug Recipe

Use Case:	Debug Recipe
Primary Actor:	User

Goal in Context:	The user wants to debug the recipe
Preconditions:	<ul style="list-style-type: none"> • User is logged in • User is on the Recipe creation page for the recipe they want to debug. • User is the author of the recipe.
Scenario:	<ul style="list-style-type: none"> • 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 2.24: Upload equipment

Use Case:	Upload equipment
Primary Actor:	User
Goal in Context:	The user wants to upload their own equipment to the game
Preconditions:	<ul style="list-style-type: none"> • The user has a png to upload • The user is on the recipe information detail page
Scenario:	<ol style="list-style-type: none"> 1. The user is on the recipe information detail page 2. The user clicks on the plus sign on the equipment section 3. 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 2.25: 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:	<ul style="list-style-type: none"> • The User is on the Mixer page • The User has made progress in making the drink
Scenario:	<ol style="list-style-type: none"> 1. User is on the mixer page 2. User clicks on the save button to save their 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:	Fourth benchmark
Frequency of use:	Many times per session
Open Issues:	N/A

Use Case 2.26: Exit Game

Use Case:	Exit Game
Primary Actor:	User
Goal in Context:	The user wants to quit the game
Preconditions:	<ul style="list-style-type: none"> • User is logged in • User is on the Mixer page

	<ul style="list-style-type: none"> User should have the Recipe open
Scenario:	<ul style="list-style-type: none"> The user is on the Mixer 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 2.27: Add steps

Use Case:	Add steps
Primary Actor:	User
Goal in Context:	The user wants to add steps to a recipe they own
Preconditions:	<ul style="list-style-type: none"> The user is on the recipe creation page
Scenario:	<ol style="list-style-type: none"> 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 2.28: 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:	<ul style="list-style-type: none"> • The user is on the recipe creation page • The step must exist in the recipe
Scenario:	<ol style="list-style-type: none"> 7. The user is on the recipe creation page 8. The user clicks on the edit steps button 9. The user selects the step they want to edit 10. 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 2.29: 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:	<ul style="list-style-type: none"> • The user is on the recipe creation page • The step must exist in the recipe
Scenario:	<ol style="list-style-type: none"> 11. The user is on the recipe creation page 12. The user clicks on the delete steps button 13. 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

Use Case 2.30: Search

Use Case:	Search
Primary Actor:	User
Goal in Context:	The user wants to search the database for a Bar, Equipment, User, or Recipe
Preconditions:	<ul style="list-style-type: none"> • User is logged in • User is on the search page
Scenario:	<ol style="list-style-type: none"> 1. User is on the search page 2. User selects the type of search they want to do 3. User types in a term 4. User presses enter to query the database
Exceptions:	<ul style="list-style-type: none"> • 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:	Fourth benchmark
Frequency of use:	Many times per session
Open Issues:	N/A

Use Case 2.31: 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:	<ul style="list-style-type: none"> • 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

2.3. 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.

2.4. 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.

2.5. Assumptions and dependencies

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

3. Specific requirements

3.1. External interfaces

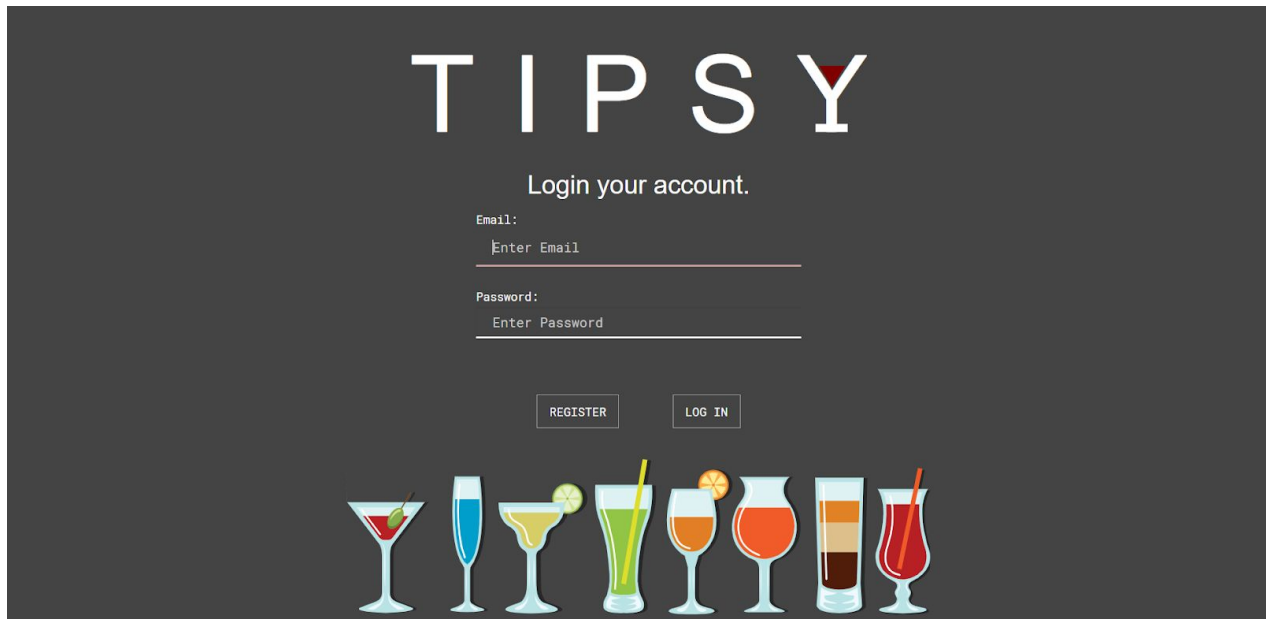


Figure 3.1: Login Page

TIPSY

Register your account.

First Name:
Enter First Name

Last Name:
Enter Last Name

Email:
Enter Email

Password:
Enter Password

Confirm Password:
Enter Password

REGISTER




Figure 3.3: Register page

You have successfully registered with **Tipsy**

Thank you for registering!

Please click [here](#) to redirect to login page.

Figure 3.4: Confirmation page

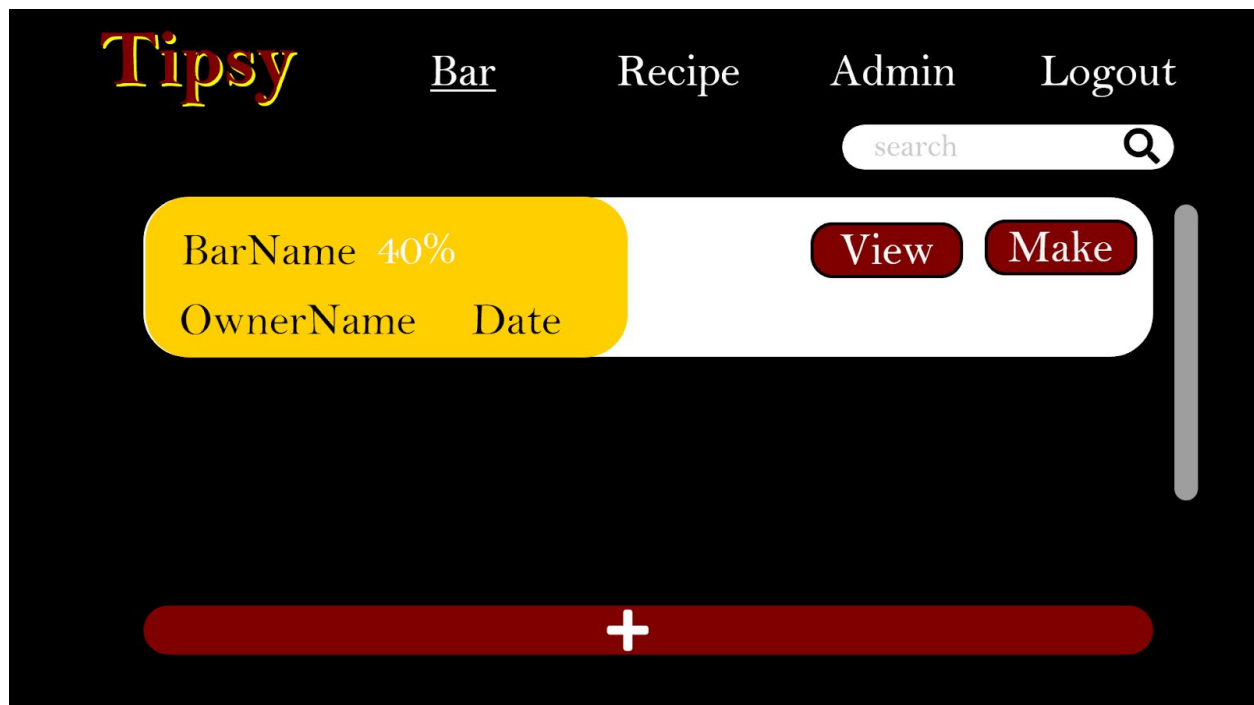


Figure 3.5: Menu Page, Bar tab



Figure 3.6: Bar information detail page

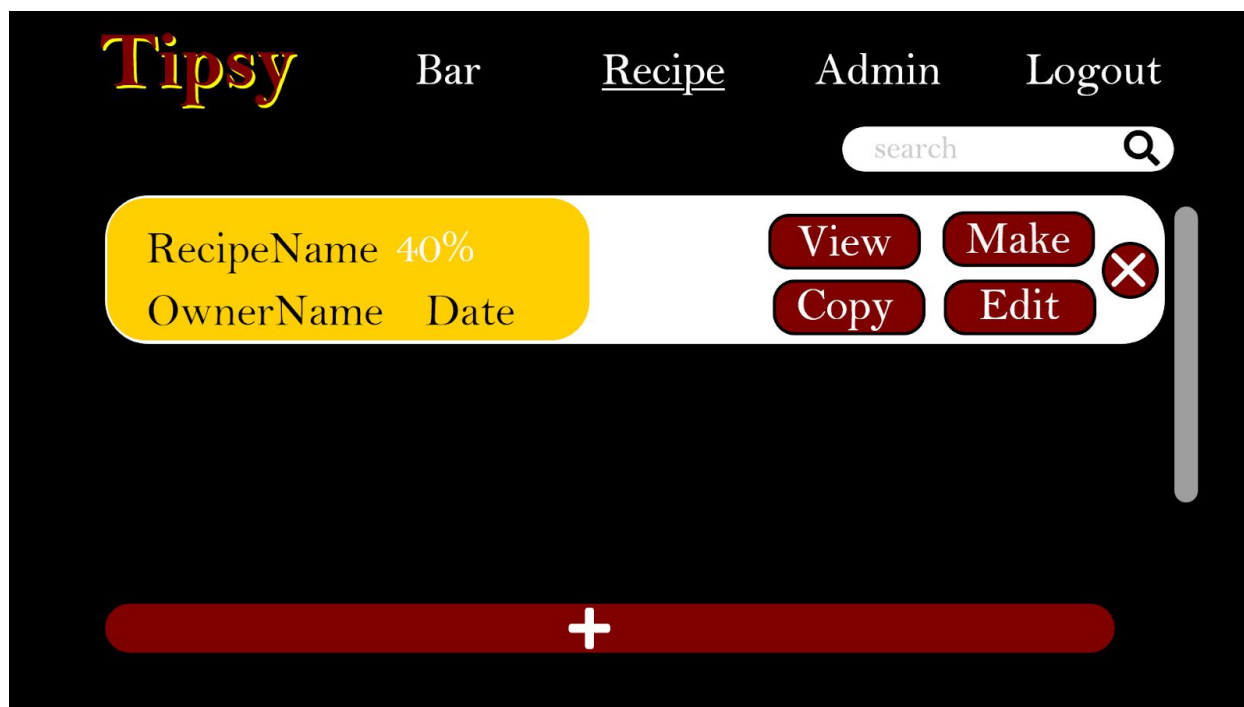


Figure 3.7: Menu Page, Recipe tab



Figure 3.8: Recipe information detail page

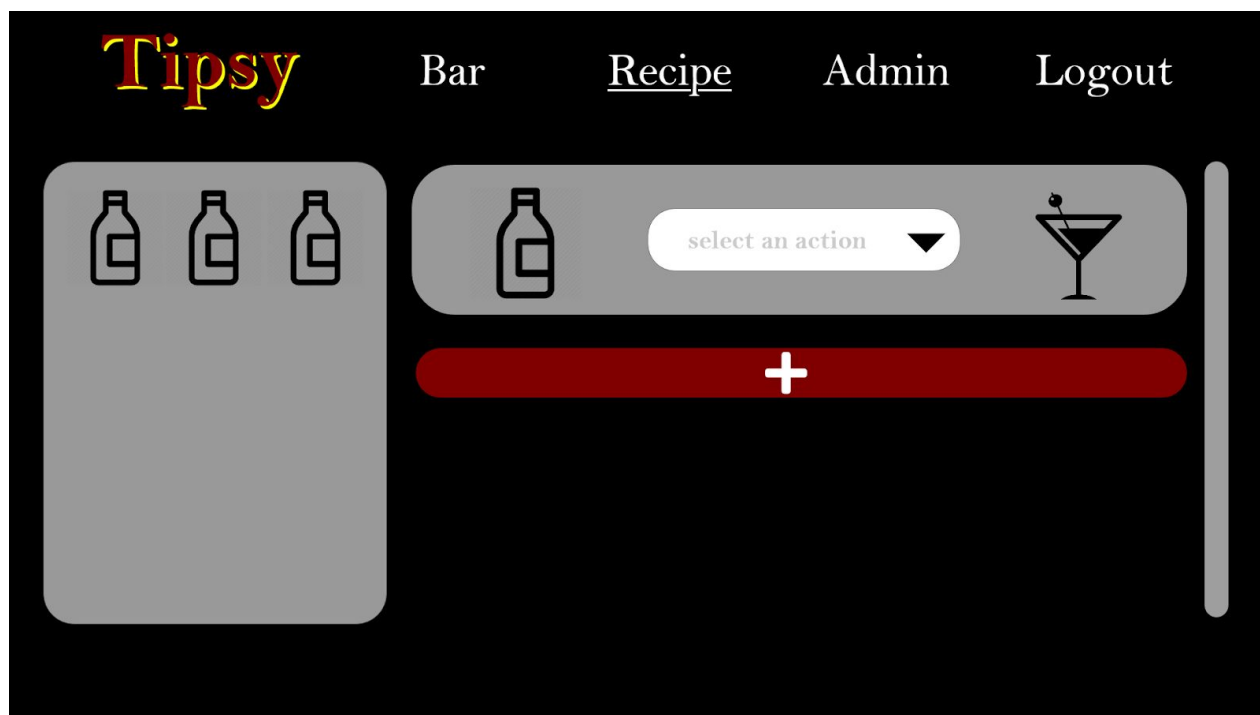


Figure 3.8: Create recipe page

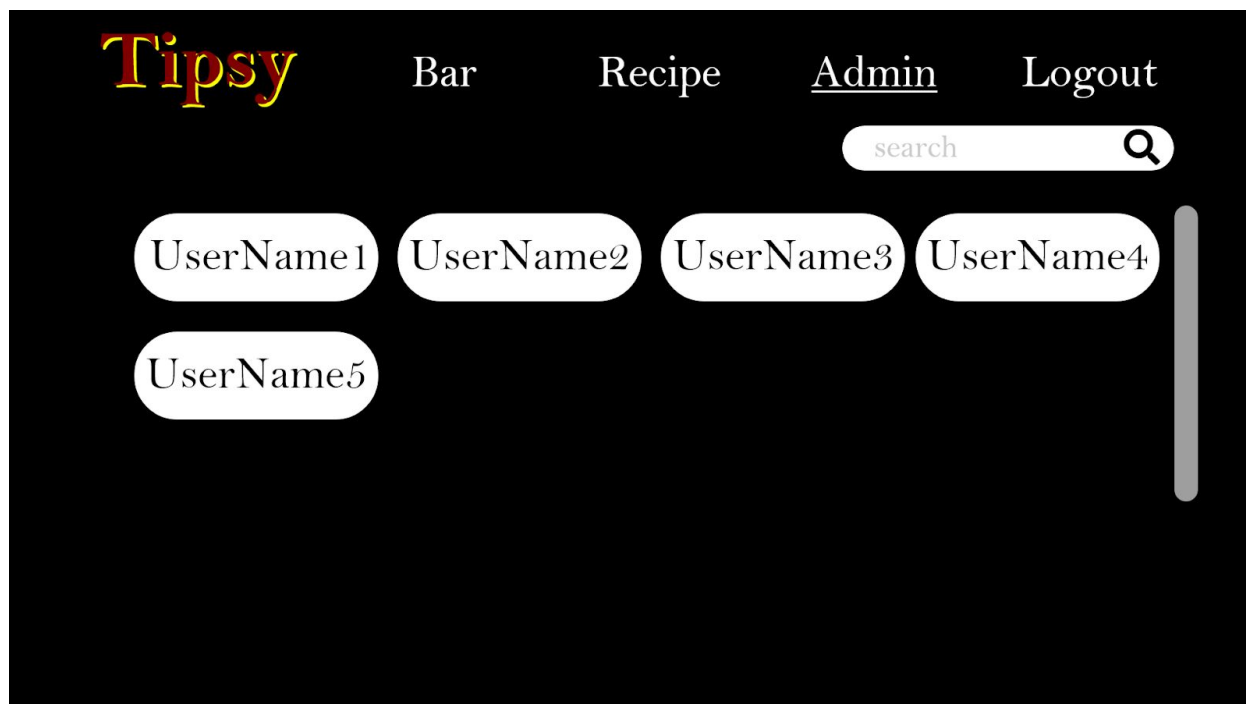


Figure 3.9: MenuPage, Admin tab (only show up for admin permission)

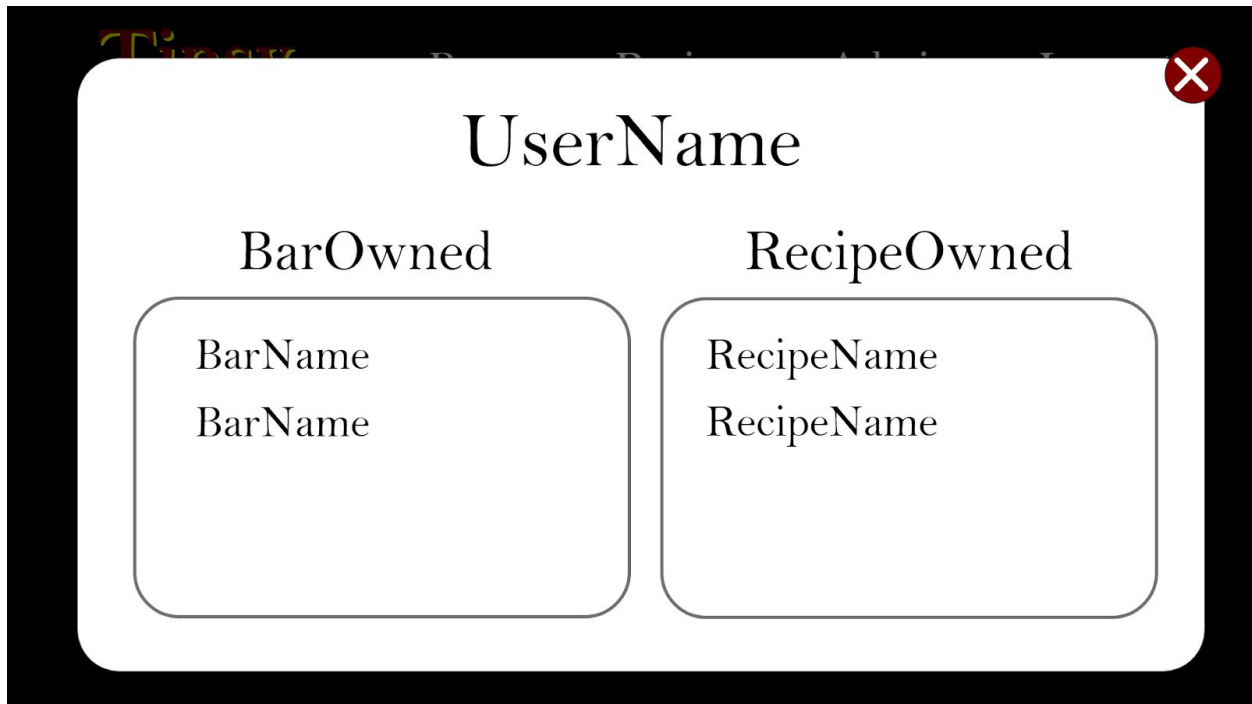


Figure 3.10: MenuPage, Admin tab, user pop-up modal

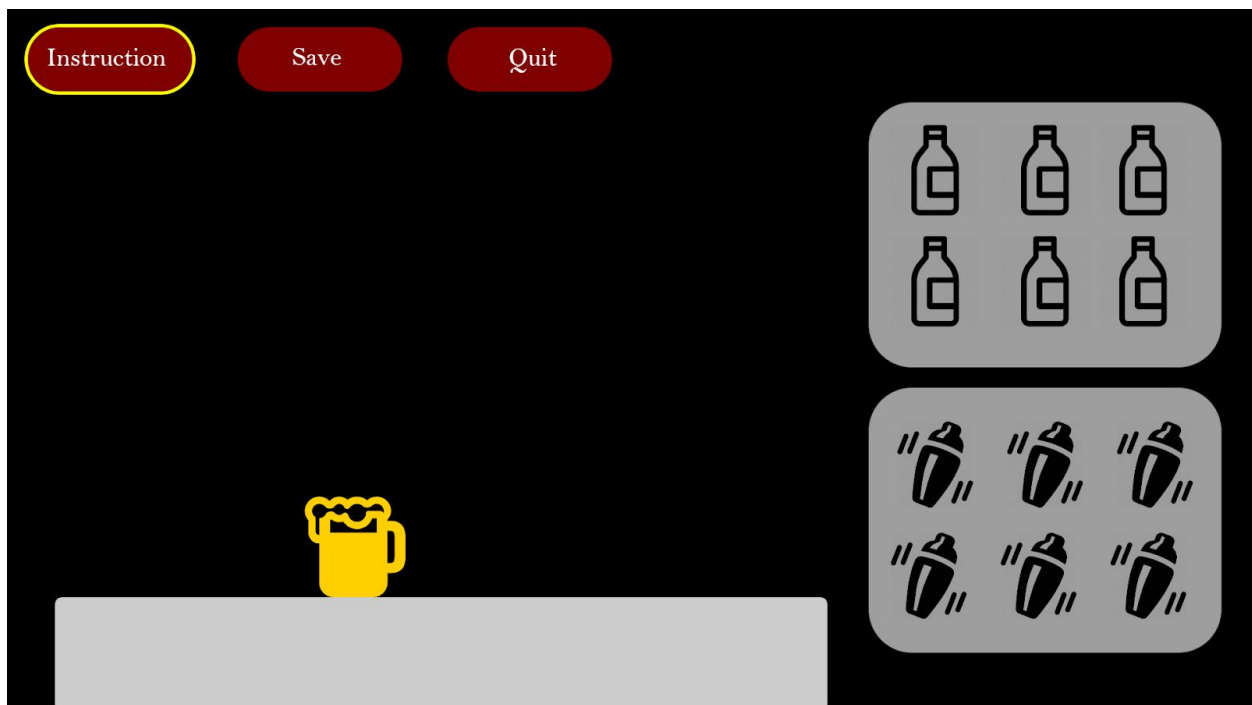


Figure 3.11: Bar Page

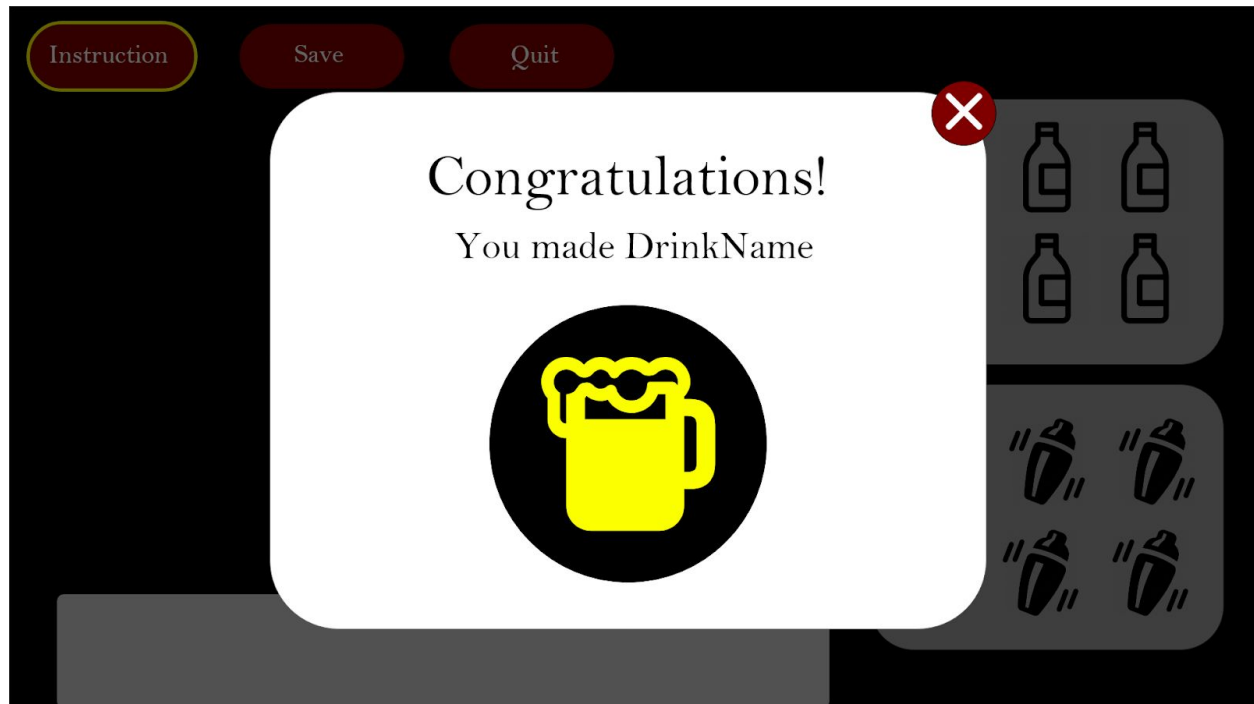


Figure 3.12: After save recipe pop-up

3.2. 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.

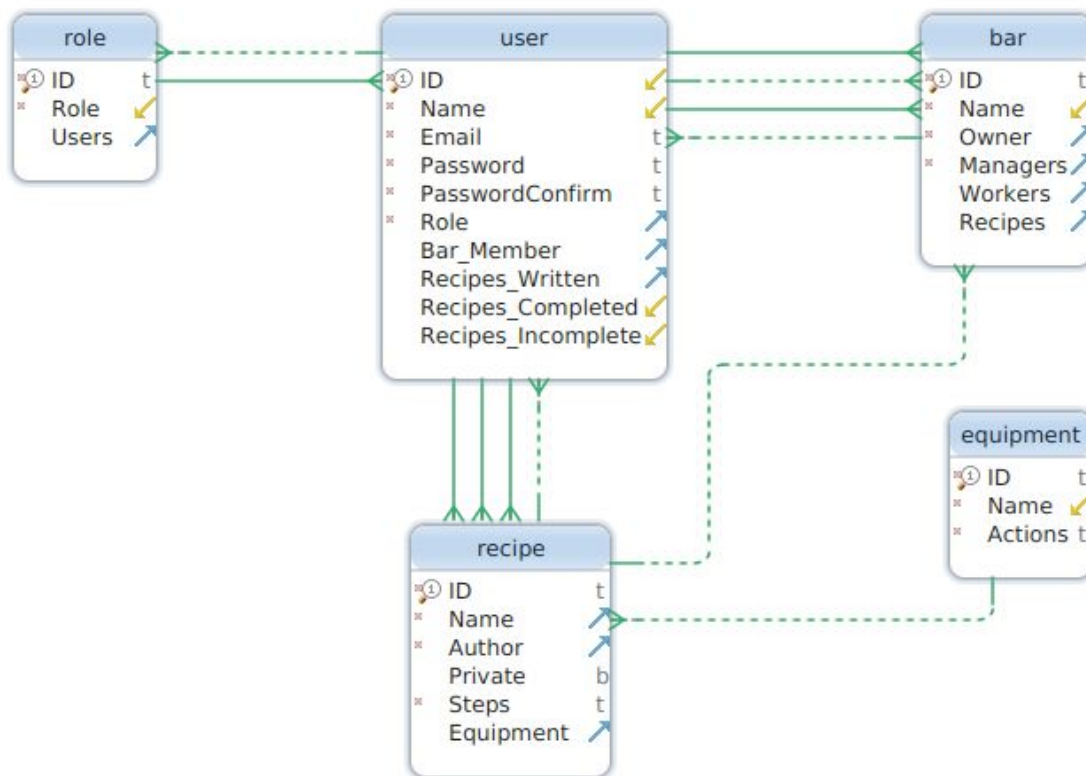
3.3. Performance requirements

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.

3.4. 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.



3.5. Design constraints

There is no specific design constraints in this project.

3.6. 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:

- 3.6.1. 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
- 3.6.2. Availability - The application should have a 99% uptime that is accessible to everyone with an internet connection through a web browser.
- 3.6.3. 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.
- 3.6.4. 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.
- 3.6.5. Portability - At launch, this application will be target web applications. This may port as a Mobile application in the future.
- 3.6.6. 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.

3.7. 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.

3.8. 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 as a tool for users to educate others in a 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.

4. Supporting Information

4.1. Table of contents

Introduction	4
Purpose	4
Scope	4
Definitions, acronyms, and abbreviations	4
References	5
Overview	5
Overall description	6
Product perspective	6
Product functions	6
User characteristics	13
Constraints	13
Assumptions and dependencies	13
Specific requirements	14
External interfaces	14
Functions	15
Performance requirements	15
Logical database requirements	16
Design constraints	16
Software system attributes	16
Organizing the specific requirements	17
Additional comments	17
Supporting Information	17
Table of contents	17
Index	18
Appendixes	18

4.2. Appendixes

N/A