

Software Requirements Specification

for

BeerButler

**Version 0.0**

**Prepared by**

**Group Name: BadScience Tech**

|  |  |  |
| --- | --- | --- |
| **Ryan Miller** | **11396551** | **ryan.d.miller@wsu.edu** |
| **Adriene Cranny** | **11452952** | **adriene.shanders@wsu.edu** |
| **Colin St. Claire** | **11416003** | **colin.stclaire@wsu.edu** |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
| **Date:** | **October 7th, 2016** |
|  |  |
|  |  |
|  |  |

**Contents**

**Revisions**

**1** **Introduction**

1.1 Document Purpose

1.2 Product Scope

1.3 Intended Audience and Document Overview

1.4 Definitions, Acronyms and Abbreviations

1.5 Document Conventions

1.6 References and Acknowledgments

**2** **Overall Description**

2.1 Product Perspective

2.2 Product Functionality

2.3 Users and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

**3** **Specific Requirements**

3.1 External Interface Requirements

3.2 Functional Requirements

3.3 Behaviour Requirements

**4** **Other Non-functional Requirements**

4.1 Performance Requirements

4.2 Safety and Security Requirements

4.3 Software Quality Attributes

**5** **Other Requirements**

**Appendix A – Data Dictionary**

**Appendix B - Group Log**

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| A | Adriene Cranny  Ryan Miller | Added:  Section 1: Introduction  Sections 2.1-2.3 | 10/05/2016 |
| B | Adriene Cranny | Revised:  Sections 2.1-Added diagram  Added:  Sections 2.4-2.7 | 10/05/2016 |
| C | Ryan Miller  Adriene Cranny | Revised:  Sections 2.5, 2.7: Refined.  3.1 - Diagram Refined  Added:  Sections 1.6, 3,4, Appendix A  Removed:  Section 5 | 10/06/16 |
| D | Ryan Miller  Colin St Claire | Revised:  Document spell/grammar check  Added:  Appendix A | 10/7/16 |

# 

# 

# Introduction

This project is designing a web app that helps users find food-beer pairings for different styles of beer. The application will suggest foods that complement, as well as contrast the flavor profile of the beer.

This section contains the documentation for:

1.1 The Document Purpose

1.2 The Product Scope

1.3 The Audience and Documentation Overview

1.4 Document Definitions, Acronyms, and Abbreviations

1.5 Document Conventions

1.6 References and Acknowledgements

## Document Purpose

BeerButler Revision A is a Web developed application. The purpose of this document is to give a detailed description of the requirements for this software. This document covers the purpose for the developed system. It will list system constraints, interface interaction, interaction with other systems such as an API. Its primary intention is to be used as a reference for developing the first version of the system for the development team.

## Product Scope

BeerButler is designed to pair different styles of beer with different styles of food described as complements or contrasts. This can benefit people who go out to have a drink and desire some style of food that can match their beer of choice. The main objective of the product is to provide two lists of food suggestions, one being complements, one being contrasts, for a selected style of beer. After the main objective is complete, other goals for the project are to expand the database of the software from not only beer but to include wine, cider, and other drinks as well. Even further than that would be to implement a social media aspect to the software that lets people “like” or “upvote” certain drink and food combination so that other people can see what combinations are the most popular.

## Intended Audience and Document Overview

The intended audience for this document is as follows. The “professor” who is reading the document for understanding of the software and grading the document’s usefulness and success in meeting a SRS’s requirements. The “client” who will read this document to understand the system requirements detailed. And the development team who will use and update this document throughout development of the software.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Admin/Administrator | System administrator who is given permission to manage/control the system. |
| API | Application program interface, a set of routines, protocols, and tools used in developing a software. |
| Graphical User Interface/GUI | A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators |
| User | Someone who is interacting with the Web Application. |
| Web Application/Web App | A client–server software application in which the client (or user interface) runs in a web browser. |
| Web page | A hypertext document connected to the World Wide Web. |
| Web/Website | A location connected to the Internet that maintains one or more pages on the World Wide Web. |

## Document Conventions

Typographical conventions followed: Arial font size 11. Single spaced with 1” margins. Section and Subsection use size 14 bold font.

## References and Acknowledgments

At this point in the development, the BeerButler SRS does not reference or acknowledge any other documents.

# Overall Description

## Product Perspective

BeerButler is a new self-contained product. It utilizes data from an API (BreweryDB database). It launches the GUI within a webpage. The API provides the system with all of the data regarding beer categories, individual beers, and food pairings.

The BeerButler system then structures this data and displays a GUI which allows the user to interact with the data. The user will be able to browse the data, search the data, or use a series of steps. The system will return data to the user based off of their selections.

The application will be launched through a webpage so it is accessible on any platform that has access to the world wide web.

## Product Functionality

Functions:

* List of different beer styles to be selected
* List different styles of food that compliment selected beer
* List different styles of food that contrast selected beer

## Users and Characteristics

There is really only one type of user for this product. The most pertinent characteristic of this user is their appreciation for beer/alcohol and food. The next key characteristic is that they must have enough technical ability to navigate a webpage on their mobile phone or computer. All users will have the same level of access to the data. They will only interact with the GUI.

## Operating Environment

Due to the nature of a web based application, this software should be able to run on most, if not all, systems including Apple, Linux, and Windows computers and Apple, Windows, and Android mobile devices. Whatever system is being used should be able to open a web browser, go to the web application site, and use the functions of BeerButler.

## Design and Implementation Constraints

Due to the fact that the BeerButler is using an API for the database of beer styles the BeerButler software is confined to using the routines and commands of the API. Therefore it will be essential that we make sure the GUI of the BeerButler software interacts with the database of the API properly (i.e. displays correct information for the user). Because BeerButler is a web app there are little to no requirements to the hardware needed for the software, besides network capabilities.

Because of the software of the web app, the coding languages that the BeerButler might have to use are Angular, Java, HTML, CSS, and JavaScript.

## User Documentation

Additional instructions and help can be found in the BeerButler User-Manual provided online through the site.

## Assumptions and Dependencies

BeerButler assumes the user has access to high speed internet. Internet connection failure or low data speeds can affect the loading time of the site, as well as rendering time for images, etc. It also assumes that the user knows the style of beer that they are drinking and can recognize said beer by name from the list the software provides.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The BeerButler web app, as of now, has been planned out as one page with lists and menus that appear on said page when certain items are selected. There will be an initial menu of styles of beer and when one is selected a second menu will appear, to the right of the first, with sub-styles of the selected style. Then after the sub-style is selected, two lists of food styles will appear, to the right, one for complementary foods and one for contrasting foods. These lists will not be selectable because they are just suggested pairing for the selected style of beer.

### Hardware Interfaces

Due to the nature of the web app software there are no hardware interfaces needed beyond input devices like keyboards, mice, etc., and output devices, such as a monitor. If the web app was native (downloaded onto the device) there might be access to hardware but as of now BeerButler has no access to the user’s hardware.

### Software Interfaces

The BeerButler web app software is ran in a web browser and as such it should be able to be run on any computer operating system (Windows, OSX, Linux, etc.) or mobile operating system (Android, iOS, etc.). The only communication needed in the software is between the BeerButler and the API that communicates with the beer and food database. All the communication between BeerButler and the API should be in the background of the program except for the fact that BeerButler will display the findings returned from the API. BeerButler will make a search request to API, the API will return and pass the result back to BeerButler, and then BeerButler will display the result to the user.

### Communications Interfaces

Communication between the API, GUI, and Web Browser are important since they depend on each other. However, for their tasks, communication is not important for the system and will be handled within the OS and web browser.

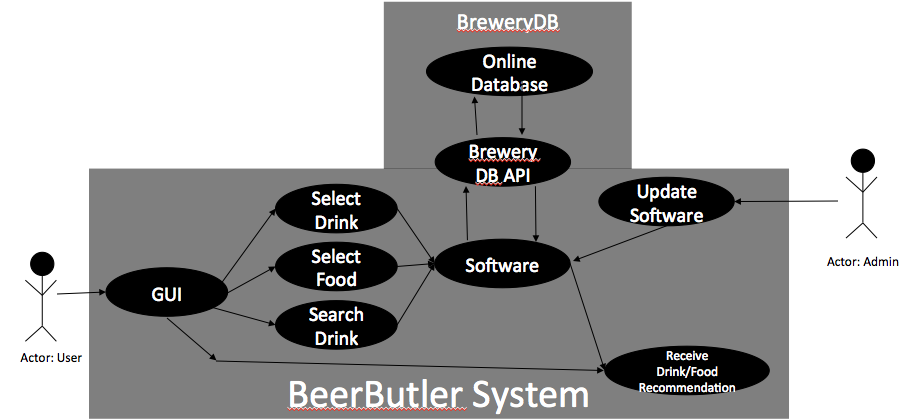
## Functional Requirements

Functions:

* List of different beer styles to be selected
  + User searches beer:
    - GUI makes calls to the database using the API, matches search input and returns associated data.
  + User browses beer:
    - Web calls pulls data from database and formats the data into a list of beer categories with subcategories. Each individual beer will be clickable and link to more information.
  + User goes through a “questionnaire”
    - GUI prompts user with multiple choice questions
    - Input from user to the GUI, the software parses through the database
    - Software returns one or more beer selections
* List food recommendations based off of beer selections:
  + User retrieves beer from one of the above options
  + User selects from the GUI the option of getting food recommendations
  + Software pulls data from database corresponding to the user’s beer choice
    - GUI will display recommendations

## Behaviour Requirements

### Use Case View



# Other Non-functional Requirements

## Performance Requirements

The performance requirements covers the expectations for the software. They are as follows:

1. The main page or home page will fully load in less than ten seconds.
2. Selecting each option at each step will not take more than ten seconds to generate a response.
3. The user will be able to navigate back and forth between categories and subcategories.
4. The user will get a message if their search does not return results.

## Safety and Security Requirements

BeerButler provides recommendation for alcohol choices, but does not condone unsafe or illegal consumption of alcohol. BeerButler provides the Government Warning (shown below) as well as advises against minors drinking and drinking and driving. BeerButler recommends food pairings, but it is the user’s responsibility to cook and drink safely. Users are advised to review health and food safety guidelines.

GOVERNMENT WARNING:

(1) According to the Surgeon General, women should not drink alcoholic beverages during pregnancy because of the risk of birth defects. (2) Consumption of alcoholic beverages impairs your ability to drive a car or operate machinery, and may cause health problems.

By federal law it is illegal for minors under the age of 21 years to consume alcohol.

By federal law it is illegal for a person to drive while intoxicated.

## Software Quality Attributes

**4.3.1 Adaptability**

BeerButler should run on multiple different web browsers (Google Chrome, Firefox, Safari, etc) because there is no guarantee the kind of web browser a user might be using.

**4.3.2 Availability or Reliability**

BeerButler should be able to be accessed on a web browser as long as the user’s device has an internet connection and javascript is not disabled on the browser.

**4.3.3 Usability**

BeerButler should be able to act quick enough for the user to be able to search their beer and get different food suggestions in the time it would take for a server to take their order at an average restaurant. BeerButler GUI should make navigating and reading data efficient for users.

**4.3.4 Maintainability**

The software calls methods or functions from an API provided through an online resource. The API updates online. The BeerButler admin will need to update the software as needed to accommodate for the updating of the API.

**Appendix A – Data Dictionary**

*<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>*

|  |  |
| --- | --- |
| *Variable/State/Function* | *Definition* |
| Actor: User | This is the person that is trying to access the BeerButler software and use the system |
| Actor: Admin | This is a person that keeps the software up to date following along with the API used. As the API updates its internal working the Admin will update the BeerButler software to keep it smoothly interacting with the API |
| GUI | A web-page that the user interacts with in order to use the BeerButler software. |
| Select Drink | A state of the software that the User will interact with. A list of drinks will be displayed for the User to click on. When a drink is clicked on the menu, the software will retrieve data from the database. |
| Select Food | A state of the software that the User interacts with. A list of food is displayed for the User to select. After the type of food is selected the software will retrieve data from the database. |
| Search Drink | A state of the software that the User interacts with. The User will use a search bar to type in the name of the beer that they have or are looking for. When the search is run (either by the enter key or the search button) the software will retrieve data from the database |
| Software | This state is where the BeerButler interacts with the database (using the API). In this state the software will return the data the User asked for (food suggestions for drinks, drink suggestions for food, or a list of drinks based on the search input) |
| BreweryDB API | The methods and functions that the software uses to interact with the BreweryDB database. |
| BreweryDB Database | The database that holds all the pairing recommendations that the User will be searching for. |
| Update Software | This is a state that only the Admin accesses. In this state the Admin makes changes to the software in order to keep it up to date and interacting properly with the API depending on if the API have changed at all. |
| Receive Drink/Food Suggestions | This state is where the GUI displays the results of its findings in the Database state (food suggestions, drink suggestions, or drink list based on search input). The User will be able to look at the results of the BeerButler search. The contents of the list will depend on the type of search that was done by the user. |

**Appendix B - Group Log**

*<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist the Teaching Assistant to determine the effort put forth to produce this document>*

|  |  |  |  |
| --- | --- | --- | --- |
| *Date* | *Participants* | *Activity* | *Minutes* |
| *09/21/16* | *Adriene Cranny*  *Ryan Miller*  *Colin St.Claire* | *Review SRS requirements, discuss roles, split up of work* | *30 minutes* |
| *09/28/16* | *Adriene Cranny*  *Ryan Miller*  *Colin St.Claire* | *Discussion of SRS*  *and Project* | *45 minutes* |
| *10/05/16* | *Ryan Miller* | *Preparing SRS RevA* | *30 minutes* |
| *10/05/16* | *Adriene Cranny*  *Ryan Miller*  *Colin St.Claire* | *Preparing SRS RevA* | *60 minutes* |
| *10/05/16* | *Adriene Cranny* | *Preparing SRS RevB* | *30 minutes* |
| *10/06/16* | *Ryan Miller*  *Adriene Cranny* | *Preparing SRS RevC* | *90 minutes* |
| *10/7/16* | *Ryan Miller*  *Colin St Claire* | *SRS revisions, adding needed information* | *90 minutes* |