Software Requirements Specification

for

DRINK SUGGESTION APPLICATION “Sip”

(Name in progress)

Version 1.0 approved

Prepared by: Katelyn Schoenberger

Eduardo Soto

Thomas Johnson

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Team Spooky | 11/6/17 | First Commit | 1 |
|  |  |  |  |

# Introduction

## Purpose

The purpose for creating this app is to give users a sense of what drinks could be paired with what foods. The database could be an already existing website, or our own personally made database, depending on whether we can properly integrate someone else’s database

## Document Conventions

We will most likely be using topological sorting methods in this app. Since we will be including and sorting several lists of beverages, it will be essential to have an algorithm built into the system that is able to match the pairings accordingly.

## Intended Audience and Reading Suggestions

The intended audience for this specific app is basically the general public. Whenever individuals go to wine tastings or vacation spots, they look forward to finding out new ways they can mix up their personal menus and get a taste of different things. For our target audience, it could range from wine tasters, cooks and chefs, as well as for events.

## Product Scope

*The app would likely be in the form of a questionnaire/checklist. For instance, the user would input his/her favorite sweet such as dark chocolate and would then be offered in return a list of different alcoholic beverages that would go similarly with the sweet. In this case, a Cabernet Sauvignon (wine) or stout (beer) would be most suitable.*

## References

N/A – Individual research by Thomas Johnson

# Overall Description

## Product Perspective

## *This product is in a sense an extenuation/improvement of an already existing app called Wine Mate. I conducted some research into this app and saw several similarities between the attributes of our product and the Wine Mate app’s. Below is a brief description of the app for reference:*

## WineMate provides food and wine pairing suggestions whether you're starting with food or with vino. This app has no preset pairings, and instead calculates pairing suggestions based on the exact ingredients in your food, giving you a much wider variety of suggestions than "oh, you're having steak? Drink Cabernet!"

What we aim to implement in order to override this current app is a checklist format and ways for the user to pick his/her preferences rather than just randomly sorted preferences as the Wine Mate app suggests.

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

(in progress)

## Operating Environment

Languages/Frameworks Used

* Java and SQL
* JavaFX or QT for graphics

As far as the software platform that will be used to construct the app, Android Studio will probably be used.

## Design and Implementation Constraints

When developing this app, the constraints that come to mind are having the necessary software installed that is compatible with the user’s system. It is important to make sure the software is updated frequently and that the commands are being used properly. When using Android Studio, several updates must be made throughout the duration of the process which can ultimately slow down the app building process and leave the developer with few other options to turn to. The reason for mainly wanting to use Android Studio is because it supports Java which is our intended language for this app.

## User Documentation

The main documentation (will be prepared once app is fully complete) will include a guide for a user (novice/beginner to software development) and a developer (a professional or experienced programmer). In addition, there will be an installation guide which will walk through step-by-step the exact instructions for installing and maintaining the necessary software used for the design of the app.

## Assumptions and Dependencies

We are still in the process of gathering the necessary data that is to be used to run the app efficiently. We plan to utilize a central database that could be an already existing website, or our own personally made database, depending on whether we can properly integrate someone else’s database. One thing that we are currently researching are what other apps are similar to the one we are building and how we can improve upon its features as well.

# External Interface Requirements

## User Interfaces

One of the key uses of the app will be that the user will be able to go into the app at any point in time and select which tastes they have and then the suggestions will pop up in the form of checklists. The app itself will contain a database of sorted selections based on the user’s preferences. One implementation to this app is to keep record of past entries into the app in order to make future suggestions for the user’s preferences. For instance, say the user selects chocolate and wants to find a wine to pair it with. The wine can also be paired with other things such as meat and will keep track of the user’s’ preferences as he/she continues to use it. The algorithm is able to pick up the user’s preferences more in depth.

## Hardware Interfaces

This app can be downloaded on any smart phone or device. We will be using Android Studio to build the app in Java and then we will be sure it is compatible with the user’s device without any issues.

## Software Interfaces

The user would input his/her pairing idea. For instance, say they enter in “dark chocolate” then just a few (curated) matches will come up for their preferences at a time. Each time the user decides they want to use dark chocolate, a new pairing will always come up, even if it’s a small alteration in the beverage or the branding/label of the drink. We want to have an infinite database of drinks and options for the user so they are able to switch it up a bit. The system will keep track of the suggestions and let the user know each time. The user will not have access to this exclusively built-in database due to the nature of the app only giving out a finite number of suggestions at a time. Another idea would be to limit the number of times the user utilizes the app each day, allowing them limited pairing sessions.

## Communications Interfaces

When the user signs up for this app, it will most likely require an email address, at least starting out. The user may also connect the app to his/her Facebook account as well.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority: Creating the database for information to be stored.

The first step in the process would be to create a database of information based on the wine/food pairing. This is probably the most important step of the process because access to this information will need to be stored somewhere and also be accessible.

## Performance Requirements

The user may need to continuously update their preferences or have a way of keeping track of them so that they are not running out of pairing suggestions. Perhaps the app can make new suggestions to the type/brand of chocolate or wine over time in order to give the user a refresher each and every time it is used.

## Safety Requirements

*In regards to safety requirements, it would need to be strict on potential food allergies, restrictions, or dietary needs. There should be a tool implemented into the app that the user can keep this information on file.*

## Security Requirements

One security concern would be using the brands of the drinks or foods that are implemented into the database. I am not familiar with any policies that restrict the use of certain brands being openly used to the public in this way, so it would be important to get familiar with legal policies regarding brands and labelling.

## Business Rules

The user will not having access to all of the drinks and foods in the database; the algorithm will randomly select the item on its own and bring the item to the user’s attention this way. They would only have access to a limited number of products at each point in the pairing process.

# Other Requirements

The app would be free and ready to download in the app store at any point in time/at anyone’s convenience. There would be no charge.

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>