Butter™
Requirements Document

Revision 1.0

March 28, 2014

Prepared by:

Bryan Chong

Kaitlyn Lee

Hyungjin Shin

Lorraine Sposto

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this software requirements specification document is to provide a detailed description of the functionalities of **Butter**. This document will cover each of the intended features, as well as a preliminary glimpse of the application's User Interface (UI). The document will also cover hardware, software, and other technical dependencies required to run **Butter**.

1.2 Document Conventions

This document contains acronyms and terminology in which the reader may be unfamiliar with. For clarification and definitions, see Appendix A (Glossary) for further details.

1.3 Intended Audience and Reading Suggestions

This document is to document **Butter**'s specific technical requirements and functionalities intended for individuals and groups participating in developing **Butter** and for members supervising **Butter**. Readers who have stumbled upon this document for the sake of boredom or interest are recommended to look at Article 1, *Introduction*, and Article 2, *Overall Descriptions*, which provides a short description of this project.

Readers who wish to explore the detailed features should explore Article 3, *Specific Requirements*, which expands on the interface and processes of major functions in running **Butter**.

Readers who are interested in the non-technical aspects and descriptions should look at Article 4, *Other Requirements*, which covers other documentations and other illogical and uncategorized information that would not belong under the main Articles.

If reading documentation simply interests the reader, we suggest reading *Outliers* by Malcolm Gladwell.

1.4 Project Scope

Butter implements both client-side and server-side applications to run on the computer. The client-side will support and interact with the server-side to track and update car positions. Scenarios may include tracking and positioning a visual car from one destination to another destination, plotting random car locations, and seeing a comprehensive traffic flow of multiple cars, etc.

2. Overall Description

2.1 Product Perspective

Butter is a cross-platform computer application. Because **Butter**'s functionality relies on its ability to retrieve real-time data, application-to-server communication is a weighty element of **Butter**'s design and thus will be discussed in detail along with the user specific aspects of design.

2.2 Product Features

The following list offers a brief outline and description of the main features and functionalities of **Butter**.

Traffic Visualization

- A map displays current cars on the roads
- Individual cars animate on the map according to their speed
- A color-coding system indicates each car's speed

Traffic Route Optimization

- Provides the user with an optimal route based on current, given the start and end locations specified by the user
- Provide the travel time for the optimal route at the speed limit and current speeds

Traffic History

- Store traffic data over time
- Displays data in a data-driven visual document
- Export data into a format such as .txt or .csv that can be imported into a spreadsheet, such as Microsoft Excel or Spreadsheets in Google Drive
- Allows the user to determine the times of day that would be best for travelling from a source to a destination

2.3 User Classes and Characteristics

Butter's interface will reflect its intention to enhance the day-to-day experience of the average driver. Because the target demographic is the general population, the interface will be minimal and simplistic to provide a user with a quick and easy experience using **Butter**.

2.4 Operating Environment

Butter will be limited to a runnable Java application and will rely on using Java's native Swing Application Programming Interface (API) and Google Maps' API to render the graphical user interface (GUI). It does not rely on any other programming language or any other external GUI components besides the APIs mentioned previously.

Butter will also connect and communicate with an external server; in this first revision demonstration, a server provided by the University of Southern California will provide JSON-formatted data that is also visually available on a web browser. The **Butter** database will be stored locally on an independent computer running Mac OS X Mavericks using MySQL.

2.5 Design and Implementation Constraints

A primary design constraint will be the different resolution the program is executed on. Because each computer displays different resolutions depending on the model and make of the monitor and screen, a suitable dimensions to encompass all screens at an appropriate non-full screen size will need to be considered. Because each operating system (OS) displays windows in slightly different manner, the compatibility of having identical looking GUIs may need to be considered, given that testing would be done on a Mac OS X and launched on the same OS. If the program tracks and plots a large number of data, limited memory and processing power will need to be considered because it may affect the latency of the program. With a fair amount of data, **Butter** should effectively and smoothly respond to the data pull and responsively update the GUI. With scalability, however, each feature must be designed so that quickness and memory usage is effectively implemented and considered.

2.6 Assumptions and Dependencies

Butter's features of traffic visualization and analysis depend on the application's ability to retrieve traffic data, and thus server communication must be implemented first. Only then can the user-oriented aspects of **Butter**'s functionality, which rely on analysis of traffic data, be implemented.

3. Specific Requirements

3.1 Interface Requirements

- Google Maps implementation
- Fastest Route Panel
    - TextField for Start Location
    - TextField for End Location
    - Enter button to launch the fastest route algorithm
- Export Data Option

3.2 Functional Requirements

The following outlines all the functionality that will be present in **Butter.**

Traffic Visualization

- An active map of Los Angeles and its transportation networks through the use of the GoogleMaps API.
    - The map will display a live flow of traffic in a given area.
- The flow of traffic will be changed by the continuously updating SQL database.
    - Between intervals of database updates, the program will predict data progression and update the map accordingly.
    - Database updates will override any "incorrect" predicted data

Traffic Route Optimization

- Takes in a "Start" and "End" location to calculate the fastest route under the current traffic circumstances.
    - Both Start and End locations must exist within a preset area of Los Angeles.
    - The Map interface will outline the shortest path to the destination as calculated.
- Specify the estimated arrival time to the destination
    - Take into account speed limits and/or speed of traffic as necessary

Traffic History

- Ability to export all data in an aggregated form
    - Ability to export all data and create organized graphs/tables as required.
    - Ability to export all data to a format readable in a spreadsheet or Excel.

3.3 Performance Requirements

Performance is largely dependant on the updating rate of the SQL server and the speed of connection to the server. Once the data is updated in the local program, performance should no longer be an issue. To cover up possible latency/slow response issues, the local program will mathematically predict a "live" update of the progress of the cars until the databases are properly updated, from which any prediction errors will be immediately corrected.

3.4 Logical Database Requirements

As **Butter** receives data from the server, all of the data will be stored locally to the running instance of the application on the user's device. **Butter** will effectively maintain its own small scale database on the user's device to store a historical pool of data (beginning at the runtime of the application) as updated information is received from the server. Outside of a local instance of the application, traffic data is not stored.

3.5 Software System Attributes

3.5.1 Reliability

Reliability of the program is mainly based on the reliability of the data being processed. To ensure reliability and correctness of the optimal routes, there will be zero tolerance for errors in the algorithm that calculates the optimal routes and travel times.

3.5.2 Quality

Butter will be presented and organized in a manner that is both visually appealing and easy for the user to navigate.

4. Other Requirements

4.1 Specifications Document: High-level document that outlines the specifications for the piece of software. Details should be sufficient to be able to write out the technical requirements. The details of implementations are not included in the specifications.

4.2 Design Document: Document to outline difficult algorithms, the exact GUI of the program, hardware and software requirements, class diagrams and inheritance hierarchies, the database schema, and other technical details.

4.3 Testing Document: Document to describe the test cases sufficient to determine that the application operates properly. The types of testing will include white box testing, black box testing, unit testing, regression testing, and more.

4.4 Deployment Document: Document that describes the steps required to take the program code from Eclipse, and deploy it on an actual server. The steps may include outside libraries, necessary server settings, download and install instructions for Java, and more.

4.5 Complete Documentation: Document including a cover page, table of contents, the requirements document, design document, testing document, and deployment document.

5. Appendix A: Glossary

API
An Application Program Interface (API) is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and be used when programming Graphical User Interface (GUI) components.

GUI
A Graphical User Interface (GUI) is a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons, and menus which can be manipulated by a mouses (and often a keyboard as well).

JSON
Javascript Oriented Notation (JSON) is a format used to store data that is visually easy to write and read by a human. Other formats include XML, DOM, etc.

MySQL
An open source data management system using Structured Query Language (SQL) to manage data, store data, retrieve data, and processing data, ultimately, managing small to large quantities of data for use.

UI
A User Interface (UI) is the space where interaction between humans and machines occur.