

Software Requirements Specification

for

SubScruple

Version <1.0>

Prepared by Kevin Black, Alex Sloan, Shane Bellika

Group Name: BSB teemMaets

|  |  |  |
| --- | --- | --- |
| Kevin Black | 11480710 | kevin.black@wsu.edu |
| Alex Sloan | 11651363 | alex.sloan@wsu.edu |
| Shane Bellika | 11639152 | shane.bellika@wsu.edu |
|  |  |  |
|  |  |  |

October 7, 2018

Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| <1.0> | Kevin Black | Software Requirements Specification section 1 (Introduction) completed. | 09/28/2018 |
| <1.1> | Kevin Black | Software Requirements Specification section 2 (Overall Description) completed. | 09/29/2018 |
| <1.2> |  | Software Requirements Specification section 3 (Specific Requirements) completed. |  |
| <1.3> |  | Software Requirements Specification section 4 (Other Non-Functional Requirements) completed. |  |
| <1.4> |  | Software Requirements Specification section 5 (Other Requirements) completed. |  |
| <1.5> |  | Software Requirements Specification appendix A and appendix B completed. |  |

# 

# Introduction

**SubScruple** is a free, subscription service, web-based application that users can purchase subscriptions through different services offered in their respective location. These subscriptions can be bought at different tier levels and provide extra benefits to the user depending on the tier. In this section, you will be introduced to the purpose of this document, the scope of this product, the intended audience/document overview, definitions/acronyms/abbreviations used throughout this document, document conventions, and references and acknowledgements.

## Document Purpose

The purpose of this document is to centralize the ideas and functionalities, provided by each group member, that ultimately define how the system operates. This will consist of an overview of how the system will function, the high-level, web-based user interface, and how this system will resonate with the team members developing said system and the interested audience and/or stakeholders. Revisions made to the document are a possibility depending on the changing vision of the project or the feasibility of certain features, therefore make sure to check the Revisions section on page iii to get an idea of what features are included within the latest version of the system. Reading over this document should provide the reader with a holistic sense of what **SubScruple** does and the benefits it provides to its subscription users.

## Product Scope

The particular scope of **SubScruple** is to offer subscriptions for various services that are congregated onto this one application. This application will strive to get the best deals possible by being constantly kept up to date with subscription pricing for the various services that the user chooses. The application can also be thought of as a hub for where all user subscription info will be stored, where they can choose to view their subscriptions by logging in to the application. In an ever-expanding market and reliance on technology for most purchases, each user can be certain to receive some benefit out of this web application.

## Intended Audience and Document Overview

This document will be written in such a way that the reader is assumed to have a knowledge of how software requirements specification documents are written, whether they are developers, project managers, marketing staff, users, testers, or documentation writers. The intended audience who are to operate **SubScruple** are those typically with an interest in technology, online shopping, or online reading, as the offered services will mainly be centered on these three categories.

The rest of this software requirements specification document past section 1 will delve into the nature of the system functionalities, with sections discussing topics like product perspective, the operating environment, the decision making behind the user-interface for the application, design and implementation constraints, specific/non-functional requirements, and documentation, just to name a few. This SRS is designed in such a way as to introduce the average reader to **SubScruple**, introduce basic functionalities to the user, and burrow all the way to the specific functionalities of how the application works, if one chooses to read that far. If you are a reader with none to a slight knowledge in how web applications operate, then feel free to just read the first section along with the basic operations for how the web application will work. Otherwise, if you are a reader with moderate to extensive knowledge in how web applications operate, please read the whole document to have a comprehensive understanding of the different functions of **SubScruple**.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| 1. WSUV | Washington State University Vancouver: A public university located in Vancouver, Washington. |

## Document Conventions

This software requirements specification document follows the writing conventions of the IEEE formatting requirements which includes but is not limited to: the use of size 11/12 Arial font, italics are used for comments, each line of the document is single spaced, the document maintains one inch margins, and the main sections begin at the top of it’s own page with a single number followed by subsections with decimal separator format that can span over multiple pages.

## References and Acknowledgments

* Standard IEEE Citation Guide:

<https://learn.wsu.edu/bbcswebdav/pid-2795651-dt-content-rid-91333015_1/courses/2018-FALL-VANCO-CS-320-7661-LEC/IEEE-Citation-StyleGuide.pdf>

* Slide reference material provided by Professor Xinghui Zhao of WSUV:

<https://learn.wsu.edu/webapps/blackboard/content/listContent.jsp?course_id=_189425_1&content_id=_2750588_1&mode=reset>

* Application coding standard (for JavaScript): <https://eslint.org/>
* Application development environments:

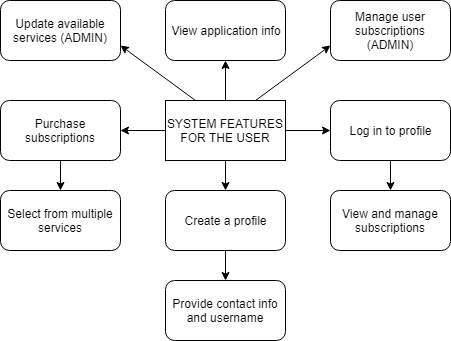
JSFiddle: <https://jsfiddle.net/>

IntelliJ: <https://www.jetbrains.com/idea/?fromMenu>

# Overall Description

## Product Perspective

As the name may entail, **SubScruple** was formed out of the idea of a subscription based business model that concentrates the subscription purchases of every user in one place. The aim of this product is to collect subscription data from the user and store it internally in such a way that reduces the time and complexity of managing subscription records. This product is simply not a follow-on member of a higher product family, but a new, self-contained product with inspiration drawn from current subscription based models/applications. The following high level diagram showcases all of the features that are to be implemented for the system.



## Product Functionality

The ideal goal of **SubScruple** is to make it easy for users to manage current subscriptions/purchase new subscriptions of various services. This will be done through a variety of functions that the user can perform, including but not limited to:

* Returning to the website will prompt the user to log in or create an account.
* If the user doesn’t not have any login information to provide, they can create a free profile so as to actually have an account for managing/purchasing subscriptions.
* Purchasing subscriptions from a variety of services, mainly from other websites that contain some form of e-commerce or digital entertainment. The services will also contain URL’s to the websites if they wish to have a more indepth look at what kinds of additional benefits the specific service provides.
* Able to view the application info, with information such as: the developers who created the application and their roles, the mission of the application, contact info and location of the developers, and so on.
* If a user logs in and they have administrative privleges, they have the additional benefits of being able to update available services offered by **SubScruple** or being able to manage normal user subscriptions through the internal database.

## Users and Characteristics

Generally speaking, there will be a few different, yet similar, classes of users who will operate **SubScruple**: the user who uses this application to keep track of the various subscriptions they currently own, the user who uses this application to purchase subscriptions through different tiers so as to receive extra benefits of some sort, the user who is a combination of the previously mentioned ones, and an administrator who oversees the database of user subscriptions and makes sure the integrity of the application is not compromised. The average user who comes across this application will more than likely have some knowledge when it comes to online purchases and managing digital subscriptions, so the user interface and functionalities are to be developed in such a way that it is to be assumed that the user doesn’t have to be initially guided through the process of how to use this application. If the user wishes to know what they can do through this application, they have the option of consulting the ‘About’ section of the application. Yet at the same time, the characteristics of this application will not be so complicated that a new user will be overwhelmed as how to navigate the system, and will be given a simplistic, slightly modern style.

## Operating Environment

Running this application should not be too intensive of a workload for modern computers to handle, although the recommended system requirements are listed below.

**User Requirements – Hardware:**

* Operating System: Windows 8 or higher/macOS 10.7 or higher
* CPU: at least 400 MHz
* RAM:at least 512 MB

**User Requirements – Software:**

* Browser: Google Chrome, Mozzile Firefox, Microsoft Edge, or Apple Safari.

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).

TO DO: In this section you need to consider all of the information you gathered so far, analyze it and correctly identify relevant constraints.>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.

TO DO: You will not actually develop any user-manuals, but you need to describe what kind of manuals and what kind of help is needed for the software you will be developing. One paragraph should be sufficient for this section.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

TO DO: Provide a short list of some major assumptions that might significantly affect your design. For example, you can assume that your client will have 1, 2 or at most 50 Automated Banking Machines. Every number has a significant effect on the design of your system. >

# Specific Requirements

## External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., Cancel) that will appear on every screen, error message display standards, and so on. Define the software components for which a user interface is needed.

TO DO: The least you can do for this section is to describe in words the different User Interfaces and the different screens that will be available to the user. Optional: You may also provide an initial Graphical User Interface design (does not have to be final).>

### Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware. You are not required to specify what protocols you will be using to communicate with the hardware, but it will be usually included in this part as well.

TO DO: Please provide a short description of the different hardware interfaces. If you will be using some special libraries to communicate with your software mention them here. In case you have more than one hardware interface divide this section into subsections.>

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems (Windows? Linux? Etc…), tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

TO DO: The previous part illustrates some of the information you would usually include in this part of the SRS document. To make things simpler, you are only required to describe the specific interface with the operating system.>

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

TO DO: Do not go into too much detail, but provide 1-2 paragraphs were you will outline the major communication standards. For example, if you decide to use encryption there is no need to specify the exact encryption standards, but rather, specify the fact that the data will be encrypted and name what standards you consider using. >

## Functional Requirements

*< Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions with specific explanations regarding every function.*

*TO DO: Break the functional requirements to several functional areas and divide this section into subsections accordingly. Provide a detailed list of all product operations related to these functional areas.*

## Behaviour Requirements

### Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration.

TO DO: Provide a use case diagram which shows the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram.>

# Other Non-functional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

TODO: Provide relevant performance requirements based on the information you collected from the client. For example you can say “1. Any transaction will not take more than 10 seconds, etc…>

## Safety and Security Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied. Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements.

TODO:

* Provide relevant safety requirements based on your interview with the client or, on your expectation for the product.
* Describe briefly what level of security is expected from this product by your client and provide a bulleted (or numbered) list of the major security requirements.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Portability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write “This software shall be maintainable…” Indicate how you plan to achieve it, etc.>

# Other Requirements

<This section is **Optional.** Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

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.>*

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>