

Software Requirements Specification

for

Django\_Web\_App

Version <1.0>

Prepared by

|  |  |  |
| --- | --- | --- |
| Muniker Aragon | 11632391 | Muniker.aragon@wsu.edu |
| Trever Hibbs | **11577861** | Trever.hibbs@wsu.edu |

|  |  |
| --- | --- |
|  |  |
| Date: | 10/25/2019 |
|  |  |
|  |  |
|  |  |

Content

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 Behavior 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 |
| --- | --- | --- | --- |
| Final Draft | Trever Hibbs  Muniker Aragon | Final version of SRS, contains all relavent information to the design specifications of the system. | 10/25/19 |

# Introduction

## Document Purpose

This document provides a descriptive view of the scope and specifications that will be implemented in the software application. This document describes the intended user interaction with the software in terms of accessibility, safety, and performance. It will also carefully explain the functionality and architecture of the software along with any present constraints. Some design components that will be analyzed in the following sections of this document will be the registration component, user profile, and the software's main feature. This document will also cover the interface requirements that will be implemented to provide an excellent user experience. Lastly, this document will cover the performance of the software in terms of speed, accessibility, and any precautions that must be followed.

## Product Scope

The objective for this software is to provide users an efficient manner of overseeing the price of any online product that they wish to purchase. Users will be able to create a personalized profile where they can store multiple products by simply entering the URL of the products site. This application will then save this URL and continuously monitor inform the user of a drop in the product’s price. The main benefits of this application is to provide users a place where they can save and monitor all the items that they wish to buy while eliminating the time that it would take to search each of these individually.

## Intended Audience and Document Overview

This document is mainly directed to the users or anyone who is interested in the design architecture of this application. The sections that will be covered in this document will provide information and diagrams to demonstrate the major components of the system and also describe how smaller subsystems  interact with one another. This document will also cover the underlying operating system of the application as well as the main functionality and purpose that this product will be able to provide. The sections of this document will examine the different levels of interfaces of the software from hardware to user interaction. Lastly, this document will dive into the non-functional requirements of the systems such as performance, quality, safety and  security.

## Definitions, Acronyms and Abbreviations

URL: Uniform Resource Locator

MVC: Model View Controller

AWS: Amazon Web Services

E2C: Elastic Compute Services

*URI: Uniform Resource Identifier*

*HTTPS: Hypertext Transfer Protocol Secure*

*HTML: Hypertext Markup Language*

*SQL: Structure Query Language*

*OWASP: Open Web Application Security Project*

*MVC: Model View Controller*

*WSGI: Web Server Gateway Interface*

## Document Conventions

<In general this document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1” margins found in this template. For Section and Subsection titles please follow the template.

TO DO: Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. Sometimes, it is useful to divide this section to several sections, e.g., Formatting Conventions, Naming Conventions, etc.>

This document follows the IEEE formatting requirements. All text is Arial font

## References and Acknowledgments

Django Documentation.Internet:https://www.djangoproject.com/, [10,25,2019].

Apache.Internet: <http://www.apache.org/>, [10,25,2019].

Oracle, “MySQL Documentation Client/Server Protocol,”

<https://dev.mysql.com/doc/internals/en/client-server-protocol.html>, [Oct. 24, 2019]

Google, “Secure your site with HTTPS,”

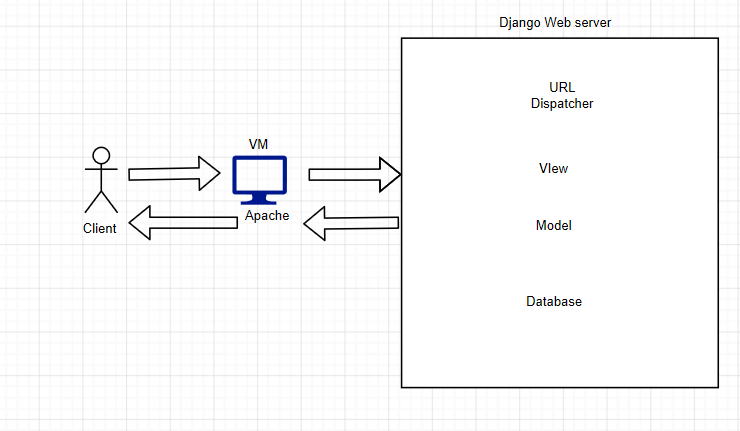
<https://support.google.com/webmasters/answer/6073543?hl=en>, [Oct 25, 2019]

OWASP, “Principle,” <https://www.owasp.org/index.php/Category:Principle> [Oct 25, 2019]

# Overall Description

## Product Perspective

This project implements an MVC architecture by implementing a Full-Stack Web application using the Django framework. This Model-View-Controller will separate the different responsibilities of our project such as databases, views and templates. An overview of the system interactions is represented in the diagram shown below. In this diagram a client will pass a request to the Web Server running an Ubuntu virtual machine. The Server will then utilized WSGI scripts to communicate with the URL dispatcher. The URL Dispatcher will decide  which views to execute and from here the views will then select which templates to render and if any models need to be accessed.



## Product Functionality

* The user interface to the app shall be an interactive form-based interface.
* All users shall have the ability to register an account using a unique email address not yet registered with the system, accounts shall be password protected.
* Users who have registered accounts shall be able to login to the web app using their credentials.
* Any user with a verified account shall have the ability to submit several online store pages for monitoring when logged into their account.
* The user shall have access to all data conected to their account. Including updates on the online store pages submitted for monitoring.
* The user shall have the ability to remove online store pages from their accounts monitoring list.
* Users shall have the ability to log out of their accounts.
* Users shall have the ability to delete their accounts.
* The system shall inform the user of what online stores are supported by the system.
* The system shall scrape any online store page that is submitted to it for the price of the relevant item. The system shall then store the relevant data on that item for later use.

## Users and Characteristics

All users shall come from the general consumer population. The most important users for our application will be those whose financial situation incentivizes them to seek discounted items. The least important user for our application will be those whose financial situation is not as severe and so would not require them to use our application as frequently. The characteristics of these users will vary drastically. Users may be disabled in many ways or have poor skills with technology these variables will need to be taken into account.

## Operating Environment

The environment that will be implemented for this software is composed of virtual machine running an Apache Web Server. This software utilize Amazon Web Services to run an E2C instance whose operating system is Ubuntu 18.04. This instance will run an Apache Web Server that will handle client requests from different users. Finally the web application utilizes the django web server to communicate with the Apache server via WSGI. The scripts contain in WSGI control the flow of communication between URLs, views, models, and database.

## Design and Implementation Constraints

2.5.1 The system will need to provide accurate information. Therefore updates on pricing data must be quick, about one update per hour when the user is not signed in and update within 10 seconds on user request.

2.5.2 All sensitive user data such as passwords and email addresses will need to be secure. This will require proper hashing and salting of relevant data. Additionally, all source code must comply with Open Web Application Security Project (OWASP) guideline and all data must be validated on the server-side.

## User Documentation

This application will provide a specific web page designed to teach users how to navigate and interact with the application. This page will contain images and text that will help instruct users on how they can add new items to their shelf, and how they can check for notifications on the drop of prices in their items.

## Assumptions and Dependencies

1. The system will assume that the prices displayed on online store listings are truthful.
2. The system will depend on the Django web application framework for much of its functionality.
3. The front end of the system will depend on bootstrap for its front end appearance and functionality.
4. For web scraping functionality the system will depend on the Beautiful Soup Python library.

# Specific Requirements

## External Interface Requirements

### User Interfaces

The different types of user interfaces that will be implemented in this application are a Homepage, Registration page,Login Page,How to use page, Add item page, My items page, and an analytics  page. A live preview of the basic layout of this page can be seen by searching the following link<http://3.91.252.175/>.

### Hardware Interfaces

This application will be contained in a virtual machine located in Amazon Web Services. This VM will provide the same components as normal computer in terms of CPU, memory, storage, and networking capacity. We will be using this virtual hardware to store all the contents of the website and also to run the Apache web server which will be in charge of receiving and maintain connections from other client machines.

### Software Interfaces

The application shall use the Django web application framework for back end systems and for front end systems the application shall use bootstrap. For data storage, the system shall use MySQL and SQLite databases. For user data and any other private or sensitive information mySQL databases will be used for storage. For non-sensitive data such as pricing information, SQLlite servers will be implemented. The MySQL database will be hosted on a Linux server. All web scraping shall be accomplished using Beautiful Soup, a python library for pulling data from HTML and XML files.

### Communications Interfaces

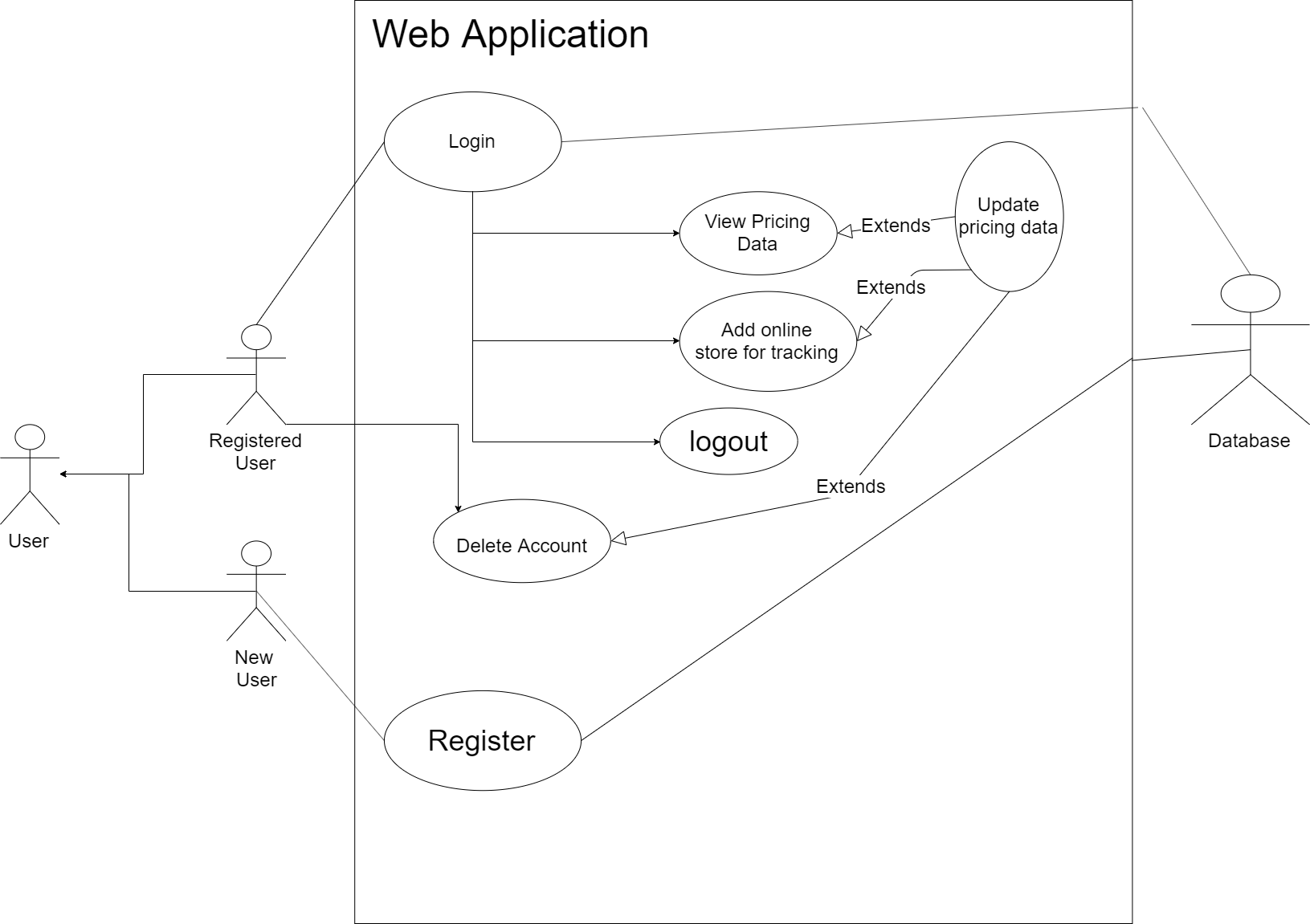
For communication between the web application and the Mysql server, the system shall use the Mysql client/server protocol. This will allow the web application to store and retrieve data as needed. For communication between the user’s browser and the web application, the Uniform Resource Identifier (URI) used shall be HTTPS. By using HTTPS the system will be resistant to man in the middle attacks and eavesdroppers.

## Functional Requirements

The functional components that will be implemented in this application will be a registration and login system that will allow users to create a personalized location to save Items that they wish to monitor. Another main component that will be implemented for this system is a model that creates a one to many relationship between a User and different products. The application will implement a web scraping component which will parse the html from a product and locate where the price is embedded. The last component that will be implemented is an Update component that will be in charge of periodically grabbing each User and updating all of their items.

## Behavior Requirements

### Use Case View



# Other Non-functional Requirements

## Performance Requirements

The system shall update price data every 10 minutes at the most. This requirement exists to ensure that users are satisfied by the responsiveness of the application. All web pages shall fully load content within 5 seconds, and the loading of pricing analytics charts shall take no more than 10 seconds. These restraints exist to ensure a quality experience for the user. Finally, while the user waits for loading times indicators should display to inform the user of what processes the system is currently performing.

## Safety and Security Requirements

The system shall comply with the Open Web Application Security Project guidelines. These guidelines will ensure that all source code is written in a secure manner. The system will also follow the HTTPS best practices laid out by Google. Finally, the system shall store sensitive user data using hashing and salting techniques. The most important user data will be passwords and email addresses. Every communication process involving this data shall be thoroughly validated. Also during the development process, every action taken with this user data will be carefully reviewed and evaluated if necessary.

## Software Quality Attributes

4.3.1 This system shall be feature a responsive interface. To achieve this requirement all user interactions with the website must give the user meaningful feedback. When a user requests analytics while the system loads information on what the system is doing shall be displayed. This will provide the user with a sense that what they have asked from the system is being handled as quickly as possible. Additionally, all wait times should be as fast as possible so that the user does not have their workflow interrupted by the application for more than 10 seconds at a time.

4.3.2 The system shall be reliable. To achieve this requirement all communication systems shall be thoroughly tested and reviewed for defects. This will ensure that the system does not lose functionality unexpectedly while attempting to send or receive data. The system shall be able to provide 24/7 service to all clients only coming offline for patching.

4.3.3 The system shall be flexible when it comes to what online store pages are compatible with the system. This means that all major online retailers should be supported by the application so that when the user inputs a URL to the system it is excepted a majority of the time. To achieve this the application must be configured individually for use with the most popular online retailers and tests on compatibility with these retailers must be conducted regularly throughout the development process.

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>