**Loyalty Crawler**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**Version 1.0**

***Place project logo here***

UTD Student Group:

Anthony Spencer (Group Leader)  
Alexander Baselice  
Alexander Lundin  
Jairo Galarza  
Joseph Samonte  
Tony Nhan

RECORD OF CHANGES

\*A - ADDED M - MODIFIED D – DELETED

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VERSION NUMBER | DATE | NUMBER OF FIGURE, TABLE OR PARAGRAPH | **A\* M D** | TITLE OR BRIEF DESCRIPTION | CHANGE REQUEST NUMBER |
| 1.0 | 02/15/2019 |  |  | Second Deliverable |  |
| 1.1 | 02/23/2019 | All pages | M | Fixed all formatting minutiae | DCR-SRS-1.1 |
| 1.1 | 02/23/2019 | Page 1 | M | Fixed team name spellings | DCR-SRS-1.1 |
| 1.1 | 02/23/2019 | Page 4 | A\* | Added abstract | DCR-SRS-1.1 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**TABLE OF CONTENTS**

**Section Page**

[**SECTION 1. INTRODUCTION**](#_tyjcwt) **5**

[1.1 PURPOSE](#_3dy6vkm) 5

[1.2 SCOPE](#_1t3h5sf) 5

[1.3 DEFINITIONS](#_4d34og8) 5

[1.4 REFERENCES](#_2s8eyo1) 6

[1.5 OVERVIEW](#_17dp8vu) 6

[**SECTION 2. OVERALL DESCRIPTION**](#_26in1rg) **7**

[2.1 PRODUCT PERSPECTIVE](#_lnxbz9) 7

[2.2 PRODUCT FUNCTIONS](#_35nkun2) 7

[2.3 USER CHARACTERISTICS](#_1ksv4uv) 8

[2.4 CONSTRAINTS](#_44sinio) 8

[2.5 ASSUMPTIONS AND DEPENDENCIES](#_2jxsxqh) 8

[**SECTION 3. SPECIFIC REQUIREMENTS**](#_z337ya) **9**

[3.1 EXTERNAL INTERFACE REQUIREMENTS](#_3j2qqm3) 9

[3.2 FUNCTIONAL REQUIREMENTS](#_1y810tw) 9

[3.3 PERFORMANCE REQUIREMENTS](#_qsh70q) 20

[3.4 LOGICAL DATABASE REQUIREMENTS](#_3as4poj) 20

[3.5 DESIGN CONSTRAINTS](#_1pxezwc) 21

[3.6 SOFTWARE SYSTEM ATTRIBUTES](#_49x2ik5) 21

[**APPENDIX A. SAMPLES FILES FROM CUSTOMER**](#_2p2csry) **23**

**LIST OF FIGURES**

**Figure Page**

Figure 1. Initial design brainstorming. 7

Figure 2. Use case model 1 11

Figure 3. Use case model 2 13

Figure 4. Use case model 3 15

**LIST OF TABLES**

**Table Page**

Table 3-1. Graphical User Interface And User Experience Requirements Lookup Table 10

Table 3-2. Web Crawler Mechanism Requirements Lookup Table 12

Table 3-2. Search Session Requirements Lookup Table 14

Table 3-4. Artifact File Reserve Requirements Lookup Table 17

Table 3-5. AWS Database Requirements Lookup Table 19

# ABSTRACT

This document is the starting point of our requirements elicited from the customer. Here, we define specifically what the product will do. Before describing the requirements, we outline all assumptions and constraints. From there, we move into the specific functional requirements. The requirements are organized by system feature. Each feature has an acronym hyphenated appropriately, along with tagging information on the type of requirement. The tag information contains functional or nonfunctional, priority and number. This scheme ensures traceability between documents and simple modifications to the requirements. Each feature is decoupled from other features so editing one set will not require modifications to any other set. Each system feature comes complete with a requirements lookup table for ease of reading.

# SECTION 1. INTRODUCTION

## 1.1 PURPOSE

In this document, the UTD Student group provide an outline framework for which technical development is to follow in the oncoming implementation phase. This framework details the requirements, or constraints on behavior, that are desired by the client for the web crawler product. More specifically, this framework includes the primary (and secondary) objectives of the product, a broad description of desired user interaction with the product, and key attributes necessary to product efficiency. Furthermore, this document has been drafted in a fashion to record future additions and revisions to the product requirements throughout the project lifecycle, which the UTD Student Group anticipates.

## 1.2 SCOPE

As part of a course facilitated by the University of Texas at Dallas, the UTD Student Group will have the remainder of the semester, or approximately three months, to complete the project; the intended date of delivery is May 3rd, 2019. In this project, the group will create a single web application, whose three core components consists of a web crawler search mechanism, a database to hold found artifacts, and a graphical user interface for the user to interact with these artifacts. As specified by the client, this tool is only to retrieve relevant artifacts (or hard data) pertaining to the loyalty rewards programs offered by vendors.

## 1.3 DEFINITIONS

1. artifacts

*The term artifacts refers to any and all documents retrieved by the web crawler, as well as the documents stored in the crawler cache and artifact file reserve. This includes, but is not limited to, HTML files, image files (.png, .jpeg, etc.), and text files (.txt, .pdf, .docx, etc.).*

1. search session

*The term refers to an instance of activating the web crawler for a duration of time, and the resulting artifacts for which it finds.*

1. crawler cache

*This term refers to the architectural component (GUI & database structures) that shall store artifacts found during a single search session.*

1. artifact file reserve

*This term refers to the architectural component (GUI & database structures) that the user shall use to organize and store artifacts deemed relevant from the crawler cache.*

## 1.4 REFERENCES

## 1.5 OVERVIEW

This specification document is organized as follows:

1. Section 1, Introduction
2. Section 2, Overall Description
3. Section 3, Specific Requirements
4. Appendix A.  *SAMPLES FILES FROM CUSTOMER*
5. Appendix B.  *DESCRIPTION OF PROBLEMS SOLVED*
6. Appendix C. *PACKAGING INSTRUCTIONS FOR CODE*
7. Appendix D. [*COST ANALYSIS STUDIES*](#_23ckvvd)

Section 1’s purpose is to give the reader some quick information about the requirements document. In Section 2 the UTD Student Group describes important assumptions and constraints about how the final product will function. In Section 3 we list the requirements. In Section 3.2 we list specific functional requirements. Section 3.2 gives a specific and easy to see breakdown of the requirements. This section shows a broad feature (like a User Interface) and breaks it down further into specific small requirements. At the end is a document change request (DCR) that can be used to request changes to this document.

# 

# SECTION 2. OVERALL DESCRIPTION

## 2.1 PRODUCT PERSPECTIVE

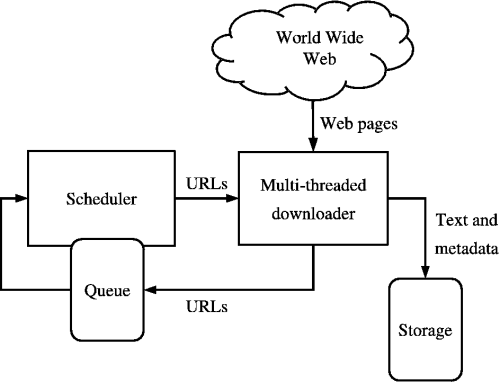
The loyalty web crawler product is a web crawler program that searches the world wide web pages for information regarding possible loyalty programs. The framework of the web crawler allows for a vast number of web page content to be collected with data specifically curated for the user’s need. This loyalty web crawler product currently consists of three major components: the web crawler itself, the database used to store artifacts found, and the web application user interface that hosts, controls, and displays the information retrieved by the web crawler.

* The loyalty web crawler program is a software program that employs special software robots, called spiders, to build indexed lists of words found on website pages. These focused lists contain the words within the web page and where the words are found; here is where the automated search and collection for loyalty program information will occur.
* The database is a simple database using Amazon Web Services’ database service. This database will store all the artifacts found by the web crawler search program.
* The website application user interface is the website interface that hosts, organizes, controls, and adjusts the data and data collecting behavior of the loyalty crawler. This is the only part of the software product that the user interacts with.

## 2.2 PRODUCT FUNCTIONS

The loyalty crawler will be gathering information from the internet. Multiple web pages will be searched then sent to a multi-threaded downloader. The data will then be sent to a queue. Once in the queue, the scheduler will analyze views, clicks, and judge importance. The URLs are then sent back to the multi-threaded downloader. Finally, the text and metadata are sent to storage. See Figure 1 on the next page for a visual representation of our brainstorming so far.

Figure 1. Initial Design Research



## 2.3 USER CHARACTERISTICS

Any user familiar with basic concepts of using a personal computer is normally able to use the program. Experience with search engines or experience with searching for anything on a computer is a plus. As a result, no specific requirements are affected by the user’s characteristics.

## 2.4 CONSTRAINTS

The project loyalty crawler must be complete with a limited time, limited budget/resources, and within the scope of the project. The constraint is set to make sure the project meet those limit.

1. The project’s time constraint is set at the schedule deadline. All project documentations and software build must be delivered by the schedule deadline and there will be no time extension. Team members must constantly keep check with the deadline to make sure the job is deliver on time.
2. The project have no budget, therefore resources available for this project is limited to team member’s personal computer and database provide by AWS RDS free tier. Any additional resources add to the project must all be free product and adhere to the project scope.
3. The project must adhere to the scope and not deviate outside of it. The project scope is a web crawler application that is web based. The application will “crawl” through the web and obtain any artifacts determined to be relevant to loyalty rewards programs for customers. These artifacts will be store in a database with predetermined requirements and view with an easy to use UI. To not deviate from the scope, team member must build many prototype to test the application build.

## 2.5 ASSUMPTIONS AND DEPENDENCIES

Assumptions and dependencies list:

1. The database service is provided by AWS RDS free tier. Uptime and downtime of the database are solely dependent upon Amazon Web Service.
2. Artifact discovery by the web crawler is dependent on the web page structure. The web crawler discovery mechanism must be able to locate and distinguish the different sections of a web page. Some common sections are web page headings, bodies and content. Correctly identification of web page structure will allow the crawler to discard low quality web pages and store high quality web pages.
3. The application will be built on local personal devices of team members. All team members must be present during the implementation phase.

# 

# SECTION 3. SPECIFIC REQUIREMENTS

## 3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 Inputs

3.1.1.1 Valid inputs for the user interface shall be text input for a filename, text inputs for an additional key word or phrase for search criterion.

3.1.2 Outputs

3.1.2.1 Valid outputs for the user interface shall be the artifacts found by the web crawler mechanism. The format of these artifacts shall be HTML pages and text files.

## 

## 3.2 FUNCTIONAL REQUIREMENTS

3.2.1 Graphical User Interface and User Experience

Acronym for requirements lookup table:

GUI-UX

3.2.1.1 Introduction of feature

The graphical user interface and user experience pertain to the psychological acceptability of the web application. Requirements described in this feature set will ensure a degree of professional attention to button placement and menu intuitiveness. These requirements will define consistent patterns of organization for every graphical feature which is part of this software system. These are general gui requirements that shall be applied to all menus.

3.2.1.2 Associated Functional Requirements

**GUI-UX-F-1-E**

All fonts shall be Arial, 10 point size.

**GUI-UX-F-2-E**

All preferences that are immediately editable by the user in the current state, shall save when the user makes a change.\

Example: An options window might have multiple tabs where the user can configure their preferences. So, if the user makes a change on the currently active tab, that change shall be saved immediately. This requirement ensures the user will not have to click save after making changes. This requirement also makes sure the users changes are saved immediately after each change.

Table 3-1. Graphical User Interface and User Experience Lookup Table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Req. type** | **prio.** | **Desc** | **Name** |
| **GUI-UX-F-1-E** | **Functional** | **Essential** | **Font** | **AL** |
| **GUI-UX-F-2-E** | **Functional** | **Essential** | **Preference saves** | **AL** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

3.2.2 Graphical User Interface Functional Requirements

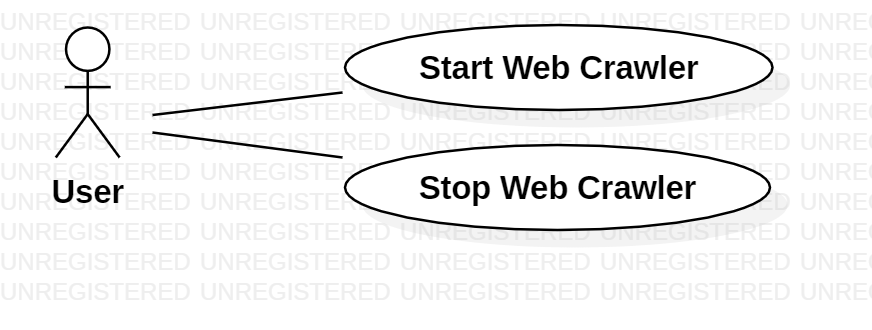
Acronym for requirements lookup table:

GUI-F

3.2.2.1 Introduction of feature

The command the user interface request of the web crawler to start for scouring the web.

Figure 2. Use case model 1.



3.2.2.2 Associated Functional Requirements

**GUI-F-1-E**

The user interface shall provide a simple button to activate and deactivate the web crawler mechanism, which instigates a search session that returns artifacts found by the web crawler.

**GUI-F-2-E**

The user interface shall visibly display the artifacts found by the web crawler in the **crawler cache.**

**GUI-F-3-E**

The user interface shall provide a user defined **artifact file reserve.** See 3.2.4 for the feature description.

**GUI-F-4-E**

The user interface shall provide options to view and organize these artifacts into the **artifact file reserve.**

Table 3-2. Web crawler mechanism requirements lookup table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Req. type** | **prio.** | **Desc** | **Name** |
| **GUI-F-1-E** | **Functional** | **Essential** | **start and stop** | **AS/AL** |
| **GUI-F-2-E** | **Functional** | **Essential** | **include cache** | **AS/AL** |
| **GUI-F-3-E** | **Functional** | **Essential** | **include file resv** | **AS/AL** |
| **GUI-F-4-E** | **Functional** | **Essential** | **include art resv** | **AS/AL** |
|  |  |  |  |  |
|  |  |  |  |  |

# 

3.2.3 Web Crawler Mechanism

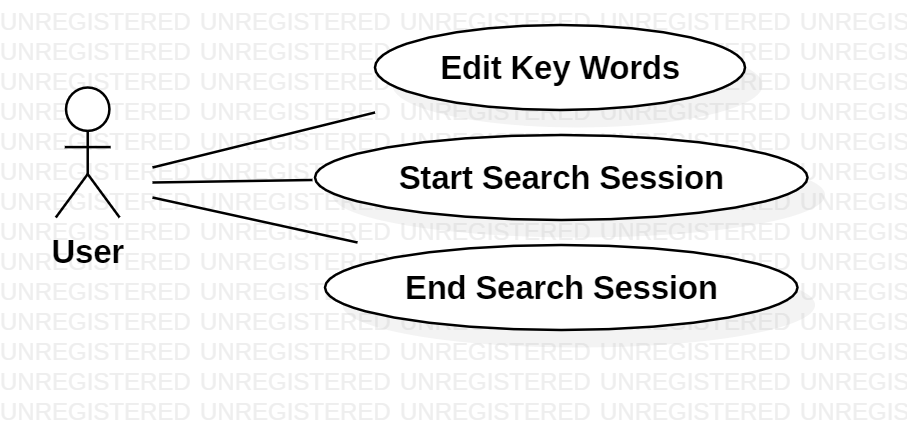
Acronym for requirements lookup table:

WCM-

3.2.3.1 Introduction of feature

The search session feature set describes how the software will present the search results.

Figure 3. Use case model 2.



3.2.3.2 Associated Functional Requirements

**WCM-F-1-E**

The web crawler component shall interact with, but not be constituent to, the graphical user interface component of the application.

**WCM-F-2-E**

The web crawler component shall sift the internet for **artifacts** according to a predetermined search criteria.

**WCM-F-3-E**

The search criteria by which the web crawler component conducts its search shall be customizable and adjustable by the user.

**WCM-F-4-E**

The constraints of each **search session**, such as duration of search and maximum number of retrieved items, shall be adjustable by the user.

Table 3-3 .Search Session Requirements Lookup Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Req. type** | **prio.** | **Desc** | **Name** |
| **WCM-F-1-E** | **Functional** | **Essential** | **Interact with GUI** | **AS** |
| **WCM-F-2-E** | **Functional** | **Essential** | **Sift Web** | **AS** |
| **WCM-F-3-E** | **Functional** | **Essential** | **custom Criteria** | **AS** |
| **WCM-F-4-E** | **Functional** | **Essential** | **adj. constraints** | **AS** |
|  |  |  |  |  |
|  |  |  |  |  |

3.2.4 Artifact File Reserve

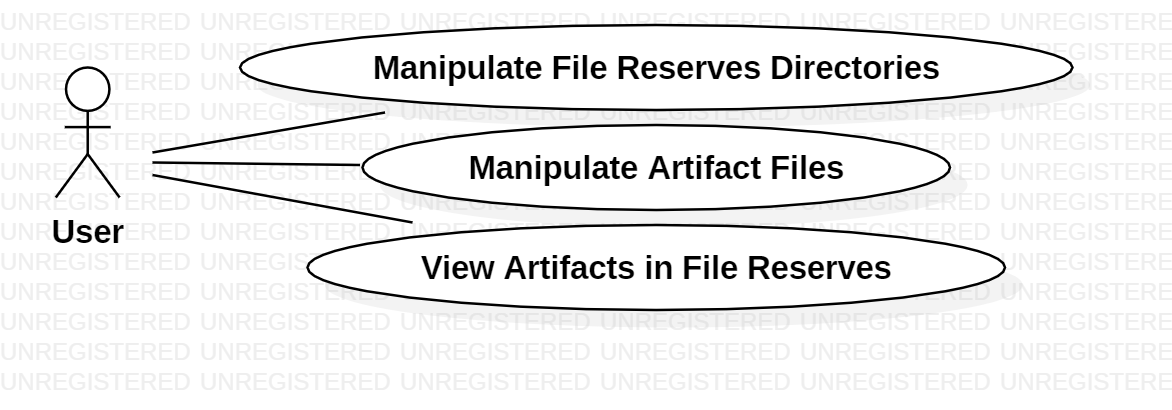
Acronym for requirements lookup table:

AFR-

3.2.4.1 Introduction of feature

The artifact file reserve is relative to the current user. It provides functions similarly to an average file management system, where the user may organize artifacts as they wish. This is the local copy of whatever the user decides to keep from the cache.

Figure 4. Use case model 3.



3.2.4.2 Associated Functional Requirements

**AFR-F-1-E**

The user shall be able to create directories in the artifact file reserve.

**AFR-F-2-E**

The user shall be able to delete directories in the artifact file reserve.

**AFR-F-3-E**

The user shall be able to rename directories in the artifact file reserve.

**AFR-F-4-E**

The user shall be able to copy directories in the artifact file reserve.

**AFR-F-5-E**

The user shall be able to move directories in the artifact file reserve.

**AFR-F-6-E**

The user shall be able to delete artifacts in the artifact file reserve.

**AFR-F-7-E**

The user shall be able to rename artifacts between directories in the artifact file reserve.

**AFR-F-8-E**

The user shall be able to copy artifacts between directories in the artifact file reserve.

**AFR-F-9-E**

The user shall be able to move artifacts between directories in the artifact file reserve.

**AFR-F-10-E**

The user shall be able to view artifacts in the artifact file reserve.

Table 3-4. Artifact file reserve requirements lookup table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Req. type** | **prio.** | **Desc** | **Name** |
| **AFR-F-1-E** | **Functional** | **Essential** | **Create dir** | **AS/AL** |
| **AFR-F-2-E** | **Functional** | **Essential** | **delete dir** | **AL** |
| **AFR-F-3-E** | **Functional** | **Essential** | **rename dir** | **AL** |
| **AFR-F-4-E** | **Functional** | **Essential** | **copy dir** | **AL** |
| **AFR-F-5-E** | **Functional** | **Essential** | **move dir** | **AS/AL** |
| **AFR-F-6-E** | **Functional** | **Essential** | **delete artifacts** | **AS/AL** |
| **AFR-F-7-E** | **Functional** | **Essential** | **rename artifacts** | **AL** |
| **AFR-F-8-E** | **Functional** | **Essential** | **copy artifacts** | **AL** |
| **AFR-F-9-E** | **Functional** | **Essential** | **move artifacts** | **AS/AL** |
| **AFR-F-10-E** | **Functional** | **Essential** | **view artifacts** | **AS/AL** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 

3.2.5 Amazon Web Service Database

Acronym for requirements lookup table:

AWS-DB-

3.2.5.1 Introduction of feature

The database performs common CRUD operations that databases commonly provide.

3.2.5.2 Associated Functional Requirements

**AWS-DB-F-1-E**

The database shall be able to create entries

**AWS-DB-F-2-E**

The database shall be able to retrieve entries

**AWS-DB-F-3-E**

The database shall be able to update entries

**AWS-DB-F-4-E**

The database shall be able to delete entries

Table 3-5. AWS database requirements lookup table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Req. type** | **prio.** | **Desc** | **Name** |
| **AWS-DB-F-1-E** | **Functional** | **Essential** | **Create Entity** | **AS/AL** |
| **AWS-DB-F-2-E** | **Functional** | **Essential** | **Retrieve entity** | **AS/AL** |
| **AWS-DB-F-3-E** | **Functional** | **Essential** | **Update Entity** | **AS/AL** |
| **AWS-DB-F-4-E** | **Functional** | **Essential** | **Delete Entity** | **AS/AL** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 3.3 PERFORMANCE REQUIREMENTS

3.3.1 In case of an error, the error pop-up messages should show up within 2 seconds of found component error.

3.3.2 The saving and export of settings should not take more than 7 seconds after the initiation of the action.

3.3.3 Start and end command for web crawling search session should take no longer than 1 second to execute

## 3.4 LOGICAL DATABASE REQUIREMENTS

3.4.1 Types of Information

The database will store artifact files, which have been defined in section 1.3.

3.4.2 Frequency of Use

The database will be expected to be accessed at most 10,000 times a week.

3.4.3 Accessing capabilities

Accessing through AWS RDS Free Tier service

3.4.4 Data entities and their relationships

Data entities: Artifact & Artifact Categories, User File Reserve Backup, Search Session History

Relationships: TBD, waiting until sample files are received from client. We require these to defined relationships between entities stored.

3.4.5 Integrity constraints

Integrity constraints shall be used to ensure accuracy and consistency of data in a relational database. Any changes made to the database tables shall cascade to all tables. The client shall not have access to the database. The UTD Student group shall write a process specification for testing to ensure live data is not contaminated or deleted.

3.4.6 Data retention requirements

Deleted search session files shall be stored on the user’s machine in a local recycle bin. The database shall not retain deleted items.

## 3.5 DESIGN CONSTRAINTS

3.5.1 Database Constraints

3.5.1.1 Retrieved Artifacts

The artifacts retrieved by a single session of a web crawler search scrap will be limited to 1,000 indexed artifacts.

3.5.1.2 Database In Use

The database is constrained to that provided by the free-tier of Amazon Web Services

3.5.2 Web Interface

3.5.2.1 Protocol

The protocol used shall be HTTP.

3.5.2.2. User Interface

The user interface for the software shall be compatible to any browser such as Microsoft Edge, Mozilla Firefox, Safari, or Google Chrome.

3.5.2.3 Web Application Interaction

The web application shall permit complete navigation and item selection using the keyboard and mouse; touch actions should be allowed if user hardware supports it.

## 3.6 SOFTWARE SYSTEM ATTRIBUTES

3.6.1 Availability

The site should check that the system always functions properly and always pop ups an error message in case of component failure. The error messages appear when something goes wrong so to help locate and prevent availability problems.

3.6.2 Maintainability

The site should be reasonable easy to maintain for new developers who want to make edits to the software after the semester ends. This means when writing any code an effort should be made what the code does easily readable.

3.6.3 Portability

The site should work on all modern browsers updated within the last year.

3.6.4 Reliability

AWS has a Service Level agreement that they will make all possible attempts to have 99.99% uptime.

3.6.5 Security

No security implemented as per client request.

3.6.6 Usability

The site system is easy to handle and navigates in the most expected way with no delays. The system program reacts accordingly and transverses quickly between its states. For example, when pressing the start crawler button, the crawler should start with minimum delay.

**APPENDICES**

# APPENDIX A. SAMPLES FILES FROM CUSTOMER

***Guidance***

This section contains sample files that describe search input parameters and expected output.

**DOCUMENT CHANGE REQUEST (DCR)**

|  |  |
| --- | --- |
| Document Title: **Software Requirements Specification** | Tracking Number:  DCR-SRS-1.1 |
| Name of Submitting Organization:  UTD Student Group | |
| Organization Contact:  amlundin88@gmail.com | Phone: |
| Mailing Address: | |
| DCR Description:  Formatting, general maintenance and deliverable compliance. | Date: 02/23/2019 |
| Change Location:  Page 1, Page 4 | |
| Proposed change:  All formatting minutiae. Fixed team name spellings on page 1. Added abstract section page 4. | |
| Rationale for Change:  Incremental improvement towards final deliverable. Also starting proof of configuration management. | |
| Note: For the ***appropriate authority*** to take appropriate action on a change request, please provide a clear description of the recommended change along with supporting rationale.  Email to:  anthonygaganovspencer@gmail.com  alexbaselice2@gmail.com  amlundin88@gmail.com  jgalarza303@gmail.com  josephisnt@gmail.com  xclearzx@gmail.com  Submit online:  ***Print this sheet and store in this Google Drive folder***  *SE Senior Project/Deliverables/Document Change Request Folder*  DCR Form 1/2009 | |