

Software Requirements Specification Template

The following annotated template shall be used to complete the Software Requirements Specification (SRS). The instructor must approve any modifications to the overall structure of this document.

Template Usage:

Text contained within angle brackets ('<', '>') shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either 'Smart Home' or 'Sensor Network'.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

Acknowledgements:

Sections of this document are based upon the IEEE Guide to Software Requirements Specification (ANSI/IEEE Std. 830-1984).

Static_Code_Checker

Software Requirements Specification

V1.0.0

Feb 1 2020

Matt Brown: Designer | Programmer

Will Fitzgerald: Commenter | Dev Ops | Programmer

Jake Gadaleta: Supreme Leader | Dev Ops | Programmer

Nick Hertzog: Programmer | Designer | Dev Ops

Revision History

Date	Description	Author	Comments
Feb 1 2020	V1.0.0	All	Created the initial document

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

Table of Contents

1. INTRODUCTION	1
1.1 PURPOSE	1
1.2 SCOPE	1
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	1
1.4 OVERVIEW	1
2. GENERAL DESCRIPTION	1
2.1 PRODUCT PERSPECTIVE	1
2.2 PRODUCT FUNCTIONS	2
2.3 USERS AND CHARACTERISTICS	2
2.4 GENERAL CONSTRAINTS	2
2.5 ASSUMPTIONS AND DEPENDENCIES	2
2.6 OPERATING ENVIRONMENT	2
3. SPECIFIC REQUIREMENTS	2
3.1 EXTERNAL INTERFACE REQUIREMENTS	3
3.1.1 <i>User Interfaces</i>	3
3.1.2 <i>Hardware Interfaces</i>	3
3.1.3 <i>Software Interfaces</i>	3
3.1.4 <i>Communications Interfaces</i>	3
3.2 FUNCTIONAL REQUIREMENTS	3
3.2.1 <i><Functional Requirement or Feature #1></i>	3
3.2.2 <i><Functional Requirement or Feature #2></i>	3
3.3 USE CASES	4
3.3.1 <i>Use Case #1</i>	4
3.3.2 <i>Use Case #2</i>	4
3.4 NON-FUNCTIONAL REQUIREMENTS	4
3.5.1 <i>Performance</i>	5
3.5.2 <i>Reliability</i>	5
3.5.3 <i>Availability</i>	5
3.5.4 <i>Security</i>	5
3.5.5 <i>Maintainability</i>	5
3.5.6 <i>Portability</i>	5
3.5 DESIGN CONSTRAINTS	5
3.6 LOGICAL DATABASE REQUIREMENTS	5
3.7 OTHER REQUIREMENTS	5
4. ANALYSIS MODELS	5
4.1 SEQUENCE DIAGRAMS	6
5. CHANGE MANAGEMENT PROCESS	6
A. APPENDICES	6
A.1 APPENDIX 1	7
A.2 APPENDIX 2	7

LIST OF FIGURES

Figure 1: Data Flow Diagram Example 1	6
Figure 2 Data Flow Diagram Example 2	6

LIST OF TABLES

Table 1: Use case 1	4
---------------------	---

1. Introduction

<The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document.>

1.1 Purpose

This SRS document is meant to explain the what this software does, and how it does it. This is mostly a document for our clientele, Teaching Professionals and Interviewers / Interviewees

1.2 Scope

This product runs a localized server using the Python package, Flask. Flask hosts a webserver with the ECMAscript(JavaScript) files: Bootstrap and JQuery, which is heavily used in our front end design. This product will be able to compile Java source code (.java files) and compare their outputs against that of a master file. We wish to take this time to stress that this is a localized project please do not put it on a server as it opens you up to potential security flaws. This can be used to speed up the general grading process or for an interview to constantly check code. This application works much like [LeetCode](#) or [HackerRank](#) to compare and contrast code, but is run locally which can a) speed up runtime and b) keep code private

1.3 Definitions, Acronyms, and Abbreviations

<This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.>

1.4 Overview

This document will continue to list out the functionality of this application along with explaining use cases and explain the dependency choices that we have made. For more information on how this document is formatted please see the above table.

2. General Description

A system built to help in the process of grading either for teachers or in an interviewing process.

- Teacher Grading
 - Allows teachers to instantly grade a class
- Student
 - Allows students to test their code against inputs / outputs given by a teacher
- Interviewer Grading
 - Allows Interviewers to inspect and grade code
- Interviewee

- Allows interviewers to test their code against inputs / outputs given by a teacher

2.1 Product Perspective

This product stands on its own independent from any existing product. It implements functionality present in the Java Runtime Environment to compile code but that's all

<. “WE NEED A DIAGRAM” >

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

This product can:

- Compile Code and Compare the Outputs
 - Teachers and Interviewers can Compile a list of src files and compare their outputs against a master output to return grading information
 -

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate is often effective.>

2.3 Users 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.>

2.4 General Constraints

<This subsection of the SRS should provide a general description of any other items that will limit the designer/developer's options for designing/developing the system.>

2.5 Assumptions and Dependencies

<This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.>

2.6 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

3. Specific Requirements

<This will be the largest and most important section of the SRS. The customer requirements are embodied within Section 2 (functions), but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- Correct
- Traceable (both forward and backward to prior/future artifacts)
- Unambiguous
- Verifiable (i.e., testable)
- Prioritized (with respect to importance and/or stability)
- Complete
- Consistent (with other requirements)
- Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefully organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.>

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communications Interfaces

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 REQ-1: Upload input file

3.2.1.1 Description and Priority

This is a feature for users to use where they can upload a text file containing a series of inputs that will be used to check to see if the code is correct. This requirement is of high priority as it is key to the basic functionality of this software. Rating 9.

3.2.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

The user shall be able to click an upload input file button, navigate through their file explorer to locate the file and then upload the file to the software.

3.2.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute a use case. Include how the product should respond to anticipated error conditions or invalid inputs.

Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

Requirement of uploading a .txt file

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

3.2.1.3.1 REQ-1: For the input file, the user needs to be able to navigate their systems file explorer to locate the file.

.....

3.2.2 REQ-2: Upload Output File

3.2.2.1 Description and Priority

This feature will be available for users to allow the upload of an output file that will be used alongside the dedicated input file. The output file will just be a formatted text file which will be used to check the output of the users compiled code. This feature is of high priority, rated at a 9.

3.2.2.2 Stimulus/Response Sequences

The user shall be able to click an upload output file button, navigate through their file explorer to locate the file and then upload the file to the software.

3.2.1.3 Functional Requirements

Requirement of uploading a .txt file

3.2.2.3.1 REQ-1: For the output file, the user needs to be able to navigate their systems file explorer to locate the file.

3.2.3 REQ-3 - Teacher View

3.2.3.1 Description and Priority

This feature will be utilized by the teacher to view student output. Also the students code will be linked creating the option to click their code and have it open with the teachers default code editor by the file extension.

3.2.3.2 Stimulus/Response Sequences

The user shall click on a button that will direct them to the teacher view part of the application

3.2.3.3 Functional Requirements

Laptop to access the application and then clicking the “Teacher View” button

3.2.4 REQ-4 - Student View

3.2.4.1 *Description and Priority*

This feature will be used to allow students to view their own code as well as their teacher's code as the example output. The feature will allow students to view any errors they have and compare to the example output.

3.2.4.2 *Stimulus/Response Sequences*

The user shall click on a button that will direct them to the student view part of the application

3.2.4.3 *Functional Requirements*

Laptop to access the application and then clicking the "Student View" button

3.2.5 REQ-5 - Interviewer View

3.2.5.1 *Description and Priority*

This feature will allow Interviewer's to look at an interviewee's code and to compare that code to the example output provided by the interviewer.

3.2.5.2 *Stimulus/Response Sequences*

The user shall click on a button that will direct them to the student view part of the application.

3.2.5.3 *Functional Requirements*

Laptop to access the application and then clicking the "Interviewer View" button.

3.2.6 REQ-6 - Interviewee View

3.2.6.1 *Description and Priority*

This feature will be utilized by the teacher to view student output. Also the students code will be linked creating the option to click their code and have it open with the teachers default code editor by the file extension.

3.2.6.2 *Stimulus/Response Sequences*

The user shall click on a button that will direct them to the interviewee view part of the application.

3.2.6.3 *Functional Requirements*

Laptop to access the application and then clicking the "IntervieweeView" button.

3.2.7 REQ-7 - Exporting CSV/PDF Files

3.2.7.1 *Description and Priority*

This feature will allow the teacher or interviewer to export a file containing the score of each submission. This is of high importance because it will allow the teacher to save time by seeing which students have perfect output or have messed up. Priority level is an 8.

3.2.7.2 *Stimulus/Response Sequences*

The user shall be able to click an export file button, navigate through their file explorer to select where the file should be saved.

3.2.7.3 *Functional Requirements*

Having run submissions to create a file.

3.2.8 REQ-8 - Highlight Differences in Output

3.2.8.1 Description and Priority

This feature will be available for all users being accessed once the user submits their code. Its purpose is to let the user see the differences within the submission and given example output.

3.2.8.2 Stimulus/Response Sequences

The user shall submit their work and see a highlighted difference in the output if they are wrong.

3.2.8.3 Functional Requirements

Submit a working file, we will work out an output if the code does not compile.

3.2.9 REQ-9 - Run Student/Interviewee view code

3.2.9.1 Description and Priority

This is a feature available for all users and can be accessed from each view. A user will be able to upload some code files which they can compile. Priority is high with a rating of 9.

3.2.9.2 Stimulus/Response Sequences

The user shall be able to click the upload button and be able to provide a code file through navigating their file explorer.

3.2.9.3 Functional Requirements

Allow for uploading of python, ruby, java, and Go files.

3.2.10 REQ-10 - Comparing code to given output within output file

3.2.10.1 Description and Priority

This is a feature available to all users that will compare the output generated from their code files and input files to the information within the output file provided. Priority is high with a rating of 9.

3.2.10.2 Stimulus/Response Sequences

The user shall be able to view a results screen/document that will tell them their score that they have received from testing their code.

3.2.10.3 Functional Requirements

Allow for output window or file (xlsx/pdf)

...

...

3.3 Use Cases

3.3.1 Use Case #1

Use Case Name	Upload Input File
Reference	Section 3.2.1
Trigger	The reader presses the upload input file button

Precondition	The user has already selected their designated view
Basic Path	<ol style="list-style-type: none"> 1. The user can select the upload input file button. 2. The user will then be prompted with the file explorer where they can navigate their computer to find the proper input file. 3. The user can then select their designated file and it will be uploaded into the software.
Alternative Paths	None
Postcondition	The selected file is uploaded to the software and it visually shows the user that the upload was successful.
Exception Paths	The user may exit the file explorer and abandon the file upload.
Other	None

3.3.2 Use Case #2

Use Case Name	Upload Output File
Reference	Section 3.2.2
Trigger	The reader presses the upload outputfile button
Precondition	The user has already selected their designated view
Basic Path	<ol style="list-style-type: none"> 1. The user can select the upload output file button. 2. The user will then be prompted with the file explorer where they can navigate their computer to find the proper output file. 3. The user can then select their designated file and it will be uploaded into the software.
Alternative Paths	None
Postcondition	The selected file is uploaded to the software and it visually shows the user that the upload was successful.
Exception Paths	The user may exit the file explorer and abandon the file upload.
Other	None

3.3.3 Use Case #3

Use Case Name	Teacher View
Reference	Section 3.2.3
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.4 Use Case #4

Use Case Name	Student View
Reference	Section 3.2.4
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.5 Use Case #5

Use Case Name	Interviewer View
Reference	Section 3.2.5
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.69 Use Case #6

Use Case Name	Interviewee View
Reference	Section 3.2.6
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.7 Use Case #7

Use Case Name	Exporting CSV/PDF Files
Reference	Section 3.2.7

Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.8 Use Case #8

Use Case Name	Highlight Differences in Output
Reference	Section 3.2.8
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.9 Use Case #9

Use Case Name	Run Student/Interviewee view code
Reference	Section 3.2.9
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	
Exception Paths	
Other	

3.3.10 Use Case #10

Use Case Name	Comparing code to given output within output file
Reference	Section 3.2.10
Trigger	
Precondition	
Basic Path	
Alternative Paths	
Postcondition	

Exception Paths	
Other	

...

3.4 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc.).

3.5.1 Performance

3.5.2 Reliability

3.5.3 Availability

3.5.4 Security

<Example: The server on which the Online Journal resides will have its own security to prevent unauthorized *write/delete* access. There is no restriction on *read* access. The use of email by an Author or Reviewer is on the client systems and thus is external to the system. The PC on which the Article Manager resides will have its own security. Only the Editor will have physical access to the machine and the program on it. There is no special protection built into this system other than to provide the editor with *write* access to the Online Journal to publish an article.>

3.5.5 Maintainability

3.5.6 Portability

3.5 Design Constraints

<Specify design constraints imposed by other standards, company policies, hardware limitation, etc. that will impact this software project. Example, the software is required to have a login screen based on company policies.>

3.6 Logical Database Requirements

<Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc?>

3.7 Other Requirements

<Catchall section for any additional requirements that did not belong to the previous sections. If there are none, exclude this section>

4. Analysis Models

<List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.>

3.1 Sequence Diagrams

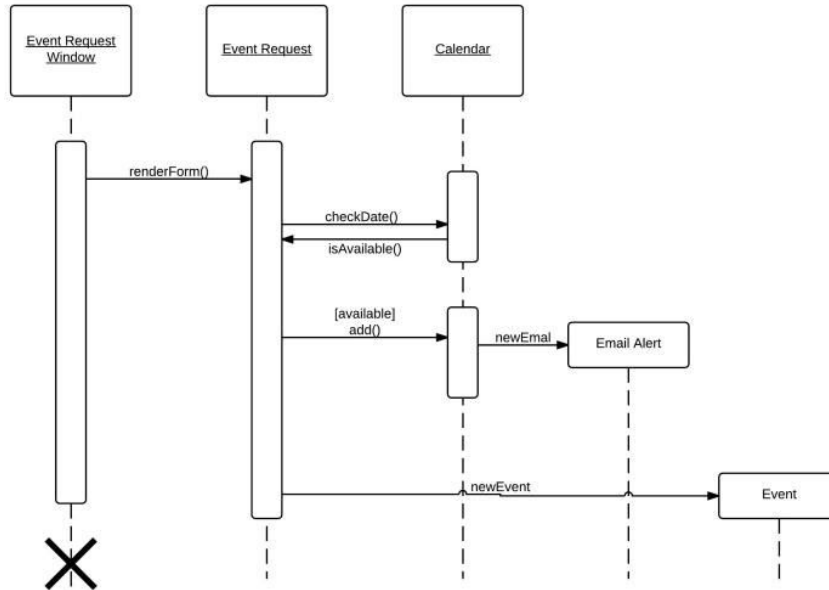


Figure 1: Data Flow Diagram Example 1

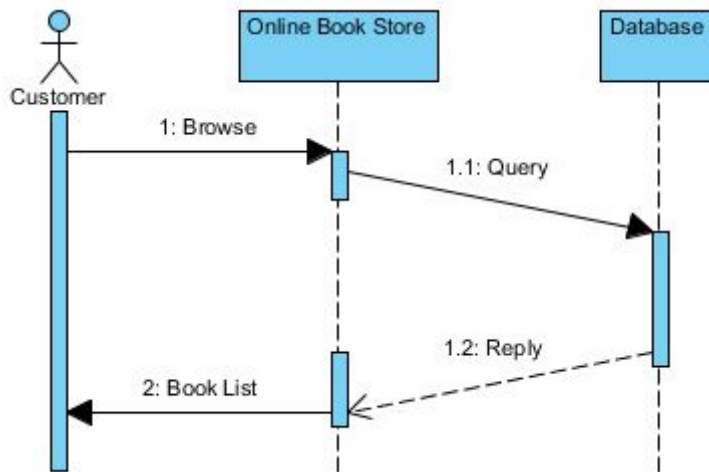


Figure 2 Data Flow Diagram Example 2

<At least one sequence diagram should be included for each requirement or use case.>

5. Change Management Process

<Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.>

References

A. Appendices

<Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements. Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.>

A.1 Appendix 1

A.2 Appendix 2