Software requirement specification (SRS) document template

Project name: UVSim

Date: 2024-09-23

Version: 0.1

By:

Revision history			
Version	Author	Verson description	Date completed

Review history			
Approving party	Version approved	Signature	Date

Approval history			
Reviewer	Version reviewed	Signature	Date

Table of contents

1 Introduction

Describe the purpose of the document.

1.1 Product scope	List the bene□ts, objectives, and goals of the product.

Our product is for students to learn to use the UVSim simulator it must be able to read from files the correct words to read.

1.2 Product value	Describe how the audience will □nd value in the product.

They will find it helpful that it runs the code without error!

1.3 Intended audience	Write who the product is intended to serve.

It is intended to serve students and software writers

1.4 Intended use	Describe how will the intended audience use this product.

It is intended that one writes the code to operate on in a .txt file and then our software will perform the operations

1.5 General description	Give a summary of the functions the software would perform and the features to be included.

There are the different words denoted by the first two digits of a word.

2 Functional requirements

List the design requirements, graphics requirements, operating system requirements, and constraints of the product.

1. READ words from the keyboard, so that I can perform operations on a number of my chosen at run time without editing source code.
2. WRITE words to the screen, so that I can see the results of the code I've written
3. LOAD words from memory into the accumulator, so that I can operate on memory
4. STORE words in accumulator to the memory, so that I can save the operations I've done
5. ADD words to the accumulator, so that I can find the sum of any two numbers between -9999 & 9999 *that dont overflow
6. SUBTRACT words (from memory) from the accumulator, so that I can compare (if negative then <, if positive then >)
7. DIVIDE the accumulator by a word from memory, so that I can determine if a number is even or odd, based on if there is a remainder
8. MULTIPLY the accumulator by a word from memory so that I can square numbers by multiplying a word by itself
9. BRANCH to a specific location, so that I can loop the program
10. BRANCH to a location if the accumulator is NEGative, so that I can loop if A>B as described in 6.
11. BRANCH to a location if the accumulator is ZERO, so that I can loop a specific number by subtracting the accumulator by one each iteration.
12. HALT the program, so that it doesn't run forever
13. Differentiate between command values and data values so that the program treats each appropriately.

USER STORIES

I am a mathematician who needs to STORE words to the accumulator, and then ADD words from the memory with the accumulator so that I can perform repeated operations.

As a student of software writing, I need to use the different BRANCH words to create conditional operations.

External interface requirements

3.1 User interface requirements

Describe the logic behind the interactions between the users and the software (screen layouts, style guides, etc).

There needs to be a terminal interface that is able to take input and read specified files and the perform the operations.

3.2 Hardware interface requirements

List the supported devices the software is intended to run on, the network requirements, and the communication protocols to be used.

3.3 Software interface requirements

Include the connections between your product and other software components, including frontend/backend framework, libraries, etc.

3.4 Communication interface requirements

List any requirements for the communication programs your product will use, like emails or embedded forms.

4 Non-functional requirements

4.1 Security	Include any privacy and data protection regulations that should be adhered to.
4.2 Capacity	Describe the current and future storage needs of your software.
4.3 Compatibility	List the minimum hardware requirements for your software.
4.4 Reliability	Calculate what the critical failure time of your product would be under normal usage.
4.5 Scalability	Calculate the highest workloads under which your software will still perform as expected.
4.6 Maintainability	Describe how continuous integration should be used to deploy features and bug □xes quickly.
4.7 Usability	Describe how easy it should be for end-users to use your software.
4.8 Other	List any additional non-functional requirements.

5 Definitions and acronyms