**Software Requirements Specification (SRS) Document for UVSim**

**1. Introduction**

**1.1 Purpose**

This document specifies the requirements for the UVSim simulator, a tool designed for computer science students to learn and execute BasicML machine language programs. The document outlines both the functional and non-functional requirements necessary for the development and operation of the UVSim.

**1.2 Scope**

UVSim is a virtual machine simulator that will allow students to load, execute, and interact with BasicML programs. It features a CPU, an accumulator register, and a 100-word memory. The simulator will interpret BasicML instructions and provide functionalities for I/O operations, load/store operations, arithmetic operations, and control operations.

**1.3 Definitions, Acronyms, and Abbreviations**

* **BasicML**: Basic Machine Language used by UVSim.
* **CPU**: Central Processing Unit of the simulator.
* **Accumulator**: A special register used for arithmetic operations and storing temporary results.
* **Instruction**: A command in BasicML consisting of an opcode and an operand.
* **Memory**: Storage area of UVSim, comprising 100 words.
* **UI**: User Interface.

**1.4 Overview**

This document provides a detailed description of the UVSim, including system functionalities. The functional requirements specify what the system should do, while non-functional requirements define system attributes such as performance and usability.

**2. Overall Description**

**2.1 Product Perspective**

UVSim is a standalone educational tool intended to run on personal computers and simulate a virtual machine environment for learning BasicML. It will provide a graphical interface for students to input programs, execute instructions, and view results.

**2.2 Product Functions**

* Load BasicML programs into memory.
* Execute BasicML instructions.
* Provide real-time feedback and error reporting.
* Support for arithmetic and control operations.
* I/O operations for interaction with the user.

**2.3 User Characteristics**

The primary users are computer science students and educators. Users should have a basic understanding of BasicML concepts but are not expected to have advanced technical skills.

**2.4 Constraints**

* The system must be able to run on Windows and macOS.
* It must handle BasicML instructions efficiently and provide accurate results.
* The memory size is fixed at 100 words.

**2.5 Assumptions and Dependencies**

* Users have basic familiarity with computer systems and programming concepts.
* The simulator will be used primarily for educational purposes.

**3. Functional Requirements**

**3.1 Memory Management**

**FR1**: The system shall initialize with a 100-word memory.

**FR2**: The system’s memory shall be able to store a signed 4-digit word at any given memory location.

**FR3**: The system shall load instructions from a txt file into memory.

**3.2 Instruction Handling**

**FR4**: The system shall interpret the first two digits of a BasicML word as the opcode.

**FR5**: The system shall interpret the last two digits of a BasicML word as the operand.

**FR6**: The system shall have the capability of reading a word from the keyboard into a specific location in memory.

**FR7**: The system shall have the capability of writing a word from a specific location in memory to the screen.

**FR8**: The system shall have the capability of loading a word from a specific location in memory into the accumulator.

**FR9**: The system shall have the capability of storing a word from the accumulator into a specific location in memory.

**FR10**: The system shall have the capability of adding a word from a specific location in memory to the word in the accumulator.

**FR11**: The system shall have the capability of subtracting a word from a specific location in memory from the word in the accumulator.

**FR12**: The system shall have the capability of dividing the word in the accumulator by a word from a specific location in memory.

**FR13**: The system shall have the capability of multiplying a word from a specific location in memory by the word in the accumulator.

**FR14**: The system shall be able to jump (branch) to a specified memory location when conditions are met.

**FR15**: The system shall stop the program when the specified opcode is read.

**FR16:** The system shall display the current contents of the accumulator on the GUI.

**FR17:** The system shall display the current contents of all memory location on the GUI.

**4. Non-Functional Requirements**

**4.1 Reliability**

**NFR1**: The system shall provide an error message in the event of an invalid instruction.

**4.2 Usability**

**NFR3**: The system shall be accessible through a graphical interface.

**NFR4**: The system’s graphical interface shall be designed with user-friendliness in mind, ensuring that all functionality is discoverable and accessible from the main screen of the graphical interface.

NFR5:The app must let users set a primary and secondary color through a config file or in-app setting, applying changes immediately or on restart without needing a reinstall. The default colors should be UVU dark green and white.