**Functional Requirements**

**UVSim**

File load button: allow the user to click the file destination through their Operating system.

File Save button: allow the user to save the file to their Operating system.

File SaveAs button: allow the user to save the file to their Operating system as a new file.

Run System button: allow the user to run the file selected from a button in the GUI.

Runs Assembly-Like code: Takes BasicML, machine-like code, and runs it like a basic computer might.

**GUI**

Memory display: displays the memory from the file

Accumulator display: displays what is stored in the accumulator

A step-through button: steps through the function and highlights each element of the

register/console/memory as it encounters it in the function

Error messages: messages can be displayed in the console portion with the error’s line number. The user can re-enter, or re-load whatever the case may be

File viewer: A viewport for the user to see the loaded file that is ready to be run

Change Colors Button: An button that allows users to change the colors of the system, for the primary reason of accessibility.

Main Window GUI: Contains the main file viewport, console, and toolbar.

Toolbar GUI: Contains main function buttons such as File-Load/Save/Run/Etc.

Program Visualization: The UVSim should offer a visual representation of the program execution, displaying the current instruction, memory, and accumulator.

**IO Operations**

Console input: Allow user to type input codes into the console through a clean GUI box part of the main screen

Console output: Display for the user the output of the machine during and after the code is run.

**Arithmetic Operations**

Adding: The system shall be able to execute the add instruction and store the result in the accumulator

Subtracting: The system shall be able to execute the subtract instruction and store the result in the accumulator

Multiplying: The system shall be able to execute the multiply instruction and store the result in the accumulator

Dividing: The system shall be able to execute the divide instruction and store the result in the accumulator

**Branch Operations**

Branching: The system shall be able to branch to a specific location in memory based on input code

**Non-Functional Requirements**

Visual appeal: The program is aesthetic but does not sacrifice too much efficiency or usability.

Efficient - simple and quick to use, doesn’t have unnecessary load time (longer than 2 seconds)

Intuitive - the user should not require many instructions to use it, but it does provide instructions that are easy to follow.