**Functional Requirements:**

1. UVSim should provide instructions to read a word from the keyboard and store it in memory.
2. UVSim should multiply a word from a specific location in memory by the word in the accumulator.
3. UVSim should provide an instruction to write a word from a specific location in memory to the screen.
4. UVSim should provide instructions to load a word from a specific location in memory into the accumulator.
5. UVSim should provide instructions to store a word from the accumulator in a specific location in memory.
6. UVSim will use a file picker pop-up to select the file they would like to run in the program and retrieve input numbers.
7. UVSim should provide instructions to add a word from a specific location in memory to the word in the accumulator.
8. UVSim should provide instructions to subtract a word from a specific location in memory from the word in the accumulator.
9. UVSim should keep track of the accumulator values, memory locations, and positive/negative signs in the code.
10. UVSim should provide instructions to divide the word stored in the accumulator by a word from a specific location in memory.
11. UVSim will have a pop-up input window with a text box for entry whenever the user is expected to provide BasicML code input.
12. UVSim should provide an instruction to branch to a specific location in memory normally and in specific cases of negative or zero values.
13. UVSim should manage the main memory of 100 words, ensuring proper allocation and deallocation of memory for instructions and data.
14. UVSim will allow the user to make edits to the file before execution.
15. UVSim should correctly read the user’s file and dissect BasicML code.
16. The file editor will not allow the user to exceed 100 lines of BasicML code.
17. UVSim should provide a way for the user to run, pause and quit the program.
18. UVSim should accept numbers without signs and provide an assumed positive sign.

**Non-Functional Requirements:**

1. The UVSim should have a user-friendly, colorblind friendly interface, allowing students to interact with the simulator easily; including separate information sections, a display of the accumulator value, and provide test files for the user.
2. The UVSim should execute BasicML programs efficiently, with minimal processing and response times on Windows and Apple operating systems.
3. The UVSim should be reliable, providing accurate results and gracefully handling errors or invalid instructions, with appropriate error messages and behavior.
4. The UVSim should allow the user to enter hexadecimal numbers to change the primary and secondary colors of the interface.
5. The UVSim text editor should be easy-to-use for writing BasicML code with line numbers for the user to keep track of where they are and how much space is left in writing the program.