Functional Requirements

1. The system shall allow the user to load a .txt file containing BasicML code from the local file system.
2. The system shall validate that the text file can be read by UVSim.
3. The system shall parse and validate each line of the loaded file to ensure it is a valid BasicML instruction or data word.
4. The system shall display an error message if a line in the file is malformed or invalid.
5. The system shall load the text file into memory.
6. The system shall allow the user to run the loaded BasicML program via a “Run” button.
7. The system shall iterate through memory and execute commands sequentially.
8. The system shall halt execution when a HALT (opcode 43) instruction is encountered.
9. The system shall execute READ (opcode 10) by displaying an input prompt via a graphical input dialog, capturing and storing user input as a signed four-digit integer, and placing the input in the correct memory location (XX when the instruction is 10XX).
10. The system shall execute WRITE (opcode 11) by displaying the value from memory location (XX when the instruction is 11XX) in a GUI output area or message box.
11. The system shall execute LOAD (opcode 20) by transferring data from memory location (XX when the instruction is 20XX) into the accumulator.
12. The system shall execute STORE (opcode 21) by transferring data from the accumulator into memory location (XX when the instruction is 21XX).
13. The system shall visually display the current state of the accumulator during execution.
14. The system shall execute ADD (opcode 30) by adding the value in memory location (XX when the instruction is 30XX) to the accumulator.
15. The system shall execute SUBTRACT (opcode 31) by subtracting the value in memory location (XX when the instruction is 31XX) from the accumulator.
16. The system shall execute MULTIPLY (opcode 33) by multiplying the accumulator by the value in memory location (XX when the instruction is 33XX).
17. The system shall execute DIVIDE (opcode 32) by dividing the accumulator by the value in memory location (XX when the instruction is 32XX).
18. The system shall execute BRANCH (opcode 40) by jumping to memory location (XX when the instruction is 40XX).
19. The system shall execute BRANCHNEG (opcode 41) by jumping to memory location (XX when the instruction is 41XX) if the accumulator is negative.
20. The system shall execute BRANCHZERO (opcode 42) by jumping to memory location (XX when the instruction is 42XX) if the accumulator is zero.
21. The system shall display all memory values in a table or grid format, updating dynamically as values change during execution.
22. The system shall highlight or indicate the currently executing instruction in the GUI (optional but recommended for user clarity).
23. The system shall allow the user to reset the simulator to its initial state via a “Reset” button.

Non-Functional Requirements

1. The system shall respond to user input (e.g., file selection, input dialogs) within 1 second.
2. The system shall be compatible with Windows operating systems and require .NET 6.0 or higher.
3. The system shall provide a user interface that is visually clean, with labels for all controls, and usable without prior training.
4. The system shall maintain separation of concerns in its design.
5. The system shall use object-oriented design principles.
6. The system shall be scalable to handle larger text files efficiently.