(Due to only three of our team members be present at the meeting, we each came up with our own and then did one three-way merge.)

**Final Software Requirements Specification**

1. The system shall allow the user to load a .txt file containing BasicML code.
2. The system shall validate the syntax of each instruction during file load.
3. The system shall store instructions in main memory starting at address 00.
4. The system shall set the program counter to 0 upon program load.
5. The system shall fetch, decode, and execute instructions sequentially.
6. The system shall support the READ instruction by prompting the user in the GUI.
7. The system shall store keyboard input into memory from the READ instruction.
8. The system shall support the WRITE instruction to display memory content in the GUI.
9. The system shall allow the LOAD instruction to store a memory value in the accumulator.
10. The system shall support STORE to write the accumulator value into memory.
11. The system shall support arithmetic instructions: ADD, SUBTRACT, MULTIPLY, DIVIDE.
12. The system shall handle HALT by stopping execution cleanly.
13. The system shall support conditional branching via BRANCHNEG and BRANCHZERO.
14. The system shall reset the memory and CPU state when the user clicks “Reset.”
15. The system shall raise an error message when an unknown opcode is encountered.

**Non-Functional Requirements:**

1. The system shall respond to user input within 1 second under normal operating conditions.
2. The system shall be operable on Windows, macOS, and Linux via a command-line interface.
3. The system shall include a user guide documenting all supported opcodes and usage examples.

**Individual SRS**

The system shall execute the LOAD opcode (10xx) to load a value from memory location xx into the accumulator.

 The system shall validate the syntax of each instruction during file load.

The system shall store instructions in main memory starting at address 00.

The system shall fetch, decode, and execute instructions sequentially.

The system shall execute the ADD opcode (20xx) to add the value at memory location xx to the accumulator.

 The system shall set the program counter to 0 upon program load.

The system shall execute the STORE opcode (30xx) to store the accumulator value into memory location xx.

 The system shall support the READ instruction by prompting the user in the GUI.

The system shall store keyboard input into memory from the READ instruction.

The system shall execute the BRANCH opcode (40xx) to jump to memory location xx for the next instruction.

The system shall execute the HALT opcode (4300) to terminate the program.

The system shall display the accumulator value after each arithmetic operation (e.g., ADD, SUBTRACT).

 The system shall support the WRITE instruction to display memory content in the GUI.

The system shall allow the LOAD instruction to store a memory value in the accumulator.

The system shall support STORE to write the accumulator value into memory.

The system shall support arithmetic instructions: ADD, SUBTRACT, MULTIPLY, DIVIDE.

The system shall handle HALT by stopping execution cleanly.

The system shall support conditional branching via BRANCHNEG and BRANCHZERO.

The system shall reset the memory and CPU state when the user clicks “Reset.”

The system shall raise an error message when an unknown opcode is encountered.

The system shall support a memory model of 100 locations (00–99) for storing instructions/data.

The system shall initialize all memory locations to +0000 on startup.

The system shall allow users to input a complete program before execution (batch mode).

**Non-Functional Requirements:**

The system shall respond to user input within 1 second under normal operating conditions.

The system’s codebase shall include inline documentation to facilitate debugging.

The system's source code shall require no platform-specific modifications during compilation.

The system shall be operable on Windows, macOS, and Linux via a command-line interface.

The system shall validate all user inputs to prevent dangers like buffer overflows or command injection.

The system shall include a user guide documenting all supported opcodes and usage examples.

The system shall have an uptime of 99.9%.

The system shall be able to support up to 100 simultaneous users with full performance.

The system shall provide clear, concise error messages to the user when input correction is needed.