## 1. Functional Requirements

*1.1 User Interface (UI) Requirements*

1. The GUI shall have a text editor for entering BasicML instructions from a file.
2. The GUI shall have a text editor for entering BasicML instructions manually, one at a time.
3. The GUI shall display the memory contents of 100 memory locations, CPU state, PC, and Accumulator dynamically.
4. The GUI shall provide buttons to Run, Step, Halt or Reset the program.

*1.2 Memory Management Requirements*

1. The program shall allocate a 100 word memory space for instructions.
2. Memory instructions shall be represented as four-digit words (e.g. +1234, -1234)
3. Instructions shall be validated before they are stored in the memory.
4. The memory shall be color coded to show any changes from the last memory state.

*1.3 CPU Functionality Requirements*

1. The CPU shall support I/O operations: READ (10), WRITE (11) and also support
2. LOAD (20), and STORE (21) operations.
3. The CPU shall support arithmetic operations: ADD (30), SUBTRACT (31), DIVIDE (32), MULTIPLY (33).
4. The CPU shall support Control Operations: BRANCH (40), BRANCHNEG (41), BRANCHZERO (42), and HALT (43).

*1.4 Error Handling Requirements*

1. The program shall handle invalid file paths by displaying an error message in the GUI
2. The Program shall handle invalid instructions by displaying an error message in the GUI.
3. The Program shall handle invalid I/O operations by displaying an error message in the GUI.
4. The program shall check for memory overflows before executing the next instruction.
5. Invalid instructions shall not affect the CPU state, or memory contents.

## 2. Non-Functional Requirements

***2.1 Usability***

1. The UI shall provide descriptions on how to use instructions.

***2.2 Performance***

1. The program shall finish executing at most 100 instructions per execution cycle in less than 1 second with real-time updates through the UI.

***2.3 Maintainability***

1. The project shall have documentation for all functions and classes and ensure that each module is loosely coupled.

# ~~Functional Requirements~~

1. ~~Allow users to input a basicML program via a text editor / file uploader~~
2. ~~Allow for the program to be inputted into memory starting from location 00~~
3. ~~Will execute BasicML instructions sequentially, unless control instructions modifies execution flow~~
4. ~~Will support reading a word from the keyboard into memory via the READ command 10XX~~
5. ~~Will support writing a word from the memory to the screen via the WRITE command 11xx~~
6. ~~Will support loading a word from memory into the accumulator via LOAD 20XX~~
7. ~~Will support storing a word from the accumulator into memory via STORE 21XX~~
8. ~~Will support the addition instruction via ADD 30XX~~
9. ~~Will support the subtraction instruction via SUBTRACT 31XX~~
10. ~~Will support the multiplication instruction via MULTIPLY 33XX~~
11. ~~Will support the division instruction via DIVIDE 32XX~~
12. ~~Will support unconditional branching to a new memory location via BRANCH 40XX~~
13. ~~Will support conditional branching if the accumulator is negative via 41XX~~
14. ~~Will support conditional branching if the accumulator is zero via 42XX~~
15. ~~Will support halting successfully when the HALT command is executed via 43XX~~
16. ~~The system will display memory contents in a structured format in the GUI~~
17. ~~The system will allow for saving/loading of BasicML programs~~
18. ~~The system will support validation of all BasicML syntax before execution~~

# ~~Non-Functional Requirements~~

1. ~~The system will have a responsive and intuitive GUI to allow users to interact with the program~~
2. ~~Will perform instructions within 10 ms per operation~~
3. ~~Will provide error messages for invalid instructions or invalid memory writes~~