Functional Requirements

1. The system shall support a 100-word memory for BasicML words.
2. The system shall read a word from the keyboard into a specific location in memory if the first two digits of the BasicML word equals 10.
3. The system shall write a word from a specific location in memory to the screen if the first two digits of the BasicML word equals 11.
4. The system shall load a word from a specific location in memory into the accumulator if the first two digits of the BasicML word equals 20.
5. The system shall store a word from the accumulator in a specific location in memory if the first two digits of the BasicML equals 21.
6. The system shall add a word from a specific location in memory to the word in the accumulator, leaving the result in the accumulator if the first two digits of the BasicML word equals 30.
7. The system shall subtract a word from a specific location in memory from the word in the accumulator, leaving the result in the accumulator if the first two digits of the BasicML equals 31.
8. The system shall divide the word in the accumulator by a word from a specific location in memory, leaving the result in the accumulator if the first two digits of the BasicML equals 32.
9. The system shall multiply a word from a specific location in memory to the word in the accumulator, leaving the result in the accumulator if the first two digits of the BasicML equals 33.
10. The system shall branch to a specific location in memory if the first two digits of the BasicML equals 40.
11. The system shall branch to a specific location in memory if the accumulator is negative and the first two digits of the BasicML equals 41.
12. The system shall branch to a specific location in memory if the accumulator is zero and the first two digits of the BasicML equals 42.
13. The system shall stop the program if the first two digits of the BasicML equals 43.
14. The program must read the words in memory line-by-line unless a branch operation performs a jump.

15. The system shall halt program execution when a halt instruction is encountered, indicating the end of program execution.
16. During input operations, the system shall provide clear instructions to the user, guiding them on the format and type of input expected.
17. The system shall allow users to customize the color scheme of the application interface.
18. The system shall allow users to load program files into the GUI for inspection and editing before execution.
19. The system shall support loading program files from any user-specified directory.
20. The system shall allow users to save edited program files to a user-chosen directory.
21. The application shall support data files containing up to 250 lines, with each line corresponding to a memory register ranging from 000 to 249.
22. The application shall use three-digit memory addresses to reference lines within the expanded address space.
23. The application shall enforce the limitation of not allowing more than 250 lines in any loaded or edited file.
24. The application shall ensure that any command referencing a line number outside the range of 000 to 249 is considered invalid.
25. The application shall handle six-digit math operations correctly, including overflow handling, for the new word size.
26. The application shall append a zero to the beginning of each functional code to represent operations in the new six-digit format (e.g., 010 instead of 10 for a READ command).
27. The application shall support both old and new file formats, allowing users to load, edit, and execute files with either four-digit or six-digit words.
28. The application shall not allow mixing and matching of four-digit and six-digit words within an individual file, ensuring that each file is consistent with one format.

Non-functional Requirements

1. The system's user interface shall be designed following principles of minimalism, with intuitive controls and concise instructions to facilitate ease of use.

2. The system's reliability shall be demonstrated by achieving at least 99.9% uptime during continuous operation, ensuring consistent and accurate execution of programs.
3. The system's response time to user interactions shall be under 1 second, ensuring efficient execution of instructions and a seamless user experience.
4. The program shall take no more than 20 minutes to be installed.
5. The application's user interface shall provide a clear distinction between old and new file formats, allowing users to differentiate between them at load or runtime.
6. The application shall be efficient in processing and executing commands, providing responsive performance even when handling large files with the expanded address space.
7. The application shall maintain data integrity and reliability when handling multiple open files, ensuring that changes made to one file do not affect the integrity of others.