**Software Specification Requirement**

**User Friendly description and definition.**

This program’s intended audience is any student learning the BasicML language. It simulates a BasicML IDE. The program is limited to programs made by the user of 100 instructions, keeping it small and simple for learning purposes. The student’s user experience is made easier with a header which describes the format of instructions to be loaded. The simulator will also ask the student for new input if the student tries to input an incorrect instruction, or one not handled by the system. When the student is done entering instructions for their own program they will enter “-99999” to stop programming and the simulator will compile and execute the student’s program.

**Description of simulator structure.**

The simulator implements two software design patterns, the façade pattern, and the Model View Controller or MVC pattern. Starting with the model, in this case the model is simply an array working as our memory which holds either BasicML instructions or 4-digit integers. The only way of accessing this model object is through a controller. Controller is a base type which can be implemented in many ways if certain functions are implemented. In this simulator we have a StoreController and a RunController. Each of which performs its own type of operations on the model. Each controller has a model that it can work on. Which model that is to be worked on is instantiated during the creation of the controller or can otherwise be set later. The view in this case acts as a façade. It takes two controllers of any type. The ViewClient takes this view (façade) and uses it to access the functionality of the more complex functionality of the controllers. This gives the view the flexibility to ask different controllers to act on the model in different ways. This keeps different types of operations separated and decoupled from one another. The view client does not even need to know what types of controllers it is getting. The ViewClient asks for instructions to be executed o the model or gets data from the model through the controllers.

**Functional Requirements**

FR1  The system should provide a brief header to introduce the basics of the system to the    user.

FR2    The user must be able to input BasicML instructions which consist of 4 digits preceded by either a + or - symbol.

FR3    The system must implement a memory space with 100 locations capable of holding 4 digit integers.

FR4    The system must transition from programming mode to compilation and execution with input of a specific value.

FR5    The system must distinguish between word instruction and operand (memory location).

FR6    The system must write any word from memory to the screen.

FR7    The system must load a word from memory into the accumulator.

FR8    The system must store a word from accumulator into memory.

FR9    The system must add a word from accumulator and a word in memory saving result in accumulator.

FR10   The system must branch to a new memory location depending on the value in the accumulator register.

FR11   The user must be able to use a keyboard to input data into and operate the system.

FR12   The user should be able to perform simple math operations on the value in the accumulator with the value in a specific memory location.

FR13   The system should output all the values in memory once the instructions are read.

FR14   The system should be able to prompt the user for an integer to read into memory.

FR15   The system must output the state of the Accumulator after execution is complete.