Team Meeting Log

|  |  |  |  |
| --- | --- | --- | --- |
| Team Name | Meeting Log # | Date | Duration: 1 hr 30 mins |
| Team Members | Name | Contribution | Signature |
| Tyler Merritt | Backlogs, Meeting Log | Tyler Merritt |
| Bryan Edman |  |  |
| Julia Hoffmann | UML Class Diagram | Julia Hoffmann |
|  |  |  |
|  |  |  |
| Topics Discussed | | Requirements needed to be completed for Milestone 2  What was left to be done with the MVC and Facade software design  Discussed how to define user friendly  defining user friendly in SRS | |
| Obstacles Encountered | | Missing feedback from the teacher about Milestone 1 | |
| Finished Items | | Finished condensed FR document with the group  defining what user-friendly means in our program  ReadMe | |
| Unfinished Items | | last cleanup of refactored code  Wireframe | |
| Notes | | ask (email) about SRS number 2 in teams doc | |

Final Condensed FR Doc also potential 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.

**Functional and Non-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.