Mohawk Engineering Project

Requirements List

Project #36 - PLC Simulator for Students Training

John Van Loon

Mohawk College

Group #19

Jasper Hanlon

Jesus Perez

Stephen Kim

## Requirements Analysis - FURPS

### Functionality

* Create an API that connects to PLC simulator
  + API Requirements:
    - Must detect whether PLC simulator is running or stopped
    - Must be able to switch PLC simulator between run and stop mode
    - Must send user input to PLC sim and receive output from PLC sim back to user
  + PLC simulator uses ladder logic from Siemens TIA portal
    - Can assume ladder logic is already created
    - Can assume TIA portal logic is named something constant
* Proof of concept game that uses ~16 inputs and ~16 outputs using our API
  + Basic simulation/control of a cylinder (connect, control, pass digital I/O)
  + Basic simulation/control of a tank level control (add analog I/O)
  + Perform basic simulation/control of a marble sorter (add simple graphics and interaction)

### Usability

* User must be able to reliably start the game
* Game/API installation must connect easily to PLC sim
* Proof of concept game must be aesthetic
* API documentation must be accurate and complete

### Reliability

* API must send correct user input and receive correct output from PLC sim
* Game should properly display and work as intended
* Friendly error messages should show in case of crashes
* Logging of what users do is a nice to have

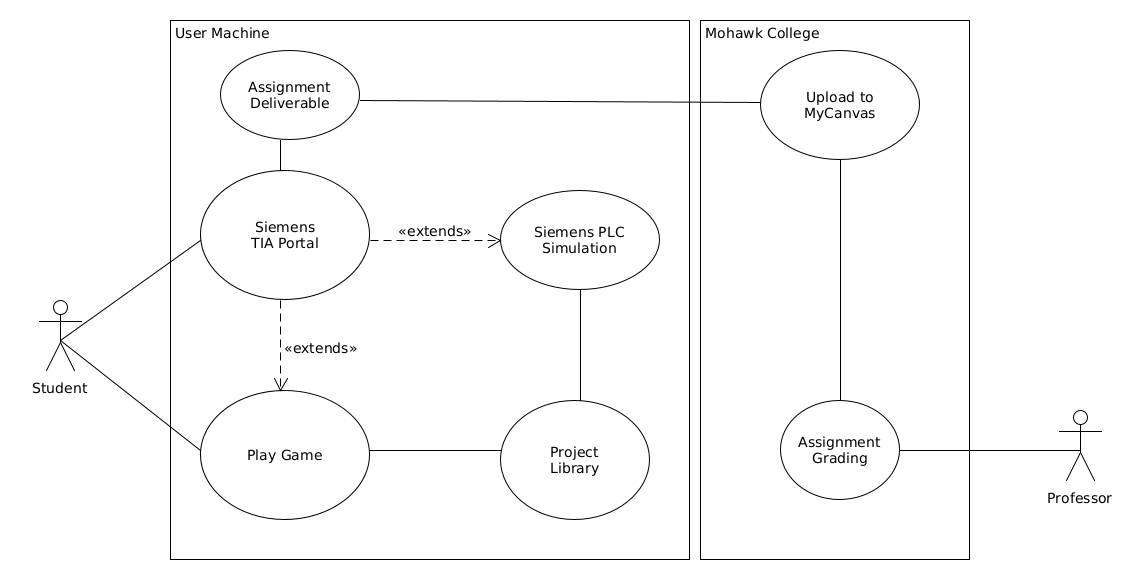
### Performance

* API must receive input/output in real time from PLC sim
* Game must then display API info within a reasonable time

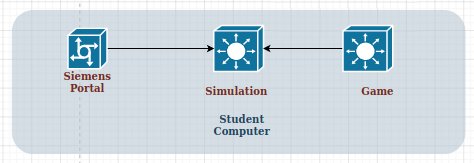
### Supportability

* Proper documentation necessary for API for proper installability and potential new games
* Logging capabilities and debug mode
* User activity monitoring is a nice to have

## Use Case Diagrams



## System Architecture/Communication Diagram



## Risk Analysis

Include a list of Risks to project success and the mitigation strategy if these arise. It is important here to ensure that you have a back-up plan if one of the requirements runs into difficulty. For example, if your project must get access to a database residing on another system but cannot get access for some technical reason, a backup plan may be to build a simple substitute database to allow the other aspects of the project to continue.

**Potential Risks**

1. The Simulator API is buggy or broken for the C++ interface, C# interface, or both
2. The Simulator API is consistently unreliable or unstable
3. The Simulator API cannot be greatly simplified
4. The Simulator API library takes longer than expected to produce, and there is not enough time to develop a game or POC
5. There is difficulty running and using the TIA Portal while the game is running. For example, it may not update adequately quickly, or the simulator must be refreshed manually

**Mitigation Strategy**

1. Use a lower-level interface and avoid using the provided language-specific API libraries directly. Eg. communicate with the simulator directly over TCP-IP
2. Find alternative ways of achieving the same functionality and apply “workarounds” by avoiding code paths or features that invoke simulator instability
3. Shift emphasis on project to documentation. Reduce scope of library, and provide a set of “starter templates” for using the API with a set of minimal functionality
4. Work on the game/POC and the simulator library in parallel. Design the POC based on the planned design of the library before it is complete.
5. Look for workarounds in the API. In the worst case scenario, include instructions in-game on how to use the simulator/portal most effectively while the game is running.