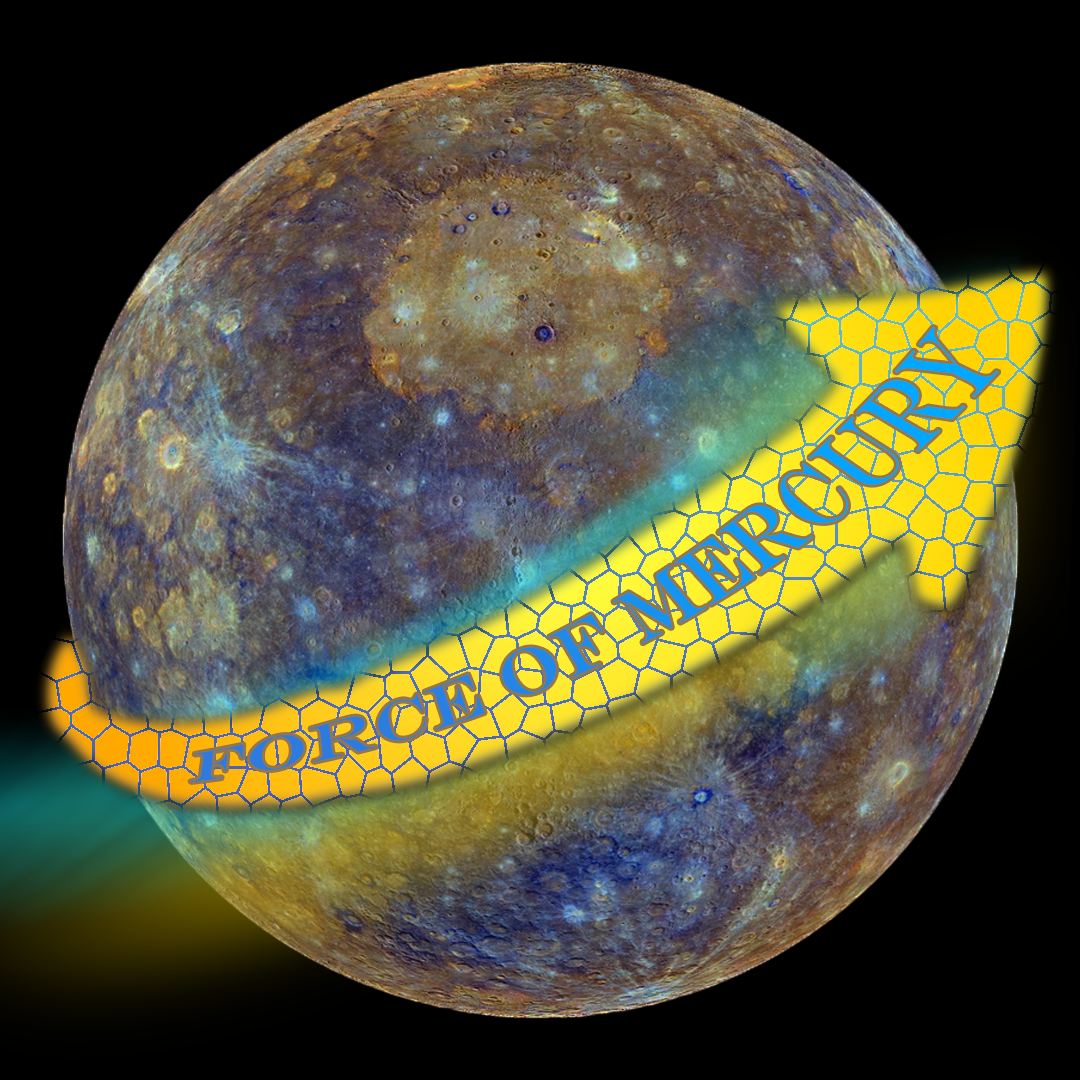
**SolarProx**

Design Document #2 Rough Draft

Version 1.0

March 22, 2021

Force Of Mercury

Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description | Date Completed |
| 1.0 | Trevor Ryan  Jacob Sanford  Austin Carper  Kurt Neimayer | This is the original draft of the Design Document #2 for the SolarProx project | 03/21/2021 |

**Table of Contents**

[**1.0 Introduction**](#_heading=h.b8nu5sd4vsq6) **4**

[1.1 Overview](#_heading=h.shdozn3yuc3c) 4

[1.2 Outlined Modules](#_heading=h.shdozn3yuc3c) 4

[1.3 Glossary of Terms](#_heading=h.567jy1e6m24d) 4

[**2.0 HTML Site**](#_heading=h.tkt7q2h6grlh) **5**

[2.1 Classification](#_heading=h.amkdedctq849) 5

[2.2 Purpose](#_heading=h.amkdedctq849) 5

[2.3 Constraints](#_heading=h.amkdedctq849) 5

[2.4 Interactions](#_heading=h.amkdedctq849) 5

[2.5 Processing](#_heading=h.amkdedctq849) 6

[**3.0 Active Directory Integration**](#_heading=h.3bscrtvb3wr2) **7**

[3.1 Classification](#_heading=h.3lmqgfg0r09) 7

[3.2 Purpose](#_heading=h.3fxwu3eonm7b) 7

[3.3 Constraints](#_heading=h.3fxwu3eonm7b) 7

[3.4 Interactions](#_heading=h.3fxwu3eonm7b) 7

[3.5 Processing](#_heading=h.3fxwu3eonm7b) 8

[3.6 Data](#_heading=h.3fxwu3eonm7b) 8

[**4.0 Proxmox**](#_heading=h.1tc4tf54rwp3) **9**

[4.1 Classification](#_heading=h.s3ozkf90nh9) 9

[4.2 Purpose](#_heading=h.s3ozkf90nh9) 9

[4.3 Constraints](#_heading=h.s3ozkf90nh9) 9

[4.4 Interactions](#_heading=h.s3ozkf90nh9) 9

[4.5 Processing](#_heading=h.s3ozkf90nh9) 9

[**5.0 Point Tracker**](#_heading=h.g6lwhe6bxjym) **10**

[5.1 Classification](#_heading=h.uocxuf47yte5) 10

[5.2 Purpose](#_heading=h.uocxuf47yte5) 10

[5.3 Constraints](#_heading=h.uocxuf47yte5) 10

[5.4 Interactions](#_heading=h.uocxuf47yte5) 10

[5.5 Processing](#_heading=h.6cc9s9dkbnu4) 11

[5.6 Data](#_heading=h.uocxuf47yte5) 11

# 1.0 Introduction

## 1.1 Overview

The purpose of this document is to show a detailed design of the SolarProx application and a description of what the 4 major modules of the application do. These descriptions of the modules will include a detailed explanation of each module’s classification, specific purpose, constraints, interactions, processing, and data.

## 1.2 Outlined Modules

**HTML Site:** User Interface

**Active Directory:** Connection for authentication

**Proxmox:** Connection for network interactions

**Point System:** File interaction for storing user scores

## 1.3 Glossary of Terms

**Proxmox:** Server connection that connects to the REST API within Proxmox to access machine status and the ability to revert machine snapshots.

**API:** Application Programming Interface.

**REST API:** An API that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.

**JSON:** Stands for JavaScript Object Notation. It is a lightweight data-interchange format. Easy for humans to read and write while being easy for machines to parse and generate.

**Active Directory:** Database and set of services that connect users with the network resources they need to get their work done. Contains information about your environment, including what users and computers there are and who’s allowed to do what.

# 2.0 HTML Site

## 2.1 Classification

The classification of the HTML Site is a Graphical User Interface (GUI)

## 2.2 Purpose

The purpose of the HTML Site is to provide an interface for users to interact with while making calls to all other modules. All user interactions will be handled in the HTML and forwarded to the server.

## 2.3 Constraints

Must limit credential reference of data to keep it secure and avoid attacks. This can be done by using a session ID in the finished project.

## 2.4 Interactions

* Login
* Add/Edit Machines
* Add/Edit Groups
* Edit Users
* Save/Restore Snapshots
* Submit Flag Values
* Logout

## 2.5 Processing

## 

## 

# 3.0 Active Directory Integration

## 3.1 Classification

The classification for Active Directory Integration is a database connection.

## 3.2 Purpose

The purpose of this module is to set permissions for users based on what they use for their login credentials. If they are in the administrator group they will be logged in as an administrator. If they are in the user group they will be logged in as a user.

## 3.3 Constraints

Must find a secure way to send credentials.

## 3.4 Interactions

* Authentication
* Get and modify user info
* Get and modify group info
* Get and modify machine info

## 3.5 Processing

## 

## 3.6 Data

* Authentication (username, password)
* Users Info(username, total points)
* Machine Info(ip, name, notes, flag key, flag value)
* Group Info (group name, users, and machines)

# 4.0 Proxmox

## 4.1 Classification

The classification of Proxmox would be a REST api Handler.

## 4.2 Purpose

The purpose of this module is to establish a connection with the proxmox server. This module will also execute commands on the server to create snapshots and revert boxes.

## 4.3 Constraints

Need to securely send credentials over the network.

Needs to execute commands on the server in a timely manner.

## 4.4 Interactions

* Establish Connection
* Revert back to snapshot
* Start machine
* Stop machine

## 4.5 Processing

public class ProxmoxHandler(){

private string boxID;

public ProxmoxHandler(string id){

boxID = id;

}

public void startBox(){

Process p = new ProcessBuilder("./start.sh", boxID).start();

}

public void stopBox(){

Process p = new ProcessBuilder("./stop.sh", boxID).start();

}

public void rollbackBox(String snapshotName){

Process p = new ProcessBuilder("./rollback.sh", boxID, snapshotName).start();

}

}

# 5.0 Point Tracker

## 5.1 Classification

The classification of the point tracker would be a function.

## 5.2 Purpose

The purpose of the Active directory module is used to manage the tracking of individual points on machines. This is used to allow verification of what users have interacted with

## 5.3 Constraints

* Requires read and write permissions within the local file directory to read and write JSON files.
* Is dependent on the JSON Simple API to read and write JSON files.
* Can only take machines names (Strings), and user ids.

## 5.4 Interactions

* Write users into JSON
* Write machine into JSON
* Read if user is in a machine JSON
* Read if a machine is in the JSON

## 

## 5.5 Processing

function isMachineInJson(Machine){

readJson(file)

for(every line section head){

check if machine is in file.{

return true

}

}

return false

}

function isUserInMachine (UserContainer, MachineContainer){

readJson(file)

get machineSection

for(every user in the machine){

check if user if user is the ID{

return true

}

}

return false

}

## 5.6 Data

Data will be stored in a similar format to this:

## 

## 