**LFO Server thread:**

**Singleton pattern creation member functions and variables:**

public:

/// @brief Public static method getInstance(). This function is

// responsible for object creation.

static ServerHandler& GetInstance();

private:

/// @brief Define constructor in the private section to make this class as singleton.

ServerHandler() = default;

/// @brief Define destructor in private section, so no one can delete the instance of this class.

~ServerHandler() = default;

/// @brief Define copy constructor in the private section, so that no one can voilate the singleton policy of this class

ServerHandler(const ServerHandler& obj){}

/// @brief Define assignment operator in the private section, so that no one can voilate the singleton policy of this class

void operator=(const ServerHandler& obj){}

**LFO member functions:**

public:

/// @brief create a REST server at port as per config.json file

/// @return true if k3s is running, false otherwise

bool Init();

/// @brief start LFO server

void StartLfOrchestratorServer();

/// @brief start pod

bool ApplyPod();

/// @brief get list of pods

bool GetPodName();

/// @brief check if pod name matches

bool CheckPodName(string& pod\_name);

private:

/// @brief delete pod

bool DeletePod();

/// @brief starve pod by reducing CPU and Memory resources

bool StarvePod();

/// @brief describe specific pod

bool PodUsage(string& pod\_name);

**Resource monitoring thread:**

**Singleton pattern creation member functions and variables:**

public:

/// @brief Public static method getInstance(). This function is responsible for object creation

static ResourceMonitorHandler& GetInstance();

private:

/// @brief Define constructor in the private section to make this class as singleton.

ResourceMonitorHandler() = default;

/// @brief Define destructor in private section, so no one can delete the instance of this class.

~ResourceMonitorHandler() = default;

/// @brief Define copy constructor in the private section, so that no one can voilate the singleton policy of this class

ResourceMonitorHandler(const ResourceMonitorHandler& obj){}

/// @brief Define assignment operator in the private section, so that no one can voilate the singleton policy of this class

void operator=(const ResourceMonitorHandler& obj){}

**Resource monitoring member functions:**

public:

/// @brief initialize LFO to check for pods status.

void Init();

private:

/// @brief Function to Calculate CPU, Memory utilization of host machine in a time interval

/// @param *resource\_utilization* - address of vector of pair to store CPU, Memory utilization,

/// @param *time\_delay* in seconds

void GetResourceUtilization(vector<pair<float, float>>& *resource\_utilization*, int *time\_delay*);

/// @brief Function to calculate average CPU, Memory utilization of the host machine

void AverageResourceUtilization();