**Low Level Design**

**Overview of Modules**

**[ TODO: Create a UML diagram showing the relationship of all our modules with an explanation** ]

**Modules**

**Container Runtime Module**

*Purpose*

This module is responsible for handling service requests that are sent by consumers. The Container Runtime isolates running services by creating a specific container instance for the service based on the consumer’s security levels that it receives in the request. This module also handles shutting down its container instances once the consumer has disconnected from the service.

*Responsibilities*

* Handles the REST API endpoint for service requests from consumers.
  + ~~GET request~~ POST request
  + endpoint: {base\_path}/api/container/runtime/{service}/
  + in body of request:{SELinux labels}
  + The specific *service* and the consumer’s *SELinux labels* are sent with the GET request as query string parameters
* Creates a container instance for the requested service
  + Each service will have its own Docker Image with the needed dependencies, configurations, and files for running this service
  + Enforces SELinux labels on any processes/services running on the container
  + Starts the RESTful service on the container instance
* Shut downs running container instances
  + The container instance stops a running service when the consumer has disconnected from the service
  + Destroys the container instance when notified by the container of the service being stopped
* Logs all transactions

**Postgres Service Module**

*Purpose*

This module is responsible for handling REST requests from the consumer, communicating these requests to the Data Storage, and returning the response to the consumer.

*Responsibilities*

* Establishes connection with the consumer to the PostgreSQL database
  + Connect to the database using credentials that are based on the consumer’s SELinux security context
  + Send response to consumer letting them know whether connection has been successfully established
* Handles read requests to the database from the consumer
  + GET request
  + endpoint: {base\_path}/api/read/{table}
  + Returns REST response of the data (if successful) or error message if not
* Handles write requests to the database from the consumer
  + PUT request
  + endpoint: {base\_path}/api/write/{table}/{data}
  + Returns REST response of success or error message if not successful
* Handles view aggregate report requests to the database from the consumer
  + GET request
  + endpoint: {base\_path}/api/view/reports/{table}/{filter}
  + Returns REST response of the aggregated report (if successful) or error message if not
* Shuts down service when notified consumer disconnects
  + Disconnects from the database
* Logs all transactions

**Logging Util Module**

*Purpose*

This utility module is responsible for logging all transactions that occur in other modules in this system and sending the logs to the Log Aggregrator.

*Responsibilities*

* Each module contains its own Logging Util module with the static codes and descriptions for the specific logging transactions it handles.
* Sends all log transactions to the Log Aggregrator
  + Logs are sent to the running Rsyslog software utility
* Notifies an operator when a log cannot be sent
  + Send an alert when the sending of the log to Rsyslog fails