JJ Boolean
Joshua Calder V00849953
Julien Godding V00877834

Seng330 Assignment 3 Design Document

This assignment was completed using the Spring MVC framework and Gradle as the build tool. Our code solution is based off of Spring's *gs-serving-web-content* starter repo. It uses certain functions and classes from our own assignment 2 solutions as well as Professor Ernst's.

Spring MVC Processing Sequence

The application implements an HTML templating language, *JSP*, to render views to the client after the Dispatcher Servlet receives an HTTP request and has successfully invoked a handler. This handler then calls the specified controller, which communicates with their corresponding services (otherwise known as models) to access and update the device's behaviour and attributes. Since Spring provides the necessary handling and mapping of these requests, we had to implement the remaining Controller, Service (Model) and Data Access classes. Link 1 in the appendix shows a more in-depth look at the Spring MVC processing sequence.

Running the Application

To run the IOT application, cd into the `complete` directory and type `gradle bootRun`. After 'bootRun' is executed, the application will initialize and then begin serving on port 8080 (default) of localhost. If the server starts on another port, there will be a message displayed in the command line. After the application has successfully started, navigate to **`localhost:8080`** (or 127.0.0.1:8080) on the web browser to begin using the application. Use the following login info to test the application.

Default Login Information

Username: user Password: password

Test Coverage

Since we were required to complete 17 acceptance tests for assignment 3, we left out specific application functionality that has not yet been covered by extensive tests. The functionality that will be completed for assignment 4 includes persistent data storage, multiple user authentication, and random event triggering (ie. camera detecting random movement). For assignment 3, the device functionality has been implemented with matching acceptance tests. Running `gradle test` will show the testing results.

UML Class Diagram

Please see the JJBoolean_UML_deliverable.png in the repository to see a full-size of the UML Class diagram divided into Model-View-Controller sections.

Appendix

[1]https://terasolunaorg.github.io/guideline/1.0.1.RELEASE/en/Overview/SpringMVCOverview.html