Project 5

ECE 528: Application Software Design

Alan Palayil

Professor: Won-Jae Yi

Acknowledgement: I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

Electronic Signature: Alan Palayil A20447935 (Due Date: 4/9/2023)

Table of Contents

Abstract:

This project aims to improve the server backend of an IoT hub by introducing group management that allows users to control multiple smart plugs. The system will implement five user stories, which include the ability to create and remove groups, query the state of a group, obtain the states of all groups, and control a group by switching on/off or toggling all the plugs in a group. The implementation will follow the red-green cycle by adding unit tests and utilizing MQTT and HTTP communications to test the classes. The design and implementation of the classes are left to the discretion of the developer. Overall, the goal of this project is to enhance the functionality and usability of the IoT hub by introducing group management, allowing for more efficient and streamlined control of smart plugs.

Introduction:

In this project, we aim to enhance the functionality of our IoT hub server backend by introducing group management capabilities. With this new feature, users can create groups of smart plugs and control them collectively, simplifying the process of managing multiple devices. This project involves implementing the necessary classes and unit tests to ensure that the new functionality is robust and meets the user's requirements. While we will discuss possible class designs and implementations in lectures, the final design and implementation decisions are left to the individual's discretion. Additionally, MQTT and HTTP communication may be utilized to test some of the classes, and GradeP5.java provides useful code for this purpose.

Background:

The goal of this project is to enhance the server backend for an IoT hub by introducing group management functionality. With this feature, users can create, remove, query the state, and control multiple smart plugs as a single unit, which will provide greater convenience and efficiency in managing their IoT devices. The requirements for the group management functionality are described through a set of user stories:

- Create a Group: The end-user desires to create a group of smart plugs named "groupName" through a POST request sent to /api/groups/groupName. This is to enable managing multiple plugs as a single unit. The POST request should contain a JSON array of plug names to be included in the group. If a group with the same name already exists, all its members will be replaced. It's important to note that a single plug can be assigned to multiple groups.
- Remove a Group: I want to be able to remove a group of plugs named "groupName" by sending a DELETE request to /api/groups/groupName as an end-user. This functionality is important to me as it allows me to easily delete a group through a web application.
- State of a Group: I want to be able to check the status of a group named "groupName" and its member plugs by sending a GET request to /api/groups/groupName as an end-user. This will allow me to view the states of the member plugs in a web application. The response should consist of a JSON object that includes the "name" key for the group name and a "members" key for a JSON array of objects, each representing the state of a member plug.
- States of All Groups: As an end-user, I would like to be able to obtain the states of all member plugs for every group in a single request by sending a GET request to /api/groups. This will allow me to conveniently view all the relevant information in a single web application. The server should respond with a JSON array of objects, with each object representing the state of a group.
- Control a Group: I want the ability to control all the smart plugs within a group named "groupName" simultaneously. To achieve this, I need to be able to send a GET request to /api/groups/groupName with a query string, just like the one used to control a single plug. This will enable me to turn on/off or toggle the power state of all the plugs in the group at once, providing a more convenient way to manage my IoT devices.

The topics and messages for each of these user stories are defined based on the plug name and configuration string. The project also includes testing procedures implemented in ece448.grading.GradeP5 to ensure that all user stories are covered.

Results and Discussion:

The project required me to create the files GroupsModel and GroupsResource. During the implementation of the project, I had to modify the following classes: iot_hub>Main.java. The GroupsModel class has the following methods:

• getGroups: returns a list of names of all locally-stored groups.

• getGroupMembers: returns a list of names of the members of a specific locally-stored group, which is provided as an argument.

• setGroupMembers: sets the names of the members of a specific locally-stored group to the names provided by the calling method as an argument.

• removeGroup: removes the specified group from the locally-stored collection of groups.

The GroupsResource class has the following methods:

• getGroups: on a JSON request to "/api/groups", returns properly-formatted information about all the existing groups and their corresponding members that are locally-stored in the GroupsModel.

• getGroup: on a JSON request to "/api/group/<group name>":

If no action is specified, returns the information about the group, including individual members and their states/powers, contained in the locally-stored groups collection in GroupsModel.

• createGroup: on a JSON POST request to "/api/group/<group name>", creates a group in GroupsModel and sets its members to those provided in the request.

• removeGroup: on a JSON DELETE request to "/api/group/<group name>", removes the provided group from the locally-stored groups in GroupsModel.

• makeGroup: a helper method for getGroup – given a group name, it returns the information for all the plugs in that group, including their state and power, as obtained from the MqttController.

The following are the screenshots of the results I got during the project.

Screenshots of the Tests results:

The screenshot of the acceptance testing ‘Gradle’ results is added which was the base testing criteria for project 5 with 10 Local Testing passed.

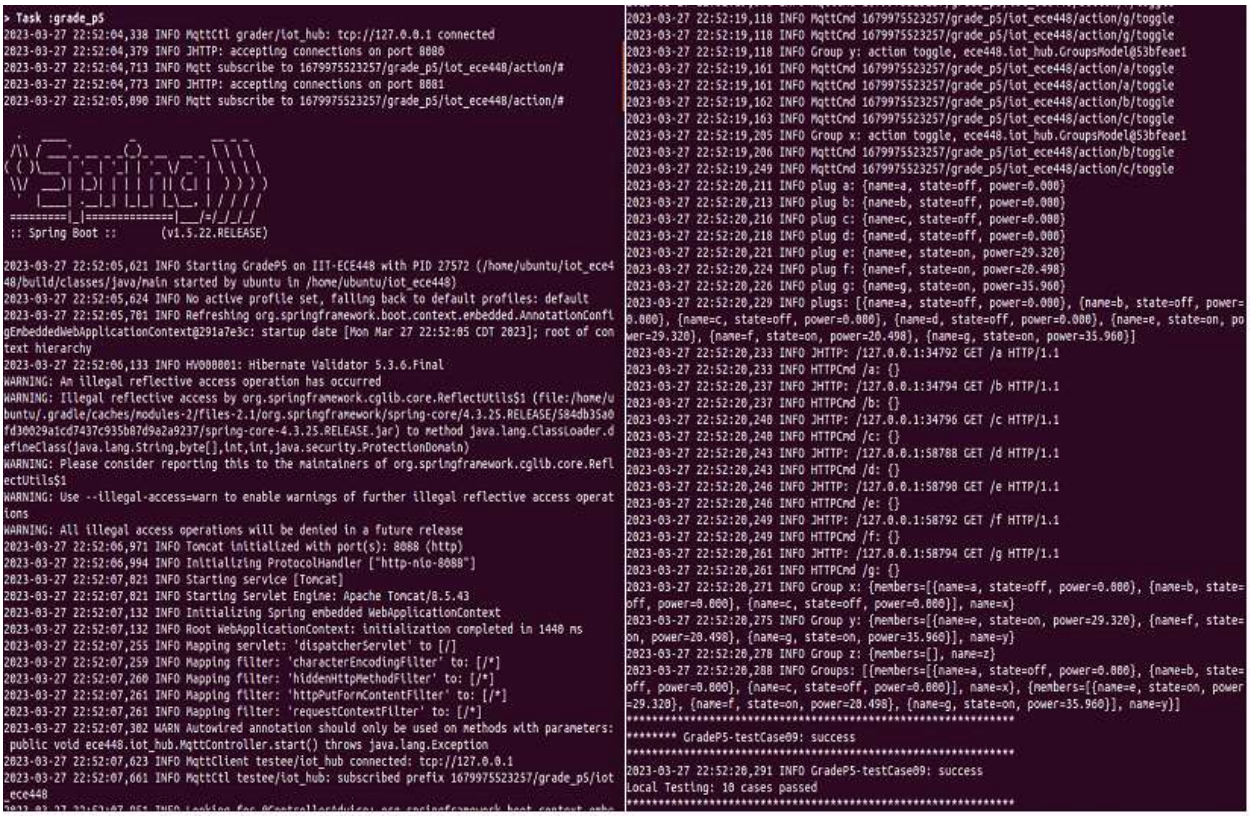


Figure 1: GradleP5- Test cases Results

Screenshots of the HTML results:

To completely utilize the GroupsModel and GroupsResource, I referred to grade_p5 and checked each of the elements within the IoT_Sim program to work over the unit testing. The test cases were designed to test the utilization of the program.

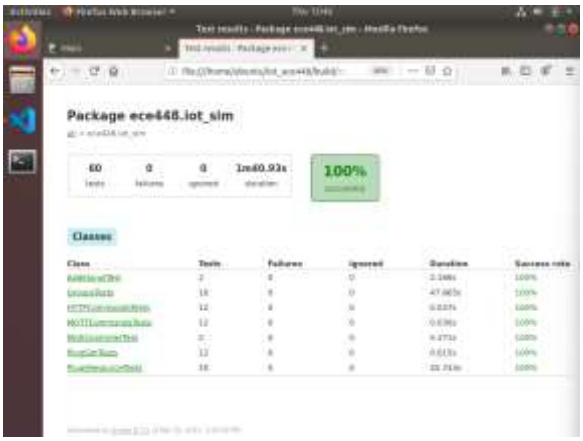


Figure 2: IoT-Sim Test Cases

Figure 3: Utilization of ece448.iot_sim

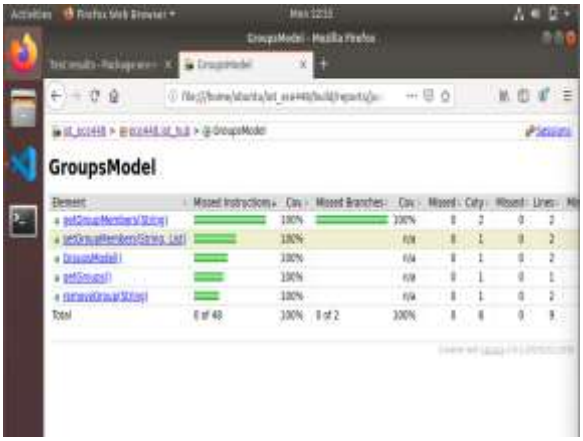


Figure 4: Utilization of Elements in GroupsModel

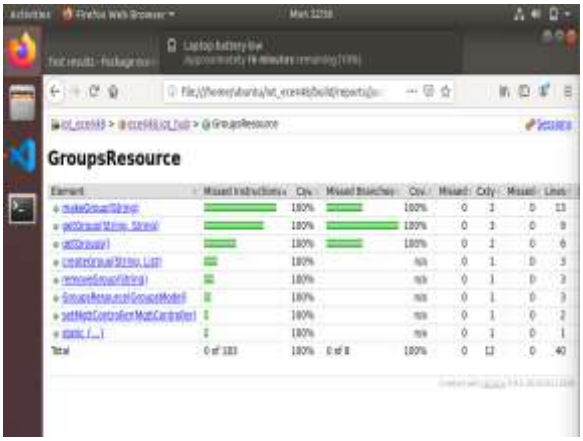
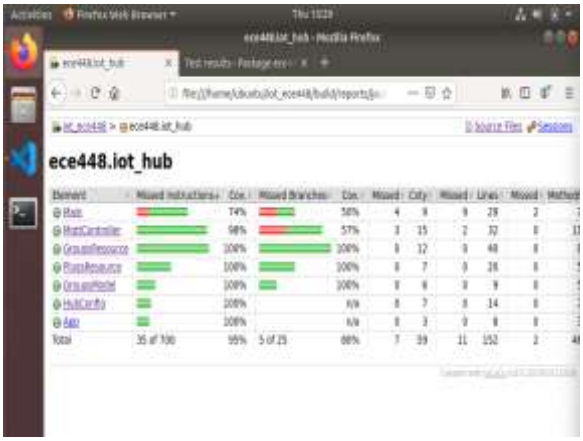


Figure 5: Utilization of Elements in GroupsResource



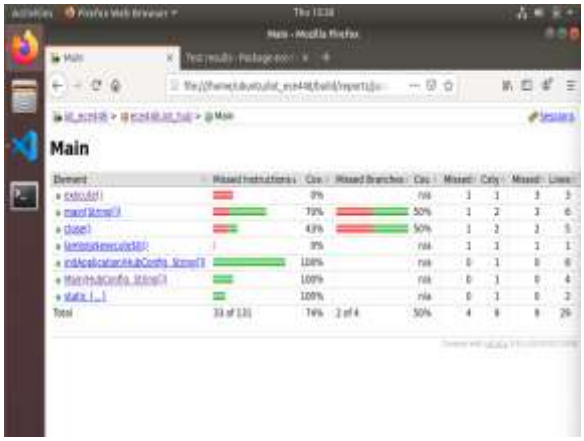


Figure 6: Utilization of ece448.iot_hub Figure 7: Utilization of Elements in Main.java

Conclusion:

In conclusion, in this project, we have introduced group management to our IoT hub, enabling users to define and control groups of smart plugs. We have followed the red-green cycle to add unit tests and implement our classes, utilizing MQTT and HTTP communications for testing purposes. While we have discussed possible class designs and implementations in lectures, we have had the freedom to choose designs and implementations we are comfortable with. Overall, this project has enabled us to enhance the server backend of our IoT hub and provide a more comprehensive and user-friendly experience for our users.

Appendix:

Source Code of the edited programs within the project:

GroupsModel.java

```
package ece448.iot_hub;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;

import org.springframework.stereotype.Component;

@Component
public class GroupsModel {
    private HashMap<String, HashSet<String>> groups = new HashMap<>();

    synchronized public List<String> getGroups() {
        return new ArrayList<>(groups.keySet());
    }

    synchronized public List<String> getGroupMembers(String group) {
        HashSet<String> members = groups.get(group);
        return (members == null)? new ArrayList<>(): new
ArrayList<>(members);
```

```
    }

    synchronized public void setGroupMembers(String group, List<String>
members) {
        groups.put(group, new HashSet<>(members));
    }

    synchronized public void removeGroup(String group) {
        groups.remove(group);
    }
}
```

GroupsResource.java

```
package ece448.iot_hub;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GroupsResource {

    private final GroupsModel groups;

    @Autowired
    private MqttController mqtt;

    public GroupsResource(GroupsModel groups) {
        this.groups = groups;
    }
    public void setMqttController(MqttController mqtt) {
```

```
        this.mqtt = mqtt;
    }

    @GetMapping("/api/groups")
    public Collection<Object> getGroups() throws Exception {
        ArrayList<Object> ret = new ArrayList<>();
        for (String group: groups.getGroups()) {
            ret.add(makeGroup(group));
        }
        logger.info("Groups: {}", ret);
        return ret;
    }

    @GetMapping("/api/groups/{group}")
    public Object getGroup(
        @PathVariable("group") String group,
        @RequestParam(value = "action", required = false) String action) {
        if (action == null) {
            Object ret = makeGroup(group);
            logger.info("Group {}: {}", group, ret);
            return ret;
        }

        // modify code below to control plugs by publishing messages to
MQTT broker
        List<String> members = groups.getGroupMembers(group);

        for (String plug: members)
            mqtt.publishAction(plug, action);
        logger.info("Group {}: action {}, {}", group, action, groups);

        return null;
    }

    @PostMapping("/api/groups/{group}")
    public void createGroup(
        @PathVariable("group") String group,
        @RequestBody List<String> members) {
        groups.setGroupMembers(group, members);
        logger.info("Group {}: created {}", group, groups);
    }

    @DeleteMapping("/api/groups/{group}")
    public void removeGroup(
        @PathVariable("group") String group) {
```

```
        groups.removeGroup(group);
        logger.info("Group {}: removed", group);
    }

    protected Object makeGroup(String group) {
        List<String> memberNames = groups.getGroupMembers(group);
        List<Map<String, Object>> members = new ArrayList<>();

        for (String name: memberNames) {
            HashMap<String, Object> plug = new HashMap<>();
            plug.put("name", name);
            plug.put("state", mqtt.getState(name));
            plug.put("power", mqtt.getPower(name));

            members.add(plug);
        }

        HashMap<String, Object> ret = new HashMap<>();
        ret.put("name", group);
        ret.put("members", members);
        return ret;
    }

    private static final Logger logger =
LoggerFactory.getLogger(GroupsResource.class);
}
```

GroupsTests.java

```
package ece448.iot_sim;
import static org.junit.Assert.*;
import java.util.Arrays;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.client.fluent.Request;
import org.apache.http.entity.ContentType;
import org.junit.Test;
import ece448.grading.GradeP3.MqttController;
import ece448.iot_hub.HubConfig;

public class GroupsTests {

    private Object[] runSimAndHub() throws Exception {
        String broker = "tcp://127.0.0.1";
```

```
        String topicPrefix =
System.currentTimeMillis()+"/test/iot_ece448";

        SimConfig config = new SimConfig(
            8080, Arrays.asList("xx", "yy", "zz.666"),
            broker, "testee/iot_sim", topicPrefix);
        HubConfig hubConfig = new HubConfig(
            8088, broker, "testee/iot_hub", topicPrefix);

        Thread simThread = new Thread() {
            public void run() {
                try (ece448.iot_sim.Main m = new
ece448.iot_sim.Main(config))
                {
                    // loop forever
                    for (;;)
                    {
                        Thread.sleep(60000);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        };
        simThread.start();
        Thread hubThread = new Thread() {
            public void run() {
                try (ece448.iot_hub.Main m = new
ece448.iot_hub.Main(hubConfig, new String[0]))
                {
                    // loop forever
                    for (;;)
                    {
                        Thread.sleep(60000);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        };
        hubThread.start();

        MqttController mqtt = new MqttController(broker, "grader/iot_hub",
topicPrefix);
        mqtt.start();
```

```
        Thread.sleep(3000);

        return new Object[]{mqtt, simThread, hubThread};
    }

    private void close(Object[] materials) throws Exception {
        ((Thread)materials[1]).interrupt();
        ((Thread)materials[2]).interrupt();
        ((MqttController)materials[0]).close();
    }

    static String getHub(String pathParams) throws Exception {
        return Request.Get("http://127.0.0.1:8088" + pathParams)
            .userAgent("Mozilla/5.0").connectTimeout(1000)
            .socketTimeout(1000).execute().returnContent().asString();
    }

    static void postGroup(String group, List<String> members) throws
Exception {
        Request.Post("http://127.0.0.1:8088/api/groups/" + group)
            .bodyByteArray(new ObjectMapper().writeValueAsBytes(members),
ContentType.APPLICATION_JSON)
            .userAgent("Mozilla/5.0").connectTimeout(1000)
            .socketTimeout(1000).execute();
    }

    static void delGroup(String group) throws Exception {
        Request.Delete("http://127.0.0.1:8088/api/groups/" + group)
            .userAgent("Mozilla/5.0").connectTimeout(1000)
            .socketTimeout(1000).execute();
    }

     @Test
     public void testGroupsRequest() throws Exception {
         Object[] materials = runSimAndHub();
         assertTrue(getHub("/api/groups").equals("[]"));

         close(materials);
     }

     @Test
     public void testGroupMembers() throws Exception {
         Object[] materials = runSimAndHub();
```

```
        postGroup("g", Arrays.asList("xx", "zz.666"));
        Thread.sleep(1000);

        String rsp = getHub("/api/groups/g");
        assertTrue(rsp.contains("g")
            && rsp.contains("xx") && !rsp.contains("yy") &&
rsp.contains("zz.666"));

        close(materials);
    }

    @Test
    public void testGroupPlugOn() throws Exception {
        Object[] materials = runSimAndHub();
        MqttController mqtt = (MqttController)materials[0];

        postGroup("g", Arrays.asList("xx", "zz.666"));
        getHub("/api/groups/a?action=on");
        Thread.sleep(1000);

        assertTrue("off".equals(mqtt.getState("xx"))
            && "off".equals(mqtt.getState("yy"))
            && "off".equals(mqtt.getState("zz.666")));

        close(materials);
    }

    @Test
    public void testTwoGroupsPlugOn() throws Exception {
        Object[] materials = runSimAndHub();
        MqttController mqtt = (MqttController)materials[0];

        postGroup("g", Arrays.asList("xx", "zz.666"));
        postGroup("a", Arrays.asList("xx", "yy"));
        getHub("/api/groups/g?action=on");
        getHub("/api/groups/a?action=toggle");
        Thread.sleep(1000);

        assertTrue("off".equals(mqtt.getState("xx"))
            && "on".equals(mqtt.getState("yy"))
            && "on".equals(mqtt.getState("zz.666")));

        close(materials);
    }
```

```
    @Test
    public void testMQTTTwoPlugsOn() throws Exception {
        Object[] materials = runSimAndHub();
        MqttController mqtt = (MqttController)materials[0];

        postGroup("g", Arrays.asList("xx", "zz.666"));
        getHub("/api/groups/g?action=on");
        Thread.sleep(1000);

        assertTrue("on".equals(mqtt.getState("xx"))
            && "off".equals(mqtt.getState("yy"))
            && "on".equals(mqtt.getState("zz.666")));

        close(materials);
    }

    @Test
    public void testJSONTwoPlugsOn() throws Exception {
        Object[] materials = runSimAndHub();
        postGroup("g", Arrays.asList("xx", "zz.666"));
        getHub("/api/groups/g?action=on");
        Thread.sleep(1000);

        String rsp = getHub("/api/plugs/xx");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));
        rsp = getHub("/api/plugs/yy");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));
        rsp = getHub("/api/plugs/zz.666");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));

        close(materials);
    }

    @Test
    public void testMQTTTwoPlugsOnOff() throws Exception {
        Object[] materials = runSimAndHub();
        MqttController mqtt = (MqttController)materials[0];
        postGroup("g", Arrays.asList("xx", "zz.666"));
        getHub("/api/groups/g?action=on");
        Thread.sleep(1000);

        assertTrue("on".equals(mqtt.getState("xx"))
            && "off".equals(mqtt.getState("yy"))
            && "on".equals(mqtt.getState("zz.666")));
```

```
        getHub("/api/groups/g?action=off");
        Thread.sleep(1000);

        assertTrue("off".equals(mqtt.getState("xx"))
            && "off".equals(mqtt.getState("yy"))
            && "off".equals(mqtt.getState("zz.666")));

        close(materials);
    }

    @Test
    public void testJSONTwoPlugsOnOff() throws Exception {
        Object[] materials = runSimAndHub();

        postGroup("g", Arrays.asList("yy", "zz.666"));
        getHub("/api/groups/g?action=on");
        Thread.sleep(1000);
        String rsp = getHub("/api/plugs/xx");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));
        rsp = getHub("/api/plugs/yy");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));
        rsp = getHub("/api/plugs/zz.666");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));

        getHub("/api/groups/g?action=off");
        Thread.sleep(1000);

        rsp = getHub("/api/plugs/xx");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));
        rsp = getHub("/api/plugs/yy");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));
        rsp = getHub("/api/plugs/zz.666");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));

        close(materials);
    }

    @Test
    public void testGroupRemove() throws Exception {
        Object[] materials = runSimAndHub();
        postGroup("g", Arrays.asList("xx", "zz.666"));
        Thread.sleep(1000);

        String rsp = getHub("/api/groups/g");
        assertTrue(rsp.contains("g")
```

```
            && rsp.contains("xx") && !rsp.contains("yy") &&
rsp.contains("zz.666"));

        delGroup("g");
        Thread.sleep(1000);

        rsp = getHub("/api/groups");
        assertTrue(!rsp.contains("g"));
        rsp = getHub("/api/groups/g");
        assertTrue(!rsp.contains("xx") && !rsp.contains("yy") &&
!rsp.contains("zz.666"));

        close(materials);
    }

    @Test
    public void testEditTwoPlugs() throws Exception {
        Object[] materials = runSimAndHub();
        MqttController mqtt = (MqttController)materials[0];

        postGroup("a", Arrays.asList("xx", "yy"));
        getHub("/api/groups/a?action=on");
        Thread.sleep(1000);
        mqtt.publishAction("zz.666", "on");
        mqtt.publishAction("xx", "toggle");
        Thread.sleep(1000);

        String rsp = getHub("/api/plugs/xx");
        assertTrue(rsp.contains("off") && !rsp.contains("on"));
        rsp = getHub("/api/plugs/yy");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));
        rsp = getHub("/api/plugs/zz.666");
        assertTrue(rsp.contains("on") && !rsp.contains("off"));

        close(materials);
    }

}
```

HTTPCommandsTests.java

```
package ece448.iot_sim;

import static org.junit.Assert.*;
```

```
import java.util.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.junit.Test;

import ece448.iot_hub.GroupsModel;
import ece448.iot_hub.GroupsResource;
import ece448.iot_hub.MqttController;
import org.junit.Before;
import java.util.Collection;

import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public class HTTPCommandsTests {

    private GroupsResource groupsResource;
    private GroupsModel groupsModel;
    private MqttController mqttController;

    @Before
    public void setUp() {
        groupsModel = new GroupsModel();
        mqttController = mock(MqttController.class);
        groupsResource = new GroupsResource(groupsModel);
        groupsResource.setMqttController(mqttController);
    }
    @Test
    public void testInit() throws Exception {
        PlugSim plugA = new PlugSim("a");
        PlugSim plugB = new PlugSim("b");
        PlugSim plugC = new PlugSim("c");
        ArrayList<PlugSim> plugList = new ArrayList<PlugSim>();
        plugList.add(plugA);
        plugList.add(plugB);
        plugList.add(plugC);
        HTTPCommands cmd = new HTTPCommands(plugList);
        String nameList = cmd.listPlugs();
        System.out.println(nameList);
        assertFalse(nameList == null);
```

```
        groupsModel.setGroupMembers("group1", List.of("device1"));
        groupsModel.setGroupMembers("group2", List.of());
        when(mqttController.getState("device1")).thenReturn("on");
        when(mqttController.getPower("device1")).thenReturn("100.0");
        Collection<Object> groups = groupsResource.getGroups();
        assertNotNull(groups);
        assertEquals(2, groups.size());
        for (Object groupObj : groups) {
            Map<String, Object> group = (Map<String, Object>) groupObj;
            String groupName = (String) group.get("name");
            List<Map<String, Object>> members = (List<Map<String,
Object>>) group.get("members");
            if (groupName.equals("group1")) {
                assertEquals(1, members.size());
                Map<String, Object> device1 = members.get(0);
                assertEquals("device1", device1.get("name"));
                assertEquals("on", device1.get("state"));
                assertEquals("100.0", device1.get("power"));
            } else if (groupName.equals("group2")) {
                assertEquals(0, members.size());
            } else {
                fail("Unexpected group found: " + groupName);
            }
        }
    }

    @Test
    public void testListPlugs() {
        PlugSim plugA = new PlugSim("a");
        PlugSim plugB = new PlugSim("b");
        PlugSim plugC = new PlugSim("c");
        ArrayList<PlugSim> plugList = new ArrayList<PlugSim>();
        plugList.add(plugA);
        plugList.add(plugB);
        plugList.add(plugC);
        HTTPCommands cmd = new HTTPCommands(plugList);
        assertTrue(cmd.listPlugs().equals("<html><body><p><a
href='/a'>a</a></p><p><a href='/b'>b</a></p><p><a
href='/c'>c</a></p></body></html>"));
    }

    @Test
    public void testReport() {
        PlugSim plugA = new PlugSim("a");
        PlugSim plugB = new PlugSim("b");
```

```
        PlugSim plugC = new PlugSim("c");
        ArrayList<PlugSim> plugList = new ArrayList<PlugSim>();
        plugList.add(plugA);
        plugList.add(plugB);
        plugList.add(plugC);
        HTTPCommands cmd = new HTTPCommands(plugList);
        String report = cmd.report(plugB);
        System.out.println(report);
        assertFalse(report.equals(""));
    }

    @Test
    public void testHandleNoPlug() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        assertEquals(cmd.handleGet("/", new HashMap<>()),
cmd.listPlugs());
    }

    @Test
    public void testHandleFakePlug() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        assertEquals(cmd.handleGet("/b", new HashMap<>()), null);
    }

    @Test
    public void testHandleRealPlug() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        assertTrue(cmd.handleGet("/a", new HashMap<>()) != null);
    }
```

```
    @Test
    public void testHandleActionOn() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        HashMap<String, String> params = new HashMap<>();
        params.put("action", "on");
        assertTrue(cmd.handleGet("/a", params).equals(cmd.report(a)));
    }

    @Test
    public void testHandleActionOff() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        HashMap<String, String> params = new HashMap<>();
        params.put("action", "off");
        assertTrue(cmd.handleGet("/x", params).equals(cmd.report(x)));
    }

    @Test
    public void testHandleActionToggle() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        HashMap<String, String> params = new HashMap<>();
        params.put("action", "toggle");
        assertTrue(cmd.handleGet("/x", params).equals(cmd.report(x)));
    }

    @Test
    public void testHandleActionNone() {
        PlugSim a = new PlugSim("a");
        PlugSim x = new PlugSim("x");
        ArrayList<PlugSim> plugList = new ArrayList<>();
        plugList.add(a);
```

```
        plugList.add(x);
        HTTPCommands cmd = new HTTPCommands(plugList);
        HashMap<String, String> params = new HashMap<>();
        params.put("action", "none");
        assertTrue(cmd.handleGet("/x",
params).equals("<html><body></body></html>"));
    }

    @Test
    public void testPowerOn() {
        PlugSim plug = new PlugSim("a");
        List<PlugSim> plugs = Arrays.asList(plug);
        HTTPCommands httpCommands = new HTTPCommands(plugs);
        Map<String, String> params = new HashMap<>();
        // When
        params.put("action", "on");
        httpCommands.handleGet("/a", params);
        // Then
        assertTrue("The plug should be on", plug.isOn());
        assertEquals("The power reading should be 0.0", 0.0,
plug.getPower(), 0.0001);
    }

    @Test
    public void testPowerOff() {
        PlugSim plug = new PlugSim("a");
        List<PlugSim> plugs = Arrays.asList(plug);
        HTTPCommands httpCommands = new HTTPCommands(plugs);
        Map<String, String> params = new HashMap<>();
        // When
        params.put("action", "off");
        httpCommands.handleGet("/TestPlug", params);
        // Then
        assertFalse("The plug should be off", plug.isOn());
        assertEquals("The power reading should be 0.0", 0.0,
plug.getPower(), 0.0001);
    }
}
```