Project 2

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: 2/19/2023)

Table of Contents

Abstract:

This project aims to develop features that enable the control of IoT simulator through web pages using the HTTP protocol. The proposed solution will eliminate the need for installing apps by providing access to the web page via a browser on a smartphone. The user stories are used as a tool for requirements analysis, which describes the desired features in natural language, balanced between end-user communication and developer evaluation. The implementation of the features will be carried out in the HTTPCommands class and will include the creation of unit tests in the HTTPCommandsTests.java file. The red-green cycle will be followed to add unit tests and implement desired features efficiently.

Introduction:

The purpose of this project is to develop features for controlling an IoT simulator through web pages using the HTTP protocol. The goal is to enable users to control smart plugs, such as turning on or off a light, from a browser on their smart phone without the need to install any apps. This project follows the Agile software development method and uses user stories to analyze requirements. The user stories are written in a natural language and are specific, measurable, achievable, relevant, and time-bound to ensure clear communication between end users and developers. The features will be implemented in the HTTPCommands class and unit tests will be created to ensure their functionality. The project requires careful attention to the method report and the creation of unit tests in the src/test/java/ece448/iot_sim/HTTPCommandsTests.java file.

Background:

The project's main objective is to understand the skills in Java programming via web pages using HTTP protocol. The project involves the implementation of three user stories: Plug Report, Toggle or Switch a Plug On/Off, and Control Feedback. These user stories enable end-users to access a report of a plug, control a plug using a query string, and receive an up-to-date report as a response. The specific requirements for this project include:

- Plug Report: The user wants to access a web page report of a plug named "plugName" without any query string, to view the switch status and power reading of the plug using a browser.
- Toggle or Switch a Plug On/Off: The end-user wants to control a plug named "plugName" through a web page accessible at the path /plugName using a browser. The control actions include toggle, turn on, and turn off and are specified in the query string with values action=toggle, action=on, and action=off respectively.
- Control Feedback: The end-user wants to receive an updated report of the plug after making changes to its state using the browser. The report will be sent as a response to the path "/plugName" with a query string and will help the end-user verify if the plug is functioning properly.

Results and Discussion:

The HTTPCommands class was the only one that underwent modification. I made the following changes to the handleGet () method within the class. If a request is made to a path of an existing plug and if there is a specified action, the method will carry out the designated action on the plug and then return a report including information about the plug and its available actions. The available actions are toggle, switch on, switch off.

I created different queries in the HTTPCommands test program for Project 2 testing which is separated into two parts: unit testing and acceptance testing. 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 2.

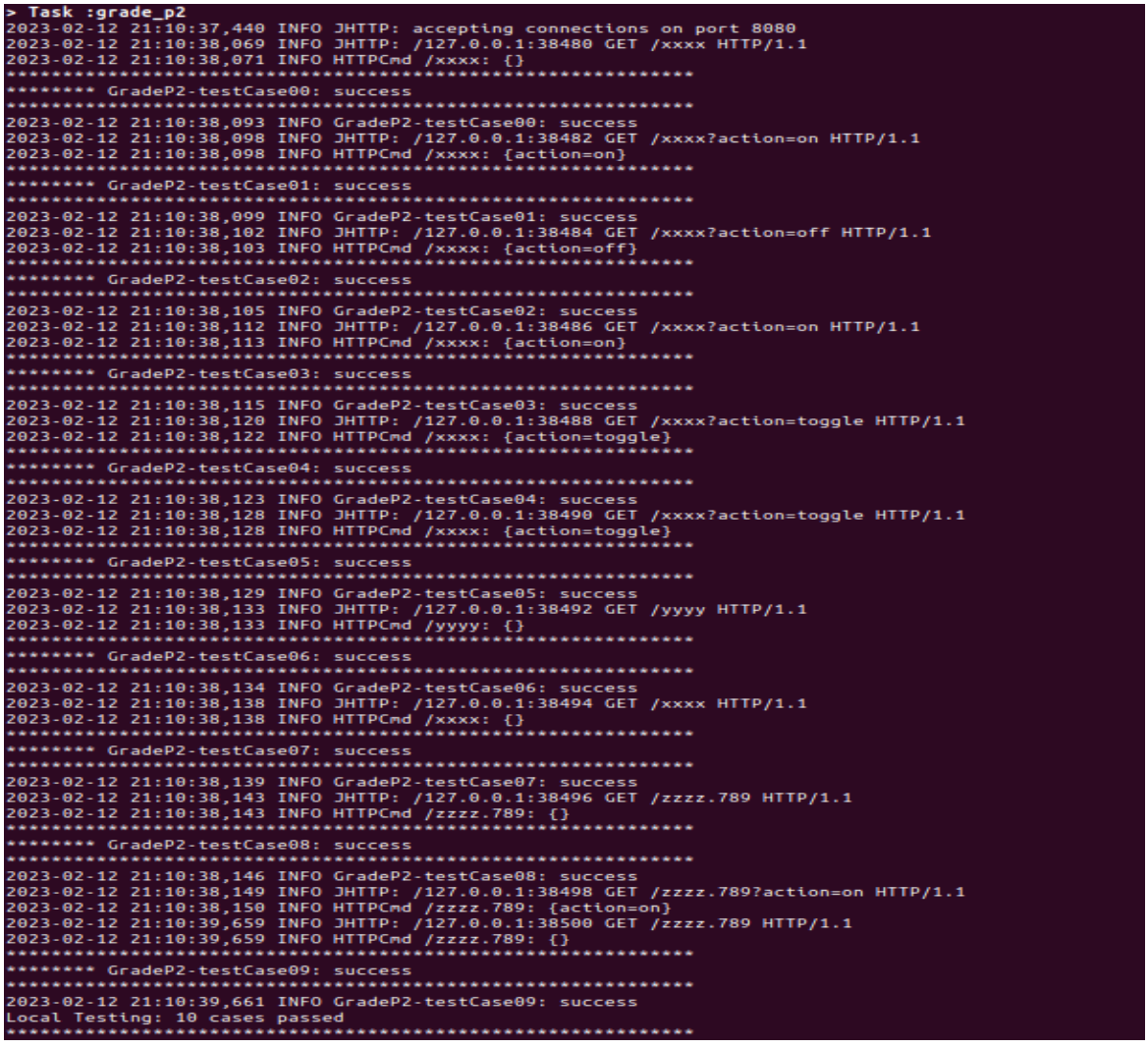

Figure 1: GradleP2- Test cases Results

Screenshots of the HTML results:

To completely utilize the HTTPCommands, I referred to the HTML result page 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.

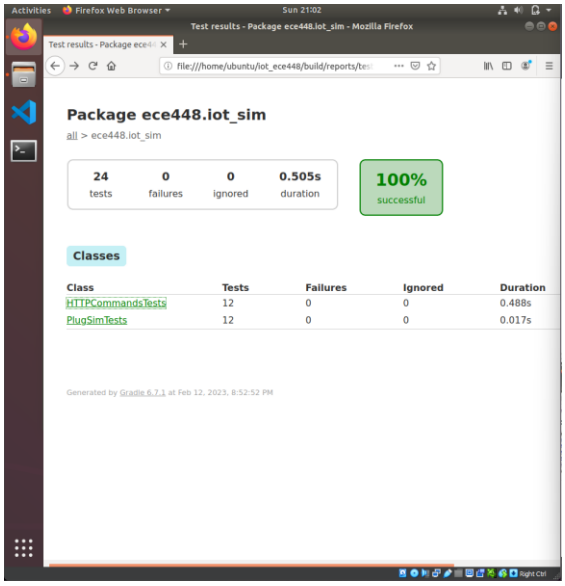

Figure 2: IoT-Sim Test Cases

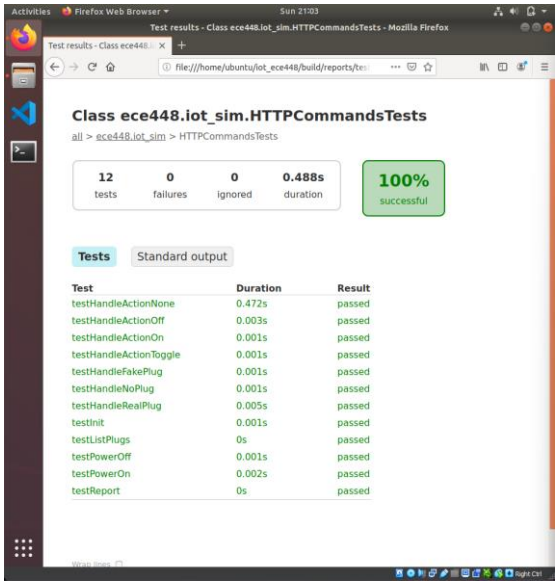

Figure 3: HTTPCommands Test Cases

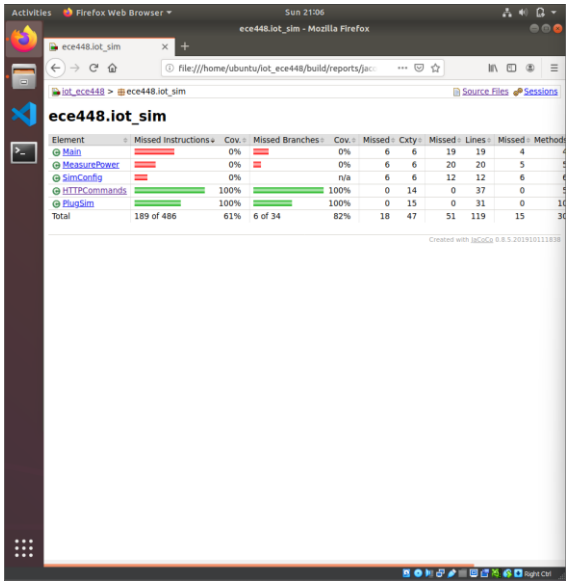

Figure 4: Complete HTTPCommands Instructions

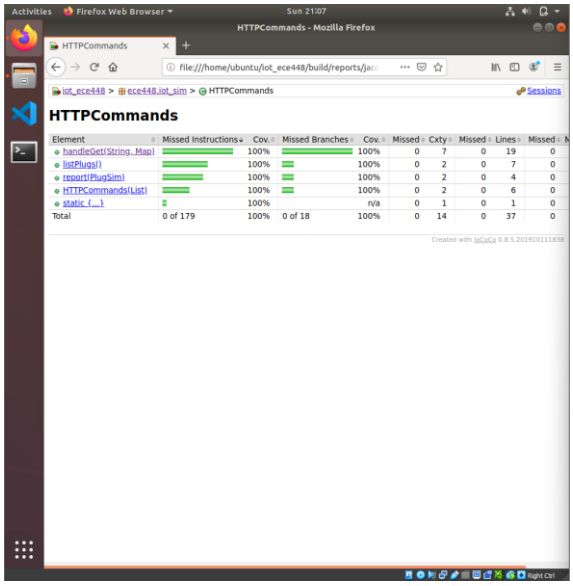

Figure 5: Utilization of Elements

Conclusion:

In conclusion, this project focuses on developing features that enable control of the IoT simulator through web pages using the HTTP protocol. This is a crucial aspect of smart plugs, as it allows for the control of lights through a browser on a smartphone without the need for any additional apps. The user stories, which are a crucial tool for requirements analysis in Agile software development, help to strike a balance between natural language preferred by end-users and the specific requirements needed by developers. The implementation of these features will take place in the HTTPCommands class, and it is important to pay attention to the report method as it generates web pages. Unit tests will also be developed in the HTTPCommandsTests.java file, following the red-green cycle.

Appendix:

Source Code of the edited programs within the project:

HTTPCommands.java

```
package ece448.iot_sim;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ece448.iot_sim.http_server.RequestHandler;
public class HTTPCommands implements RequestHandler {

   // Use a map so we can search plugs by name.
   private final TreeMap<String, PlugSim> plugs = new TreeMap<>();

   public HTTPCommands(List<PlugSim> plugs) {
      for (PlugSim plug: plugs)
      {
         this.plugs.put(plug.getName(), plug);
      }
   }

   @Override
   public String handleGet(String path, Map<String, String> params) {
      // list all: /
      // do switch: /plugName?action=on|off|toggle
      // just report: /plugName

      logger.info("HTTPCmd {}: {}", path, params);
```

```
    if (path.equals("/"))
    {
        return listPlugs();
    }

    PlugSim plug = plugs.get(path.substring(1));
    if (plug == null)
        return null; // no such plug

    String action = params.get("action");
    if (action == null)
        return report(plug);

    // P2: add your code here, modify the next line if necessary
    //turn on plug
    if (action.equals("on"))    {
        plug.switchOn();
        return report(plug);
    }
    //turn off plug
    if (action.equals("off"))   {
        plug.switchOff();
        return report(plug);
    }
    //toggle plug
    if (action.equals("toggle"))    {
        plug.toggle();
        return report(plug);
    }
    return "<html><body></body></html>";
}

protected String listPlugs() {
    StringBuilder sb = new StringBuilder();

    sb.append("<html><body>");
    for (String plugName: plugs.keySet())
    {
        sb.append(String.format("<p><a href='/%s'>%s</a></p>",
            plugName, plugName));
    }
    sb.append("</body></html>");

    return sb.toString();
```

```
    }

    protected String report(PlugSim plug) {
        String name = plug.getName();
        return String.format("<html><body>"
            +"<p>Plug %s is %s.</p>"
            +"<p>Power reading is %.3f.</p>"
            +"<p><a href='/%s?action=on'>Switch On</a></p>"
            +"<p><a href='/%s?action=off'>Switch Off</a></p>"
            +"<p><a href='/%s?action=toggle'>Toggle</a></p>"
            +"</body></html>",
            name,
            plug.isOn()? "on": "off",
            plug.getPower(), name, name, name);
    }

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

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;

public class HTTPCommandsTests {

    @Test
    public void testInit() {
        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);
```

```
    }

    @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);
    }
}
```