Smart Hub for the Internet of Things (IoT)
Project Instructions

I.

Introduction

The goal is to build a Smart Hub for the Internet of Things (IoT) using full-stack web application frameworks. As we will spend time developing this Smart IoT Hub, projects are broken down into 7 manageable projects. Projects are organized in a way that you do not need to understand the whole system before you can demonstrate your progress.

Project documents will indicate what you should accomplish for each project. Note that they will mostly specify the requirements instead of describing step-by-step procedures/implementations. Though discussions of the projects are in the lecture notes, you are required to design and implement the system by combining the knowledge learned from the lecture, other sources, and problem-solving and troubleshooting skills.

II.

The IoT Hub

Our IoT hub consists of a frontend web application that enables users to monitor and control their IoT devices from their phones, pads, and/or computers, and a server backend deployed on-premises or in the cloud that communicates with the IoT devices. Open-source solutions like Home Assistant exist, but we would like to build our own to learn application software development.

It is quite tempting for someone to prototype such an IoT Hub using actual IoT devices. We choose not to do so in part because that is not quite practical for our course. More importantly, we would like to develop an IoT hub that can work with any IoT device from any device vendor. In other words, we want to avoid any dependency on proprietary services from those vendors, i.e. to prevent vendor lock-in. Our solution is to first build an IoT simulator in Projects 1 to 3 that communicates with our IoT hub using open protocols. This simulator not only will help us to develop and test our IoT hub later but also may give you some idea of how future IoT devices should be created. The server backend of our IoT hub will be created in Projects 4 and 5. In addition to providing services over individual IoT devices, the server backend would further allow users to group devices into groups and control devices belonging to the same group at the same time. The frontend web application will be created in Project 6, providing a user interface to our IoT hub via browsers

available on most mobile devices and computers. Overall, our IoT hub may work with multiple IoT simulators/devices and multiple users at the same time, enabling the distributed operation of the whole system.

With all the experiences gained from Projects 1 to 6, you will be able to extend the IoT hub in the open-ended Project 7. Sample ideas you may explore include TLS/HTTPS support, Alexa integration, database integration, and data visualization. Feel free to come up with your idea but try to base it on what you can complete before the deadline.

III.

Working with Your Projects

A.

Directory and File Conventions

We will utilize the Gradle build tool to build, test, and grade your projects, which in turn depends on a proper layout of directories and files in your project submissions. The desired layout is provided with the initial project package as follows and is widely used for Java development today.

• The file build.gradle: this is the Gradle script controlling all aspects of building, testing, and grading your projects.

• The directory src: this is where you should put your source files for the IoT simulator and the server backend.

o src/main/java: Java source code organized into packages

o src/main/resources: files used by Java packages, e.g., logback.xml is the default configuration for logging

o src/test/java: Java source code for testing purposes organized into packages

o src/test/resources: files used by Java packages for testing purposes. It is empty initially but you may find it useful later

• The directory public: this is where you should put your source files for the frontend web application

• The .json files: these are the configuration files to run the IoT simulator and the server backend

• The file iot_ece448_code-workspace: this is used by VS Code to locate your projects. VS Code will then read build.gradle to load your projects

• The hidden files .gitignore: these files keep the Git repository clean

It is important for you to follow those conventions as otherwise your projects won't be graded. Feel free to add/modify files as long as you feel comfortable doing so but please be advised that build.gradle and all files under src/main/java/ece448/grading will be overwritten to their initial version for building, testing, and grading your projects.

B.

Building and Testing Your Projects

Simply use Gradle to build and test your projects from a Terminal in the project directory.

gradle

This command first downloads third-party libraries if they have not been downloaded yet. Once all libraries are downloaded, it builds the projects, executes all unit tests, and creates the coverage report. If anything fails, Gradle will produce error messages and you should read and understand what went wrong. It could be a broken Internet connection preventing a required third-party library from downloading – in such case, you will need to connect to the Internet and run Gradle again. It could also be some compilation problem with your Java code and you must correct it. Or, it could be a failed unit test case, and you will need to make it pass.

If Gradle builds and tests the projects successfully, it creates the unit test report at

build/reports/tests/test/index.html

and the coverage report at

build/reports/jacoco/test/html/index.html

Please open and explore them with a web browser. You'll need to take some screenshots of them later to be included in your project report. You may run grading test cases the same way using Gradle for troubleshooting.

gradle grade_p1

You will need to change grade_p1 into grade_p2 and so on for Project 2-5.

We will use the same scripts to build, test, and grade your projects. Please make sure to check your Git status and log to make sure that, first, all local modifications and new files are committed and pushed, and second, build.gradle and all files under src/main/java/ece448/grading have not been changed. Please be advised that all the scripts are provided to you only for your convenience and we will only use the grading report from our end to decide your project grade.

Please be aware that while the unit tests may run in any order and therefore should be independent of each other, our grading test cases are executed in their specific order and a later grading test case may depend on the results from previous grading test cases.

C.

Interact with IoT Simulator and IoT Hub

Many times it will be very helpful for you to interact with the IoT simulator and the IoT hub to understand how they work. We again use Gradle here to simplify the task of bundling your projects with all necessary Java libraries and to run them.

To start the IoT simulator with the configuration from simConfig.json, run

gradle iot_sim

If you have not changed anything in simConfig.json, you will be able to interact with the simulator at http://127.0.0.1:8080 using a web browser in the VM. Not all functionalities may work as some may not be implemented yet. As you click through the web pages, pay attention to the log messages from the terminal to understand how various parts of the IoT simulator work together.

Starting from Project 3, you may open a new terminal and start a second IoT simulator with the configuration from simConfigEx.json using

gradle iot_sim_ex

You will be able to interact with this simulator at http://127.0.0.1:8081. Feel free to run more simulators for fun by studying how the two commands are created in build.gradle.

The IoT hub with the configuration from hubConfig.json can be started with

gradle iot_hub

You will be able to interact with the IoT hub at http://127.0.0.1:8088. Internet connection may be required as your web browser may need to download supporting libraries for our web application. You may or may not see anything on the webpage depending on your progress with Projects 4 and 5 but you should always be able to read the log messages from the terminal. Please also start at least one IoT simulator so the IoT hub will have something to display.

V.

Deliverables and Grading

Unit Test Cases (10 points): You are required to create at least 10 new unit test cases located in src/test/java. 1 point will be awarded per new unit test case that passes, up to 10 points total.

▪ Please make sure to add the source files to your Git repository before committing and pushing your projects.

▪ Note that we require 10 new unit tests for each project – e.g., by Project 4, we should see 40 unit tests from your Test Summary page, and all of them should be different.

o Unit Test Coverage (10 points): Your unit tests should achieve an average coverage of 90% for all the classes you have added/modified. We will deduct 1 point for each 5% you are missing (e.g., 9 points for 88%, 8 points for 81%).

• Grading Test Cases (70 points): 7 points per grading test case that your project passes as indicated by the given grading programs.

o Your code should perform reasonably well for the grading test cases – they should not need more than 15 minutes to complete them all together, and should not consume more than 8 GB of memory.

o Please be advised that no partial credits will be given. If your program passes a test, you will be awarded 7 points. If your program fails a test, no points will be awarded (e.g., 49 points will be awarded for 7 passes and 3 fails, 35 points will be awarded for 5 passes and 5 fails)

o Please be advised that you should NOT alter the provided grading program located in src/main/java/ece448/grading. We will be using the original grading program to grade your project source codes.