Assignment 23: Set a timer and provide spoken feedback - 10422030

Questions:

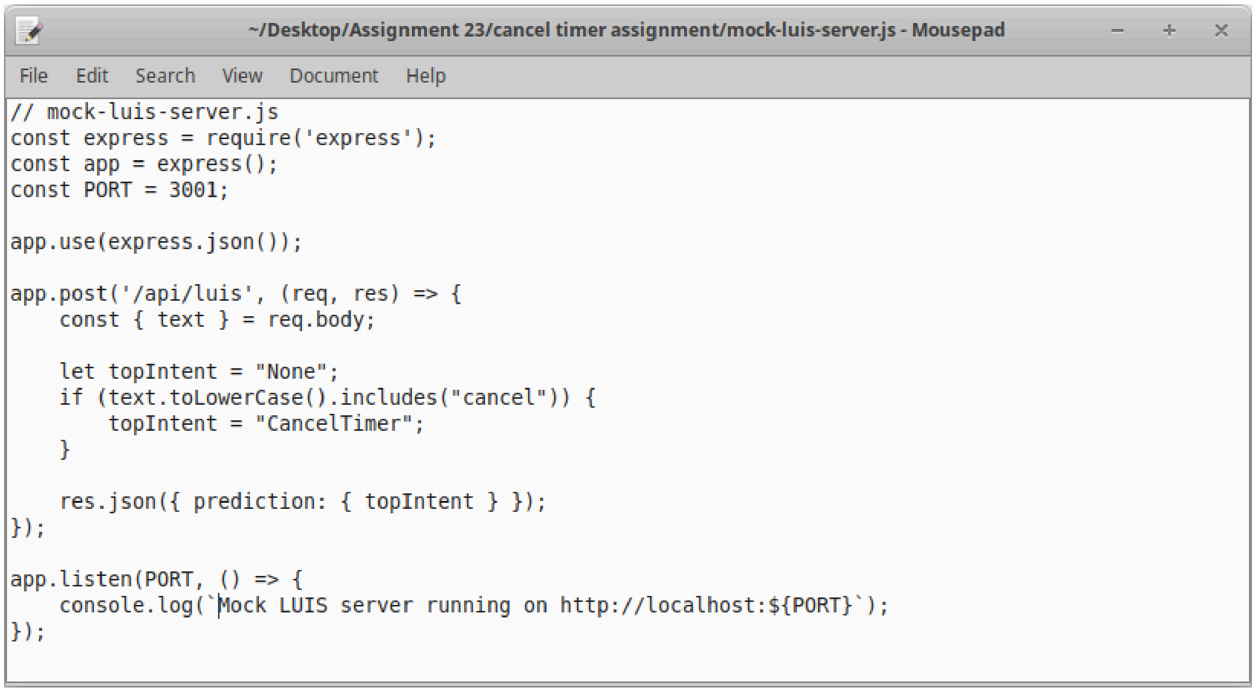
In the assignment for the last lesson, you added a cancel timer intent to LUIS. For this assignment you need to handle this intent in the serverless code, send a command to the IoT device, then cancel the timer.

Answer:

- In this Assignment, I also run on my local machine and use the simulate server as I do in Assignment 22. Because I can't access Azure so that I must use the local machine instead.

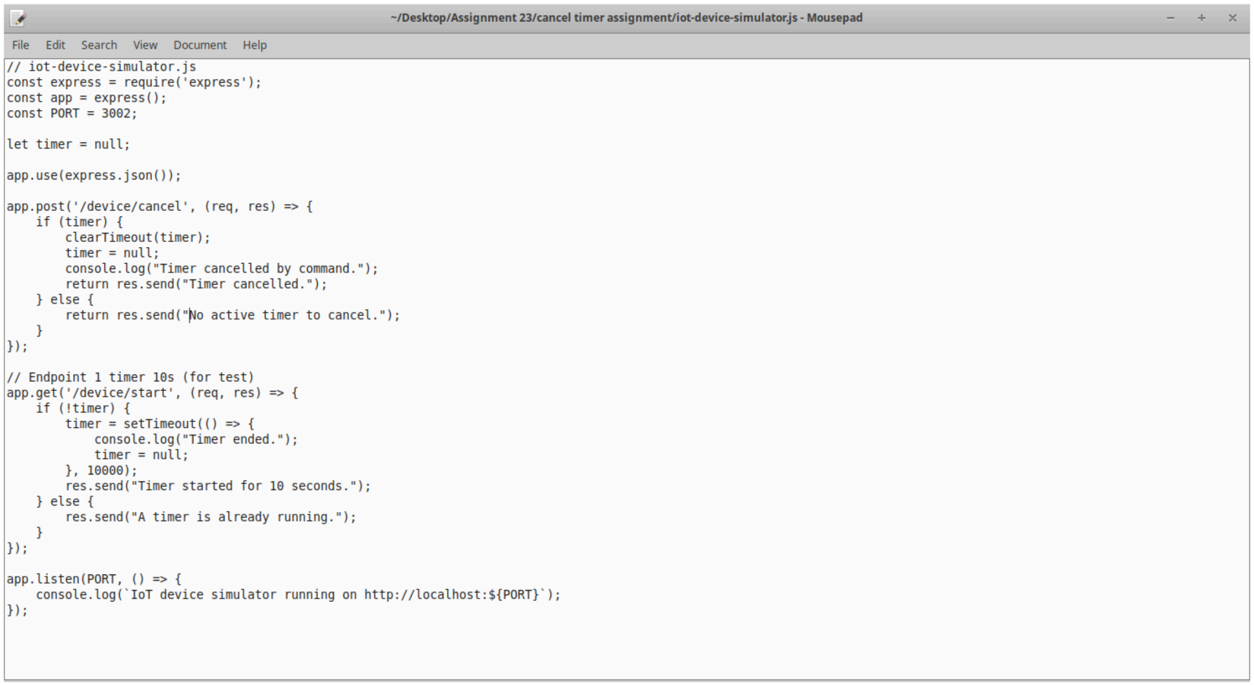
1. Create the iot-device-simulator.js, main-controller.js and mock-luis-server.js files.

- To simulate the server, we need the mock-luis-server.js file to do it.


~/Desktop/Assignment 23/cancel timer assignment/mock-luis-server.js - Mousepad
File Edit Search View Document Help

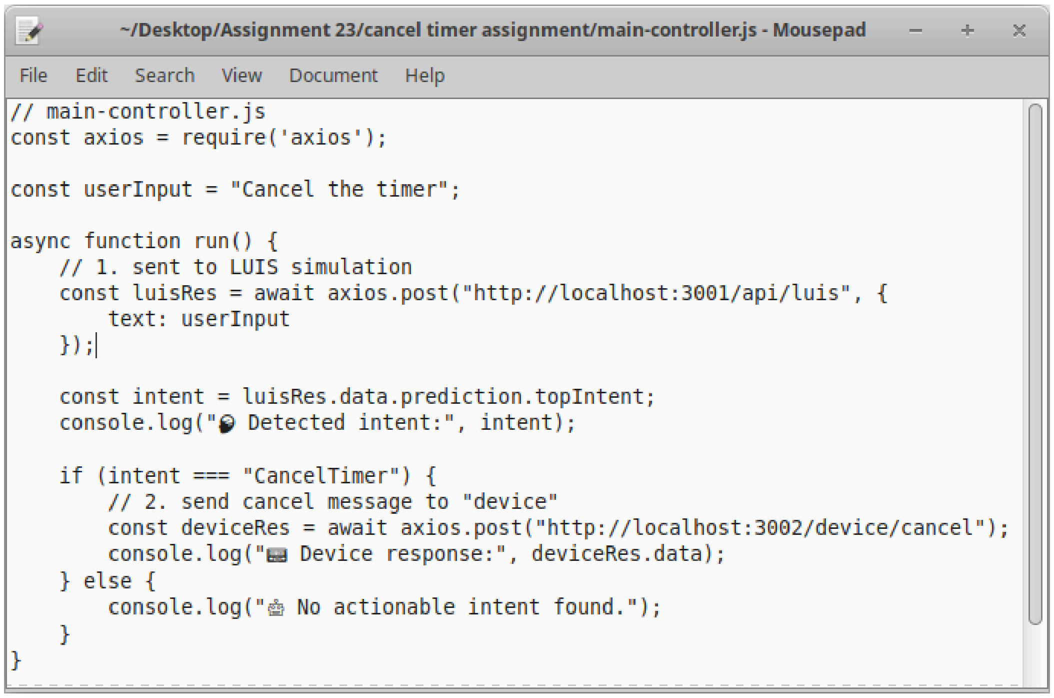
```
// mock-luis-server.js
const express = require('express');
const app = express();
const PORT = 3001;

app.use(express.json());

app.post('/api/luis', (req, res) => {
    const { text } = req.body;

    let topIntent = "None";
    if (text.toLowerCase().includes("cancel")) {
        topIntent = "CancelTimer";
    }

    res.json({ prediction: { topIntent } });
});

app.listen(PORT, () => {
    console.log(`Mock LUIS server running on http://localhost:${PORT}`);
});
```

- Then, we also need the IoT devices to send and process the data to the server LUIS.


~/Desktop/Assignment 23/cancel timer assignment/iot-device-simulator.js - Mousepad
File Edit Search View Document Help

```
// iot-device-simulator.js
const express = require('express');
const app = express();
const PORT = 3002;

let timer = null;

app.use(express.json());

app.post('/device/cancel', (req, res) => {
    if (timer) {
        clearTimeout(timer);
        timer = null;
        console.log("Timer cancelled by command.");
        return res.send("Timer cancelled.");
    } else {
        return res.send("No active timer to cancel.");
    }
});

// Endpoint 1 timer 10s (for test)
app.get('/device/start', (req, res) => {
    if (!timer) {
        timer = setTimeout(() => {
            console.log("Timer ended.");
            timer = null;
        }, 10000);
        res.send("Timer started for 10 seconds.");
    } else {
        res.send("A timer is already running.");
    }
});

app.listen(PORT, () => {
    console.log(`IoT device simulator running on http://localhost:${PORT}`);
});
```

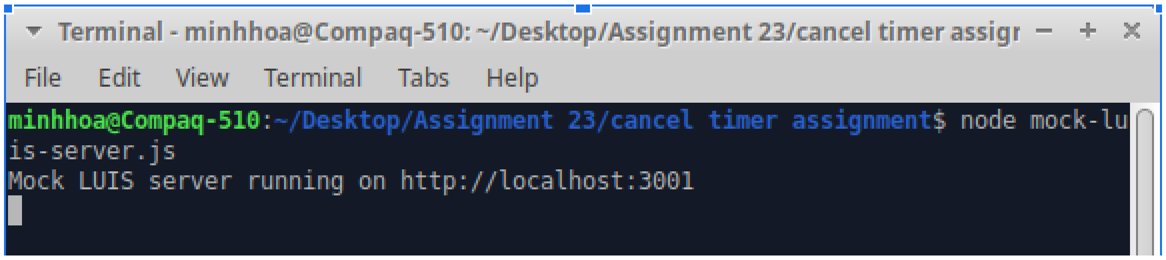
- The main controller file can help us send the cancel time to the server and the server will send this message to simulate IoT devices.


~/Desktop/Assignment 23/cancel timer assignment/main-controller.js - Mousepad
File Edit Search View Document Help

```
// main-controller.js
const axios = require('axios');

const userInput = "Cancel the timer";

async function run() {
    // 1. sent to LUIS simulation
    const luisRes = await axios.post("http://localhost:3001/api/luis", {
        text: userInput
    });

    const intent = luisRes.data.prediction.topIntent;
    console.log("🧠 Detected intent:", intent);

    if (intent === "CancelTimer") {
        // 2. send cancel message to "device"
        const deviceRes = await axios.post("http://localhost:3002/device/cancel");
        console.log("📟 Device response:", deviceRes.data);
    } else {
        console.log("🤖 No actionable intent found.");
    }
}
```

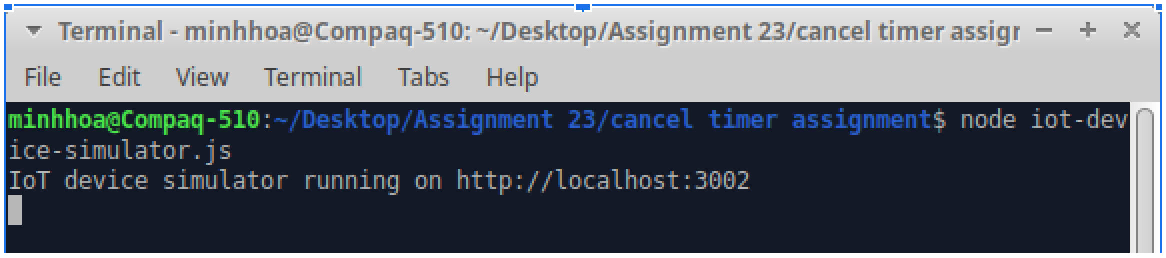
2. Run and test.

- First we will run the mock LUIS simulator server.


Terminal - minhhoa@Compaq-510: ~/Desktop/Assignment 23/cancel timer assigr
File Edit View Terminal Tabs Help

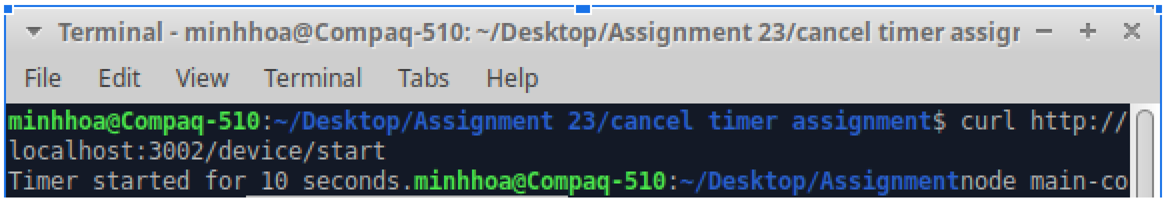
```
minhhoa@Compaq-510:~/Desktop/Assignment 23/cancel timer assignment$ node mock-lu
is-server.js
Mock LUIS server running on http://localhost:3001
```

- After that, we start the IoT device.


Terminal - minhhoa@Compaq-510: ~/Desktop/Assignment 23/cancel timer assigr
File Edit View Terminal Tabs Help

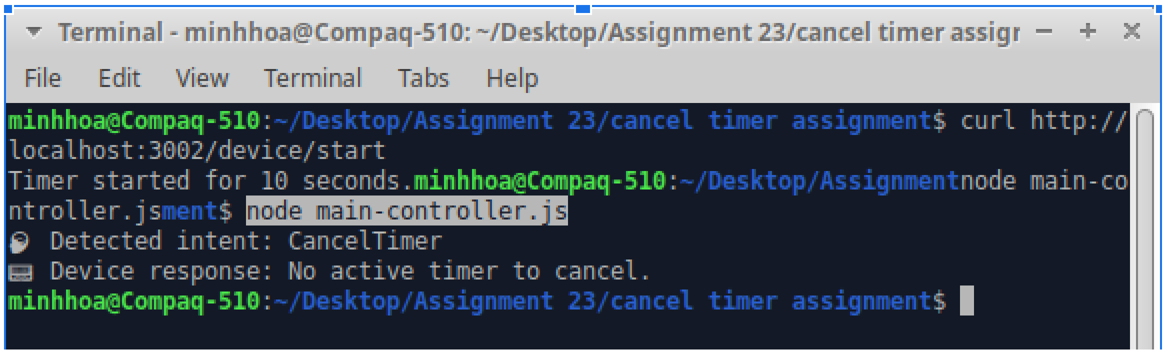
```
minhhoa@Compaq-510:~/Desktop/Assignment 23/cancel timer assignment$ node iot-dev
ice-simulator.js
IoT device simulator running on http://localhost:3002
```

- And, we can start the timer.


Terminal - minhhoa@Compaq-510: ~/Desktop/Assignment 23/cancel timer assigr
File Edit View Terminal Tabs Help

```
minhhoa@Compaq-510:~/Desktop/Assignment 23/cancel timer assignment$ curl http://
localhost:3002/device/start
Timer started for 10 seconds.minhhoa@Compaq-510:~/Desktop/Assignmentnode main-co
```

- Finally, we will run the main program to test.


Terminal - minhhoa@Compaq-510: ~/Desktop/Assignment 23/cancel timer assigr
File Edit View Terminal Tabs Help

```
minhhoa@Compaq-510:~/Desktop/Assignment 23/cancel timer assignment$ curl http://
localhost:3002/device/start
Timer started for 10 seconds.minhhoa@Compaq-510:~/Desktop/Assignmentnode main-co
ntroller.jsment$ node main-controller.js
🧠 Detected intent: CancelTimer
📟 Device response: No active timer to cancel.
minhhoa@Compaq-510:~/Desktop/Assignment 23/cancel timer assignment$
```

- As we can see above, the mock device is reponse with the message “No active timer to cancel” when it receives “CancelTimer”.

3. Conclusion

In this assignment, we implemented a local simulation of handling the CancelTimer intent without using Azure services. The objective was to recognize the intent, send a command to a simulated IoT device, and handle the timer cancellation logic.

To achieve this, we created a mock LUIS recognizer to detect the CancelTimer intent, and a serverless-style controller (main-controller.js) to interpret and route the command. A simulated device module was used to process incoming commands and respond accordingly. The system was tested locally on a Linux PC using Node.js v12, ensuring full compatibility without requiring cloud infrastructure.

The simulation successfully demonstrated intent recognition, command dispatch, and appropriate device response. When the CancelTimer intent was triggered, the system returned a message confirming whether a timer was active and canceled, or that no timer was set.

This assignment showcases the ability to develop and test intent-based IoT interactions using only local resources, which is valuable for offline development, testing, or constrained environments.