Coding Activity 1.1

Set Up Express Server

**Express Server**

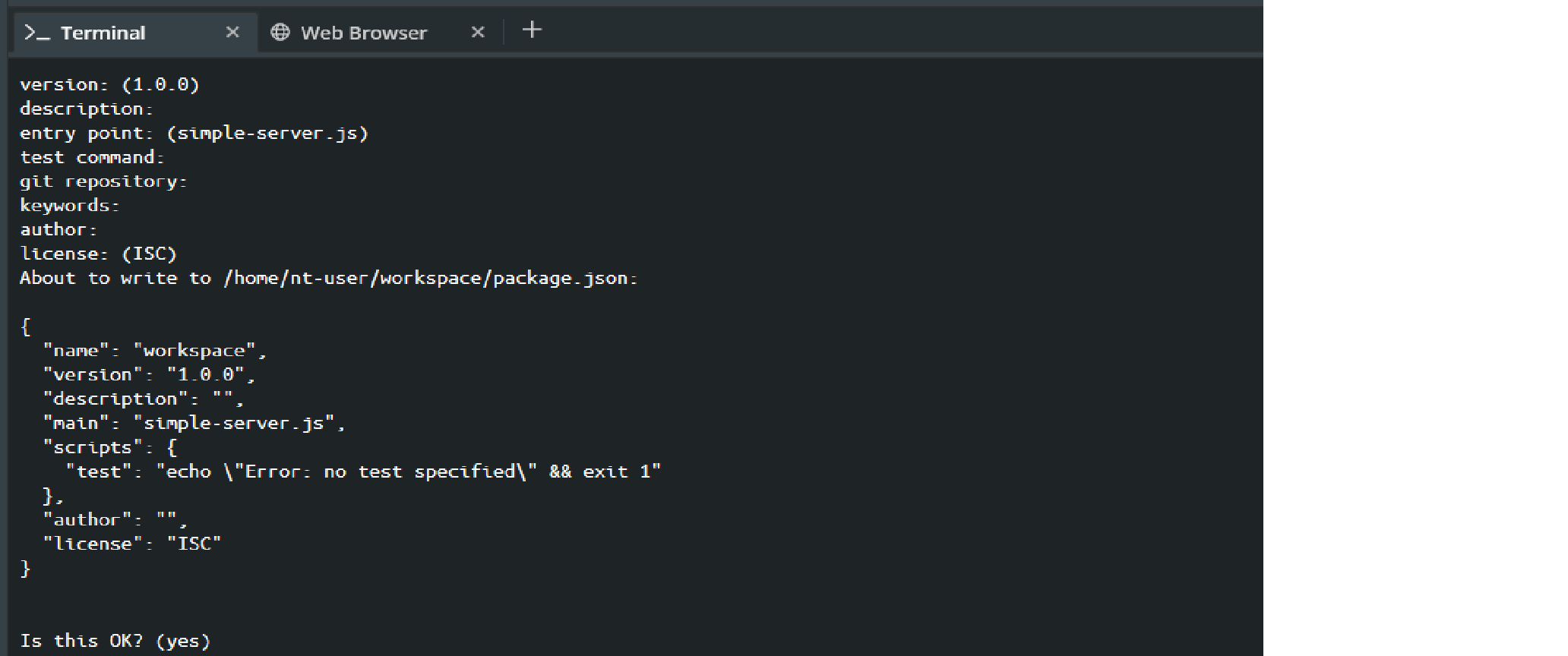
**In this activity, you will focus on the middle-tier application and run your own server. You will initialize and run a new node project and then write handlers for different routes on that server.**

**Step 1: Project initialization**

First, you need to initialize a node project using the simple-server.js file as the entry point, ensuring that you've installed all dependencies in the new project's node\_modules folder.

After you initialize the project, you can use the node simple-server.js command in your Terminal tab to run simple-server.js as your server. However, this command won't work until you initialize the node project.

In order to initialize the project, run npm init from the current directory in your Terminal tab. Click enter when you are prompted to do so. You should then see something like this:

[](https://cdn.filestackcontent.com/jtB9p81DSwexIEa2J08y)

Next, install the dependencies for this simple server. You will need to work out what the dependencies are by looking at the simple-server.js file and reviewing the contents of the require() statement at the top.

You can then take that value and run the following command:

npm install <package listed in the require section>

Note: Don't add quotation marks to the npm install command.

**Step 2: Run your simple server**

Now that you've initialized the project using its own package.json file and a node\_modules folder for all its external dependencies, you can now run the project by running the following command: node simple-server.js

**Step 3: Add the npm start script to package.json**

Adding a start-up script will help you to run this project more easily in the future. In your workspace, navigate to the package.json file. In the scripts section, add a script with the key start and a value of node simple-server.js. It should look like this:

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node simple-server.js"

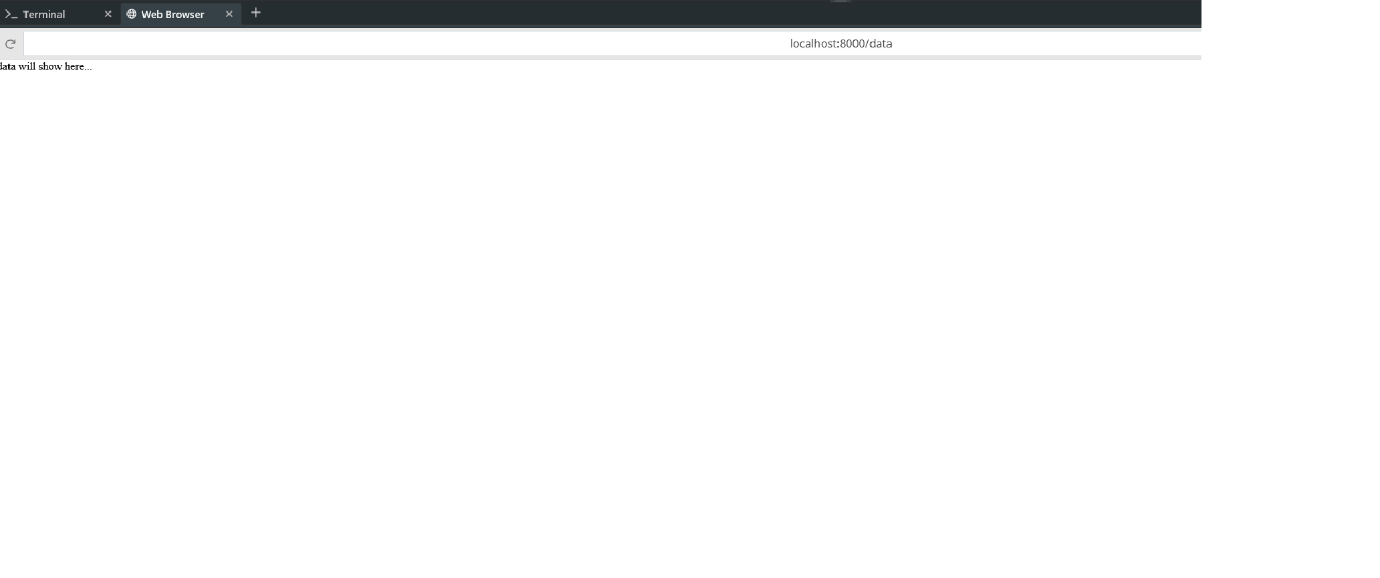
},

You can use npm start to re-run your Express server after you have stopped it by hitting CTRL+C or CMD+C (depending on which OS you’re using).

**Step 4: Add /data route with specific response text**

Now it's time to add a response for another route in your Express server. You will add a response for the /data route when navigating to localhost:8000/data in your Web Browser tab. In the simple-server.js file you will see two code blocks that start with app.get. Add your own response using res.send('...') in the block for the /data route; this response should say data will show here...

After adding the code, restart your Express server and navigate to localhost:8000/data; you should see something like this:

[](https://cdn.filestackcontent.com/NTZfNaQjSBSvxFJoDqEp)

That’s it, you have completed the activity—nice work.

Note: Unlike a React project created with create-react-app (which will watch project files and automatically recompile to show your changes in browser) this Express server does not automatically refresh. In order to see your code changes reflected in the Express server, you need to stop it in the Terminal tab, restart it, and refresh the Web Browser tab to see your changes.

Tasks

Add the necessary dependencies for simple-server.js with npm install. They should be showing under the dependencies block in package.json.

**0**

Add an npm start script to the package.json file so you can run the express project with that standard command rather than node simple-server.js every time.

**0**

Run server on port 8000 and send response for /data route so that the specified text shows in the DOM.