CMSC335 ("The Forgotten Class") Fall 2024

Project #4, SuperMarket App (Due Thu, Nov 14, 11:55 pm)

Objectives

To practice Node.js, setting routes using express and JSON data processing.

This project is dedicated to the around 25 students that show up to lecture of which 3 are paying attention and the rest making sure I record the lecture :).

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. Any case of academic dishonesty will be referred to the University's Office of Student Conduct. **Please don't post your code online where others can see your code.**

You can discuss this project with other students, but you may not implement code together and you may not share code.

Overview

For this project you will implement an app called Supermarket that allow users to buy food online. You will define a server (using Node.js) that will allow users to see the items for sale and to place an order. Information about items available will be provided in JSON files that the server will access. A video showing some of the system functionality can be found at Application Video. You should watch this video before you continue reading. **Please read the submission instructions below (they are not the usual ones).**

Grading

1. (9%) One ES6 class (like in Java) defined and used. The class has least one **private** instance variable, a constructor and at least one method (besides the constructor). What you used the class for, it is up to you.
2. (10%) Server command line interpreter
    - (2%) Stopping server with "stop".
    - (4%) Displaying items list when "itemsList" is entered.
    - (2%) "Invalid command: " followed by comand is displayed for a command other than "stop" or "itemsList".
    - (2%) Displaying the message "Usage supermarketServer.js jsonFile" when an incorrect number of command line arguments are provided.
3. (16%) Provided templates files are used to generate the application pages.
4. (5%) Not using iteration statements (i.e., for(i = 0; i <), while, or do-while) while processing an array. You need to use array functions similar to forEach, reduce, find, etc.

5. (40%) Placing Order Functionality
    - (10%) Correct item's list is displayed after information is provided (using one of the JSON files provided).
    - (10%) Correct total cost generated (using one of the JSON files provided).
    - (10%) Correct item's list is displayed after information is provided (using our own JSON file).
    - (10%) Correct total cost generated (using our own JSON file).
6. (20%) Catalog Functionality
    - (10%) Correct item's list is displayed after information is provided (using one of the JSON files provided).
    - (10%) Correct item's list is displayed after information is provided (using our own JSON file).

Code Distribution

The project's code distribution is available at Supermarket.zip. When you unzip the file, you will find a folder called **Supermarket**. This folder includes:

1. **templates folder** - Template files for the application. **Do not modify these files.**
2. **itemsList.json** - Items JSON file.
3. **itemsListTwo.json** - Another items JSON file.
4. **supermarketServer.js** - Empty file where you will implement your server.
5. **Support files for node (e.g., package.json)**

Specifications/Requirements

Module Installations

The modules you need for this project are ejs and express. You don't need to use nodemon, but it helps the development process as nodemon restarts the server after any server code change.

Items Information

Information about the items a supermarket sells will be provided via a JSON file (e.g., itemsList.json). The "itemsList" property represent an array of the items the supermarket sells. The "itemsListName" property is the name of the list (you can ignore it as it will not be used in the project).

Command Line Interpreter

You should start developing your application by implementing the command line interpreter that is part of the supermarketServer.js file. Users will start the server by executing **node supermarketServer.js** followed by a file with JSON information about items sold by the supermarket. For example, to start the server with the data available in itemsList.json the user will

type on the command line: **node supermarketServer.js itemsList.json**
About the command line interpreter:

- The command line interpreter will use the prompt "Type itemsList or stop to shutdown the server: ".
- The server will stop when "stop" is entered. The message "Shutting down the server" should be displayed when the server is stopped. Use process.exit(0) to stop the server.
- The items in the JSON file loaded are displayed when "itemsList" is entered.
- "Invalid command: " followed by the command entered, is displayed for a command other than "stop" or "itemsList".
- Displays the message "Usage supermarketServer.js jsonFile" when an incorrect number of command line arguments are provided.

Definining and Processing Endpoints (path and specific HTTP method)

Using **express**, you will define endpoints that will generate the expected processing. Assumming **app** is the express request handler function, the endpoints you need to implement are:

1. app.get("/", (request, response) => { /* You implement */}); - This endpoint renders the main page of the application and it will display the contents of the index.ejs template file.
2. app.get("/catalog", (request, response) => { /* You implement */ }); - This endpoint displays the displayItems.ejs template with the table of items available.
3. app.get("/order", (request, response) => { /* You implement */ }); - This endpoint displays the placeOrder.ejs template with the table of items available.
4. app.post("/order", (request, response) => { /* You implement */ }); - This endpoint will process the submission of the placeOrder form, retrieving the order values and processing the order. Processing an order requires displaying the orderConfirmation.ejs template with a table that includes the items to be purchased, along with their cost. The last table row has the sum of all the items in the order.

Template files

We have provided the template files you need to use while processing endpoints. In those files, except for index.ejs, you will find variable(s) (e.g., <%- orderTable %>) that will be replaced with the appropriate contents. Class examples illustrate how to use the render method (e.g., response.render()) with an ejs file and an object that has values that will replace the ejs file variables. **Do not modify the provided template files.**

Format for Output Messages and Tables

Take a look at the application video to identify the format expected for output messages and tables.

Data Validity

You an assume the provided JSON files are correct. Also, you can assume users will provide the expected data (the application does not have to generate any error messages for invalid data provided by the user).

Port Number

Your server will use 5000 as the port number. The application main page will be accessed via the endpoint http://localhost:5000/

Miscellaneous

1. Check the Piazza clarifications for this project often.
2. Do not use fetch for this project.
3. Feel free to use any code presented in lecture.
4. You can used the method toFixed() (e.g., cost.toFixed(2)) to generate a currency value with two decimal places.
5. There is no need to use express.Router in this project.
6. To run nodemon in powershell (PC) you need to change execution permissions by:
    - Opening "Windows Powershell (Admin)"
    - Set-ExecutionPolicy Unrestricted
    - **Warning: previous step allows any script to execute**
7. If you need to change the port number (e.g., using port 5001), make sure you modify placeOrder (it'll try to use 5000 instead of 5001)
8. Truncate the sum using toFixed() after you have added the costs.
9. Do not worry if the user doesn't select any items in their order (you can assume they will always make a selection).
10. Make sure you have the most recent versions of Node.js and npm. Some students reported problems defining class while using old versions of Node.js/npm
11. **Please read the submission instructions below (they are not the usual ones).**

Submission

- **Remove the node_modules directory** from the Supermarket folder that has all the files for your project. We can install any modules based on your package.json file (just make sure it has the correct module dependencies).
- Compress the Supermarket folder.
- Upload the verified zip file to the submit server using the **Supermarket** entry.
- Verify your submission is correct by downloading your submission from the submit server and:
    - Extract your project.
    - run **npm install**
    - Verify you can run the application you were expected to develop.
    - **You will be penalize at least -20 pts for an incorrect submission.**

- **Carefully Read - As we don't have automatic testing performed by the submit server, we will only grade the last submission you provide. If you provide both an on time and a late submission, we will only grade the last submission and apply a 12 pts penalty.**

Web Accessibility