COMP 2406 B - Fall 2022
Tutorial #9

Due Sunday, December 4, 23:59

Objectives

- Add session support to an existing Express-based web application
- Implement a simple recommender system

Expectations

You need to submit solutions to all the problems in this tutorial. Our TAs will mark your submission. Remember to use the available resources (w3schools, Node.js, documentation, lecture materials, etc.) for more information if you are struggling to complete the problems.

Marking scheme. For each tutorial, you will receive:

- 2/2 for submitting high-quality solutions to all problems. “High-quality” means that your code works (solves the problem) and is also neat and concise (no overengineering, please).
- 1/2 for submitting solutions for all problems but some need improvement, or you are missing a problem.
- 0/2 if you are missing several problems or your solutions are poorly done; or you do not make a submission, or your submission cannot be executed.

Problem Background

This tutorial will involve adding session functionality to the existing store example we have seen in earlier lectures. To start, download the **T09-Base-Code.zip** file, extract the contents, and run **npm install** to install the dependencies. The provided code is a completed version of the store database that uses Express. Take a few minutes to look through the **store-server.js** and **products-router.js** files to get an idea of how the server operates. The initial data for the server is read from the **products.json**, **reviews.json**, and **users.json** files. You should be able to run the **store-server.js** file with Node, access http://localhost:3000, and navigate through the store's data.

Problem 1 (Adding Session Data)

The first thing you will do in this tutorial is add session data into the application. This will allow you to track the actions of a client as they use the application. To start, you will need to install and use the Express session module. To install the module, use **npm install express-session**. You can then require the module in your server and configure your Express app to use the express-session module using:

```
const session = require('express-session');
app.use(session({
    secret: 'some secret key here',
    resave: true,
    saveUninitialized: true
}));
```

After you have done this, run the server and access the page. You should be able to see cookie data in the Chrome developer tools that shows a **`connect.sid`** property. You can also try logging the session data for each request in your server by adding the following basic middleware function before the router code:

```
app.use(function (req, res, next) {
    console.log(req.session);
    next();
});
```

Problem 2 (Using Session Data)

You now have an object on the server (**`req.session`**) that you can use to remember anything about the client making a request. For example, if you support a user logging in, you can remember the username of the client and whether they are currently logged in or not. This can then be used for authorization purposes. For this application, let's assume you want to remember the IDs of products the client has viewed while the session is active.

Modify the **`sendSingleProduct`** middleware function in the **products-router.js** file to update the **`req.session`** object so that it remembers a list of product IDs that have been viewed. For now, you can just log the list of the current session's products every time a request is made. Request some different products and make sure the list is updated correctly. **It can be difficult to simulate multiple different clients using a single computer. Opening an "incognito" browser window will allow you to establish a second, separate session.**

Problem 3 (Recommending Products)

Now that you know what products a client is viewing, you can update your server to make product recommendations. Update the **`sendSingleProduct`** handler and the **product.pug** template to provide a list of 5 recommended products to the user whenever they are viewing a product page. You can come up with whatever rules you want to use to select recommendations. For a simple solution – recommend products the user has previously viewed, if there are any. If you are looking for something more complex – try finding products that share common buyers with the current product. That is, if the client is viewing product X, find the 5 products that have been purchased the most by people who have already purchased X. The data required to do so is present within the application. You can access the products, users, and reviews from middleware using **`req.app.locals.products`**, **`req.app.locals.users`**, and **`req.app.locals.reviews`** respectively. The rest is just a matter of extracting the required data and computing the correct result. If you hate Pug and refuse to use it – sorry! You can print out the recommended products in the server console and call it a day.

Problem 4 (Save Your Work & Submit)

Once you have completed all the problems for this tutorial, **zip** (compress) all your files into a single **.zip** file and submit it to the Tutorial submission on Brightspace. Name your file **T9-YourName.zip**. Your dependencies must be installable via the **npm install** command. You should not include your **node_modules** folder in your submission. You must include a **README** file with detailed instructions on how to run your server and test your application. Make sure you download your .zip file and check its contents after submitting. If your .zip file is missing files or corrupt, you will lose marks.