Objectives

The first step in your journey towards the implementation of your application's back end is to begin thinking about the REST API endpoints that your Express server will support.

Instructions

1. Write out the routers you will need for your Express server, and what endpoints each will support, in a planning document (not code - just regular text).
   * This should at minimum include each path that your application will support (such as /campsites), any route parameters it will support, and the desired behavior in response to the most common HTTP methods (GET, PUT, POST, DELETE).
2. Set up a new subfolder for your portfolio project inside your main course folder (5-NodeJS-Express-MongoDB).
   * If using Git, initialize it as a Git repository and create a .**gitignore**file that contains "node\_modules".
   * Install **Express**into this folder (along with body-parser and morgan middleware libraries if you intend to use them).
   * Create a **package.json** file in this folder and fill it in appropriately.
   * Create a **routes** subfolder to contain your routers.
3. Using the planning document, you wrote in step 1, now create router module files inside the routes subfolder, and write the basic code for each router and REST API endpoint. As you did in the exercises this week, you do not have to write out the full implementation for each endpoint, but you can send a response body message that says what you plan to do for that endpoint.

**Answers:**

1. Below are my routes. I think this may need some editing, but this is my initial thoughts. I think I’m a bit confused if this is for the end user or a site administrator. End user would not be deleting most of these, but an administrator would.
   1. Home (GET)
   2. Shop (GET, POST, DELETE)
   3. Members (GET, POST, DELETE)
   4. About us (GET, POST, DELETE)
   5. FAQ (GET, POST, DELETE)
   6. Contact (GET, POST, DELETE)
   7. login (GET, POST, DELETE)
2. done – files viewable via git link.
3. done – files viewable via git link. I added routes for GET, POST, PUT, & DELETE on all of these; however, I returned comments/errors if it was a method that was not available.