Storefront Backend Project

Getting Started

This repo contains a basic Node and Express app to get you started in constructing an API. To get started, clone this repo and run `yarn` in your terminal at the project root.

Required Technologies

Your application must make use of the following libraries:

- Postgres for the database
- Node/Express for the application logic
- dotenv from npm for managing environment variables
- db-migrate from npm for migrations
- jsonwebtoken from npm for working with JWTs
- jasmine from npm for testing

Steps to Completion

1. Plan to Meet Requirements

In this repo there is a `REQUIREMENTS.md` document which outlines what this API needs to supply for the frontend, as well as the agreed upon data shapes to be passed between front and backend. This is much like a document you might come across in real life when building or extending an API.

Your first task is to read the requirements and update the document with the following:

- Determine the RESTful route for each endpoint listed. Add the RESTful route and HTTP verb to the document so that the frontend developer can begin to build their fetch requests.
  Example: A SHOW route: 'blogs/:id' [GET]

- Design the Postgres database tables based off the data shape requirements. Add to the requirements document the database tables and columns being sure to mark foreign keys.
  Example: You can format this however you like but these types of information should be provided
  Table: Books (id:varchar, title:varchar, author:varchar, published_year:varchar, publisher_id:string[foreign key to publishers table], pages:number)

NOTE It is important to remember that there might not be a one to one ratio between data shapes and database tables. Data shapes only outline the structure of objects being passed between frontend and API, the database may need multiple tables to store a single shape.

2. DB Creation and Migrations

Now that you have the structure of the databse outlined, it is time to create the database and migrations. Add the npm packages dotenv and db-migrate that we used in the course and setup your Postgres database. If you get stuck, you can always revisit the database lesson for a reminder.

You must also ensure that any sensitive information is hashed with bcrypt. If any passwords are found in plain text in your application it will not pass.

3. Models

Create the models for each database table. The methods in each model should map to the endpoints in `REQUIREMENTS.md`. Remember that these models should all have test suites and mocks.

4. Express Handlers

Set up the Express handlers to route incoming requests to the correct model method. Make sure that the endpoints you create match up with the enpoints listed in `REQUIREMENTS.md`. Endpoints must have tests and be CORS enabled.

5. JWTs

Add JWT functionality as shown in the course. Make sure that JWTs are required for the routes listed in `REQUIUREMENTS.md`.

6. QA and `README.md`

Before submitting, make sure that your project is complete with a `README.md`. Your `README.md` must include instructions for setting up and running your project including how you setup, run, and connect to your database.

Before submitting your project, spin it up and test each endpoint. If each one responds with data that matches the data shapes from the `REQUIREMENTS.md`, it is ready for submission!