13. Object-Relational Mapping (ORM) Challenge: E-commerce Back End

Start Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a text entry box or a website url

Your Task

Internet retail, also known as e-commerce, is the largest sector of the electronics industry, having generated an estimated US$29 trillion in 2017 (Source: United Nations Conference on Trade and Development). E-commerce platforms like Shopify and WooCommerce provide a suite of services to businesses of all sizes. Due to the prevalence of these platforms, developers should understand the fundamental architecture of e-commerce sites.

Your challenge is to build the back end for an e-commerce site. You'll take a working Express.js API and configure it to use Sequelize to interact with a MySQL database.

Because this application won't be deployed, you'll also need to create a walkthrough video that demonstrates its functionality and all of the following acceptance criteria being met. You'll need to submit a link to the video and add it to the README of your project.

Before you start, clone the starter code.

User Story

```
AS A manager at an internet retail company
I WANT a back end for my e-commerce website that uses the latest technologie
SO THAT my company can compete with other e-commerce companies
```

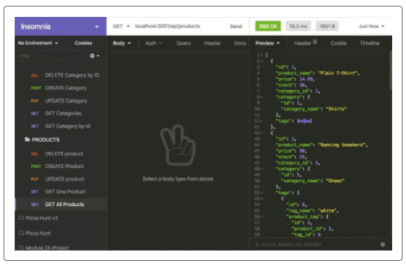
Acceptance Criteria

```
GIVEN a functional Express.js API
WHEN I add my database name, MySQL username, and MySQL password to an enviro
THEN I am able to connect to a database using Sequelize
WHEN I enter schema and seed commands
THEN a development database is created and is seeded with test data
WHEN I enter the command to invoke the application
THEN my server is started and the Sequelize models are synced to the MySQL d
WHEN I open API GET routes in Insomnia Core for categories, products, or tag
THEN the data for each of these routes is displayed in a formatted JSON
WHEN I test API POST, PUT, and DELETE routes in Insomnia Core
THEN I am able to successfully create, update, and delete data in my databas
```

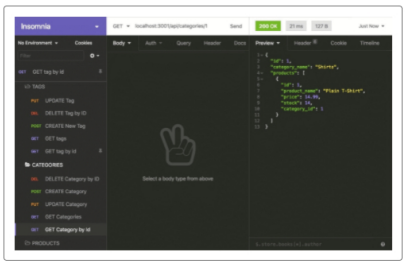
Mock-Up

The following animations show examples of the application's API routes being tested in Insomnia Core.

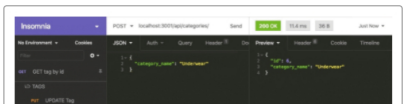
The first animation shows GET routes to return all categories, all products, and all tags being tested in Insomnia Core:



The second animation shows GET routes to return a single category, a single product, and a single tag being tested in Insomnia Core:



The final animation shows the POST, PUT, and DELETE routes for categories being tested in Insomnia Core:



Your walkthrough video should also show the POST, PUT, and DELETE routes for products and tags being tested in Insomnia Core.

Getting Started

You'll need to use the MySQL2 and Sequelize packages to connect your Express.js API to a MySQL database and the dotenv package to use environment variables to store sensitive data, like your MySQL username, password, and database name.

Use the `schema.sql` file in the `db` folder to create your database using MySQL shell commands. Use environment variables to store sensitive data, like your MySQL username, password, and database name.

Database Models

Your database should contain the following four models, including the requirements listed for each model:

- `Category`
  - `id`
  - Integer
  - Doesn't allow null values
  - Set as primary key
  - Uses auto increment
  - `category_name`
  - String
  - Doesn't allow null values
- `Product`
  - `id`
  - Integer
  - Doesn't allow null values
  - Set as primary key
  - Uses auto increment
  - `product_name`
  - String
  - Doesn't allow null values
  - `price`
  - Decimal
  - Doesn't allow null values
  - Validates that the value is a decimal
  - `stock`
  - Integer
  - Doesn't allow null values
  - Set a default value of 10
  - Validates that the value is numeric
  - `category_id`
  - Integer
  - References the `category` model's `id`
- `Tag`
  - `id`
  - Integer
  - Doesn't allow null values
  - Set as primary key
  - Uses auto increment
  - `tag_name`
  - String
- `ProductTag`
  - `id`
  - Integer

- Doesn't allow null values
- Set as primary key
- Uses auto increment
- `product_id`
- Integer
- References the `product` model's `id`
- `tag_id`
- Integer
- References the `tag` model's `id`

Associations

You'll need to execute association methods on your Sequelize models to create the following relationships between them:

- `Product` belongs to `Category`, as a category can have multiple products but a product can only belong to one category.
- `Category` has many `Product` models.
- `Product` belongs to many `Tag` models. Using the `ProductTag` through model, allow products to have multiple tags and tags to have many products.
- `Tag` belongs to many `Product` models.

HIDE HINT

Make sure you set up foreign key relationships that match the column we created in the respective models.

Fill Out the API Routes to Perform RESTful CRUD Operations

Fill out the unfinished routes in `product-routes.js`, `tag-routes.js`, and `category-routes.js` to perform create, read, update, and delete operations using your Sequelize models.

NOTE

> The functionality for creating the many-to-many relationship for products is already done for you.

SHOW HINT

Seed the Database

After creating the models and routes, run `npm run seed` to seed data to your database so that you can test your routes.

Sync Sequelize to the Database on Server Start

Create the code needed in `server.js` to sync the Sequelize models to the MySQL database on server start.

Grading Requirements

This Challenge is graded based on the following criteria:

Deliverables: 10%

- Your GitHub repository containing your application code.

Walkthrough Video: 37%

- A walkthrough video that demonstrates the functionality of the e-commerce back end must be submitted, and a link to the video should be included in your README file.
- The walkthrough video must show all of the technical acceptance criteria being met.
- The walkthrough video must demonstrate how to create the schema from the MySQL shell.
- The walkthrough video must demonstrate how to seed the database from the command line.
- The walkthrough video must demonstrate how to start the application's server.
- The walkthrough video must demonstrate GET routes for all categories, all products, and all tags being tested in Insomnia Core.
- The walkthrough video must demonstrate GET routes for a single category, a single product, and a single tag being tested in Insomnia Core.
- The walkthrough video must demonstrate POST, PUT, and DELETE routes for categories, products, and tags being tested in Insomnia Core.

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:
- Uses the MySQL2 and Sequelize packages to connect to a MySQL database.
- Uses the dotenv package to use environment variables to store sensitive data, like a user's MySQL username, password, and database name.
- Syncs Sequelize models to a MySQL database on the server start.
- Includes column definitions for all four models outlined in the Challenge instructions.
- Includes model associations outlined in the Challenge instructions.

Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.
- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
- Repository contains multiple descriptive commit messages.
- Repository contains a high-quality README with description and a link to a walkthrough video.

How to Submit the Challenge

You are required to submit BOTH of the following for review:

- A walkthrough video demonstrating the functionality of the application and all of the acceptance criteria being met.
- The URL of the GitHub repository. Give the repository a unique name and include a README describing the project.

NOTE

> You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Submit, then indicate you are skipping by typing "I choose to skip this assignment" in the text box.