Node.js, Express, MySQL Exercise

A store wants to use a RESTful Web Service that can

- Create/Retrieve/Update/Delete Products. The store has multiple products.
- Create/Retrieve/Update/Delete Reviews for products. A product has many reviews.

Build this API using Node.js + Express, along with MySQL as the database.

Models

The MySQL database for this application has 2 tables – one for **Products**, and one for **Reviews**. A product can have many reviews, but every review belongs to a single product.

Product Model specifications

- id: Primary key, Auto-generated
- name: String
- description: String
- price: Number
- rating: Number between 1 and 5
- urls: An array of image URLs (each URL is a String)

Review Model specifications

- id: Primary key, Auto-generated
- reviewer: String [required]
- rating: Integer between 1 and 5 [required]
- title: String
- text: String [required]
- createdAt: Date, set automatically to the date review was created on [required] - this can be set via DB schema for the table, or in code when adding a review.

API

The API has endpoints for working with products and reviews. The specifications are listed below. A sample implementation of this server can be found at
https://awesome-store-server.herokuapp.com/

Note: This sample implementation was created using a third-party library and its API and schema may not exactly be as per the specs here – this is provided only as a reference.

+91-9448441478
www.digdeeper.in
puranik@digdeeper.in

Products Resource

Action	HTTP Verb (Method)	URL	What is sent in request	HTTP Status Code (success)	What is received in response (success)
Retrieve all products	GET	http://localhost:4201/products	-	200	Array of products
Retrieve product with given id	GET	http://localhost:4201/products/1	-	200	Object with details of product with id = 1
Create a new product	POST	http://localhost:4201/products	Object with details of new product	201	Object with details of new product (incl. id)
Update details of product with given id	PUT	http://localhost:4201/products/1	Object with new details for product with id = 1	200	Object with new details for product with id = 1
Delete product with given id	DELETE	http://localhost:4201/products/1	-	204	-

Reviews Resource

Action	HTTP Verb (Method)	URL	What is sent in request	HTTP Status Code (success)	What is received in response (success)
Retrieve reviews for product with given id	GET	http://localhost:4201/products/1/reviews	-	200	Array of reviews for product with id = 1
Create a new review for product with given id	POST	http://localhost:4201/products/1/reviews	Object with details of new review (includes productid)	201	Object with details of new review (incl. id)

Note:
The tables show response codes in case of success only. The API will need to handle all error cases appropriately. Some of the popular codes in case of error are.

1. 400 – Bad Request. The data sent in the request was malformed.
2. 403 – Not Authorized
3. 404 – Resource not found
4. 409 – Duplicate resource exists and hence new resource was not created
5. 500 – Internal server error