# 6.NodeJS Assignment + Notes —-Normalization, Associations, GoCardless Payment Integration

CENTRALOGIC

12/5/24

> 💡 **Github code reference using modular structure**
>
> https://github.com/gauravwani127/NODEJS_training/tree/main/5NormalizationInSequelize

## Assignment

### Objective:

Develop a RESTful API for a bookstore with features including user authentication, book reviews, ratings, external book information integration, and a payment system. This will challenge students to implement secure payment handling, manage complex relationships, and integrate with external services.

### Scenario:

You are hired as a backend developer for a new online bookstore. The bookstore wants to provide detailed book information, user reviews, ratings, and a secure payment system for purchasing books. Your task is to create an advanced RESTful API to meet these requirements.

**Design schemas for Books, Authors, Users, Reviews, Ratings, and Orders.**

A Book should have the following properties:
**id, bookCode, title, description, publishedYear, price, authors, externalId.**
An Author should have the following properties:
**id, name, bio, birthdate, isSystemUser**
A User should have the following properties:
**id, username, password, email.**
A Review should have the following properties:
**id, userId, bookId, content.**
A Rating should have the following properties:
**id, userId, bookId, rating.**
A Payment should have the following properties:
**id, userId, bookId, amount, status, createdAt.**

- **API Endpoints:**
  - **Books:**
    - `GET /books` - Retrieve a list of all books.
    - `GET /books/:id` - Retrieve a specific book by ID.
    - `POST /books` - Create a new book (Admin only).
    - `PUT /books/:id` - Update an existing book by ID (Admin only).
    - `DELETE /books/:id` - Delete a book by ID (Admin only).
  - **Authors:**
    - `GET /authors` - Retrieve a list of all authors.
    - `GET /authors/:id` - Retrieve a specific author by ID.
    - `POST /authors` - Create a new author (Admin only).
    - `PUT /authors/:id` - Update an existing author by ID (Admin only).
    - `DELETE /authors/:id` - Delete an author by ID (Admin only).
  - **Users:**
    - `POST /register` - Register a new user.
    - `POST /login` - User login.
    - `GET /users/me` - Get current user details (Authenticated users only).
  - **Reviews:**
    - `GET /books/:bookId/reviews` - Get reviews for a book.
    - `POST /books/:bookId/reviews` - Add a review for a book (Authenticated users only).
    - `DELETE /reviews/:id` - Delete a review by ID (Admin or review author only).
  - **Ratings:**
    - `GET /books/:bookId/ratings` - Get ratings for a book.
    - `POST /books/:bookId/ratings` - Add a rating for a book (Authenticated users only).
  - **Payment:**
    - `POST /orders` - Create a new order (Authenticated users only using role of user).
    - `GET /orders/:id` - Retrieve a specific order by ID (Authenticated users only using role of user).

- **Relationships:**
  - Ensure that books can have multiple authors and that an author can write multiple books.
  - When fetching a book, include author details, reviews, and average rating in the response.
  - When fetching an author, include a list of books they have written.
  - Ensure orders are linked to users and books.
- **User Authentication:**
  - Implement JWT-based authentication.
  - Protect routes that require authentication.
  - Use bcrypt for hashing passwords.
- **External API Integration(Enhancement)**
  - Integrate with an external book information service (**e.g., Google Books API**) to fetch additional book details.
- **Payment Integration:**
  - Integrate with GoCardless for payment processing.
  - Implement secure payment handling and create orders upon successful payment.
  - Ensure sensitive data is handled securely.

Exploratory Questions (Just evaluation) ——————————————————————→

1. Like we have covered , one to one and one to many, Is there something like Many to many case. Can we use Junction Table here?
2. Do we need migration if we want to change associations among tables?
3. How goCardless exceptions are handled?
4. How webhooks can be utilized? Can they be setup on localMachine?
5. Is there compulsion if you are using hasMany , then you need to make use of  belongsTo? Like In Contract-Account case? What if  we dont use ?

Notes ——————————————————————→

**Normalization in Sequelize (Check in  the above git repository for real life example)**

## 1. One-to-One Association

In a one-to-one relationship, one record in a table is associated with one and only one record in another table.

Example: Let's consider a Contract and a Account model where each Contract has one Account.

```
Contract.belongsTo(Account, { foreignKey: 'accountUId' });
Account.belongsTo(Address, { foreignKey: 'addressUId' });
Account.belongsTo(BankDetails, { foreignKey: 'bankDetailsUId' });
```

## 2. One-to-Many Association

In a one-to-many relationship, one record in a table is associated with many records in another table.

Example: Let's consider an `Author` and a `Book` model where each author can write many books.

```
Account.hasMany(Contract, { foreignKey: 'accountUId' });
```

## 3. Many-to-Many Association

In a many-to-many relationship, many records in one table are associated with many records in another table. This is typically implemented using a junction table.

Example: Let's consider a `Student` and a `Course` model where students can enroll in many courses and each course can have many students.

```
Account.belongsToMany(Project, { through: AccountProject });
Project.belongsToMany(Account, { through: AccountProject });
```

**Use of GoCardless Payment Integration (Check repo for reference for example covered in the session )**

1. **Create a sandbox project** for GoCardless - https://manage-sandbox.gocardless.com/sign-up
2. **Create Access Token**: Generate a read-write access token through the Developer section of your sandbox dashboard.
3. Check out documentation for different apis (Customer Creation, Bank Account Creation,  Mandate creation , Payment Initiation apis)

https://developer.gocardless.com/api-reference/#core-endpoints-mandates

1. Check out what is webhooks , how you can set it up on local machine ?

***JavaScript basics***

I suggest solving atleast one question every day-

https://exercism.org/tracks/javascript

https://javascript.info/

https://learning.postman.com/docs/sending-requests/requests/

**How to submit assignment guidelines?**

**Theoretical questions are not to be submitted, Check documentations for finding answers to them**

**For 1. Coding question to be sent by pushing it into a public git respository . Exclude Node_Modules using gitignore**

Maintain a  diffeernt folder for Outputs , put here ss of the resposes of the apis given in the requirement

***Submit a detailed :***

- ***Screen capture (Record ) the flow of the Project , using debugger***
- ***Put the code in a public git repository and share it***
- ***Attach screenshots of the outputs***
- ***Use best practices as they were followed in last session (Modular structure)***

**Deadline: 12/6/2024**

**Note:** Feel free to reach out for any clarifications or assistance during the assignment

+

*Documentation is the key, Happy Learning---*

CentraLogic

***Gaurav Wani***

***Team Lead | CentraLogic Consultancy***