**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 11-07-2024 |
| Team ID | SWTID1719997659 |
| Project Name | Book-Nest |
| Maximum Marks |  |

**Project Title**: Book-Nest

**Date**: 11-07-2024

**Prepared by**: J. Marthand Bhargav, G. Aakash , G. Harshith and V.Nagendra

**Objective**

The objective of this report is to outline the database design and implementation details for the Book-Nest project, including schema design and database management system (DBMS) integration.

**Technologies Used**

**Database Management System (DBMS):** As the DBMS, MongoDB stores all your application's data, including books, user profiles, orders, and more. Its document-oriented nature allows for easy storage and retrieval of complex data structures without rigid schema requirements.

**Object-Document Mapper (ODM):** Acts as the ODM layer between your Node.js application and MongoDB. It simplifies interactions with MongoDB by providing a structured way to define schemas and models, manage relationships, and perform database operations using native MongoDB queries under the hood.

**Design the Database Schema**

The database schema is designed to accommodate the following entities and relationships:

**1. Users**

* **Attributes**:
  + \_id: ObjectId (automatically generated)
  + username: String (required)
  + email: String (required, unique)
  + password: String (required, hashed for security)
  + createdAt: Date (default: current timestamp)
  + updatedAt: Date (default: current timestamp, updated automatically)

**2. Books**

* **Attributes**:
  + \_id: ObjectId (automatically generated)
  + title: String (required)
  + author: String (required)
  + description: String
  + price: Number (required)
  + quantity: Number (default: 0)
  + createdAt: Date (default: current timestamp)
  + updatedAt: Date (default: current timestamp, updated automatically)

**3. Orders**

* **Attributes**:
  + \_id: ObjectId (automatically generated)
  + user: ObjectId (references User)
  + books: Array of Objects:
    - bookId: ObjectId (references Book)
    - quantity: Number
  + totalAmount: Number
  + createdAt: Date (default: current timestamp)
  + updatedAt: Date (default: current timestamp, updated automatically)

**Implement the Database using MongoDB**

The MongoDB database is implemented with the following collections and structures:

Database Name: mern-book-store

Here is an example for the following collection:

**1. Collection: users :**

{

"\_id": ObjectId("..."),

"username": "john\_doe",

"email": "john.doe@example.com",

"password": "$2a$10$...",

"createdAt": ISODate("2024-07-13T12:00:00Z"),

"updatedAt": ISODate("2024-07-13T12:00:00Z")

}

**2. Collection: books :**

{

"\_id": ObjectId("..."),

"title": "Book Title",

"author": "Author Name",

"description": "Description of the book.",

"price": 29.99,

"quantity": 100,

"createdAt": ISODate("2024-07-13T12:00:00Z"),

"updatedAt": ISODate("2024-07-13T12:00:00Z")

}

**3. Collection: orders :**

{

"\_id": ObjectId("..."),

"user": ObjectId("..."),

"books": [

{

"bookId": ObjectId("..."),

"quantity": 2

},

{

"bookId": ObjectId("..."),

"quantity": 1

}

],

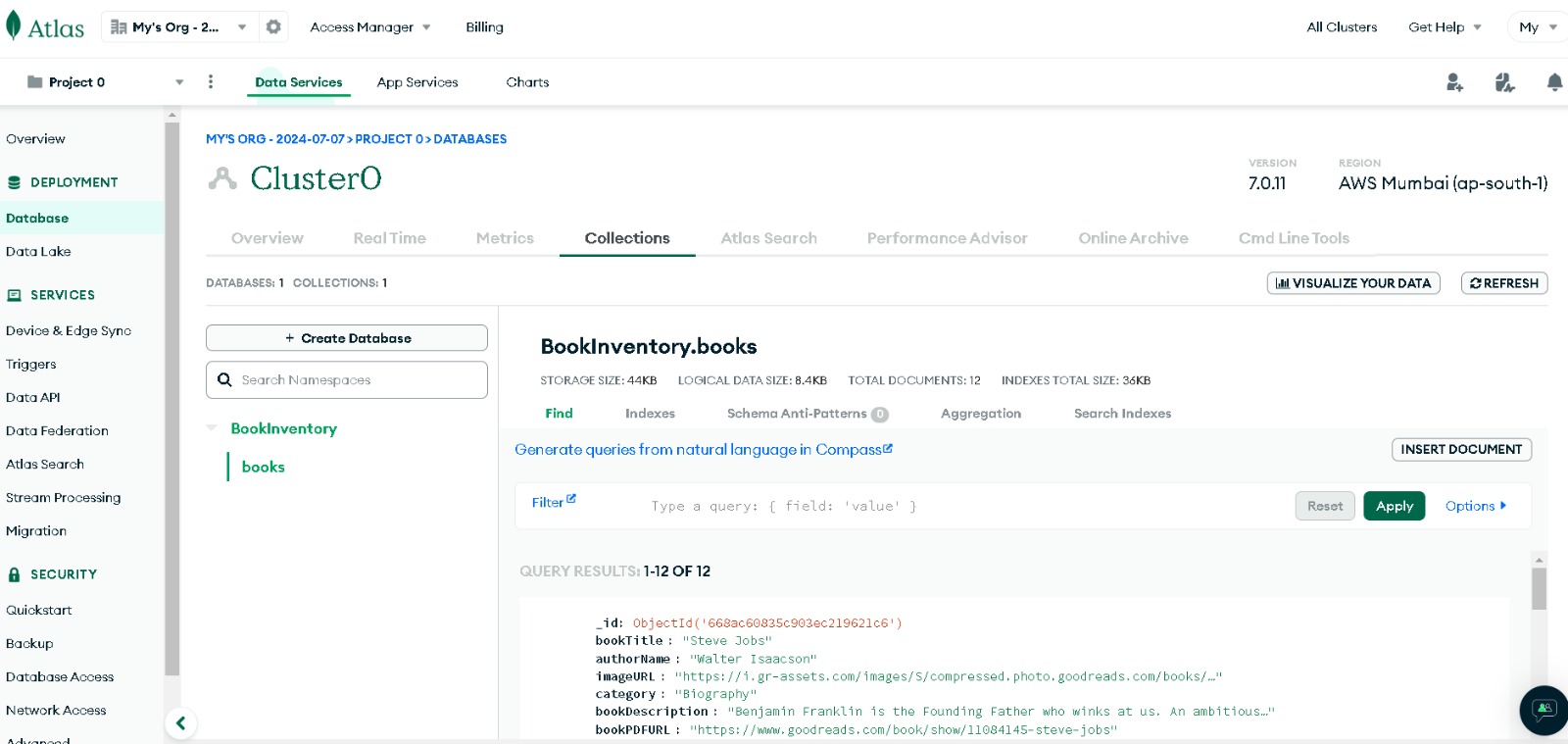
"totalAmount": 89.97,

"createdAt": ISODate("2024-07-13T12:00:00Z"),

"updatedAt": ISODate("2024-07-13T12:00:00Z")

}

**Integration with Backend**

****

* **Mongoose Models**: Each interaction involves using Mongoose models (User, Book, Comment) to interact with MongoDB collections (users, books, comments). These models define the schema, provide validation, and offer methods for querying and updating data.
* **Error Handling**: Error handling is implemented to catch and respond appropriately to errors that occur during database interactions, ensuring robustness and reliability of the API endpoints.
* **Middleware**: Middleware functions like authentication middleware (authMiddleware) are used to protect routes that require user authentication, ensuring secure access control to sensitive operations.

These interactions form the backbone of our backend API implementation, providing structured access to MongoDB data through Mongoose ODM while ensuring data integrity, security, and performance in your online bookstore application.