**Full Stack Development with MERN**

**API Development and Integration Report**

| Date | 18/07/2024 |
| --- | --- |
| Team ID | SWTID1720014974 |
| Project Name | Project – BookStore |
| Maximum Marks |  |

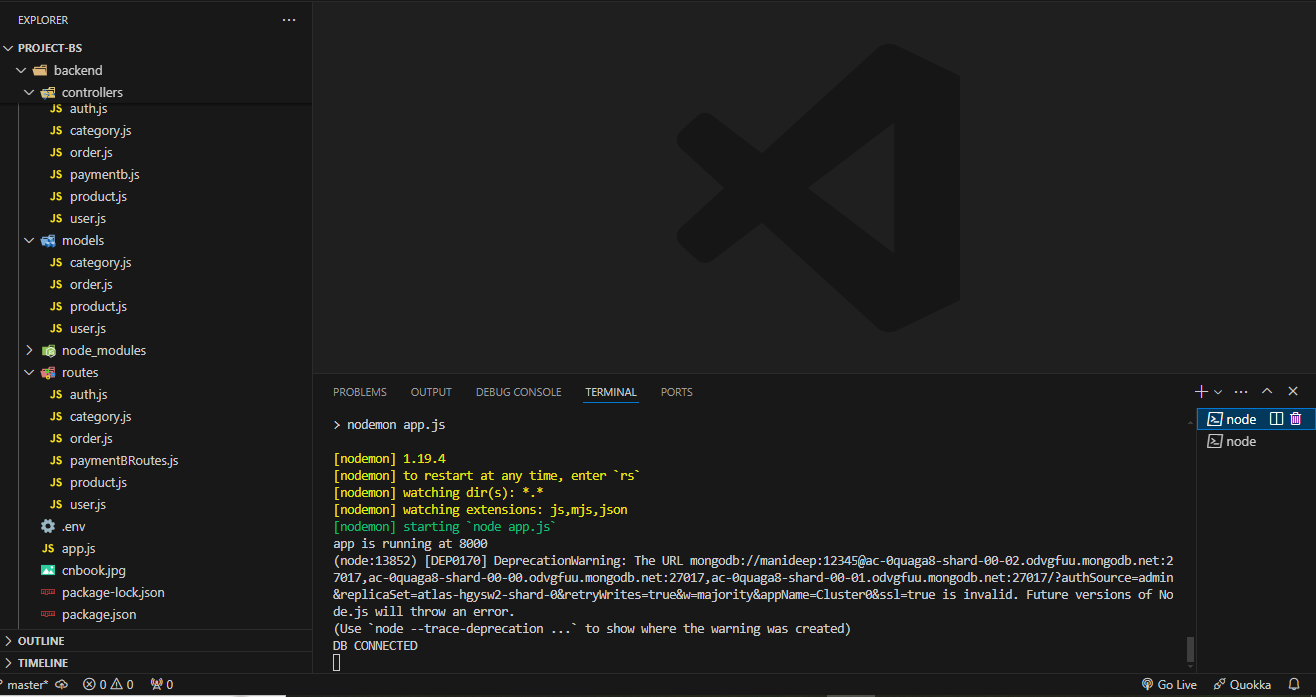
**Project Title:** BookNest  
**Date:** 18/07/2024  
**Prepared by:** Manideep , Pardhiv ,Shrinivasan M, Keerthi

**Objective**  
The objective of this report is to document the API development progress and key aspects of the backend services implementation for the BookStore project.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** JWT (JSON Web Tokens)

**Project Structure**  
Provide a screenshot of the backend project structure with explanations for key directories and files.



**Key Directories and Files:**

**/controllers:** Contains functions to handle requests and responses for various modules such as user, product, order, and payment.

**/models:** Includes Mongoose schemas and models for MongoDB collections.

**/routes:** Defines the API endpoints and links them to controller functions.

**/middlewares:** Custom middleware functions for request processing, authentication, and authorization.

**/config:** Configuration files for database connections, environment variables, etc.

**API Endpoints**

**User Authentication**

* **POST /api/signup**: Registers a new user with validation.
* **POST /api/signin**: Authenticates a user and returns a token.
* **GET /api/signout**: Signs out the user.

**User Management**

* **GET /api/user/**

: Retrieves user information by ID (requires authentication).

* **PUT /api/user/**

: Updates user information by ID (requires authentication).

* **GET /api/users**: Retrieves all users (requires admin authentication).
* **DELETE /api/user/deleteUser/**

: Deletes a user by ID (requires admin authentication).

**Product Management**

* **POST /api/product/create/**

: Creates a new product (requires admin authentication).

* **GET /api/product/**

: Retrieves product details by ID.

* **GET /api/product/photo/**

: Retrieves product photo by ID.

* **DELETE /api/product/**

**/**

: Deletes a product by ID (requires admin authentication).

* **PUT /api/product/**: Updates a product by ID (requires admin authentication).
* **GET /api/products**: Retrieves all products.
* **GET /api/products/categories**: Retrieves all unique product categories.

**Order Management**

* **POST /api/order/create/**

: Creates a new order (requires authentication).

* **GET /api/order/all/**

: Retrieves all orders for a user (requires admin authentication).

* **POST /api/order/**

: Retrieves orders for a user (requires authentication).

* **GET /api/order/status/**

: Retrieves order status (requires admin authentication).

* **PUT /api/order/**

**/status/**

: Updates order status (requires admin authentication).

**Payment Processing**

* **GET /api/payment/gettoken/**

: Retrieves payment token (requires authentication).

* **POST /api/payment/braintree/**

: Processes a payment (requires authentication).

**Integration with Frontend**

The backend communicates with the frontend via RESTful APIs. Key points of integration include:

* **User Authentication:** Tokens are passed between frontend and backend to handle authentication.
* **Data Fetching:** Frontend components make API calls to fetch necessary data for display and interaction.

## Error Handling and Validation

### Error Handling

Centralized error handling is implemented using middleware to catch and process errors uniformly.

### Validation

Input validation is performed using the express-validator library to ensure data integrity and security.

## Security Considerations

### Authentication

Secure token-based authentication is implemented using JWT to ensure that only authenticated users can access certain endpoints.

### Data Encryption

Sensitive data is encrypted both at rest and in transit to protect it from unauthorized access.