

ShoppYGlobe RESTful API - Project Documentation

Table of Contents

1. Overview
 2. Installation
 3. Technologies Used
 4. Project Structure
 5. Controllers
 6. Middlewares
 7. Models
 8. Routes
 9. Utils
 10. Server Configuration
 11. API Endpoints
 12. Key Features
 13. Testing
-

Overview

ShoppYGlobe is a backend RESTful API project built using Node.js, Express.js, and MongoDB (Mongoose). It provides core functionalities for an e-commerce application, including user authentication (JWT-based), product management (single and bulk), and a shopping cart system with complete CRUD operations.

GitHub Repository: https://github.com/SravanGunaganti/ShoppYGlobe_Backend.git

Installation

Clone the Repository

```
git clone https://github.com/SravanGunaganti/ShoppYGlobe_Backend.git
cd ShoppYGlobe_Backend
```

Install Dependencies

```
npm install
```

MongoDB Setup

Ensure MongoDB is running either locally or using MongoDB Atlas.

Environment Configuration

Create a .env file in the root directory and add the following fields:

```
PORT=5050
MONGO_URI=mongodb://127.0.0.1:27017/shoppyglobe
JWT_SECRET=your_jwt_secret
```

Start the Server

```
npm start
```

Technologies Used

- Node.js
 - Express.js
 - MongoDB + Mongoose
 - JWT for Authentication
 - Thunder Client for API Testing
-

Project Structure

```
├── ShoppyGlobe_Backend
│   ├── controllers
│   │   ├── cart.controller.js
│   │   ├── product.controller.js
│   │   └── user.controller.js
│   ├── middlewares
│   │   ├── errorHandler.js
│   │   ├── userValidation.js
│   │   ├── validateBulkProducts.js
│   │   ├── validateObjectId.js
│   │   ├── validateProduct.js
│   │   └── verifyToken.js
│   ├── models
│   │   ├── Cart.model.js
│   │   ├── Product.model.js
│   │   └── User.model.js
│   ├── routes
│   │   ├── cart.routes.js
│   │   ├── product.routes.js
│   │   └── user.routes.js
│   └── utils
│       ├── generateToken.js
│       ├── sendErrorResponse.js
│       ├── sendSuccessResponse.js
│       └── validators.js
```

```
|— .env
|— index.js
|— package-lock.json
|— package.json
|— products.json
|— README.md
|— Screenshots_Submission.pdf
|— Project Documentation.pdf
```

Controllers

- **user.controller.js**: Manages user registration and login.
- **product.controller.js**: Handles single and bulk product operations .
- **cart.controller.js**: Manages cart operations including add, increment,decrement and delete and clear cart.

Middlewares

- **errorHandlers.js**: Centralized error handling.
- **userValidation.js**: Validates user registration/login data.
- **validateBulkProducts.js**: Checks incoming bulk product data.
- **validateObjectId.js**: Validates MongoDB ObjectId in route parameters.
- **validateProduct.js**: Validates individual product data.
- **verifyToken.js**: JWT token verification middleware.

Models

- **User.model.js**: Schema for storing users with validation for name, email, and hashing the password pre saving.
- **Product.model.js**: Schema for products including name, description, price, stock.
- **Cart.model.js**: Stores userId of cart including an array of product references and quantities.

Routes

- **user.routes.js**: Registration and login routes.
- **product.routes.js**: Product CRUD and bulk operations.
- **cart.routes.js**: All cart-related API routes.

Utils

- **generateToken.js**: Generates JWT token.
- **sendErrorResponse.js**: Standardized error response formatter.
- **sendSuccessResponse.js**: Success response formatter.
- **validators.js**: Utility functions for validation and type checks.

Server Configuration

.env File

Holds sensitive configuration variables:

```
PORT=5050
MONGO_URI=mongodb://127.0.0.1:27017/shoppyglobe
JWT_SECRET=your_jwt_secret
```

index.js

- Connects to MongoDB
 - Applies global middlewares
 - Mounts all routes
 - Starts the Express server
-

API Endpoints

User Routes

- **POST /api/register** - Register a new user.
- **POST /api/login** - Login and receive a JWT token.

Product Routes

- **POST /api/products/bulk** - Bulk insert products (protected).
- **POST /api/product** - Add single product (protected).
- **GET /api/products** - Get all products.
- **GET /api/products/:id** - Get product by ID.
- **PUT /api/products/:id** - Update product by ID (protected).
- **DELETE /api/products/:id** - Delete product by ID (protected).

Cart Routes

- **POST /api/cart** - Add to cart or increment existing quantity.
 - **GET /api/cart** - Fetch Logged in user's cart.
 - **PUT /api/cart/increment/:productId** - Increase cart item quantity.
 - **PUT /api/cart/decrement/:productId** - decrease cart item quantity.
 - **DELETE /api/cart/:productId** - Remove product from cart.
 - **DELETE /api/cart** - Clear entire cart.
-

Key Features

Authentication

- JWT-based user authentication.
- Token stored in header: Authorization: `JWT <token>`.
- Protected routes validated via `verifyToken` middleware.

Product Management

- Single and bulk product creation.
- Product validation with error handling.
- Product update, retrieval, and deletion.

Cart Management

- Add/increment/decrement/delete cart items.
- Clear cart.

Middleware Validation

- Field-level validation for all inputs.
 - `ObjectId` checks.
 - Centralized error formatting.
-

Testing

All endpoints tested using Thunder Client:

You can test inserting products using data from `products.json` file

Positive Tests

- Successful user registration and login.
- Valid JWT for protected routes.
- Product creation and retrieval.
- Cart add/update/delete functionality.

Negative Tests

- Missing or invalid fields.
- Invalid JWT token.
- Non-existent endpoints or `ObjectIds`.

Test results are documented in the [Screenshots_Submission.pdf](#).