🌍 ShoppyGlobe E-commerce Backend API

A complete Node.js, Express, and MongoDB backend for the ShoppyGlobe e-commerce application with JWT authentication, cart functionality, and comprehensive testing.

🚀 Live Demo

- **Backend API**: https://shoppyglobe-nodejs-23-11-2025.vercel.app/
- **GitHub Repository**: https://github.com/JKS-sys/shoppyglobe-nodejs-23-11-2025

📋 Table of Contents

✨ Features

- 🔐 **User Authentication**: JWT-based registration and login system
- 📦 **Product Management**: Complete CRUD operations for products
- 🛒 **Shopping Cart**: Add, update, and remove items from cart
- 🛡️ **Input Validation**: Robust validation using custom middleware
- 🚨 **Error Handling**: Comprehensive error handling and validation
- 🔒 **Security**: Password hashing with bcryptjs, protected routes
- 🧪 **Testing**: Complete test suite with Jest and Supertest (100% passing)
- 📊 **MongoDB Integration**: Efficient data storage with Mongoose ODM

🛠️ Technology Stack

Layer	Technology
Backend Framework	Node.js, Express.js

Layer	Technology
Database	MongoDB with Mongoose ODM
Authentication	JWT (JSON Web Tokens)
Password Hashing	bcryptjs
Validation	Custom validation middleware
Testing	Jest, Supertest
Environment	dotenv
Development	Nodemon

📊 API Endpoints

🔐 Authentication Endpoints

Method	Endpoint	Description	Auth Required	Status
POST	/auth/register	Register a new user	❌	✅ Implemented
POST	/auth/login	Login user and get JWT token	❌	✅ Implemented

📦 Product Endpoints

Method	Endpoint	Description	Auth Required	Status
GET	/products	Get all products	❌	✅ Implemented
GET	/products/:id	Get single product by ID	❌	✅ Implemented

🛒 Cart Endpoints

Method	Endpoint	Description	Auth Required	Status
GET	/cart	Get user's cart	✅	✅ Implemented
POST	/cart	Add product to cart	✅	✅ Implemented
PUT	/cart/:productId	Update cart item quantity	✅	✅ Implemented
DELETE	/cart/:productId	Remove item from cart	✅	✅ Implemented

ℹ️ Info Endpoint

Method	Endpoint	Description	Auth Required	Status
GET	/	API information and documentation	❌	✅ Implemented

🚀 Installation & Setup

Prerequisites

- Node.js (v18 or higher)
- MongoDB (local installation or MongoDB Atlas)
- npm or yarn package manager

1. Clone the Repository

```
git clone https://github.com/JKS-sys/shoppyglobe-nodejs-23-11-2025.git
cd shoppyglobe-nodejs-23-11-2025
```

2. Install Dependencies

```
npm install
```

3. Environment Configuration

Create a `.env` file in the root directory:

```
MONGODB_URI=mongodb://127.0.0.1:27017/shoppyglobe
MONGODB_URI_TEST=mongodb://127.0.0.1:27017/shoppyglobe_test
JWT_SECRET=your_super_secure_jwt_secret_key_here_make_it_long_and_secure
PORT=5002
NODE_ENV=development
```

4. Database Setup

```
# Start MongoDB service (if using local MongoDB)
brew services start mongodb-community

# Seed the database with sample products
npm run seed
```

5. Start the Application

```
# Development mode with auto-restart
npm run dev

# Production mode
npm start
```

The server will start on `http://localhost:5002`

⚙️ Environment Variables

Variable	Description	Default Value
`MONGODB_URI`	MongoDB connection string for development	`mongodb://127.0.0.1:27017/shoppyglobe`
`MONGODB_URI_TEST`	MongoDB connection string for testing	`mongodb://127.0.0.1:27017/shoppyglobe_test`
`JWT_SECRET`	Secret key for JWT token generation	Required (no default)
`PORT`	Server port number	`5002`
`NODE_ENV`	Application environment	`development`

🗃️ Database Models

User Model

```
{
  username: { type: String, required: true, unique: true, minlength: 3,
maxlength: 30 },
  email: { type: String, required: true, unique: true, lowercase: true },
  password: { type: String, required: true, minlength: 6 },
  timestamps: true
}
```

Product Model

```
{
  name: { type: String, required: true, maxlength: 100 },
  price: { type: Number, required: true, min: 0 },
  description: { type: String, maxlength: 1000 },
  stockQuantity: { type: Number, required: true, min: 0 },
  timestamps: true
}
```

Cart Model

```
{
  user: { type: ObjectId, ref: 'User', required: true, unique: true },
  items: [{
```

```
    product: { type: ObjectId, ref: 'Product', required: true },
    quantity: { type: Number, required: true, min: 1 }
  }],
  totalAmount: { type: Number, default: 0 },
  timestamps: true
}
```

🧪 Testing

Test Results

✅ **All 13 tests passing** - Complete test coverage including authentication, products, cart operations, and error handling.

Run Test Suite

```
# Run all tests
npm test

# Run tests in watch mode
npm run test:watch

# Run tests with coverage
npm run test:coverage
```

Test Coverage

The comprehensive test suite covers:

- ✅ User registration and login
- ✅ Product retrieval (all and single)
- ✅ Cart operations (add, update, remove)
- ✅ Authentication middleware
- ✅ Error handling and validation
- ✅ Protected route access
- ✅ 404 handling

📖 API Documentation

🔐 Authentication

Register a New User

```
POST /auth/register
Content-Type: application/json

{
```

```
  "username": "john_doe",
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```
{
  "success": true,
  "message": "User registered successfully",
  "data": {
    "user": {
      "id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

User Login

```
POST /auth/login
Content-Type: application/json

{
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

📦 Products

Get All Products

```
GET /products
```

Response:

```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "_id": "6925905e36893c0357e2a0cc",
      "name": "Apple iPhone 15",
      "price": 999.99,
      "description": "Latest Apple iPhone with A17 chip.",
      "stockQuantity": 50
    }
  ]
}
```

Get Single Product

```
GET /products/6925905e36893c0357e2a0cc
```

🛒 Cart Operations (Protected)

Add to Cart

```
POST /cart
Authorization: Bearer <your_jwt_token>
Content-Type: application/json

{
  "productId": "6925905e36893c0357e2a0cc",
  "quantity": 2
}
```

Update Cart Item

```
PUT /cart/6925905e36893c0357e2a0cc
Authorization: Bearer <your_jwt_token>
Content-Type: application/json

{
  "quantity": 3
}
```

Remove from Cart

```
DELETE /cart/6925905e36893c0357e2a0cc
Authorization: Bearer <your_jwt_token>
```

ℹ️ API Information

```
GET /
```

🚀 Deployment

Vercel Deployment

The application is deployed on Vercel. The deployment process includes:

1. **Automatic deployments** from the main branch
2. **Environment variables** configured in Vercel dashboard
3. **MongoDB Atlas** for production database
4. **CORS configuration** for frontend integration

Local Production Build

```
# Install dependencies
npm install

# Set production environment variables
export NODE_ENV=production
export MONGODB_URI=your_production_mongodb_uri
export JWT_SECRET=your_production_jwt_secret

# Start production server
npm start
```

📸 Screenshots

MongoDB Database

- Database Collections
- Products Collection
- Users Collection
- Carts Collection

API Testing

- Thunder Client Tests
- Test Results

🛡️ Security Features

- **Password Hashing**: bcryptjs with salt rounds
- **JWT Tokens**: Secure authentication with configurable expiry
- **Input Validation**: Custom validation middleware for all endpoints
- **Protected Routes**: Authentication middleware for sensitive operations
- **Error Handling**: Secure error messages without exposing sensitive data
- **CORS**: Configured for cross-origin requests

🔧 Scripts

Script	Description
`npm start`	Start production server
`npm run dev`	Start development server with nodemon
`npm run seed`	Seed database with sample products
`npm test`	Run test suite
`npm run test:watch`	Run tests in watch mode
`npm run test:coverage`	Run tests with coverage report

🐛 Troubleshooting

Common Issues

1. **MongoDB Connection Error**

```
# Ensure MongoDB is running
brew services start mongodb-community

# Check connection string in .env file
```

2. **JWT Authentication Errors**
    - Verify JWT_SECRET is set in environment variables

- Check token in Authorization header format

3. **Validation Errors**
   - Review request body format
   - Check required fields and data types
4. **Test Failures**

```
# Clear test database
npm run test -- --detectOpenHandles --forceExit
```

Getting Help

- Check console for detailed error messages
- Review API documentation for expected request formats
- Check test cases for implementation examples

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

📄 License

This project is proprietary and confidential. Unauthorized copying, modification, distribution, or use of this project in any form without explicit permission from the author is strictly prohibited.

👥 Contributors

- **JKS** - Initial development and implementation

📞 Support

For support, email jagadeesh_2k17@proton.me or create an issue in the GitHub repository.

🎉 Happy Coding with ShoppyGlobe!

Build something amazing with our robust e-commerce API!


```

## 🎯 Project Structure Summary
```

```
shoppyglobe-nodejs-23-11-2025/
├── 📁 config/                  # Database configuration
├── 📁 middleware/             # Custom middleware
│   ├── auth.js               # JWT authentication
│   ├── validation.js         # Zod validation
│   └── validation-simple.js # Custom validation
├── 📁 models/                # MongoDB models
│   ├── User.js
│   ├── Product.js
│   └── Cart.js
├── 📁 routes/                # API routes
│   ├── auth.js
│   ├── products.js
│   └── cart.js
├── 📁 screenshots/           # Documentation screenshots
│   ├── MongoDB Database/
│   ├── get/
│   ├── post/
│   ├── put/
│   └── delete/
├── 📁 tests/                 # Test files
│   └── shoppyglobe.test.js
├── 📄 server.js              # Main application file
├── 📄 seedProducts.js        # Database seeder
├── 📄 package.json
├── 📄 .env
└── 📄 README.md
```

✅ Project Status

- ✅ **Backend API**: Complete and fully functional
- ✅ **Authentication**: JWT-based auth implemented
- ✅ **Database**: MongoDB with proper models
- ✅ **Testing**: 13/13 tests passing (100%)
- ✅ **Documentation**: Comprehensive README
- ✅ **Deployment**: Live on Vercel
- ✅ **Validation**: Input validation implemented
- ✅ **Error Handling**: Comprehensive error handling

ShoppyGlobe API is now production-ready with complete documentation! 🚀