Ecom API Project — Setup and Usage Documentation

1. Project Overview

This project is a backend API for an e-commerce system built with Laravel.
Features include:

- Product management (CRUD)
- Shopping cart operations
- Checkout & payment integration with Razorpay
- Order management (listing, viewing, status update)
- Payment verification

---

2. Project Setup Instructions

Prerequisites:

- PHP >= 8.0 installed
- Composer installed
- MySQL or compatible database installed
- Git (optional)
- Postman (for API testing)

Step 1: Clone or download the project

```
bash
Copy code
git clone https://github.com/your-repo/ecom-api.git
cd ecom-api
```

Or download and extract ZIP file.

Step 2: Install dependencies

Run composer install:

```
bash
```

```
Copy code
composer install
```

Step 3: Environment configuration

Copy `.env.example` to `.env`:

```
bash
Copy code
cp .env.example .env
```

Edit `.env` file and set your database and Razorpay credentials:

```
env
Copy code
APP_NAME=EcomAPI
APP_URL=http://127.0.0.1:8000

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your_database_name
DB_USERNAME=your_db_username
DB_PASSWORD=your_db_password

RAZORPAY_KEY=rzp_test_yourkey
RAZORPAY_SECRET=your_secret_key
```

Step 4: Generate application key

```
bash
Copy code
php artisan key:generate
```

Step 5: Run migrations

Create the database first in MySQL.

Then run:

```
bash
Copy code
php artisan migrate
```

This will create necessary tables (`products`, `cart`, `orders`, `order_items`, etc.).

Step 6: Seed initial data (optional)

If you have seeders for products or users, run:

```
bash
Copy code
php artisan db:seed
```

Step 7: Start Laravel server

```
bash
Copy code
php artisan serve
```

By default, the app will run at:

```
cpp
Copy code
http://127.0.0.1:8000
```

---

3. API Endpoints Overview

All API routes are prefixed with `/api/v1`

Products

Method	Endpoint	Description
GET	`/products`	List all products
POST	`/products`	Create a new product
GET	`/products/{id}`	Get product details
POST	`/products/{id}`	Update product
DELETE	`/products/{id}`	Delete product

Cart

Method	Endpoint	Description
POST	`/cart`	Add item to cart
GET	`/cart`	List cart items
PUT	`/cart/{cartItem}`	Update cart item quantity
DELETE	`/cart/{cartItem}`	Delete cart item

Checkout & Payment

Method	Endpoint	Description
POST	`/checkout`	Create Razorpay order & get payment details
POST	`/razorpay/verify`	Verify Razorpay payment & complete order

Orders (Admin)

Method	Endpoint	Description
GET	`/orders`	List all orders
GET	`/orders/{order}`	View single order details
PUT	`/orders/{order}/status`	Update order status

---

4. How to Test APIs Locally

- Use Postman application (download from https://www.postman.com/downloads/)
- Set base URL: `http://127.0.0.1:8000/api/v1`
- Test endpoints by choosing appropriate HTTP methods (GET, POST, PUT, DELETE)
- For POST/PUT requests, set **Body** as JSON with the required parameters.
- Example:
  To add a product:
  - POST `/products`
  - Body (raw JSON):

```
json
Copy code
{
  "name": "Sample Product",
  "description": "Product description",
  "price": 1000,
  "stock": 10
}
```

---

5. Notes

- User ID is hardcoded as `1` for testing purposes (replace with real user authentication later).
- Razorpay keys must be valid for the payment to work.
- On successful payment verification, cart is cleared and order items saved.
- Order status can be updated using admin endpoints.

---

6. Troubleshooting

- Ensure `.env` is configured correctly.
- Run `php artisan config:clear` if env variables are not reflecting.
- Check database connection.
- Check logs in `storage/logs/laravel.log` for errors.

Enjoy building your e-commerce backend API!