

E-Commerce API Setup Guide

Prerequisites

- Python 3.8+
- PostgreSQL
- Redis

Installation Steps

1. Install Dependencies

```
bash  
pip install -r requirements.txt
```

2. Fix requirements.txt

Update line in requirements.txt:

```
python-decouple==3.8
```

3. Database Setup

Make sure PostgreSQL is running and create the database:

```
bash  
createdb ecommerce_db
```

4. Run Migrations

```
bash  
python manage.py makemigrations  
python manage.py migrate
```

5. Create Superuser

```
bash  
python manage.py createsuperuser
```

Enter email, password, first name, and last name when prompted.

6. Run the Server

```
bash
```

```
python manage.py runserver
```

7. Start Redis (in separate terminal)

```
bash
```

```
redis-server
```

8. Start Celery Worker (in separate terminal)

```
bash
```

```
celery -A ecommerce_api worker -l info
```

API Endpoints

Authentication Endpoints

Register a New User

```
http
```

```
POST /api/users/register/
```

Content-Type: application/json

```
{
  "email": "customer@example.com",
  "password": "SecurePass123!",
  "password2": "SecurePass123!",
  "first_name": "John",
  "last_name": "Doe",
  "phone": "+1234567890",
  "role": "customer"
}
```

Login

http

POST /api/users/login/

Content-Type: application/json

```
{  
  "email": "customer@example.com",  
  "password": "SecurePass123!"  
}
```

Get User Profile

http

GET /api/users/profile/

Authorization: Bearer <access_token>

Update User Profile

http

PATCH /api/users/profile/

Authorization: Bearer <access_token>

Content-Type: application/json

```
{  
  "first_name": "Jane",  
  "phone": "+0987654321"  
}
```

Change Password

http

POST /api/users/change-password/

Authorization: Bearer <access_token>

Content-Type: application/json

```
{  
  "old_password": "SecurePass123!",  
  "new_password": "NewSecurePass456!",  
  "new_password2": "NewSecurePass456!"  
}
```

Refresh Token

http

POST /api/users/token/refresh/

Content-Type: application/json

```
{  
  "refresh": "<refresh_token>"  
}
```

Logout

http

POST /api/users/logout/

Authorization: Bearer <access_token>

Content-Type: application/json

```
{  
  "refresh_token": "<refresh_token>"  
}
```

Product Endpoints

List All Categories

http

GET /api/products/categories/

Create Category (Admin only)

http

POST /api/products/categories/

Authorization: Bearer <admin_access_token>

Content-Type: application/json

```
{  
  "name": "Electronics",  
  "description": "Electronic devices and accessories"  
}
```

Get Category Details

http

GET /api/products/categories/<slug>/

List All Products

http

GET /api/products/

Query parameters:

- `(category)`: Filter by category ID
- `(is_featured)`: Filter featured products (true/false)
- `(min_price)`: Minimum price
- `(max_price)`: Maximum price
- `(in_stock)`: Show only in-stock items (true/false)
- `(search)`: Search by name or description
- `(ordering)`: Sort by price, created_at, name (use - for descending)

Examples:

http

GET /api/products/?category=1&min_price=100&max_price=500

GET /api/products/?search=laptop&ordering=-price

GET /api/products/?is_featured=true

Create Product (Admin only)

http

POST /api/products/
Authorization: Bearer <admin_access_token>
Content-Type: multipart/form-data

```
{  
  "name": "Laptop",  
  "description": "High-performance laptop",  
  "category": 1,  
  "price": "999.99",  
  "stock": 50,  
  "is_active": true,  
  "is_featured": true,  
  "image": <file>  
}
```

Get Product Details

http

GET /api/products/<slug>/

Update Product (Admin only)

http

PATCH /api/products/<slug>/
Authorization: Bearer <admin_access_token>
Content-Type: application/json

```
{  
  "price": "899.99",  
  "stock": 45  
}
```

Delete Product (Admin only)

http

DELETE /api/products/<slug>/
Authorization: Bearer <admin_access_token>

Search Products

http

GET /api/products/search/?q=laptop

Get Featured Products

http

GET /api/products/featured/

Upload Product Images (Admin only)

http

POST /api/products/<product_id>/images/

Authorization: Bearer <admin_access_token>

Content-Type: multipart/form-data

```
{  
  "image": <file>,  
  "alt_text": "Product image",  
  "is_primary": false  
}
```

Review Endpoints

List Product Reviews

http

GET /api/products/<product_id>/reviews/

Create Review (Authenticated users)

http

POST /api/products/<product_id>/reviews/

Authorization: Bearer <access_token>

Content-Type: application/json

```
{  
    "rating": 5,  
    "comment": "Excellent product!"  
}
```

Update Review

http

PATCH /api/products/reviews/<review_id>/

Authorization: Bearer <access_token>

Content-Type: application/json

```
{  
    "rating": 4,  
    "comment": "Good product, updated review"  
}
```

Delete Review

http

DELETE /api/products/reviews/<review_id>/

Authorization: Bearer <access_token>

Admin Endpoints

List All Users (Admin only)

http

GET /api/users/

Authorization: Bearer <admin_access_token>

Get User Details (Admin only)

http

GET /api/users/<user_id>/
Authorization: Bearer <admin_access_token>

Update User (Admin only)

http

PATCH /api/users/<user_id>/
Authorization: Bearer <admin_access_token>
Content-Type: application/json

```
{  
  "is_active": false,  
  "role": "admin"  
}
```

Testing with Postman/Thunder Client

1. Register a new user

- Use the register endpoint
- Save the access token

2. Login

- Use the login endpoint
- Save both access and refresh tokens

3. Create some categories (as admin)

- Login with superuser credentials
- Create categories like "Electronics", "Clothing", "Books"

4. Create products (as admin)

- Create products under different categories
- Upload product images

5. Test as customer

- Login as regular user
- Browse products
- Add reviews

6. Test filtering and search

- Try different query parameters
 - Test pagination
-

Common Issues & Solutions

Issue: Migration errors

Solution: Delete all migration files (except `__init__.py`) and run:

```
bash  
  
python manage.py makemigrations  
python manage.py migrate
```

Issue: Redis connection error

Solution: Make sure Redis is running:

```
bash  
  
redis-server
```

Issue: Database connection error

Solution: Check PostgreSQL is running and credentials in `.env` are correct

Issue: Permission denied errors

Solution: Make sure you're using the correct access token and user has proper role

Next Steps

You've now completed:

- User Authentication System (JWT-based)
- Role-Based Access Control (Admin, Customer)
- Product Management (CRUD operations with categories)
- Product Reviews
- Redis Caching (Product detail view)

Still to implement:

- Shopping Cart
- Order Management
- Payment Integration (Stripe)
- Celery Tasks (Email notifications)
- API Documentation

Would you like to continue with the Shopping Cart implementation next?