Here’s a **ready-made** README.md **draft** for your **Nexora Backend Project**.

# 🛒 Nexora – E-commerce Backend (Python + Django + DRF + MySQL)

## 📌 **Overview**

**Nexora** is a complete **backend-only e-commerce system** built with **Python, Django, Django REST Framework (DRF), and MySQL**.  
It provides **role-based access** (Admin, Seller, Customer) and covers all key features of a real-world e-commerce platform, including authentication, product management, cart, checkout, orders, payments, wishlist, and coupons.

This project demonstrates **backend developer skills** in **API design, database modeling, authentication, and deployment**.

## 🚀 Features

### 🔑 Authentication & Users

* JWT-based authentication (djangorestframework-simplejwt)
* Role-based access (**Admin, Seller, Customer**)
* User profile management (update email, phone, address)

### 📦 Catalog Management

* Products, Categories, Brands (CRUD APIs)
* Search & filter products
* Role restrictions: Admin/Seller manage catalog, Customers read-only

### 🛒 Cart & Checkout

* Add, update, and remove cart items
* Checkout API → generates Order
* Auto stock deduction after purchase

### 📜 Orders & Payments

* Order workflow: **Pending → Paid → Shipped → Delivered**
* Payment API (simulated Razorpay/UPI/Stripe)
* Transaction logging with unique IDs
* Customer order history
* Admin order status management

### ❤️ Wishlist & Coupons

* Customers can manage wishlist
* Admin can create coupons (discount %)
* Apply coupon logic during checkout

### 🛡 Extra Features

* API Documentation → Swagger & Redoc
* Rate limiting (prevent abuse)
* Logging & error tracking
* Ready for **deployment on AWS EC2 + RDS (MySQL)**

## 🛠 Tech Stack

* **Backend Framework:** Django, Django REST Framework
* **Database:** MySQL
* **Auth:** JWT (SimpleJWT)
* **API Docs:** Swagger / Redoc (drf-yasg)
* **Deployment:** AWS EC2, Nginx, Gunicorn
* **Others:** Logging, Rate limiting

## 📂 Project Structure

nexora/

│── accounts/ # Authentication, Roles, Profile

│── products/ # Categories, Brands, Products, Wishlist

│── cart/ # Cart & Checkout

│── orders/ # Orders, Payments, Coupons

│── nexora/ # Core settings & configs

│── manage.py

│── requirements.txt

## 🔗 API Endpoints (Sample)

| Module | Method | Endpoint | Role |
| --- | --- | --- | --- |
| **Auth** | POST | /api/auth/register/ | Register user |
|  | POST | /api/auth/login/ | Login (JWT) |
|  | GET/PUT | /api/auth/me/ | Profile |
| **Products** | GET | /api/catalog/products/ | All users |
|  | POST | /api/catalog/products/ | Admin/Seller |
| **Cart** | GET | /api/cart/ | Customer |
|  | POST | /api/cart/ | Customer |
| **Orders** | GET | /api/orders/ | Customer/Admin |
|  | PATCH | /api/orders/{id}/ | Admin |
| **Payments** | POST | /api/payments/ | Customer |
| **Wishlist** | GET | /api/catalog/wishlist/ | Customer |
| **Coupons** | POST | /api/coupons/ | Admin |

👉 Full API documentation available at:

* Swagger → /swagger/
* Redoc → /redoc/

## ⚡ Installation & Setup

### 1. Clone Repo

git clone https://github.com/yourusername/nexora-backend.git

cd nexora-backend

### 2. Create Virtual Env

python -m venv .venv

.venv\Scripts\activate # Windows

# source .venv/bin/activate # Linux/Mac

### 3. Install Dependencies

pip install -r requirements.txt

### 4. Configure MySQL

CREATE DATABASE nexora CHARACTER SET utf8mb4;

CREATE USER 'nexora\_user'@'%' IDENTIFIED BY 'StrongPassword123!';

GRANT ALL PRIVILEGES ON nexora.\* TO 'nexora\_user'@'%';

FLUSH PRIVILEGES;

Update settings.py with DB credentials.

### 5. Run Migrations

python manage.py makemigrations

python manage.py migrate

### 6. Create Superuser

python manage.py createsuperuser

### 7. Start Server

python manage.py runserver

## 🌍 Deployment Notes

* Use **Gunicorn + Nginx** on AWS EC2
* Store secrets in .env using django-environ
* Database on **AWS RDS (MySQL)**
* Enable HTTPS with **Let’s Encrypt**

## 📜 License

This project is for educational/job portfolio purposes. Free to use & modify.

✅ That’s your **GitHub-ready README**.  
It will look clean and professional — exactly what recruiters expect.