Software Documentation

> **Software Name:** E-Shop
> **Version:** 1.0.0
> **Date:** Apr 24, 2024
> **Author:** Suhas Shetty

TABLE OF CONTENTS

Software Summary

E-Shop is a simple basic demonstration of an online shopping portal for the purpose of assignment towards the role of Backend Engineer position at Artifact.

The E-Shop project has two modules:

- Authentication
- Shopping Cart

Authentication

The authentication module covers the aspects of the user of the application, and those of login and logout of the user.

User

The User of the application is an implementation of the Django AbstractUser interface, with additional attributes added, such as Phone Number, Address, Gender, Date of Birth and Display Name.

It's business logic prevents the creation and modification of the user from reusing the username, phone number and address of an existing user.

Login and Logout

Login to the application is implemented using JWT (JSON Web Token). A Django-specific library for JWT is used. It generated an access token which is required to authenticate the user for all the APIs in the application.

A refresh token is also provided to the user to refresh the logged in session of the user.

Logout is taken care of by blacklisting the refresh token, as access token is short-lived.

Shopping Cart

The Shopping Cart is the main module of the application. It is for implementing the actual business logic of the application.

It contains three main entities:

- Product
- Order
- Payment

It also contains two more lesser entities:

- Discount
- Review

API endpoints for all CRUD operations along with listing, have been implemented for all the models.

Product

Product is the OOPS representation of a product sold on an e-commerce platform.
It contains attributes such as *Product Code, Name, Rating, Price, Discounted Price and Description*. It also has a foreign key to the Discount model which is how it obtains its discounted price.

The business logic of products prevents it from creating and updating with a product code that is already in use. The discount foreign key provides it with a percentage or fixed value amount that can be deducted from its price attribute to obtain the discounted price.

Order

An order is a relationship between a *collection of products*, with two users each, i.e. *the buyer and the seller*. It also contains attributes such as *value (sum total of prices of all products), status, created and updated dates.*

The *Order status* value is obtained from an Enum which contains values such as *placed, packed, shipped, delivered and their counterparts for the return process* of the order.

Its business logic merely adds up the discounted prices of the products, for the value field of order.

Payment

The Payment model is to represent the process of paying money for the products purchased. It contains a f*oreign key to an Order and a Discount*, along with attributes such as *mode of payment, payment status, gross amount i.e. the sum of the prices of the products in the order, the net amount, after deducting the discount*.

The business logic includes validation to ensure a valid order, and if there exists a discount, then the calculation of the net amount and the compatibility of the mode of payment with that of the discount.

The payment status is obtained from an Enum which contains values such as *pending, completed and failed*. The Enum for payment mode consists of *cash, debit card, credit card,*

UPI, net banking, EMIs for debit and credit cards, pay later, in-house and third party wallets.

Discount

The Discount model is created above and beyond the requirements, to add a little flavour to the amount calculation. It contains attributes like *discount code, type of discount (like coupon or offer by the seller, platform or payment mode), value type, such as fixed or percentage, value of the fixed or percentage amount, and the valid dates from and to*.

The Discount model is foreign key to both the Product and Payment models.

The business logic dictates that a discount cannot be created or updated to have a code already used by another discount.

Review

The Review model is another model outside of the requirements, to add some flavour to the Product model. It contains a *foreign key to the Product model and the User model, along with attributes for title, description rating and verified status*.

The review business logic prevents the creation and update of review with product and user foreign keys same as of an existing review. It also prevents a review without a product, user and rating.

Requirements

As described at
https://docs.google.com/document/d/1ZShsYv4sbOJ0wulaJOGAR1PLTX-Qz--12sRwbb75tKU/edit

https://docs.google.com/document/d/1ZShsYv4sbOJ0wulaJOGAR1PLTX-Qz--12sRwbb7...

How-to Guide

Step 1

Clone repository from Github:

https://github.com/suhasshetty/eshop.git

Step 2

Install Python

Step 3

Install project requirements using the command:

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

Step 4

Run Django server using the command:

```
python manage.py runserver
```

FAQs

Answer and document frequently asked questions below.

Is the application complete?

The requirements are implemented. The application cannot be said to be complete until it a fully functional e-commerce platform.

Does the application implement bonus requirements?

The application implements certain bonus requirements such as Django REST Framework and JWT, but not the unit tests.

What are the next steps for the application?

The next steps are implementation of unit testing, along with additional models for Seller, Addresses and Product Listing.

Special thanks to my fiancee Kajal Rai, Senior E-Commerce Manager, for providing me with guidance for developing an application for e-commerce. I have best attempted to simplify her vast knowledge of e-commerce into minimal amount of code.