Design and Development 1page Report

My Data Source(Customer Subscription Data)

https://www.kaggle.com/datasets/mkechinov/ecommerce-purchase-history-from-electronics-store

My Render URL

https://kzelectronics.onrender.com/

Design

The project is an e-commerce store developed using Django Python framework. This application is built on open source data from Kaggle.com. The application was implemented using the following models/tables products, customer(Django.contrib.auth.User), order, orderItem and shippingAddress.

Tools and Implementation

The first requirement was to load open source data into the application. The database was populated using a script/command which I developed "parse_csv.py" , some values were also generated using the Faker library. The Database I implemented was built on SQL and used ORM relations. During the period of development, I practiced Test Driven Development (TDD) approach, meaning that I would write tests for each unit before they are developed. I also used Behave tests to implement Behavior Driven Development using other tools such as selenium and the chrome driver. Implementing the behave steps was rather challenging but a valuable experience nonetheless.

I also choose to use GitHub for version control and code maintenance. From my experience using GitHub, I got more familiar with running commands in the terminal. I also felt less anxiety about my work being lost because it was stored in a remote repository online. Over time I built the habit of committing and pushing my code to the repository frequently. Using GitHub gave me confidence to experiment with my codebase because I could easily restore a previous save point from the git repository. I will definitely be using git for all my projects in the future.

The web application is rendered/hosted on render.com. I choose this platform because it is free and also allows for easy deployment. Render automatically creates a webhook that redeploys the live website when there is a push to the Github repository.

The Costumer model was created by adding a one to one relationship to the default Django user model. I learned to use @post_save to automatically create a customer object when a user object was created while also initializing the properties of the customer model. The User model was structured to have the following roles: Admin, Guest and Customer/User. For assessment purposes, I created the following dummy credentials:

- Admin (Username: dave, Password: dave)
- Customer (Username: angel, Password: angel)

The website is designed to restrict access to certain pages based on roles, also some links become hidden when a user that is not an Administrator logs in to the website.