**Django E-Commerce Application Documentation**

For the web application task, I used some data on men clothing wears which I got from Kaggle. This data includes name of products, brand, image and price. I decided to use this data as it was really simplified well formatted in a csv file therefore ensuring a bug free migration of the data to the database. The data was also very ideal for an ecommerce project.

**Database Architecture**

I created four models. One for Product, Customer, Order and Line item. The product model is to store each item in the csv sheet. The customer model is for each registered customer. The model has a one-to-one relationship with Django’s built in User model and then accepts other details such as the name, email and address of the user. The order model is for each customer order and has a foreign key relationship to the customer. The line item maps the product selected by the user with the appropriate order model. In other words, it has foreign key relationships with the logged in customer and the order.

I used this schema structure because I felt it would be most efficient to handle the data especially for a logged in user, it is flexible, and maintainable. For guest users, I use sessions to hold their cart until their ready to make a purchase, then the items in the carts are turned into line items and orders and then stored in the database.

**Templates and Routing**

I created multiple routes to handle all the created templates and map them to the controller. I created templates for product listing, product detail, login, register, product add/edit, cart, checkout, dashboard and contact pages. I also have a base.html that has html code which is then shared amongst all other template files. Some templates such as the dashboard, product add/edit have permission protection such that only logged in staff and superuser can view those pages. Normal users are redirected to product listing if they try to access those special templates.

**Installation and running the application**

Please refer to the readme.md for information on running the application locally.

**Testing**

Behaviour driven development and unit testing were used to test that interactions on the website worked as they should. Behave and selenium libraries were used to ensure that pages and routes worked as predicted on user clicks. For instructions on how to run tests, please refer to the readme.md file.

**Maintenance & Deployment**

The application was deployed successfully to Heroku and PostgreSQL is being used in production. Continuous deployment has also been set up so once the file is pushed to GitHub, it automatically deploys to Heroku. You can visit the Heroku link here:

https://agile-bastion-30369.herokuapp.com/