Solutions in terms of content and concept.

The app was designed to allow an entrepreneur to source products and have a platform to sell the products on. The target market was local and therefore didn't require a payment platform to be added right at the beginning. The app can however be expanded to incorporate a payment platform as an add on to the cart. Products are uploaded in the admin panel by the site admin, and they are grouped under categories. When the customer is on the home page they can browse products and sort them via category, which will be very helpful when the list of products get large.

The initial database is an sqlite3 database, but Django has other options for databases that can be used in case the business grows and a larger database server is required.

Making of the project

The Django framework Is a "batteries included" framework where it handles the database via models, where each database table is a model class that defines all the columns and data types required in the database.

Initially the database would have had a customer, supplier, product, and orders tables. But when I started implementing the database the supplier did not make sense and was replaced by a category table. The orders also had to have cart items as a single order can have many items. The customer can have many orders and is the foreign key in the orders database. And the product is linked to the cart items as a foreign key.

The models are set up in a way that will make it easy to maintain when the application grows, each table has its own file (except the order items as they are inherently linked). And that makes the models very modular and easy to maintain.

The HTML side is set up via templates. This contains all the HTML, CSS, JavaScript and bootstrap code. The functionality is handled by the views, which allows communication to the models and the templates. The templates have a base template where every page can have the same basic theme and header which allows the site to look more cohesive.

The security and authorization side are a built-in feature of Django, so creating a sign-up and login functionality was made easier. Because it also derives from the base template, the login and register pages pulls through the same styling as the rest of the app.

The admin panel makes the loading and management of products and users very easy and is also included in Django, the layout is defined by the models.

During the development process the project can be run in debug mode, which leads to much more verbose error messages and stack traces for fault finding, this has to be turned off for deployment to production.

Django allows you to run a local server on 127.0.0.1 which makes seeing the changes made and testing a whole lot easier to see the changes you make and how they reflect on the application.

The project was approached with the idea that it can be scaled up. The setup of the models and views allows for easier maintenance and upgrading and I tried to keep it as modular as possible.

Bootstrap was used to make the pages adaptable and most of the JavaScript was handled through Django.

I have included a database file in the GitHub repository with some products pre-loaded for testing purposes. The admin login has a password of admin and a username of admin which need to be changed before it goes into production.

Lessons learned.

This is the first time I used Django, and I chose it because I am familiar with python and Django is a Python based framework. Through having to manage one to one and one to many relationships with regards to the database, I learned how to manage and setup a database and how to handle the relationships between tables. I also learned how to pull data from the database, and how to commit new data.

I learned how the views receive GET and POST requests and how to handle those requests, and how media and static files are handled. My understanding of how a web app communicates between the front end and back end was also enhanced.

I learned how to read and understand documentation a bit better, and that there is more than one way to handle an error or functionality. There is also a big difference between running the app on a local server compared to deploying it to a production server.