****

**AN E-COMMERCE WEBSITE DEVELOPMENT**

**IN FULFIMENT OF THE**

**MULTI-TIER ARCHITECTURE ASSIGNMENT**

**THEOPHILUS AWOYELE**

**MSC COMPUTER SCIENCE (SOFTWARE ENGINEERING)**

**INTRODUCTION**

This project was done to create an e-commerce web app for a clothing business.

The web page includes a home page, an about page which is in the same page as the home page, a contact page with an upload able text message of up to 200 words, an FAQ page and a store to sell the products.

The store was built to include a shopping basket with items selected listed, quantity, price per unit, total price per items and total price including tax at 20%.

This web app was built using Node.js and express for the backend. The data layer was built using Mongo DB and hosted in the cloud on Mongo Atlas. The data layer was linked to mongoDB using Mongoose and Express. Below is a list of dependencies for the backend build-up.

**Backend Dependencies:**

* body-parser
* cors
* express
* mongoose
* nodemon

For the **frontend**, the app was built using **HTML5**, **CSS3** and **Vanilla Javascript**.

The API was hosted on **HEROKU APP**, and the **FETCH** function was used to call data to display in the frontend.

**User Stories**

* As a user, I want to be able to access store products and view details and prices of each product.
* As a user, I want to be able to add products of interest into cart.
* As a user, I want to be able to see a cart containing my selected products and the total price of selected products, including 20% VAT added to the price.

*Details of each page of the web app include*:

**Store page**

**Views and Features:**

* Returns a list of ALL products to the user and each listed product with properties of an image, title, and price.
* Ability to add a product to cart.

**Cart view**

* Returns data about selected products a user wants to buy.
* Allow users to see total price of selected products.

**Home page view**

* Allows users to access all pages in the application from the navbar.

**About page view**

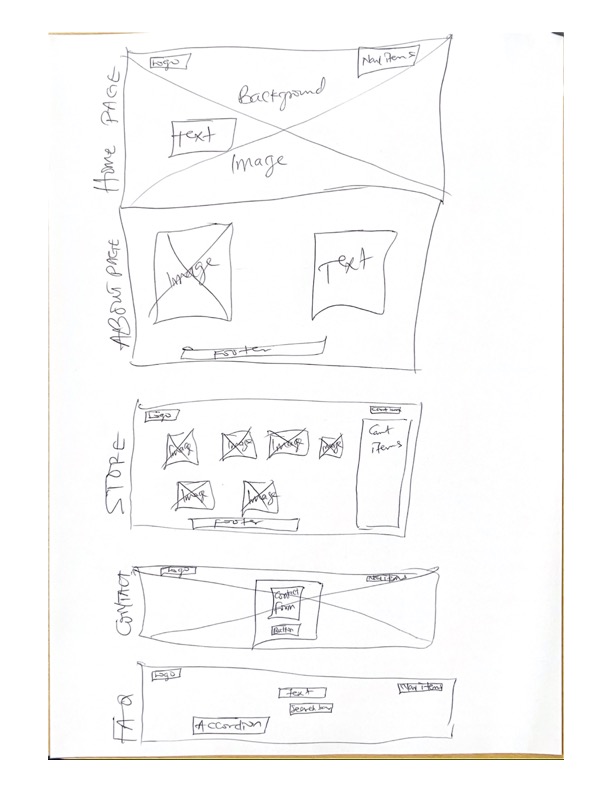
* Displays description about the business.

**Contact page view**

* Returns a contact form for users to send messages to business owner.
* Returns an error if user doesn’t fill form appropriately e.g, leaving a form field blank or not entering the email in the right format.
* Allow user to send message contents of 200 words maximum.

**FAQ page view**

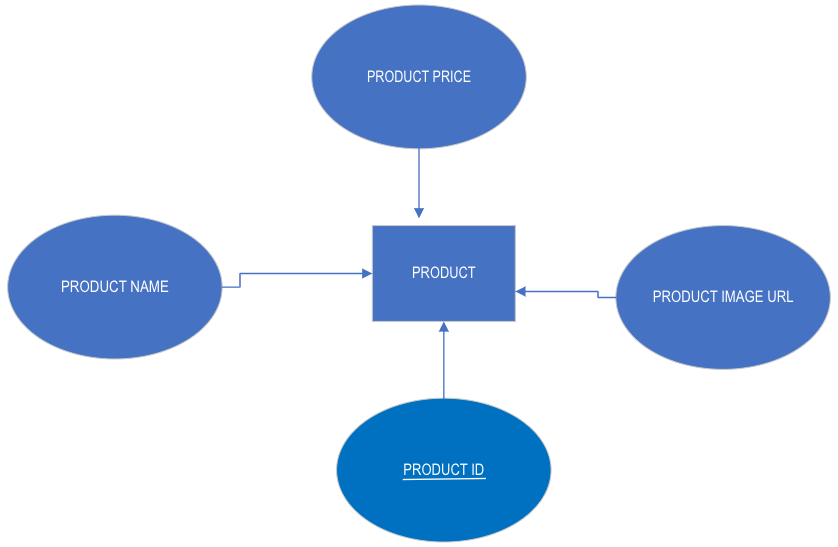
* Displays frequently asked questions a user might want to know.
* Question lists should be displayed in the form of an accordion.
* Displays answers to each question on the click of each accordion.

****

**Low fidelity diagram showing the layout of all the pages in the web app**

**Database design**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| REQUEST | URL | http Method | Request data format | Response data format |
| Get a list of products | /products | GET | None | Json object holding data about all products.  Product properties:  {  "\_id": "Product ID",  "name": "Product name",  "imageUrl": "Image link”  "price": “product Price”  }  Database Schema:  name:{  type: String,  required: true  },  price: {  type: Number,  required: true  },  imageUrl: {  type: String,  required: true  } |



Entity relationship diagram

**Test Plan**

For this project, testing was scheduled for the end of the project.

The lighthouse testing was done. This was run in google chrome browser dev tools, to test for performance, accessibility, best practice and SEO. The diagrams below show the results of the tests for each page of the web app.

Also, W3C validation test was done on all all CSS and HTML files and they all returned ‘**Valid**’ results.

