Assignment 02: E-Commerce Flask Application

Objective: In this assignment, you will create a Flask web application with two routes:

- a homepage (/) and,
- a products page (/products).

The application will connect to a MongoDB Atlas database to fetch and display product information, which includes a product's name, tag, price, and image path (local to the Flask app). You will use Jinja2 templates to create dynamic content and Bootstrap for styling the product page.

Instructions

1. **Set Up MongoDB Atlas**
    - Create a MongoDB Atlas account and set up a new cluster.
    - Create a database named shop_db.
    - Create a collection named products.
    - Insert several product documents into the products collection, with each document containing the following fields:
        - name (string): The name of the product.
        - tag (string): A descriptive tag or category for the product.
        - price (number): The price of the product.
        - image_path (string): Path to a local image that will be served by the Flask app (e.g., static/images/product1.jpg).

2. **Set Up Flask Application**

   Set up a new Flask project with the following structure:

```
flask_app/
├── app.py
├── templates/
│   ├── base.html
│   ├── home.html
│   └── products.html
├── static/
│   ├── css/
│   │   └── styles.css
│   └── images/
│       ├── product1.jpg
│       └── product2.jpg
└── requirements.txt
```

3. **Connect to MongoDB Atlas**

- Install the necessary Python packages using pip (make sure to include these in requirements.txt):
    1. flask
    2. pymongo
- In app.py, use pymongo to connect to your MongoDB Atlas database. Fetch the product data from the products collection.

4. **Create Routes**

- **Homepage (/)**:
    - Create a basic homepage route that renders a home.html template.
    - You can provide a brief introduction to your store on this page.
- **Products Page (/products)**:
    - Create a route "/products" that queries the MongoDB Atlas products collection to retrieve product information.
    - Pass the retrieved data to the products.html template to dynamically display a table of products using Jinja2 and Bootstrap.

5. **Template and Bootstrap Integration**

- **Base Template (base.html):** Create a base.html file that serves as the layout template for all pages. Use Bootstrap for basic styling (e.g., navigation bar, container layout).
- **Home Page (home.html)**: Extend the base.html template and create content for the homepage.
- **Products Page (products.html)**:
- Use Jinja2 to loop over the product data and generate a table of products dynamically.
- Each row should display:
    - Product name
    - Product tag
    - Product price
    - Product image (display the image using the image_path field)

- **Custom Styling:** Add some custom styling in static/css/styles.css to improve the look of the product page (optional but encouraged).

6. **Testing the Application**

- Run the Flask app using python app.py.
- Test both the homepage and the products page to ensure the data is fetched and displayed correctly from MongoDB Atlas.

7. **Submission**

- Submit the following:
    - A link to your Public GitHub repository.
    - Ensure that the requirements.txt file is included in your repository.