Author
Daksh Sharma
21f3002338
21f3002338@ds.study.iitm.ac.in
I am a B.Tech. – 5th Semester engineering student from Jalandhar (Punjab). My areas of interest include working with python in fields of Data Sciences and Machine Learning.

Description

The project involved developing a Flask-based Grocery Store App with user authentication, category and product management, inventory control, search functionality, and a shopping cart feature. Admin and users can log in, the admin can manage categories and products, and users can search for and purchase products. The app will be styled using Bootstrap, with data stored in an SQLite database, and includes optional dynamic pricing.

Technologies used:

1. Flask: It is a lightweight web framework for Python that is used to build the application's structure, define routes, and handle HTTP requests and responses.
2. Jinja2: A templating engine used to generate dynamic HTML content based on the data provided by the application. It allows embedding Python code within HTML templates.
3. SQLite: A serverless, self-contained, and file-based relational database management system used to store data like user information, products, categories, and cart items.
4. Flask-SQLAlchemy: A Flask extension that simplifies working with databases by providing an object-relational mapping (ORM) layer for interacting with databases using Python objects.
5. Flask-Login: A Flask extension that handles user authentication and session management. It's used to manage user sessions, login, and access control.
6. Flask-Bcrypt: A Flask extension for password hashing. It's used to securely hash and store passwords in the database.
7. Bootstrap: A popular front-end framework that provides pre-styled components and layouts, helping to create a responsive and visually appealing user interface.
8. HTML/CSS/JavaScript: Standard web technologies used for defining the structure, styling, and interactivity of web pages.
9. Datetime: A Python module used to work with date and time information, utilized to handle product manufacture dates in the application.
10. Flash: A feature of Flask for displaying temporary messages to the user, typically used to show success, error, or warning messages after certain actions.
11. os: A Python module used to interact with the operating system. Using this code, we can construct the path to the SQLite database file.

These technologies collectively enable the creation of a functional, interactive, and user-friendly Grocery Store web application.

The purpose behind using these technologies in the code:

-Flask: To provide the core framework for building the web application, managing routes, and handling HTTP requests and responses.
- Jinja2: To enable dynamic HTML generation by integrating Python code and data into HTML templates.
- SQLite: To serve as the database backend to store and manage user information, product details, and cart items.
- Flask-SQLAlchemy: To simplify database interaction by offering an ORM for creating, querying, and updating database records using Python classes and objects.
- Flask-Login: To manage user authentication and provides a convenient way to track logged-in users and their sessions.
- Flask-Bcrypt: To enhance security by securely hashing passwords before storing them in the database.
- Bootstrap: To speed up front-end development by providing pre-styled components and responsive design features.
- HTML/CSS/JavaScript: These standard web technologies were used for creating and styling the user interface and adding interactivity.
- Datetime: To handle date-related functionality, such as managing product manufacture dates.
- Flash: To provide user feedback messages after actions like successful logins, errors, or account creations.
- os: To construct file paths in a platform-independent way, which is useful for locating the SQLite database file.

DB Schema Design

(A) Customers Table:

- `id`: Primary key, unique identifier for each customer.
- `username`: Username of the customer, must be unique.
- `password`: Hashed password of the customer.
- `cart_items`: One-to-many relationship with the `CartItem` table to represent items in the customer's cart.

(B) StoreManagers Table:

- `id`: Primary key, unique identifier for each store manager.
- `username`: Username of the store manager, must be unique.
- `password`: Hashed password of the store manager.

(C) Categories Table:

- `id`: Primary key, unique identifier for each category.
- `name`: Name of the category.

(D) Products Table:

- `id`: Primary key, unique identifier for each product.
- `name`: Name of the product.
- `manufacture_date`: Date when the product was manufactured.
- `rate_per_unit`: Rate per unit of the product.
- `quantity`: Quantity of the product available.
- `category_id`: Foreign key referencing the `id` column in the `Categories` table.
- `category`: Many-to-one relationship with the `Category` table to associate products with categories.

(E) CartItems Table:

- `id`: Primary key, unique identifier for each cart item.
- `customer_id`: Foreign key referencing the `id` column in the `Customers` table.
- `product_id`: Foreign key referencing the `id` column in the `Products` table.
- `product`: Many-to-one relationship with the `Product` table to associate cart items with products.
- `quantity`: Quantity of the product in the cart.
- `added_at`: Date and time when the item was added to the cart.

Reasons Behind the Design:

1.User Differentiation: The separate `Customers` and `StoreManagers` tables allow you to distinguish between customers and store managers, with their own unique identifiers and login credentials.

2. Category-Product Relationship: The `Categories` table helps organize products into different categories. The `category_id` column in the `Products` table establishes a relationship between products and their corresponding categories.

3. Cart Management: The `CartItems` table stores information about items in a customer's cart, including the product, quantity, and timestamp. This structure helps track customer-specific cart items and manage their contents efficiently.

4. Data Integrity: Foreign key relationships (such as `category_id`, `customer_id`, and `product_id`) maintain referential integrity by ensuring that values in child tables (e.g., `Products`, `CartItems`) reference existing values in parent tables (e.g., `Categories`, `Customers`).

5. UserMixin for Authentication:Using `UserMixin` from `flask_login` in the `Customer` and `StoreManager` models provides common attributes and methods for user authentication, making it easier to manage user sessions and access control.

6. Timestamps: The `added_at` field in the `CartItems` table records the date and time when items are added to the cart, which can be useful for order history and tracking.

Architecture:

-Controllers (Route Handlers): The route handlers (controllers) are defined as route functions within the main application file. These functions are responsible for handling various HTTP requests, rendering templates, and processing form submissions.

- Templates: HTML templates are stored in a separate directory. The templates are rendered using Jinja2 templating engine, which allows dynamic content insertion. Templates are used for rendering login forms, product lists, cart views, and more.

-Models: The database models are defined in an external module (`models.py`). The models represent different entities like `Customer`, `StoreManager`, `Category`, `Product`, and `CartItem`.

- Static Files: Static assets like CSS, JavaScript, and images are usually stored in a separate directory named "static."

- Database: The project uses SQLite as the database for data storage. SQLAlchemy is used as the Object-Relational Mapping (ORM) library to interact with the database.

Features Implemented:

1. User Authentication: Both customers and store managers can register and log in using secure password hashing. Flask-Login is used for managing user sessions.

2. Customer Actions:
   - Customers can log in and view products and categories.
   - Customers can add products to their cart and view the cart contents.

3. Store Manager Actions:
   - Store managers can log in to access the store manager dashboard.
   - Store managers can add, edit categories.
   - Store managers can add, edit products within categories.

4. UI Interaction: The UI is styled using Bootstrap for responsive and visually appealing designs.

Video Link:

https://drive.google.com/file/d/1GQQ9d4kfE5joKi7BhULlJsqQkvehQnY9/view?usp=sharing