Author:

Areeba Farooqui

22f1001053

22f1001053@ds.study.iitm.ac.in

I enrolled in the B.S Online Degree after completing my 12th. Currently I'm in my fourth term of diploma with Java, MLP and BDM Project.

Description:

Grocery Store is a dynamic multi-user web application developed using Python, SQLite and Flask.

Admins manage categories, approve manager's registration and their requests and can also do operations on products. Store Managers handle product and category management, while users enjoy a wonderful shopping experience, from product selection to order placements.

Technologies used:

FLASK - Employed as the backend framework, Flask facilitated API development, handling HTTP requests, and managing routes for seamless communication between the front end and database in Grocery Store.

VueCLI- Utilized on the frontend, VueCLI enhanced the user interface of Grocery Store, providing dynamic and responsive features for an interactive and efficient shopping experience.

SQLite - Used as the database system to store and manage the application's data.

Redis(for caching)- Integrated Redis for caching in Grocery Store, optimizing data retrieval performance by storing frequently accessed information, thereby reducing latency and enhancing overall system efficiency.

Redis and Celery (for backend jobs)- Redis and Celery for implementing backend jobs in Grocery Store, automating tasks such as sending periodic emails and generating reports to enhance system functionality and user experience

HTML and CSS: - HTML structured the content of Grocery Store web pages, while CSS was used for styling and formatting, collectively contributing to an aesthetically pleasing and user-friendly interface.

Bootstrap - Used it to enhance the user interface and design of the web application.

DB Schema Design:

Models : RolesUsers, User, Role, Category, Product, ShoppingCartItem, Order,

StoreManagerRequest.

The database has eight tables. The "users" and "products" are used to store the users and the products data respectively. The "RolesUsers" table is for connecting the roles to users. The "Categories" table is used to store the data of the present categories in the Grocery Store. The "ShoppingCartItem" table is used storing the products which are added to the cart. The table "Order" which used for orders. Then "StoreManagerRequest" is used to store the requests the manager made to the admin for doing anything with the categories.

API Design:

The API design follows a RESTful architecture using Flask-RESTful. It defines endpoints for user authentication, user profile, sections, products, section requests, and the shopping cart. The API employs standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations, adhering to REST principles for simplicity and scalability. Data is exchanged in JSON format, and the API includes authentication mechanisms using JWT for secure user access.

Architecture and Features:

- Architecture

Controllers: Defined in the controllers.py within the application folder, handling HTTP requests and defining endpoints.
APIs: Present in api.py within the application folder, utilizing Flask-RESTful for building RESTful APIs.
Models: Stored in models.py within the data folder, leveraging SQLAlchemy for defining data structures and interacting with the database.
JavaScript: JavaScript files for dynamic frontend functionality.
Styles: Defined in style.css for basic styling across the application.
Celery Jobs: Implemented in the application folder with files like email.py, tasks.py, and worker.py for asynchronous task processing
Configuration: Configuration settings are managed in DevelopmentConfig
Main Entry Point: Executed from main.py, creating a Flask app, initializing components, and defining API resources

- Features

User Authentication- The app supports user registration,login and logout functionalities . Passwords are securely hashed .

Admin Authentication: Admin can login and manage their dashboard, where they can manage the sections and products.

Product and Section Management: Admins can create, edit and delete the sections and products.

Product listing and store: Users can view the store and browse and buy products .

Cart Management: Users can add products to the carts and specify quantities . They can also view and remove items from the cart.

Order Placement: Users can proceed to checkout, view the total amount and place the orders. Products' stock quantity is updated upon order placement.

Search Functionality: Users can search for the products using keywords , and the application returns relevant results based on product names, section names, category and price.

Password Hashing: User passwords are securely hashed before storing them in the database by generate_password_hash.

User and Admin Roles: The application distinguishes between user and admin roles from the registration functionalities.

User Profile and Order details: Users can view their profile and order history.

Video: For the video demo, click here