Author

Swastik Kumar Shukla
21f3002166
21f3002166@ds.study.iitm.ac.in
I have hunger and eagerness to learn new technologies, currently I am currently focusing on backend development, and how can I implement ML in applications.

Description

The User Interface of the project can be improved, google authentication for user signup and login can be introduced, and Coupon codes can be introduced.

Technologies Used

Frontend-Vue 3; **Database-** SQLite 3; **Backend-** Flask; **Backend Jobs-** Celery, Redis **Flask Extensions**- Flask-CORS , Flask-Caching , Flask-SQLAlchemy

DB Schema Design

1) User Table
    - **id –** Primary key for the user table.
    - **name –** Stores the username of the User.
    - **email** – Stores the Email id of User.
    - **password** – Stores the password of user in hashed form.
    - **role**- Stores the role/type of user.
    - **orders**- A relationship one-to-many relationship between the 'User' and 'Order' tables.
2) Product Table
    - **id –** Primary key for the product table.
    - **name –** Stores the name of the product.
    - **price –** Stores the selling price of product.
    - **expdate –** Stores the expiry date of product.
    - **quantity –** Stores the stock-left of the product. • **ms-** Stores the measuring scale of the product.
    - **desc –** Stores a brief description about the product.
    - **category_id -**Stores the category id of the product.
3) Category Table
    - **id –** Primary key for Category table.
    - **name -** Stores the name of the category.
    - **c_item** – A one-to-many relationship between the 'Category' and 'Product' tables.
4) Request Table
    - **id-** Primary key for the Request table.
    - **cat_id –** Stores the category id of the category to be updated/deleted.
    - **type –** Stores the type of request.

- **subtype –** Stores the subtype of the request.
- **status -** Stores the status of the request i.e., pending, approved, rejected. 5) Order Table
- **id –** Primary key of the Order table.
- **user_id –** Stores the user id of the customer.
- **name –** Stores the username of the customer.
- **total_value –** Stores the total order value.
- **transactions-** A one-to-many relationship between 'Order' and 'Transaction' tables.
- **order_timestamp –** Stores the timestamp of the order.

6) Transaction Table

- **id –** Primary key of Transaction table.
- **name -** Stores the name of the product.
- **quantity –** Stores the quantity of the product purchased.
- **price –** Stores the selling price of the product.
- **order_id-** Stores the order id of the Transaction.
- **product_id** – Stores the product id of the purchased product.

API Design

I have created the API's using by routing the request URL to the respective functions. The API's have access control and only the authorized user can use them.

Architecture and Features

Inside the root folder of the project in the **src** folder there are separate folders for the **views, routers(urls), components, assets (CSS files and image files),** and the frontend **App.** Inside the root folder of the project in the **backend** folder the is the **app.py-**python file having the API's, **cache.py –** python file for caching tasks, **celeryconfig.py** – python file for celery configuration, **config.py** – python file for main app's configuration, **middleware.py** – python file for helper functions, **models.py –** python file for storing the models, **tasks.py**- python file for performing backend tasks. The app has JWT token authentication and authorization. In the app the store manager can perform CRUD operations on the products. An additional feature which I have implemented is the Administrator can add Store Managers instead of the store manager. I have sharpened my problem-solving while making this project.

Video Link

https://drive.google.com/file/d/1tFTa4Mw_hkkT8DEPi7vqo87G8Ao6nB0l/view?usp=drive_link