Grocery Store Application Project Report

Author

Name : Vibhav Jayant Pande
Roll number : 22f1000996
Student email : 22f1000996@ds.study.iitm.ac.in
I possess a strong technical aptitude, which reflects my robust problem-solving skills and optimistic approach. My adaptable personality and unwavering commitment to excellence further bolster my capabilities.

Description

This project entails developing a full-fledged web-app for a grocery store, having a Managers interface for streamlined item management (addition/removal) and a User interface facilitating seamless shopping experiences, allowing item selection, cart management, and purchases.

Technologies used

1. **Flask:** Flask is like the core organizer, ensuring everything runs smoothly in the app.
2. **SQLAlchemy:** SQLAlchemy talks to the database, storing and fetching data like a pro.
3. **Flask-Login:** Flask-Login acts as the gatekeeper, allowing only the right folks into specific app areas.
4. **Werkzeug:** Werkzeug is the app's bodyguard, keeping it safe and secure and modify the API exceptions
5. **os Module:** The os module is the helper that talks to the computer's brain, helping manage files and folders neatly using 'os.path.join'.
6. **re Module:** The re module plays detective, checking text patterns like a pro with 're.search'.
7. **datetime:** datetime keeps track of date, making sure expiry dates and purchase dates are on point.
8. **Flask_restful** : Making the API
9. **Flask_cors** : used to handle Cross-Origin Resource Sharing (CORS) issues in Flask applications, allowing controlled and secure communication between different domains or origins.

DB Schema Design

1. **User Table:**
   - Stores user details: id (primary key), username (email), hashed password, name, contact number, role_id (foreign key).
   - Role-based access control: role_id (1 for User, 2 for Manager).
2. **Role Table:**
   - Defines roles: id (primary key), role, description.
   - Facilitates role differentiation and permissions.
3. **Category Table:**
   - Manages categories: Cat_id (primary key), Cat_name, is_Deleted, image_link.
   - Supports historical tracking and visual representation.
4. **Inventory Table:**
   - Tracks products: Prod_id (primary key), Prod_name, Prod_price_rs, Quantity, Expiry_date, Cat_id (foreign key), is_Deleted, is_Expired.
   - Links products to categories, aids in product management.
5. **Cart:**
   - Handles user carts: Cart_id (primary key), user_id(foreign key), cart_total, Status, Buy_date
   - Connects users to active/inactive carts.

6. **Cart_Details:**

- Stores cart contents: Cart_id (foreign key), prod_id(foreign key), Quantity, Prod_total.
- Records product details within carts.

API Design

I've developed an API using Flask-RESTful to manage Categories and Products with CRUD operations. The API has separate endpoints and resource classes for Categories and Products.

Endpoints and Resource Classes:

1. Categories:
    - /api/manager/category/add (POST): List and create categories
    - /api/manager/category/<string:category> (GET, PUT, DELETE): Get, update, or delete a category
2. Products:
    - /products (GET, POST): List and create products
    - /products/<product_id> (GET, PUT, DELETE): Get, update, or delete a product

Custom error handling uses HTTPException for specific cases:

- 404 Not Found: Category/Product does not exist.
- 400 Bad Request: Attempting to create an existing Category/Product.
- 422 Unprocessable Entity: Expiry date is not after today.
- 201 Created: Successful creation of a Category/Product.
- 200 OK: Successful retrieval, update, or deletion.

Please refer to the attached YAML file for detailed API specifications, including request/response structures and examples.

Architecture and Features

The project structure is organized with a single Python file containing the main code, while HTML templates reside in the "templates" directory, and corresponding CSS files are stored in the "Static" directory.

This project presents a user-friendly web application that offers seamless interaction for both users and managers. The core functionalities have been meticulously implemented to ensure a smooth experience.

User Functionality:
Users and managers access distinct login and signup pages. Users can effortlessly browse categories, search products by name or price, and easily add items to their cart or make purchases. Cart management is intuitive, allowing users to adjust quantities and have expired items automatically removed. The user profile page provides a comprehensive view of their past carts.

Manager Functionality:
Managers are empowered with comprehensive control. They can seamlessly manage categories and products by adding, updating, or deleting them. A convenient option allows managers to efficiently restock product quantities. Expiry alerts facilitate prompt action for managing product lifecycles.

In conclusion, this project delivers an efficient and user-centric web application, ensuring a smooth experience for users and providing managers with essential tools for effective system management.

Video

https://drive.google.com/file/d/19O8R_joJSXCPiftCwVO7kS0F2beo00CW/view?usp=sharing