IIT Madras
BS Degree
App Dev II Project

Grocery App Version 2 Project Report

Author:

Name: Ritik Ranjan
Student ID: 21f2001147
Mail ID: 21f2001147@ds.study.iitm.ac.in
About me: I am a diploma level student in IIT Madras BS in Data Science and application degree.

Description:

It is a multi user application used for buying grocery items. A user can buy many products from one or multiple sections. Store manager can add, delete and edit products. Admin can add new categories and edit and delete them upon the request of store manager. Every category can have a number of products. System will automatically reflect the latest product/category added.

Technologies Used:

- Flask for application code
- VueJs for UI and Jinja templates Bootstrap for HTML generation and css styling
- SQLite as the database for the application
- Flask_security(Token based Authentication) is used for secure user login
- Flask_celery for async_jobs
- Flask_restful for building RESTful APIs
- Redis for caching
- Redis for celery jobs

Database Schema:

There are a total of 14 different tables for handling different functionalities of the applications:

1. **Roles Table:**
    - Represents the roles assigned to users.
    - Each role has a unique identifier (‘id’).
    - Associated with a user through the ‘user_id’ foreign key.
    - Also linked to a specific role through the ‘role_id’ foreign key.
2. **Orders Table:**
    - Stores information about user orders.
    - Each order has a unique identifier (‘id’).
    - Contains details such as ‘user_id’, ‘product_id’, ‘product_name’, ‘quantity’, and ‘total’.
    - Establishes a relationship with the ‘Products’ table through the’products’ foreign key.
3. **Role Table:**
    - Defines user roles within the application.
    - Each role has a unique identifier(‘id’), a ‘name’, and a ‘description’.
4. **User Table:**
    - Represents information about users.
    - Includes details like ‘id’, ‘username’, ‘age’, ‘city’, ‘email’, ‘lastlogin’, ‘password’, and ‘active’ status.
    - Users can have multiple roles, linked through the ‘roles’ relationship.
    - Also associated with categories through the ‘category’ relationship.

5. **Categories Table:**
    - Stores information about product categories.
    - Each category has a unique identifier(‘category_id’) and a ‘category_name’.
    - Linked to products through the ‘product’ relationship.
6. **CategoryCreated Table:**
    - Represents the creation of categories by users.
    - Contains identifiers(‘vc_id’, ‘cadmin_id’, ‘ccateg_id’) linking to user and category details.
7. **Products Table:**
    - Contains information about products.
    - Each product has a unique identifier (‘product_id’), ‘product_name’, ‘unit_price’, ‘quantity’, ‘revenue’, ‘manufacturing_date’, and ‘expiry_date’.
    - Associated with categories through the ‘categories’ relationship.
8. **ProductCreated Table:**
    - Represents the creation of products within specific categories.
    - Contains identifiers(‘sc_id’, ‘cprod_id’, ‘ccateg_id’) linking to product and category details.
9. **CartItem Table:**
    - Stores items in user shopping carts.
    - Each item has a unique identifier (‘cart_item_id’), ‘user_id’, ‘product_id’, ‘product_name’, ‘quantity’, and ‘total’.
    - Linked to products through the ‘product’ relationship.
10. **Cart Table:**
    - Represents user shopping carts.
    - Each cart has a unique identifier(‘cart_id’) and is associated with a user through ‘user_id’.
    - Contains items linked through the ‘items’ relationship, with cascading deletion.
11. **ModifyRequests Table:**
    - Stores requests to modify categories.
    - Includes information such as ‘id’, ‘categ_id’, ‘categ_name_req’, and ‘state’.
12. **CreateRequests Table:**
    - Stores requests to create categories.
    - Contains information like ‘id’, ‘categ_name_req’, and ‘state’.
13. **DeleteRequests Table:**
    - Stores requests to delete categories.
    - Contains information such as ‘id’, ‘categ_id’, and ‘state’.
14. **StoreManagerRequests Table:**
    - Records requests related to store managers.
    - Includes details like ‘id’, ‘user_id’, ‘state’, and ‘created_at’.

API design:

The table schema has been implemented for CRUD operations with API endpoints. This allows for easy manipulation of data using HTTP requests such as GET, POST, PUT and DELETE. By providing a standardized way to interact with the data, the API ensures that the database is always in sync with the application’s needs. This makes it easier to maintain and update the application over time.

Architecture and features:

VueJs is used to make website faster and more responsive. The design is focused on creating a smooth experience for users by integrating Vue components seamlessly on a single page. Data retrieval speed is boosted by incorporating Redis and caching. Additionally, automated alert notifications and monthly reports using celery jobs is also integrated. These enhancements together create a sophisticated and strong system that prioritizes speed, efficiency, and automation to provide users with a great digital experience.

Video link:

To view short video of my project click here: App dev 2 Project video link