MAD II Grocery Store V2 Project Report

Author

J.Sai Satya Narayana

21f3001439@ds.study.iitm.ac.in

Hello, I am Sai Satya.I am pursuing BTech CSE from SRMIST Chennai and BS in Data Science and Applications from IIT Madras(online degree). I am always keen to explore new things related to Tech and avidly inclined towards solvingreal-life problems by applying my knowledge of Computer Sciences.

Description

I engineered a sophisticated multi-user grocery store web app using Vue.js and Flask merging frontend and backend seamlessly. Bootstrap ensured an intuitive, user-friendly interface.The app featured dynamic category and product management. Admins/Managers effortlessly controlled Categories and products, while users enjoyed real-time stock updates and flexible pricing options.Security was paramount, enforced by Flask Security for token-based authentication. Admins had the ability to approve managers, streamlining access hierarchy. The application boasted Flask REST APIs for strict data validation, ensuring accuracy at every user interaction.Additionally, Redis and Celery were integrated for efficient batch job execution, enhancing the app's performance during heavy loads.This tech fusion resulted in a robust, secure, and responsive grocery shopping experience, seamlessly integrating functionalities while upholding data integrity.

DB Schema Design

- User Roles and Permissions
- RolesUsers **Table**
    - id: Integer, Primary Key
    - user_id: Integer, Foreign Key referencing user.id
    - role_id: Integer, Foreign Key referencing role.id
- User **Table**
    - id: Integer, Primary Key
    - username: String, Unique
    - email: String, Unique
    - password: String
    - active: Boolean
    - fs_uniquifier: String, Unique
    - Relationships:
    - Many-to-Many relationship with Role through RolesUsers
- Role **Table**
    - id: Integer, Primary Key
    - name: String, Unique
    - description: String
- Product and Category Management
- Category **Table**
    - id: Integer, Primary Key
    - name: String, Not Null
    - active: Boolean
    - image: String
    - Relationships:
    - One-to-Many relationship with Product through product_relation
- Product **Table**
    - id: Integer, Primary Key
    - name: String, Not Null
    - unit: String, Not Null
    - rate: Integer, Not Null
    - quantity: Integer, Not Null
    - manufacture_date: String, Not Null
    - expiry_date: String, Not Null
    - image: String
    - category_id: Integer, Foreign Key referencing category.id
    - Relationships:
    - One-to-Many relationship with Category
- Shopping Cart and Purchases
- Cart **Table**
    - id: Integer, Primary Key
    - user_id: Integer, Foreign Key referencing user.id
    - product_id: Integer, Foreign Key referencing product.id

        - product_name: String, Not Null
        - req_quantity: Integer, Not Null
        - product_rate: Integer, Not Null
        - product_unit: String, Not Null
- Bought **Table**
        - id: Integer, Primary Key
        - date: String, Not Null
        - user_id: Integer, Foreign Key referencing user.id
        - product_id: Integer, Foreign Key referencing product.id
        - product_name: String, Not Null
        - req_quantity: Integer, Not Null
        - product_rate: Integer, Not Null
        - product_unit: String, Not Null
- Association Table for Category-Product Relationship
    - Association **Table**
        - category_id: Integer, Foreign Key referencing category.id, Primary Key
        - product_id: Integer, Foreign Key referencing product.id, Primary Key

API Design:

Title: Grocery Store RESTful API
Description: This API manages operations for a grocery store.
Version: 1.0.0
Base Path: /
Tags:
- **Category**: Operations related to categories
- **Product**: Operations related to products
- **Cart**: Operations related to the shopping cart
- **Bought**: Operations related to purchased items

Schemes: HTTPS
Consumes:
- application/json

Produces:
- application/json

Paths:
- /api/category:
    - **GET**: Get all categories
    - **POST**: Create a category
    - **PUT**: Update a category by ID
    - **DELETE**: Delete a category by ID
- /api/product:
    - **GET**: Get all products
    - **POST**: Create a product
    - **PUT**: Update a product by ID
    - **DELETE**: Delete a product by ID
- /api/cart:
    - **GET**: Get user's cart items
    - **POST**: Add an item to the cart
    - **DELETE**: Delete a item by user ID
- /api/bought/{id}:
    - **GET**: Get purchased items by user ID

Security: Token-based authentication for all endpoints
Definitions:

Architecture and Features:

In the project python file, the model.py python file consists of different models like Category, Product , Association , User , Role , RoleUsers , Cart , Bought. The app.py contains all the creation of app and controllers. api.py consists of all the API end points.I have implemented the token based authentication for Admin ,Customer and Manager.the Admin can add, edit or delete a Category and the Manager can add items into different categories created by Admin and edit and delete their details time to time.A Manager can also add categories but it has to be approved by the Admin.A Manager can access the application only if he is authorised by the Admin.Customer can shop for products of same or different Categories and place his Order. Admin can send daily remainder emails to Customers and monthly report email to managers.

Video : https://drive.google.com/file/d/10U_f-Z1H_Sa54N8r68hY_7Z2tEC5twXc/view?usp=sharing