Grocery Store Report

Author

- **Name**: Aishwarya Anil Menon
- **Roll No.**: 21f1005945
- **Student Email**: 21f1005945@ds.study.iitm.ac.in
- **About Me**: I am a fourth-year BTech student at CCOEW, Pune, with a passion for Machine Learning and Web Development. I'm enthusiastic about exploring new technologies such as Next.js and Web3.

Description

The "GrocerEase" project involves the development of a user-friendly grocery store application by using VueJS for frontend and Flask for backend and SQLite3 database for storign the data.
The aim is to provide a seamless shopping experience for users, allowing them to browse and purchase groceries conveniently from their devices. Shop with Ease, Groceries at Your Fingertips.

Technologies Used

The project utilizes the following technologies:

- **VueJS (Frontend)**: Responsible for creating dynamic and interactive user interfaces.
- **Flask (Backend)**: Powers the backend logic and API endpoints.
- **Flask-Caching**: Enhances API performance by caching responses.
- **Flask-CORS**: Facilitates cross-origin resource sharing.
- **Flask-JWT**: Implements authentication and authorization processes.
- **ChartJS**: Enables the creation of an interactive sales dashboard.

- **Flask-SQLAlchemy**: Used for efficient database management.
- **Flask-RESTFul**: Implements RESTful architecture for API development.
- **Reportlab**: Utilized for generating aesthetically pleasing reports.
- **Bootstrap/CSS:** Provides a responsive and visually appealing design.
- **SQLite3**: Serves as the application's database.
- **Vuex**: Manages state for the shopping cart.

Tools Used:

- **Postman**: Used for testing APIs.
- **VSCode**: Primary IDE for development.
- **Git/GitHub**: Version control and code repository.
- **Swagger**: Used for creating the YAML document.

Features

1. **Animated Frontpage:** Utilizes VueJS animations for an engaging user experience.
2. **User Authentication:** Implemented login and registration using Flask-JWT.
3. **User Dashboard:**
    - Displays products with status indicators (e.g., out of stock, recently added).
    - Functionalities include adding to cart, direct purchase, and favoriting.
4. **Direct Purchase:** Displays a payment form; updates stock after checkout.
5. **Order History:** Provides a comprehensive buying history for users.
6. **Shopping Cart:** Utilizes Vuex for efficient state management.
7. **Filtering:** Allows filtering by category, product, price, manufacturing date, and name sorting.
8. **Admin/Manager Dashboard:**

    - Notification system for admin requests.
    - Interactive Sales Summary Dashboard using ChartJS.
    - CRUD operations for products and categories.
    - Managers require admin approval for category-related actions.
9. **Backend Jobs:**
    - **Monthly PDF Report:** Automatically sent to admin/manager via email.
    - **Daily Reminder:** Sends reminders to users who haven't visited the app.
    - **Manager Profile Deletion:** Automatically deletes profiles 30 days after admin declines.
    - **CSV Product Details:** Allows downloading product details in CSV format.

Architecture

The "GrocerEase" application follows the Model-View-Controller (MVC) architecture:

- **Model**: Utilized SQLite3 database for data storage and manipulation.
- **View**: Created the frontend using VueJS and Bootstrap to ensure a user-friendly interface.
- **Controller:** Developed the backend using Python and Flask to handle user requests and application logic.

Database Schema:

User:

- user_id: Integer, primary key, auto-increment
- email: Text, unique, not-null
- name: Text, not-null

- password: Text, not-null
- role: Text, not-null
- avatar: Text, nullable
- request_approval: Integer, not-null, default=0

Category:

- category_id: Integer, primary key, auto-increment
- category_name: Text, not-null
- category_approval: Integer, nullable, default=0
- category_image: String(255), nullable, default='/default_img.png'

Product:

- product_id: Integer, primary key, auto-increment
- category_id: Integer, foreign key(Category.category_id), not-null
- product_name: String(255), not-null
- manufacturing_date: Text, default=datetime.now().date(), not-null
- unit: Text, not-null
- price: Float, not-null
- hasDiscount: String(10), nullable
- product_image: String(255), nullable, default='/default_img.png'

Favourite:

favourite_id: Integer, primary key, auto-increment

user_id: Integer, foreign key(User.user_id), not-null

product_id: Integer, foreign key(Product.product_id), not-null

Cart:

- cart_id: Integer, primary key, auto-increment
- user_id: Integer, foreign key(User.user_id), not-null
- product_id: Integer, foreign key(Product.product_id), not-null
- quantity: Integer, not-null
- total_price: Float, not-null

Order:

- order_id: Integer, primary key, auto-increment
- user_id: Integer, foreign key(User-user_id), not-null
- order_date: Text, default=datetime.now().date(), not-null

OrderItem:

- order_item_id: Integer, primary key, auto-increment
- oder_id: Integer, foreign key(Order.order_id), not-null
- product_id: Integer, foreign key(Product.product_id), not-null
- category_id: Integer, not-null
- quantity: Integer, not-null
- total_price: Float, not-null

API Design

The project's RESTful API adheres to the OpenAPI Specifications and was implemented using Flask-Restful. It provides CRUD operations for all database tables and uses JSON Web Tokens (JWT) for secure authentication. For detailed information, please refer to the openapi.yaml file.

Video

To see the demo fo this web app, click here!