Grocery Store web Application

Introduction:

This is a multiuser application. Used for buying and selling groceries.

- Users can buy products from one or more sections, update profile, check order history.
- Store manager can add new products, update, delete products added by him, can request for deleting, updating and creating new sections.
- Admin can create new sections, verify the request raised by manager, authenticated manager profile.

The purpose of the application is to fulfill the requirements of one of the subjects of the BS in Data Science and application.

Description:

The application has three interfaces, one for user, and one for Store Manager and one for Admin.
Users can search for products based on categories, product name, brand name, product price, and appropriately add products to carts, and finally place orders.
The user can see the full profile, update the profile, and check past orders.
Admin and store manager can search for a product based on its name, quantity and categories. Admin can create, update, and delete categories, while store manager can create updates and delete products. New user/store manager registration, store manager can only login after admin approve him.

Technology used:

- Python flask for application logic
- Vue.js for UI and bootstrap for styling
- SQLite and SQLAlchemy for data storage and class mapping.
- Flask security too for authentication and RBAC authorization.
- Flask restful for API
- Browser LocalStorage for storing authentication token

Database Schema:

The following are the database tables with their attributes:

- **section** (*id, name, image_url, active*)
- **product** (*id, name, brand, manufacturing_date, expiry_date, price, stock, image_url, section_id, manager_id, active*)
- **user** (*id, name, username, password, email, mobile, address, city, state, pin*)
- **role**(id,name, description)
- **role_user**(id,user_id,role_id)
- **cart** (*id, user_id, product_id, quantity*)
- **order** (*id, user_id, order_date, total_amount*)
- **order_item** (*id, product_name, brand, manufacturing_date, expiry_date, price, quantity, order_id*)
- **change_request_section**(id,request_type,section_id,name,image_url,manager_id,reason,status)

API

Using the flask_restful library, API endpoints were created for the User, section and product table to perform the CRUD operation.

References

- https://flask.palletsprojects.com/en/2.3.x/
- https://flask-restful.readthedocs.io/en/latest/
- https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/
- https://flask-security-too.readthedocs.io/en/stable/
- https://chat.openai.com
- https://getbootstrap.com/