Grocery Store V1 Web App Report

Author:

- **Name** - Riya Shah
- **Roll** - 22f1000952
- **emailid** - 22f1000952@ds.study.iitm.ac.in
- **About:** Hey there! this is Riya Shah, a 21 year old web technologies enthusiast. I am also an undergrad student at IIT Madras BS Degree (Programming and Data Science). Interested in Development, AI, Data Science and Programming.

In addition, I'm a youthful and dynamic individual who constantly seeks to unearth something fresh every single day. I like playing games, listening music, visiting new places.

Description

The Grocery Store Web App is a web-based application designed to display product information to users also **creation**, **deletion**, and **deletion** of products, prices by the Admin. The purpose of this report is to provide an overview of the app, including its features, functionality, and potential areas for improvement.

Frameworks used in the project

- ***Flask***:- for backend of the application
- ***Flask-Login***:- for implementing RBAC authentication and authorization.
- ***Jinja2***:- For templates
- ***bootstrap***:- For templates

Tools and Technologies

These are tools and technologies to develop Ticket Show Web App. These include:

- ***Bootstrap***:- for styling and aesthetics of the application
- ***requests***:- Requests is a popular Python library used for making HTTP requests to APIs, websites, and other web services.
- ***os***:- for some operation related with files directory in the application
- ***session***:- Flask extension supports Server-side session to our application.
- ***redirect***:-used to redirect a user to another endpoint using a specified URL and assign a specified status code.
- ***request***:- used to handle HTTP requests and responses.
- ***render_template***:- used to render html templates based on the Jinja2 engine that is found in the application's templates folder.
- ***Flask-sqlalchemy***:- used to create database schema and tables using SQLAlchemy with Flask by providing defaults and helpers.

Database Schema

1. ***Relations:*** There are seven tables in the database namely Product, UserProduct, Category, User , Order. There is one to many relationship product and category, means one category can have many products and many to many relationship between user and product to implement add to cart feature.

Architecture and Features:

The project code is organised based on its utility in different files. I have named my project grocery-store. Inside this folder there are four folder including application, db_directory, static, templates and files main.py and requirements.txt. Images in static folder, templates in template forlder. Report folder with the report.pdf file, demo video and file to setup and run this project on windows.

Routers used in category_view.py

- @app.route("/",methods=['GET'])
- @app.route("/authorizing")
- @app.route("/admin", methods=['GET', 'POST'])
- @app.route("/admin/products/under/string:cat_name")
- @app.route("/add_category", methods=['GET', 'POST'])
- @app.route("/add_product", methods=['GET', 'POST'])
- @app.route("/update_product/int:product_id", methods=['GET', 'POST'])
- @app.route("/delete_product/int:product_id", methods=['GET', 'POST'])
- @app.route("/update_category/int:category_id", methods=['GET', 'POST'])
- @app.route("/delete_category/int:category_id", methods=['GET', 'POST'])
- @app.route("/admin")

Routers used in order_view.py

- @app.route("/user")
- @app.route('/mycart')
- @app.route('/pay', methods=['GET', 'POST'])

Routers used in product_view.py

- @app.route("/user")
- @app.route('/mycart')
- @app.route('/pay', methods=['GET', 'POST'])

Routers used in user_view.py

- @app.route("/user")
- @app.route('/mycart')
- @app.route('/pay', methods=['GET', 'POST'])

A short demo video link is here
https://drive.google.com/file/d/1MywGukCyWAGP0XO5JVknH47nJBICbqUe/view?usp=sharing .