IIT Madras
Project Title: GroceryStore
Course: Modern Application Development II – Project
Course Code: BSCCS2006P
Submitted by: Kanishk Mishra
Roll no.: 2Ff1006627

PROJECT REPORT

Introduction:

The **GroceryStore** is a web-based application developed using Vue.js and Flask, a Python-based web framework. The objective of this application is to provide a platform for users to purchase products across various categories. This project emphasizes the dynamic (asynchronous) capabilities of a web application.

Overview:

This project consists of four major components:

- Category Management- *Creation, Modification & Deletion*
- Product Management- *Creation, Modification & Deletion*
- Product Purchase
- Asynchronous Jobs- *Daily Reminders, Monthly Reports, CSV Download*

Additionally, several other components have been implemented to enrich the user experience. These include User/Store-Manager/Admin role-based access control (RBAC), login/signup processes, sending and approving requests, product and category search functionalities, a shopping cart feature, API performance and caching, product reviews, graphs depicting popular categories, and aesthetics.

Models:

The system is designed using a Model-View-Controller (MVC) architecture. There are six models used in this application: **User, Role, Category, Product, Rate as well as a Purchase model** to store information about user purchases.

1. User: Stores user's (Admin/Users/Store-Manager) credentials and their relationships with other tables like- Role, Category & Product.
2. Role: Store roles like -Admin, Users & Store-Manager their relationships.
3. Category & Product: Stores category & products details respectively.
4. Rate: Stores the product ratings (Star value- end user's input).
5. Purchase: Stores the records of purchases made by the end users.

Overall System Design:

The application follows a client-server architecture. The frontend is built using Vue, a widely used open-source JavaScript framework for developing user interfaces and single-page applications. Meanwhile, the backend is constructed in Python, utilizing the SQLAlchemy ORM and stored within an SQLite database.

The frontend handles most controls, while the server-side business logic is managed through the Flask framework using APIs. These APIs exclusively transmit data to the frontend upon request. To fortify the structure, additional Flask packages like Flask-Security-Too and Restful are integrated.

Python-Jinja usage is minimized, primarily reserved for generating monthly reports and reminder emails. No Flask controller automatically sends data to the frontend for HTML page rendering.

For executing backend tasks and maintaining an uncluttered server, Celery is employed for task queuing, with assistance from Redis-Server. Redis serves a dual role as both a server and a temporary data store, enabling caching functionality.

Functionalities:

- Users need to log in or sign up to access the application beyond basic product viewing.
- Admins have direct control over creating, updating, or deleting categories, while store managers must request these actions from the admin.
- Store managers can sign up but require admin approval for access, which can be revoked by the admin later.
- Store managers handle direct management (creation, update, deletion) of products.
- Store managers have the option to download a product details & stock CSV report from their dashboard.
- Product search functionality is accessible for both store managers and customers.
- Customers can use a shopping cart to store products for later purchase, retaining items even after logging out, based on product availability in inventory.
- Purchases can include items from various categories.
- Non-active users receive daily email reminders to visit the app for special offers, while buyers receive monthly expenditure reports.

My Presentation Video Link:

https://drive.google.com/file/d/1KnQvIeLW85F3PuPSA_C9XC82dFLIECPb/view?usp=sharing