FRESHCART APPLICATION

Name -	Mahak Thakre
Roll No:	21f2000845
Email Id:	21f2000845@ds.study.iitm.ac.in
About:	I'm from Balaghat, Madhya Pradesh. I finished my B.Sc. in Computer Science, and now I'm studying Data Science and Applications at IITM as a standalone course.

Description:

The project involves developing a web application using Flask, Celery, and Vue CLI. The application manages product information, generates monthly reports, sends reminders, and exports data to CSV files.

Technologies used:

- Flask: A Python web framework for backend development.
- Celery: An asynchronous task queue/job queue system.
- Vue CLI: A command-line interface for rapid Vue.js development.
- CSS: For frontend designing
- Flask-Security: An extension for managing security-related features.
- Flask-CORS: An extension for handling Cross-Origin Resource Sharing.
- Redis: Used as a message broker and result backend for Celery.
- SQLite: A lightweight, file-based database used for data storage.

The purpose of using these technologies is to create a scalable and efficient web application that handles background tasks, user authentication, and provides a seamless user experience.

DB Schema Design:

1. **User**: Represents users of the system with attributes like email, username, password, and roles.
2. **Role:** Defines user roles with a name and description.
3. **Request**: Stores user requests with approval status and requester's information.
4. **Category:** Represents product categories with a name and a relationship to products.

5. **Category_Requests:** Stores requests related to category creation with user and approval status.

6. **Category_Update_Request:** Manages requests for updating category details with user, approval status, and old category name.

7. **Category_Delete_Request:** Handles requests for deleting categories with user, approval status, and category details.

8. **Product:** Represents products with attributes like name, manufacturing and expiry dates, price, stock, and a relationship to categories and users.

9. **Cart:** Stores user-specific shopping cart details with product information and quantity.

10. **BoughtProducts:** Records details of purchased products including user, product, quantity, name, purchase date, and amount.

API Design:

The API facilitates user and manager sign-ups, user data retrieval, and admin dashboard access. It manages category operations, allowing addition, retrieval, modification, and deletion, with support for category requests and approvals. The product management section enables the addition, retrieval, modification, and deletion of products within specified categories. Users can interact with their dashboard, purchase products, manage their cart, and perform cart-based purchases. The API emphasizes role-based access control, ensuring secure operations. Authentication and authorization are crucial aspects to safeguard data and functionality, fostering a robust and secure e-commerce environment.Yaml File is present in Project under backend folder.

Architecture and Features:

FreshCart utilizes a Model-View-Controller (MVC) architecture, comprising three main components:

Model: Defined in models.py, it outlines the database schema using SQLAlchemy models for User, Manager, Category, Product, Cart, Requests, Roles and Bought Products. These models define the structure of the application's data and interact with the database. They handle tasks such as querying, updating, and ensuring data integrity.

View: The View component encompasses the templates and frontend components built using Vue CLI. These templates define the presentation layer and are responsible for rendering the user interface. Vue components interact with the backend through APIs, ensuring a responsive and dynamic user experience.

Controller: The Controller component is implemented using Flask, managing the flow of data between the Model and View. Flask routes serve as controllers, handling incoming requests, processing data from the models, and passing the relevant information to the views. Additionally, Celery tasks serve as controllers for background processes, such as sending reminders and generating reports.

Features: The project follows a modular structure with Flask controllers handling backend logic and Vue components managing the frontend. Templates are organized for clarity. Key features include user authentication, background tasks for reminders and reports, and exporting product data to CSV files. Additional features include monthly progress reports and automated reminders for inactive users

Video link: **https://drive.google.com/file/d/1XMJzLhbjcCL9-WtGkYSLHtOEiCdSlf65/view?usp=drive_link**