**Sahil Sayani**
21f1003203
21f1003203@ds.study.iitm.ac.in

# Grocers - Grocery Store V2

**IITM - MAD 2 Project**

## OVERVIEW

This project involves the development of a comprehensive grocery store application using Flask and its extensions, SQLAlchemy, Celery, and Redis. Flask serves as the backend server for API handling, while Flask-JWT-Extended provides user authorization using JWT tokens. SQLAlchemy is utilized as the database engine, and Celery offers asynchronous task queues for sending reminders and order summaries.

## TECHNOLOGIES USED:

1. Flask: A lightweight and flexible Python web framework used as the backend server for handling APIs. Its simplicity and extensibility make it ideal for building scalable web applications.

2. Flask-JWT-Extended: This extension adds support for JSON Web Tokens (JWT) in Flask applications, providing a secure and standardized method for user authentication and authorization.

3. Flask-SQLAlchemy: An ORM (Object-Relational Mapping) extension for Flask that facilitates the interaction with SQL databases. It simplifies database operations like creating, reading, updating, and deleting records.

4. Celery: A distributed task queue system used for handling asynchronous tasks such as sending email notifications, reminders, and processing background tasks. It enhances the application's performance by offloading resource-intensive tasks.

5. Redis: An in-memory data structure store, used as a database, cache, and message broker. It offers high performance and is particularly effective in handling tasks like caching and queuing in web applications.

6. Vue.js: A progressive JavaScript framework which has been used in this project to create dynamic and responsive front-end components.

## SCHEMA DESIGN

The schema design for the project includes several models:

1. User: Represents each user of the application. It includes fields like **id** (a unique identifier), **name**, **email** (unique for each user), **password**, and a reference to **role** to signify the user's role in the system.

2. Role: Defines different user roles. Each role has an **id** and a **name** (like '**admin**', '**customer**', etc.).

3. Product: Stores product details, including **id**, **name**, **price**, **available_quantity**, **category_id** (linking to the Category model), and an **image** URL.

4. Category: Holds information about product categories. Each category has an **id**, **name** (unique and not nullable), and a boolean **active** status to indicate if it's currently in use.

5. Cart: Represents a user's shopping cart. It has an **id**, **user_id** (linking to the User model), and a relationship with **CartItem**.

6. CartItem: Details items in a shopping cart. It includes **id**, **user_id**, **cart_id** (linking to the Cart model), **product_id** (linking to the Product model), and quantity of the product.

## Architecture and Features

The application's architecture includes models.py, containing database model classes and CRUD operations, and app.py, hosting various API endpoints and configurations. Key features include role-based access control, customer shopping pages, and dashboards for managers and admins. Celery is used extensively for asynchronous tasks like sending reminders, order summaries, and stock reports to users and managers. Redis functions as the message broker, ensuring efficient data handling.

Project Demo : [https://youtu.be/TLQDMEnlhaI](https://youtu.be/TLQDMEnlhaI)