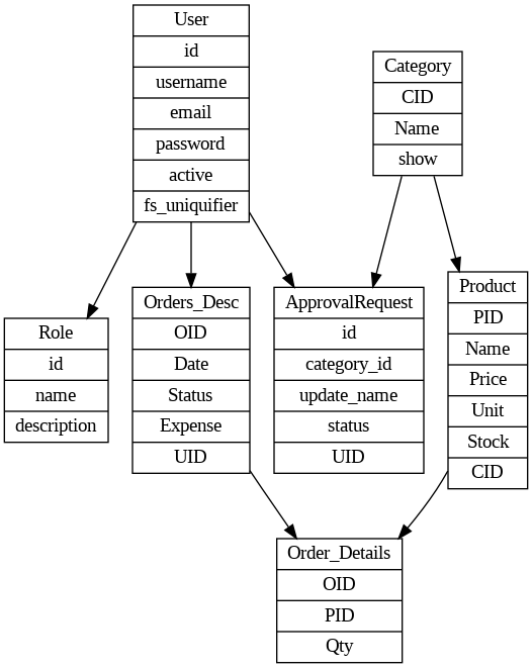
Description:

The Grocery Store Management System is a web application designed to facilitate the efficient management of a grocery store. It includes user authentication with role-based access control, allowing administrators, store managers, and buyers to perform specific tasks. Users can manage product categories, individual products, and place orders, with a comprehensive order tracking system. The app features a request and approval system for category modifications, as well as periodic reporting and reminders sent via email. Leveraging caching mechanisms and scheduled jobs, it ensures optimal performance and timely communication, making grocery store operations more streamlined and user-friendly.

Database Schema:

1. **User**: Represents users of the system with attributes like username, email, password, and roles.
2. **Role**: Defines roles that users can have, such as 'admin' or 'buyer', with a description.
3. **ApprovalRequest**: Stores requests for category updates or deletions, including the requester's ID and the request status.
4. **Category**: Represents product categories with a unique name, an ID, and a flag indicating whether it should be shown.
5. **Product**: Describes individual products with details like name, price, unit, stock, and the category it belongs to.
6. **Orders_Desc**: Represents orders made by users, including order date, status, total expense, and the user ID.



7. **Order_Details**: Stores details of each order, specifying the product ID, quantity, and linking back to the order it belongs to.

Technologies used:

1. **Backend Framework:** Flask
2. **Database:** SQLAlchemy (with SQLite for Development) - used as the Object-Relational Mapping (ORM) tool for interacting with the database.
3. **User Authentication and Authorization: Flask-Security** - used for user authentication and authorization.
4. **Asynchronous Task Queue: Celery** - integrated into your application for handling asynchronous tasks, such as sending daily reminders.
5. **Task Scheduling: Celery Beat-** a daily reminder using the **add_periodic_task** method.
6. **Frontend:** Vue.js
7. **Styling:** Bootstrap

8. **Data Serialization: Flask-RESTful -** used for building RESTful APIs and serializing/deserializing data.
9. **Caching: Flask-Caching** - configured it to use Redis
10. **Background Jobs: Flask-CeleryExt (Celery for Flask)** - used for handling background jobs.
11. **Excel Export: Flask-Excel** - integrated for handling Excel file responses.
12. **HTML Templates** - for rendering views.
13. **Frontend Styling: Custom CSS** - styling the frontend.

Features implemented:

1. User Authentication and Authorization: User signup and login using Flask Security or JWT-based Token Authentication.

2. Admin and Store Manager Roles: Separate forms for admin and store manager login.

3. Store Manager Signup and Approval: Store manager signup requests are sent to the admin for approval.

4. Section/Category Management (Admin): CRUD operations for sections/categories. Requests from store managers for category changes need approval from admin.

5. Product Management (Store Manager): CRUD operations for products. Allocate sections/categories while creating products.

6. Product Search: Implement search functionalities based on category, product, price, etc.

7. Shopping Cart: Allow users to add multiple products to the cart and buy products

9. Daily Reminder Jobs and Monthly Activity Report: Schedule daily reminders for users who haven't visited/bought anything. Generate a monthly progress report and as an email on the first day of the month.

11. Export as CSV (Store Managers): Export Sales summary details as a csv

12. Performance and Caching: Optimize API performance.

13. Recommended (graded): Design well-crafted PDF reports. Create a single responsive UI for both mobile and desktop.

Link to Video:

https://drive.google.com/file/d/1-4S05T_4uqx0QwVlSE8cMwiZP3PXMtYN/view?usp=sharing