Report

Author

Siddhardh Devulapalli
21f2000579
21f2000579@ds.study.iitm.ac.in
Tech-enthusiast with great inclination towards Software Development & AI and its applications in the near future.

Description

The project's focus is on building the backend of a grocery store application using lightweight python framework like Flask, Flask_restful, Flask_sqlalchemy, Flask_security and HTML, CSS, Bootstrap, JS (VueJS) for rendering the front-end. The project has admin level access to perform CRUD on categories, manager level access to perform CRUD on products and user level access to purchase categories/products.

Frameworks

The following are the technologies used followed by the purpose of usage:

Backend:

- Flask: Lightweight, simple and flexible python framework used for web-development of small to medium scale apps.
    - Flask object: Used to setup configurations and routes to different URL paths.
    - Request: Used to handle HTTP methods and provides attributes to access data of the response body.
- Flask_sqlalchemy: helps in working with databases using SQL Alchemy as an ORM for python.
    - SQLAlchemy: provides ORM layer to define Python classes that represent database tables.
- Flask_restful: helps in creating RESTful APIs quickly and efficiently.
- Flask_security: helps in implementing RBAC
- Celery: it's queue system for asynchronous processing of jobs
- Redis: key-value style database used for storing jobs in the queue, end-result of the jobs and endpoint cache.
- Cache: used to cache the frontend until a specified to timeout to increase performance

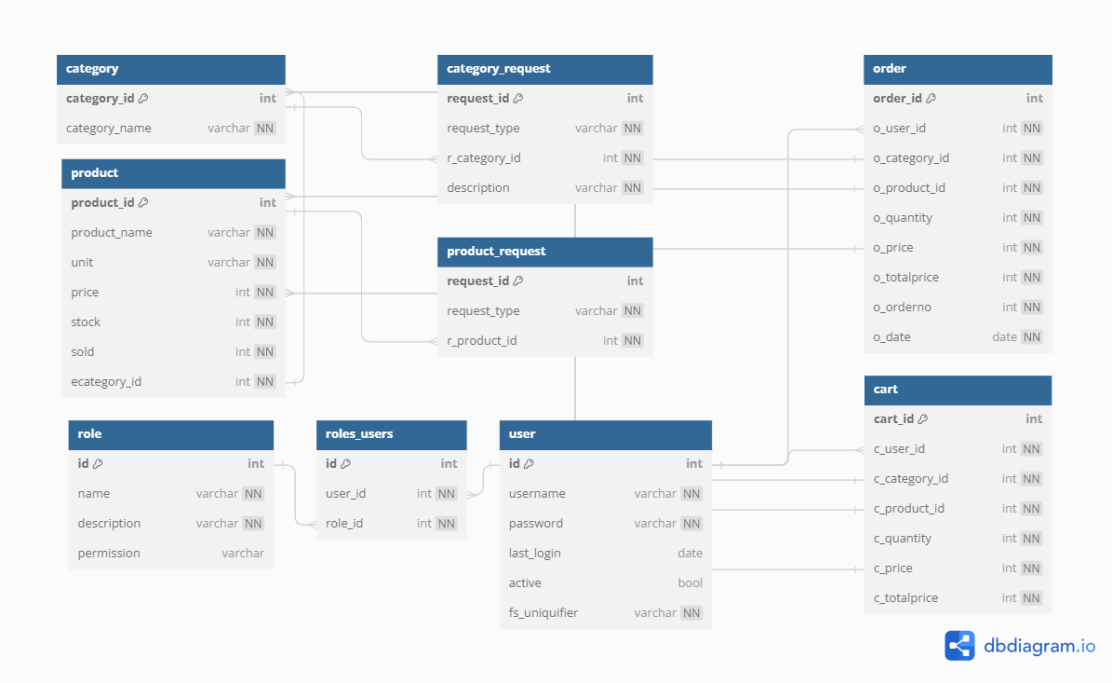
Frontend:

- HTML: used for developing applications in web and website templates.
- CSS: used for modifying the look and format of a web page.
- Bootstrap: built on top of HTML CSS to create responsive templates and designs
- JS (VueJS): JavaScript Framework for building front-end/UI of an application.

DB Schema

- The database has one table for users, one for roles and one for user_roles which defines the relationship between them.
- One table for category and one for product. There's a foreign key **ecategory_id** to link categories and products.
- One table for category requests and one for product requests. These requests are made by managers to admins. These are linked to the respective categories/ products tables using foreign keys.
- One table for cart and one for order to separate out temporary vs permanent changes in the database due to product purchases. Cart and order are linked to product, category, customer tables using foreign keys.

The following is the schema for the database:



API Design

API was created for CRUD operations of Category/section, Product and Cart. It was implemented using flask-restful library. Three classes, one for Category, one for Product and the other for Cart have been created. Appropriate status codes and status messages have been defined for different CRUD operations of each section.

Architecture and Features

The root project folder consists of Project report and Code. The code has the files app.py to configure the app and setup the controllers. It also has models.py for defining Python classes as database tables using SQL Alchemy. We link both app and model by importing model into app.py. The database created is present in the instance folder. The base html

template to be rendered is present in the template folder. The JS components which use VueJS CDN style are present in static/js folder. Each JS file is used for rendering different UIs for admin, manager, user.

The following are the features implemented:
Core:

- User/ Manager Login. Implemented using RBAC of flask_security.
- SignUp for Manager and User (Manager signup needs to be approved by admin)
- Category, Product and Cart (CRUD) for Admin, Manager, User. Implemented using flask_restful API.
- Search for Product/Category for User. Implemented using JS filters.
- Displays all the categories/products, buy multiple products from multiple categories for multiple users and show their cart value to be paid. Implemented using foreign key constraints in the database.
- Shows out of stock categories/products.
- Daily reminders to be sent to users (who have not logged/bought anything on that day). Implemented using celery jobs (workers, beat)
- Monthly active report to users. Summary of products bought in a month. Implemented using celery jobs (workers, beat)
- Download Products CSV. Implemented using celery jobs for asynchronous processing.
- Added Caching wherever possible to improve performance and reduce website latency. Implemented using flask_cache

Additional features:

- Form validation both in VueJS and at backend in flask routes before inserting into database.
- Images for Categories and Products. Styling and Aesthetics for all the pages. Implemented using Bootstrap for responsive website.

Video

Project Walkthrough –
https://drive.google.com/file/d/1cVDE-3-GSV6bM-02_VwE9nrpT7qc96tE/view?usp=sharing