Project Report

Author

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

Description

The project focuses on building the backend of a grocery store application using lightweight python framework like Flask and HTML, CSS and Bootstrap for rendering the front-end. The project has admin level access to add/edit categories/products and user level access to purchase categories/products.

Technologies used

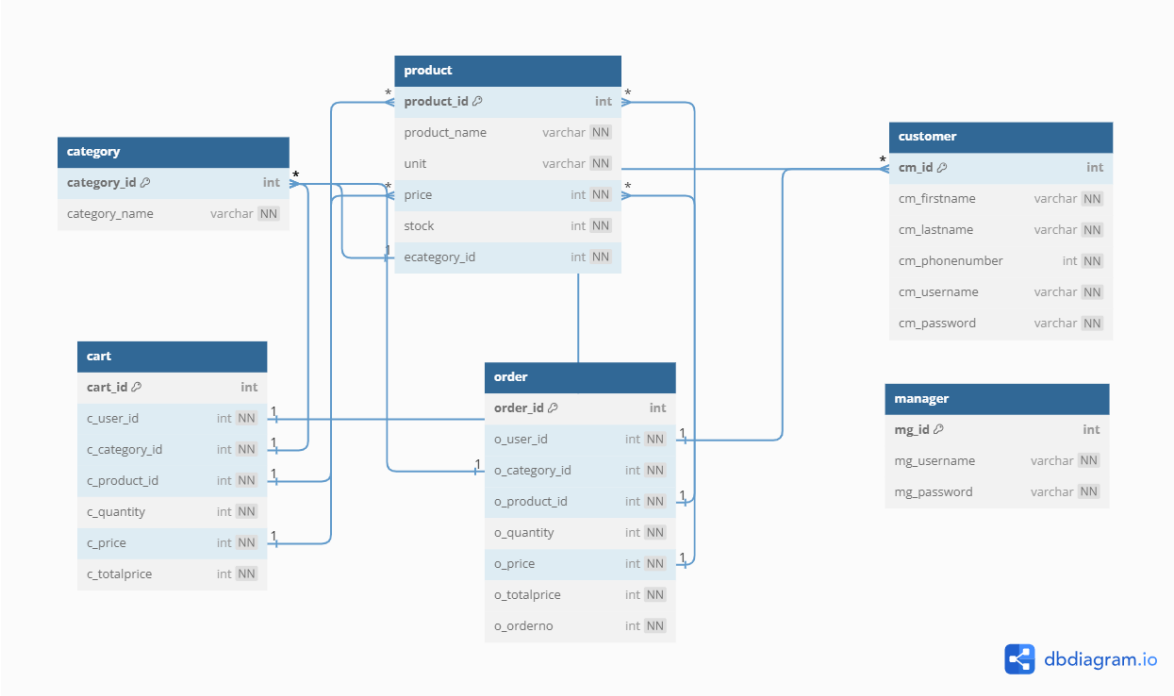
The following are the technologies used followed by the purpose of usage:

- 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.
    - Render template: templating engine to render dynamic html templates.
    - Redirect: used to redirect to specific URLs after routing.
    - url_for: Used to dynamically generate routes based on route functions defined.
- 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.

DB Schema Design

- The database has one table for customers, one for admin.
- One table for category and one for product. There's a foreign key **ecategory_id** to link categories and products.
- 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 and Product. It was implemented using flask-restful library. Two classes, one for Category and the other for Product 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 model.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 html templates to be rendered for each user input are present in the template folder. Each html file in template folder has a Bootstrap CDN for styling.

The following are the features implemented:
Default:

- User/ Manager Login. Implemented using HTTP post methods.
- Category and Product Management (CRUD) for Admin. Implemented using HTTP responses and flask_sqlalchemy database insertion in route functions.
- Search for Product/Category for both Admin & User. Implemented using HTTP response.
- 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.

Additional features:

- Form validation both in HTML attributes and at backend in flask routes before inserting into database.
- Summary of Categories bought, Products bought and valuable Customers at Admin level.
- Styling and Aesthetics for all the pages. Implemented using Bootstrap.

Video

Project Walkthrough – https://drive.google.com/file/d/1Lb6oXwWD_xO11CNiM55ag4Mi82bpVInM/view?usp=sharing