Online Grocery Store Management:

Author:

Name: Shivayokeshwari A
Roll. no: 21f1000415
E. mail: 21f1000415@ds.study.iitm.ac.in

Hi I am Shivayokeshwari, I am pursuing my online degree in data science and programming in IITM. By developing this web application I have gained more knowledge about web development.

Description:

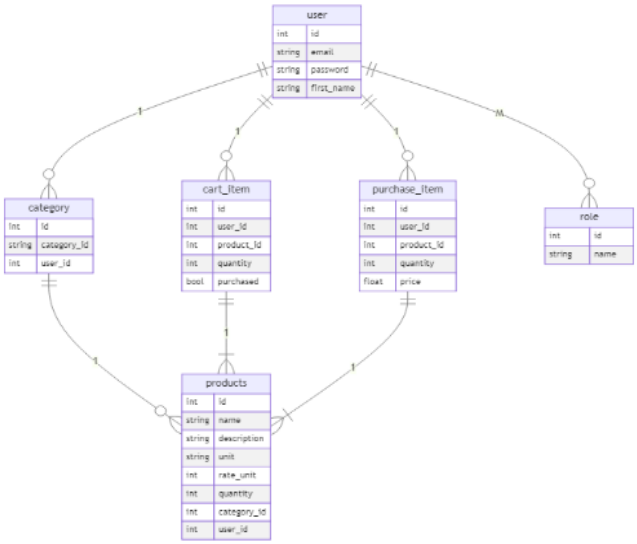
Online Grocery Shop is a Flask-based online store with user authentication and authorization using Flask-Login and Flask-Security. It allows users to manage categories and products through API endpoints. Users can register, log in, and perform CRUD operations on their own categories and products. The project is organized with separate controllers for API routes and templates for rendering HTML pages. It includes models for defining the database schema with tables for users, roles, categories, products, cart items, and purchase items.

Explanation for Technologies:

Flask: It is a micro web framework for Python used for building web applications ang APIs
Flask-RESTful: Flask-RESTful is an extension for Flask that simplifies the creation of RESTful APIs.
SQLAlchemy: SQLAlchemy is used for database connection. It is a popular Orm library.
SQLite: SQLite is used for database management system in my project.
Jinja2: Jinja2 template engine used to render dynamic HTML templates.
Flask-CORS: It is a extension used for handling Cross-Origin Resource Sharing in web application, allowing communication between different domains.
WTForms: It is used for form validation and rendering.
JSON: It is used for data interchange between frontend and backend.
HTML/CSS: It is used for creating and styling the webpage.
Bootstrap: It is used for frontend-end framework.
Flask-login: It is used to manage my login.
Flask-security: It is a extension used upon flask-login, the main purpose using this login is for role-based login system.

DB Schema Design:

I have created 6 tables namely **User, Roles, Category, Products, CartItems** and **PurchaseItems.**
To describe the relation between the tables I have attached a flow chart.



User: One-to-many with CartItem, PurchaseItem, Category, and many-to-many with Role.
Category: One-to-many with Products.
Products: Many-to-one with Category, and one-to-many with CartItem and PurchaseItem.

API DESIGN:

The API includes the following elements:

Category API: It provides endpoints to create, retrieve, update, and delete categories. Categories are associated with a user ID, and the API ensures that users can only manage categories they own.

Product API: It provides endpoints to create, retrieve, update, and delete products. Products are associated with a category, and the API ensures that products can only be managed within the context of their corresponding category and user.

DisplayAPI: It provides enpoints to retrieve all the categories and products.

Architecture and Features:

The project follows the Model-View-Controller (MVC) architecture, where models are defined using SQLAlchemy in models.py, controllers with API routes and logic are implemented in views.py, and templates reside in the templates directory, written in HTML with Jinja2 templating. The API endpoints are defined as Python functions in views.py, handling data flow between the back-end and front-end. The static assets like CSS and JavaScript are stored in the static directory. This organizational structure ensures a clear separation of concerns, making the codebase maintainable and scalable.

Online Grocery Shop
│ instances
|__database.db
│ website
|__static
|__templates
|__base.html etc.,
|__ __init__.py,auth.py, models.py, views.py
│ requirements.txt
│ main.py

All the controllers are saved in views.py and auth.py, models.py stores the database model. And main.py will run the app. Templates has the html files in it,requirements.txt has the extension that are installed in the terminal.

Features:

User Authentication: Users can register and log in securely to access personalized features.
Role-Based Authorization: Different roles (e.g., admin, user) have different levels of access and permissions.
Category Management: Users can create, view, update, and delete categories for their products.
Product Management: Users can add, view, update, and delete products within their categories.
Search Functionality: Users can search for products based on category, product name, or rate.
Shopping Cart: Users can add products to their cart, view the cart contents, and purchase items.
Purchase History: Users can view their purchase history, including details of purchased items.
API Endpoints: The project provides API endpoints for performing CRUD operations on categories and products.
Database Management: The project uses a relational database to store user data, categories, and products.
Responsive Design: The frontend templates are designed to be responsive and accessible across devices.
Error Handling: Proper error handling is implemented to provide informative responses to users.
Back Button: A back button is included to easily navigate back to the home page.
File Structure: The project is well-organized with clear separation of controllers, templates, and models.

Video link:

https://drive.google.com/file/d/1tZp65hQeLMcQXvXEbwOuXrem3HUU-0eG/view?usp=drive_link