.

Description

I had to develop a Grocery store web application by using HTML,CSS for frontend and Flask for backend.Here I created a separate login for admin and user and new user can also register.Admin can create,update and delete category as well as products.User can add to cart and order any product.

Technologies used

Flask: A Python framework for building web applications.
SQLAlchemy: An Object-Relational Mapping (ORM) library for Python, used it for interacting with database.
SQLite: A relational database management system.I used it as a database because it is a very powerful tool for small and medium scale applications.
Flask-CORS: A Flask extension for handling Cross-Origin Resource Sharing (CORS) headers.I used it for securely handling cross origin requests.
Flask-RESTful: An extension for creating RESTful APIs in Flask.Created APIs using this.
Werkzeug: A utility library for handling authentication and password hashing.
Python: The programming language used for building the application.

HTML/CSS: For front-end to make web pages.

API Design

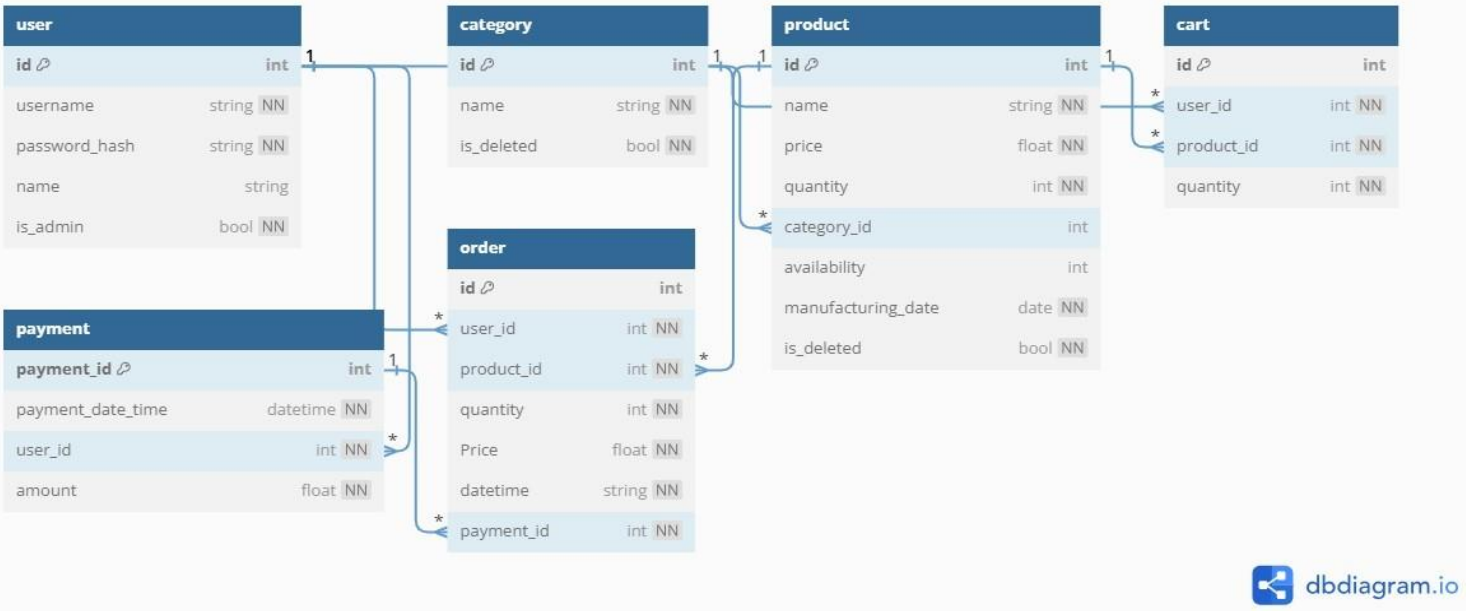
I have created a Flask RESTful API with two main resources: Api_category and Api_products.

#) Api_category is for CRUD operations of Category :- GET(Retrieves list of all categories) , POST(Creates a new category) , PUT(Update a category),DELETE(Delete a category).

#)Api_products is for CRUD operations on Product :- GET(Retrieves all products),POST(Creates a new product), PUT(Updates a product), DELETE(Delete a product).

Both resources use “reqparse” to parse and validate request data. They also interact with the database using SQLAlchemy models.

DB Schema Design



User has a one-to-many relationship with the **Cart** table (one user can have multiple items in their cart) and also with **Payment** table (one user can have multiple payments/orders) **. Category** has a one-to-many relationship with the **Product** table (one category can have multiple products). **Product** has one-to-many relationships with both the **Cart** table (one product can be in multiple users' carts) and the **Order** table (one product can be in multiple orders). **Payment** has a one-to-many relationship with the **Order** table (one payment can be associated with multiple orders).

Architecture and Features

In the project folder there is a Templates folder in which there are all HTML templates, inside Templates there are two more folders Category and Product which contain all the category and product related templates respectively.There is “app.py” file which contains all the controllers and the configuration settings for the Flask application. In “models.py” file all the database models are described using Flask-SQLAlchemy. The “api_resources.py” contains code for creating a RESTful API using Flask and Flask-RESTful for managing categories and products. “requirements.txt” contains a list of Python package dependencies along with their respective versions.

Features include that admin can create,update and delete Category as well as product associated with each category.User can add a product to cart,delete a product from cart, and order a product added to cart.When a user orders a product added to cart cart becomes empty.We can create,update and delete categories and products by API also and controllers also.