Author

Name: AZFAR FAHEEM MUSTAFA
Roll No: 21F2000478
Student Mail: 21f2000478@ds.study.iitm.ac.in

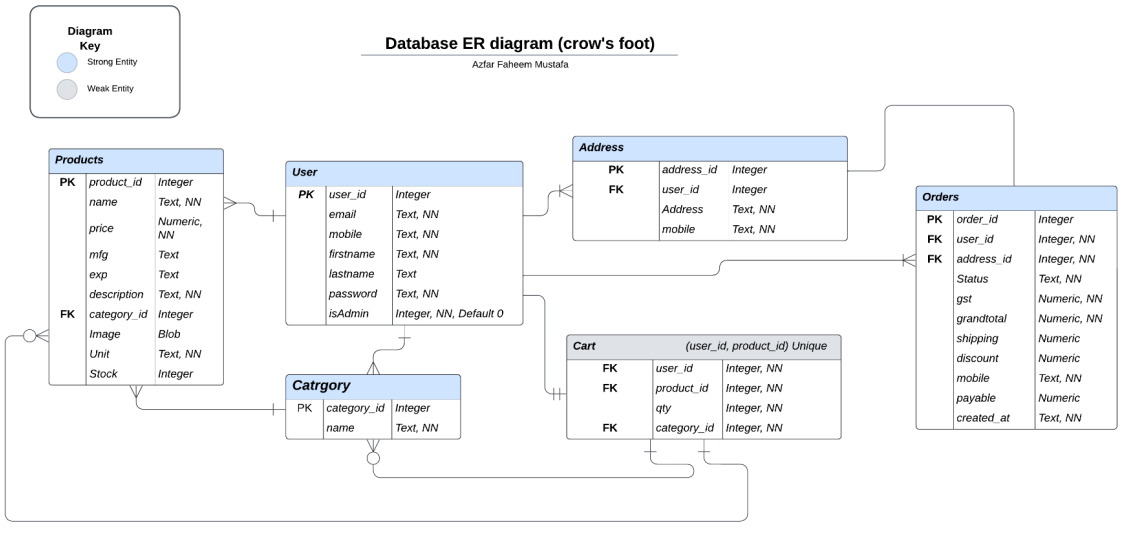
B.tech Computer Science graduate from B.S Abdur Rahman Crescent Institute of Science and Technology with excellent academic record and keen interest and practical exposure in the field of AI, NLP and cross platform mobile application development.

Description

A grocery store app using the Flask framework, Jinja2 templating engine, and SQLite database must be created. The app should allow users to browse products by section/category, add products to their cart, checkout and complete their order. The store manager should be able to add/edit/delete products that will then instantaneously reflect in the shop. Each function of the app must be carefully understood and the wireframe analyzed in depth before implementation to avoid any logical errors.

Technologies used

- **Flask, Jinja2 and SQLite:** Flask is a microframework for Python web development, Jinja2 is a templating engine for Python and SQLite is a lightweight database engine that is embedded in Python all three are lightweight and easy to use, making it the perfect choice for this project.
- **Flask-RESTful:** Flask-RESTful is an extension for Flask that was used because it makes it easy to create RESTful APIs.
- **CORS:** CORS is a mechanism that allows web applications to make cross-origin requests. This will allow users to access the API's from other domains.
- **Requests:** It was used to make HTTP requests to the CRUD API's in the application to get the sections/products to display and for various other purposes.
- **JSON:** A lightweight data-interchange format which was used to serialize and deserialize data in the application, making it easy to pass data to and receive data from the API.
- **OS**: Part of the Python standard library, It provides access to operating system functionality used primarily to generate the app secret key, so that sessions can be enabled to make the app multiuser compatible.
- **base64:** A standard encoding scheme that was used to encode binary data into ASCII characters in order to handle the SQLite blob data type in product image fetch & display.
- **datetime:** Part of the Python standard library, it provides classes for representing dates and times so that they can be stored in the database during order creation and updation.



DB Schema Design

The database is designed the way it is to make it easy to store and retrieve information about the products, categories, and orders in the grocery store. All tables except cart have a primary key, which ensures that each row in the table is uniquely identified. Foreign key relations are also defined, allowing the database to efficiently retrieve information from multiple tables. As one user can only have one cart and one product(even multiple quantities of) can be in the cart only once, (user_id,product_id) Unique is defined as a constraint. All the entities and their relationships can be seen in the above ER diagram.

API Design

I've created an API for Products, Categories, Users and orders. This includes getting and displaying the products/categories, adding products/categories, logging the user in, updating user profile/password, creating a new user and placing orders.

For each of these purposes I've created a helper function(ex. get_products()) that has the SQL statements to send and retrieve data from the database. The api itself is added as a resource using the flask_restful extension. For post functions such as adding products I get the data from a HTML form, convert it to json using json.dumps() and then send it with the api call as json, In some cases as text or even as part of the url and the api accordingly handles the data and passes it to the helper function, gets a JSON response and sends it back.

Architectures and Features

The project is organized into the following directories:

- app.py: This is the main Flask application file. It contains the code that initializes the Flask app and defines the routes. In my case it also contains all the API's and API helper functions.
- templates: This directory contains the HTML templates for the project. Templates are used to render the HTML pages that are displayed to the user.
- Static: This directory contains the images, the CSS, the bootstrap files and icons that have been used in the project.
- Store.db: The SQLite database with all the tables required for this project.

Default features:

- User registration and login, Admin login, Product browsing, Cart, Checkout, Product and Inventory management(for manager). CRUD on categories and products.

Additional features:

- User can view orders and details such as date of order, order status etc
- Managers can update stocks of products and it also auto deducted on order.
- Managers can also sort products by category, making it easy to manage.
- Application of coupon code during checkout, NEW10 for 10% discount for new users.
- Styling and aesthetics has been done using CSS and Bootstrap.

Video: **Click Here** or visit
https://drive.google.com/file/d/1NjkbPQlWfgQKJMqGYfosc48wpdSF3S1I/view?usp=sharing