Project Report: Flask Grocery Store App.

Abdul Ahad Rauf
21f3002590
21f3002590@ds.study.iitm.ac.in
8948887405

Abstract:

The Flask Grocery Store Application is a web-based platform designed to provide users with an online grocery shopping experience. The application allows users to browse and search for various grocery items, add items to their cart, place orders, and view order history. It's like a virtual store for groceries on the internet. The project utilizes the Flask framework, SQLAlchemy for database management, Bcrypt for password encryption and various other technologies to implement essential features of an e-commerce grocery store.

Table of Contents:

1. Introduction:

I, Abdul Ahad Rauf; am pursuing BS Degree in Data Science and Programming from Indian Institute of Technology, Madras. I like to code and I have made this project for my MAD-1 project.

Flask Grocery Store Application aims to provide a user-friendly platform for online grocery shopping. Users can conveniently browse through various grocery items, add them to their cart, and place orders from the comfort of their homes. The project is built using the Flask web framework, allowing for efficient development and deployment of the application.

2. Technologies Used:

- Flask: Web framework for building the application.
- SQLAlchemy: Database ORM for managing data.
- Flask-Login: User authentication and session management.
- Flask-Bcrypt: Password management and Hashing.
- SQLite: Database management system.
- HTML, CSS, JavaScript: Front-end development (Bootstrap).
- Matplotlib: Data visualization for analysis.

3. Features and Functionalities:

- User Registration and Login
- Product Browsing and Searching
- Cart Management
- Order Placement and History
- Admin Dashboard for Product Management
- Data Analysis and Visualization (Admin Only)

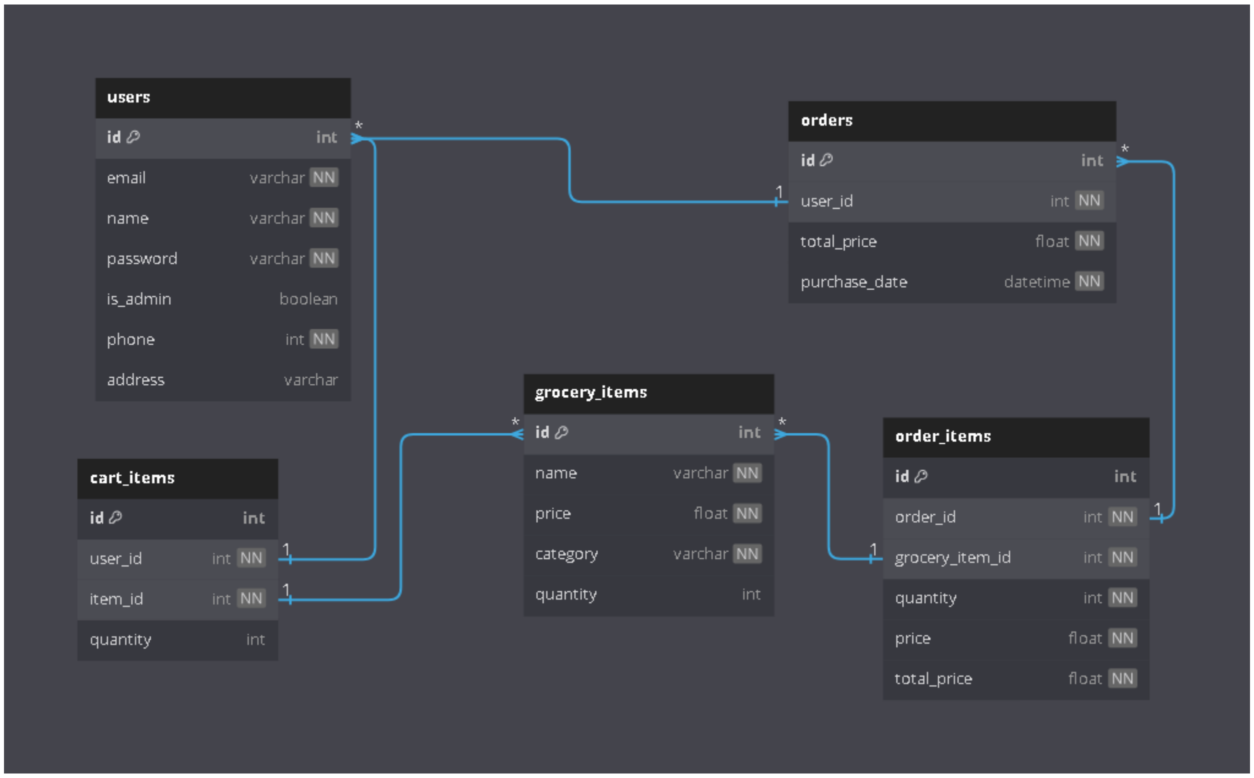
4. Video Link:

https://drive.google.com/file/d/1k22hdY5as-Mui1NxF9prtbGYy0abYIoy/view?usp=drive_link

5. System Architecture:

- APIs as well as a file to test them are located within the folder.
- The app configuration is written in app.py
- The templates are in templates folder
- Database configurations are in modles.py
- Database is located in the instance folder as database.sqlite3
- All media content, CSS files are in static folder

DB SCHMEA: Flask SQLAlchemy was used for the Database Management. The Db Schema is briefed below:



6. More Details:

- **API**

 - /api/grocery_item - Contains API calls to get, put, delete an Grocery Item [CRUD operations].

 - /api/user - Contains API call for getting users.

- **APPLICATION**

 - (app.py) - Main code file, contains routes for all pages.

 - (models.py) - Contains Database schemas.

- **ROUTES**

 - (/) - Home page / Product page . Also acts like the user dashboard when user signs in.

 - (/user/login) - Validates the id password and lets the user login or redirect back to login page.

 - (/user/signup) - Lets users Signup based on various checks (email, phone must be unqiue), (entries must not be blank) etc etc.

 - (/admin/dashboard) - Admin can CRUD all Grocery items.

 - (/admin/analysis) - Plots cumulative and total sales line graph. Creates Pie charts based on Category and each item.

 - (/logout) - logs the current user out.

- **STATIC**

 - (style.css) - Stylesheet.

 - (artbreeder-image.png) - Image for Background.

 - (cumulative_revenue.png) - Image for cumulative_revenue ([Graph] creates new everytime).

 - (daily_sales.png) - Image for daily_sales ([Graph] creates new everytime).

 - (Grocery1.jpg) - Image for home page carousel.

- (Grocery2.jpg) - Image for home page carousel.

- (Grocery3.jpg) - Image for home page carousel.

- (pchart_category.png) - Image for pchart_category (PIE [Chart] creates new everytime).

- (pchart_items.png) - Image for pchart_items (PIE [Chart] creates new everytime).

➢ **<u>TEMPLATES</u>**

- (admin_dashboard.html) - admin_dashboard where he can CRUD all Grocery Items, has all validations.

- (admin_update.html) - admin_update can updated previous items.

- (analysis.html) - shows the analysis by plotting charts.

- (base.html) - base template for all the child templates.

- (cart.html) - User cart that automatically clears once user makes the purchase.

- (checkout.html) - Final checkout page to make payment and add/change details.

- (product_page.html) - product_page/ Home page, diplays the data in table form.

- (user_login.html) - user_login with proper validations.

- (user_signup.html) - user can signup with proper validations.

7. API Documentation:

The application provides APIs for CRUD operations on grocery items. The following API endpoints are available:

- `GET /api/grocery_items`: Retrieve a list of all grocery items.

- `GET /api/grocery_items/<item_id>`: Retrieve details of a specific grocery item.

- `GET /api/user`: Retrieve details of users in the store.

- `POST /api/grocery_items`: Add a new grocery item (Admin Only).

- `PUT /api/grocery_items/<item_id>`: Update details of a grocery item (Admin Only).

- `DELETE /api/grocery_items/<item_id>`: Delete a grocery item (Admin Only).

8. References:

- Flask Documentation: https://flask.palletsprojects.com/

- SQLAlchemy Documentation: https://docs.sqlalchemy.org/

- Flask-Login Documentation: https://flask-login.readthedocs.io/

- Matplotlib Documentation: https://matplotlib.org/stable/contents.html