**Grocery Store:**

Report created by:

Name: **Ramprasad R**

Roll Number: **21f1003573**

Email: **21f1003573@ds.study.iitm.ac.in**

**Technology used:**

* Flask for application code
* Jinja2 templates + Bootstrap for HTML generation and styling
* SQLite for data storage
* SQLAlchemy for database connectivity.

**Description:**

Given the problem statement which is to create a grocery shop based app , the basic intuition developed based on the various requirements posted in the document can be, used to implement the same.

Registration page was provided for new users to register . Some background validation was performed such that no two users can have the same mail address . Once registration is done user can login using their credentials which includes email id and password. They can enter the dashboard page and choose to search based on categories or products. Based on the search query the page will be loaded dynamically . After that user can choose the products he/she wishes to buy and add the same to cart . Each user will have different cart associated with their account , if they wish to checkout they can . Similarly if a user feels that they don’t need a product in their cart they can remove them and proceed . Session has been created and connected in all the routes. So if someone tries to navigate to the dashboard route they can only do so if their email address exists in the session .

Another thing is a user can logout of the app if he wishes to.

Now moving on to the admin/manager part ,they can manage the stores. They can create new categories provide data for a category like name ,type etc and edit the categories if they wish to.They can also delete the categories if they feel they don’t need it anymore.For each category corresponding products can be added . Just like categories these products too can be edited or can be deleted based on the manager and admin. Product can have attributes like name,manufacture date ,cost,available quantity in stock etc. All these can be edited.Just like users admin side also has a login page and session management associated with it only if an admin is present in a session, the category or the product can be edited or deleted.

**Architecture and Features:**

The main python file used here which serves as the **server** for the application to run on is **app.py.** All the required functions and their corresponding routes have been configured and coded in this python file. The routes used here include:

* /  **:**  The index page of the application .
* **/test\_db :** This page/route was created just for experimental purposes to check whether a db connection is successful or not.
* **/user\_login :** The login page which ask’s for credentials for the user to log into. It also contains pointer for a new user to register .
* **/user dashboard:** This page is the dashboard page designed specifically for users.Here user can search for categories or products add products to cart . It contains pointer’s for both logout facility and cart facility .
* **/cart :** This route is where the user can take a look at their chosen products and can choose to either checkout or buy the products . The total cost of all item’s is displayed for user’s,providing insights.
* **/logout :** This is the route used to delete session of the user and logout that user.
* **/admin\_login :** This route is specifically designed for admin’s to login which asks for admin details like username and password and a session is created for that particular admin/manager.
* **/admin\_dashboard:** This route allows inventory management tools like creating a category, editing the category and deleting it if deemed necessary. Similarly product(‘s) can be created under each section/category. One category can have multiple products associated with it. Just like categories products too can be deleted and edited based on the need.
* **Finally the main function where the application is loaded and run.**

**DB SCHEMA DESIGN:**

The tables and the basic schema used here is:

**User,Admin,Category,Product,Cart.**

Here user table contains the date for user’s who have registered themselves as user’s of the application.It contains columns like email address,name,mobile number,password and importantly cart.All these values are set as not nullable and cart is set as one-many relationship between the table user and cart ie one user can have multiple rows correspondingly in cart table.

Admin table contains column’s like username and password.Both these are set as not nullable.

Category table consists of columns like id,name and type where id is primary key and basically all attributes are not nullable.

Product table contains columns like name,price,available quantity,category\_id and category,cart..Here category\_id is a foreign key referencing to categories table.Cart column references to cart table in a way such that its one-many.

Cart table consists of columns like user\_email,product\_id,quantity,product\_name,price,availability.Here all values are not nullable user\_email is a foreign key referencing to user’s table.product\_id is another foreign key referencing to product’s table.availability denotes whether a product is in stock or not.

**API DESIGN:**

The API was created and developed for category tables.It allows to perform CRUD operations on that table.

Curl command was used to test for its ingenuity in the command prompt and it worked fine. A .yaml file was created to store the routes,endpoints VERB definition .This was then read in the app.py file and all the dependencies were installed.The input was provided in JSON format and output too was received in JSON format.

**VIDEO PRESENTATION LINK:**

<https://drive.google.com/file/d/1GWdHIDg2lN0THtpoHlBz5xt_1q70eaHN/view?usp=drive_link>