Author

Name: Ayush Singh
Student roll number: 21f1003617
Student mail id: 21f1003617@ds.study.iitm.ac.in

Hi, I am Ayush Singh, a third-year engineering student at DSI, Bengaluru. I like coding and development. I also love reading books and exploring new domains.

Description
We need to create a multiuser app which provides functionalities such as Buy to user, and Add or delete to the admin. We need to create a database to store the data of products and categories of those products. The main required feature is whenever an admin adds a product the same should reflect to user. Whenever a user logout, its cart should automatically be deleted. Extra features such as a search bar to search the product is also required.
Technologies used
These are the technologies used:
● flask - for main flask app
● flask_restful - for REST API implementation
● flask_sqlalchemy - add support to image upload, code syntax highlighting and more
● sqlalchemy - Object Relational Mapper in python for SQL
● flask-Session – Managing user sessions and cart implementation

DB Schema Design
The scheme is quite simple, total 4 tables are used. The user and admin table stores the information about user and admin with minimal attributes such as user_id, user_name and password. The product table has autoincremented prod_id which is the primary key as well, it also has a foreign key cat_id which connects it to category table. The category table has cat_id as its primary key which is also auto incremented.

user		CREATE TABLE "user" ("user_id" INTEGER, "username" TEXT NOT NULL, "password" TEXT NOT NULL, PRIMARY KEY("user_id" AUTOINCREMENT))
user_id	INTEGER	"user_id" INTEGER
username	TEXT	"username" TEXT NOT NULL
password	TEXT	"password" TEXT NOT NULL
admin		CREATE TABLE "admin" ("admin_id" INTEGER, "username" TEXT NOT NULL, "password" TEXT NOT NULL, PRIMARY KEY("admin_id" AUTOINCREMENT))
admin_id	INTEGER	"admin_id" INTEGER,
username	TEXT	"username" TEXT NOT NULL,
password	TEXT	password" TEXT NOT NULL
product		CREATE TABLE "product" ("prod_id" INTEGER, "cat_id" INTEGER NOT NULL, "name" TEXT NOT NULL, "unit" INTEGER NOT NULL, "rate" INTEGER NOT NULL, "quantity" INTEGER NOT NULL, PRIMARY KEY("prod_id" AUTOINCREMENT), FOREIGN KEY("cat_id") REFERENCES "category"("cat_id") ON DELETE CASCADE)
prod_id	INTEGER	"prod_id" INTEGER
cat_id	INTEGER	"cat_id" INTEGER NOT NULL
name	TEXT	"name" TEXT NOT NULL,
unit	INTEGER	"unit" INTEGER NOT NULL,
rate	INTEGER	"rate" INTEGER NOT NULL
quantity	INTEGER	"quantity" INTEGER NOT NULL,
category		CREATE TABLE "category" ("cat_id" INTEGER, "cat_name" TEXT NOT NULL, PRIMARY KEY("cat_id" AUTOINCREMENT))
cat_id	INTEGER	"cat_id" INTEGER
cat_name	TEXT	"cat_name" TEXT NOT NULL

API Design

The REST API has been created for GET, PUT, DELETE, POST methods which allow us to perform CRUD operations on the User, Deck, Card entities.

CategoryApi:

METHOD	**API endpoints**	**Request Parameters**	**Response Parameters**
GET	/api/category/<string:name>	-	Name,id
PUT	/api/category/<string:name>	name	New_name
DELETE	/api/category/<string:name>	-	Deleted_product
POST	/api/category	name	Cat_name,cat_id

ProductApi:

METHOD	**API endpoints**	**Request Parameters**	**Response Parameters**
GET	**/api/product/<string:name>**	**-**	**Name,unit,rate,quantity,cat_id**
PUT	**/api/product/<string:name>**	**Cat_id,pname,unit,rate,quantity**	**Cat_id,pname,unit,rate,quantity**
DELETE	**/api/product/<string:name>**	**-**	**Deleted_product**
POST	**/api/product**	**Cat_id,pname,unit,rate,quantity**	**Name,unit**

AdminApi:

METHOD	**API endpoints**	**Request Parameters**	**Response Parameters**
GET	**/api/admin/<string:name>**	**-**	**Admin_id,name,password**

Architecture and Features

The architecture of the project has been kept fairly simple and standard. The main code to run and set up the application is in main.py. All the dependencies are listed in requirements.txt. db_directory is the folder that contains database. The application folder contains all the main parts of the application, like controllers,api, config, models. The static folder has the static files which are needed. The templates folder has all the jinja2 template files

Features implemented are Dashboard management, Secure login framework,User/admin management, cart management, Styling and Aesthetics and last but not least a search bar using standard ways and the technologies/dependencies listed above.

Video

Please watch this video for knowing more about the project.

https://drive.google.com/file/d/1LatbWIff-bEdP9fxOAyU31deGRpscMSg/view?usp=drive_link