Author

- Dhairy Kumar
- 22F2000893
- 22f2000893@ds.study.iiitm.ac.in
- I've interested in stargazing, nature, physics and a bit of history. And now one addition A.I.

Description

First to get start with home page and the build on the redirects & features of home page and the make all relevant routes, templates and the databases. If some functions are required, can be created simultaneously. Then move to the Api with basic auth for delete, update & add function.

Technologies Used

i. Flask: The framework used in making webapp
ii. Flask-Login: For managing the user sign-in
iii. SQLAlchemy: For managing the database
iv. Flask-HTTPAuth: For Api user verification
v. Flask-RESTful: For making restful Api
vi. tinydb: For storing Api user's details in unstructured form
vii. matplotlib: Creating the graphs
viii. Flask-Mail: Sending verification mail for password reset to users
ix. Flask-WTF, WTForms: Creating flask forms
x. ast, literal_eval: To convert str to original python literal [e.g., I used it to convert a list of lists sored in database as string back to its original structure for further computation.]
xi. PLI; Image: resize and save an image
xii. os: For getting the path to save and delete the image
xiii. secret: may some user upload image with same name, so it not conflicts with each other change name to some arbitrary name.
xiv. Datetime: To get the date and time
xv. Werkzeug: for securing the passwords
xvi. Jinja2: Creating template

DB Schema Design

I have use sqlite3 database and SQLAlchemy as ORM, model includes:

1) Users

Profile image can be left black and if left blank a default profile-pic provided.

```
❖     id = db.Column(db.Integer, primary_key=True)
❖     name = db.Column(db.String(50), nullable=False)
❖     email = db.Column(db.String(120), unique=True, nullable=False)
❖     username = db.Column(db.String(120), unique=True, nullable=False)
❖     password = db.Column(db.String(120), nullable=False)
❖     address = db.Column(db.String(255), nullable=False)
❖     image_file = db.Column(db.String(50), nullable=False,
default='/img/user/avator.png')
```

2) Products

```
❖ id = db.Column(db.Integer,primary_key=True)
❖ product_name = db.Column(db.String(50), unique=True, nullable=False)
❖ detail = db.Column(db.String(255))
❖ category = db.Column(db.String(50), nullable=False)
❖ price = db.Column(db.REAL, nullable=False)
❖ image = db.Column(db.String,default='/img/product/default.png')
❖ stock = db.Column(db.Integer, nullable=False)
❖ expiry = db.Column(db.DATE)
❖ unit = db.Column(db.String(20))
❖ section_name = db.Column(db.String(120))
```

in addition to user and product, models.py contains 3) orders, 4) section, 5) category, 6) cart, 7) contact, 8) admin, 9) store-manager, 10) approvals, 11) report and lastly 12) order-report schema

API

The REST Api is created on category, section and product. Role based operation, admin can only delete store manager can update and add, anyone can view

Category: Can only view

Section: CRUD operation, admin & manager can update and add but admin can only delete

Product: CRUD on product, admin & manager can update and add but admin can only delete

Architecture and Features

The python webapp is arrange in a package like structure:

Grocery

- all template in templates folder
- image, css and js file in static folder
- __init__.py file contains app initialization
- form.py file have all python form and validators in it
- model.py file have all models
- routes.py file have routes and function and also contains the api

instance

- app.db

api_user.json

- contains api user credentials

run.py

- responsible for running the app

Video

Link: http://youtu.be/u5x9ahliq3M