Project Report

Author

Name: Nomit Rawat, Roll no: 21f1005029, Email: 21f1005029@ds.study.iitm.ac.in,
I am currently a student at the foundation level. Working as Business analyst Tech Mahindra.

Note

To run the application install flask, flask_sqalchemy. And run the “app.py” file using the “python” command on the terminal.
Run the “/sample” url if running the application for the first time.

Description

Objective was to build a grocery store web application focusing on using Flask as the backend framework for the application. The idea is similar to Blink it and Big basket, with the additional feature of providing admin/manager based facilities on the website.

Technologies used

1. Flask: for request handling, rendering templates, defining views/routes to the application
2. Flask Sqlalchemy: defining models, doing query operations on the database, committing changes to the database.
3. Jinja: for templating, provides more flexibility to the html document.
4. Sqlite: for database tables

DB Schema Design

All the classes have an id field as the primary key.

1. User class
    a. uid: user id, integer, primary key
    b. username: string, unique, cannot have null value
    c. password: string, cannot have null value
    d. fname: firstname, string, cannot have null value
    e. lname: lastname, string, can have null value
2. Admin class
    a. aid: admin it, integer, primary key
    b. username: string, unique, cannot have null value
    c. password: string, cannot have null value
3. Category class
    a. cid: category id, integer, primary key
    b. cname: category name, string, cannot have null value
    c. Relationship defined on the product class

4. Product class (one to many relationship with Category)
    a. pid: product id, integer, primary key
    b. pname: product name, string, cannot have null value
    c. pcid: product category id, foreign key is cid (category id in the category table)
    d. pcount: product count, integer, cannot have null value
    e. pprice: product price, cannot have null value

The relationship defined in category helps for easier access of the "category of the product" from product instance itself.

API Design

The category field supports create, read, update and delete operations. In a similar fashion the products also have the same operations.

Architecture and Features

Architecture

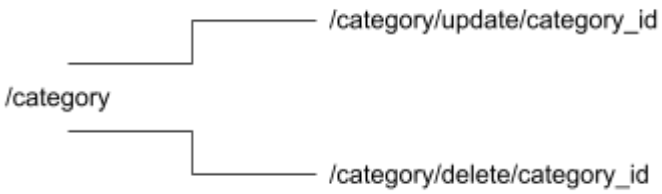
Following the recommended flask app structure.

1. The "model.py" containing the database schema related definitions is divided into a separate module from the "app.py" which contains all the views.
2. Templates folder is used to serve the html files.
3. Static folder contains the css.
4. The "Instance" folder has the database defined.
5. Venv contains the required python libraries used to build the application.

Features

1. An "admin" and a "miscellaneous" product category are created in the model at the start.
2. The flask app opens at the main page. Where person can
    a. **User login**: username and password are required fields in form. User should exist in the database or he/she can register himself/herself.
    b. **Admin login**: username and password are required fields in form. Admin cannot register, id is created at the start itself.
    If incorrect password or username is given a prompt is displayed for correct details.
    c. **Register user**: first name, username and password are required in form. Username should not exist in the database.
3. **Users**
    a. Can login and purchase items but the product count while purchasing cannot be negative.
    b. Buy products from different categories.
    c. Search for products and categories and see if they are available. The search at the backend matches the string.
4. **Admin** page gives an overview of the products and categories available in the website. And options to update products and categories separately.
5. Product category and management
    a. **Categories:**

i. can be added, renamed, deleted (except for “miscellaneous”). Same for products.
ii. If a category is deleted, products for that category move to “miscellaneous”.
iii. Delete endpoint only supports GET requests.
iv. Urls in the category looks as follows (same for product urls)



v. While creating a new category or updating them the form values are required.

b. **Products**:
   i. Create: count and price cannot be negative, a pop up is displayed showing either of them being negative. All the fields are required in the form.
   ii. Updating:
      1. All the fields are optional. If no values are given then the default value is taken.
      2. Negative price or count is not allowed
      3. The count cannot become negative while updating. A message is shown for this.

Video

https://drive.google.com/file/d/1aGolOSpu7OhfKmKvzqI9rNPYRnqWnm6P/view?usp=sharing