BOTTOMLESS BOWL GROCERY APP – PROJECT REPORT

Author

NAME - RITISH KUMAR DAS
ROLL NO. - 21f3000959
EMAIL ID - 21f3000959@ds.study.iitm.ac.in

An avid reader and a spiritual practitioner, my love for this subject is what keeps me grinding, through the pressure of doing two degrees together. I am pursuing my B. Tech in Electronics and Communication Engineering from Tezpur University. Currently I am in Diploma Level of the B. S in Data Science and Applications degree IIT Madras.

Description

To be able to complete this project one needs to implement the following -

1. Designing an effective database schema to work with, where only Admin and Store Manager can create/edit/delete categories and products as well as shop, whereas Users can only shop.

2. APIs and app routes to perform the database related and logical operations of the app.

3. Careful documentation of the APIs and other related frameworks for others to refer and learn.

Technologies used.

The different frameworks used are –

1. FLASK – A Python Micro Web Framework. Very useful and easy to implement Validation, App Routing, HTML Rendering etc.

2. FLASK SQLALCHEMY – It is a Flask extension, which helps the app to interact with the database efficiently and helps in maintaining the consistency of the database.

3. FLASK RESTful – Useful for creating REST APIs.

4. HTML, CSS (Bootstrap only) – Used for rendering the web pages, user interaction, aesthetics of the app and app styling.

5. JINJA – Used to apply certain Python logical functions inside html pages and for redirection to other pages links.

6. VUE.JS – For the UI/UX Design wherever user interaction is required.

7. REDIS-CELERY – For backend jobs and scheduling those jobs

8. FLASK-CORS – For Cross-Origin Service, where different vue components could communicate data among themselves.

9. FLASK-JWT-EXTEDNED – For RBAC Login and Tokenization.

DB Schema Design

The Database contains 5 tables –

1. Category:
    a. Category_ID: Integer, Primary Key, Autoincrement, Unique, Not Nullable
    b. Category_Name: String, Unique, Not Nullable
    c. products: relationship backreferencing to the Products table
2. Product:
    a. Product_ID: Integer, Not Nullable, Primary Key, Unique
    b. Category_Name: String, Foreign Key to Category_Name in Category table, Not Nullable
    c. Product_Name: String, Not Nullable, Unique
    d. Mfg_Date: String, Not Nullable
    e. Exp_Date: String, Not Nullable
    f. Rate_Per_Unit: Integer, Not Nullable
    g. Available_Quantity: Integer, Not Nullable
3. Role:
    a. id: Integer, Primary Key, Autoincrement, Not Nullable
    b. name: String, unique, nullable
    c. description: String
4. User:
    a. id: Integer, Primary Key, Autoincrement, Unique, Not Nullable
    b. email: String, unique, Not Nullable
    c. password: String, Not Nullable, Unique
    d. active: Integer, Not Nullable
    e. role: String, Foreign Key to name in Role table
5. Cart:
    a. Cart_ID: Integer, Primary Key, Autoincrement, Not Nullable
    b. User_ID: Integer, Foreign Key to id in User table
    c. Purchase Date: DateTime
    d. Cart_Items: JSON
    e. Total_Price: Integer
6. Request:
    a. id: Integer, Primary Key
    b. user_id: Integer, Not Nullable
    c. request_text: String, Not Nullable
    d. status: String
7. UserRequest:
    a. email: String, Primary Key, Unique, Not Nullable
    b. password: String, Not Nullable

API Design

Following are the APIs defined in the app –

1. Class CategoryAPI [For fetching, updating, deleting and creating categories]:
    a. /api/categories – Used to GET all the categories present or POST a new category.
    b. /api/categories/<int:category_id> - Used to PUT or DELETE an existing category.
2. Class ProductAPI [For fetching, updating, deleting and creating products]:
    a. /api/products - Used to GET all the products present or POST a new product.
    b. /api/products/<int:product_id> - Used to PUT or DELETE an existing product.

3. Class PurchaseAPI [Just to create a new purchase]
    a. /api/purchase – Used to create a new cart entry for a new purchase by a user.

Architecture and Features

1. The application files are present in the folder named 'application'. It contains files for api definition, validation, model definition, database initialization, configuration, jwt configurations, caching configurations.

2. The HTML templates are present in the folder named 'templates'. Very minimal CSS has been used, that too using Bootstrap.

3. App controllers are present in the main.py file.

4. There is also a readme file.

5. Features for Admin and Store Manager –
    a. Can create, update, delete categories (Only for Admin).
    b. Can create, update, delete products (Store Manager can delete products only on validation from Admin).
    c. There is a login form for them to log in.
    d. No registration form as Admins as are added when a new database is created by the app.
    e. Store Manager registrations need to be approved by the Admin.
    f. Store Manager can request Admin to create/edit/delete existing categories.

6. Features for Users –
    a. Can only shop products.
    b. There is a login form for Users to log in.
    c. There is a separate registration form for those whose details are not there in the database.
7. Others -
    a. There is a reminder that is sent to any user who does not shop anything in a day to visit the app.
    b. Monthly Activity Reports are sent to users via mail. [Have used proxy server for this]
    c. Store Managers can also see the overall Reports by clicking on downloadable link

Video –
https://drive.google.com/file/d/1v8dTDtEwIDjlBrdcA0VsAspbCosKFusY/view?usp=sharing