About the Author

Name: Bhavya Aditya
Roll no. 21f3001442
Email: 21f3001442@ds.study.iitm.ac.in
About: Based in Ghaziabad (Uttar Pradesh), I am a graduate in economics from the University of Delhi. Apart from a budding data scientist, I'm also a casual doodler.

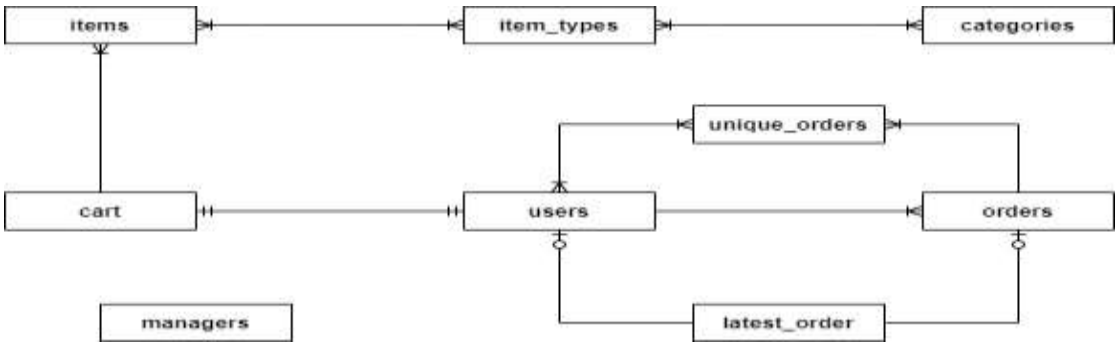
Description

The project has 2 parts: admin and user. Users are the buyers and admins are the sellers who control the product range. Users and admins have different functions and hence different controllers, but have products, categories, and orders data shared between them.

Technologies Used

Packages used: **Flask, Flask-SQLAlchemy, Jinja2, WTForms, Datetime**.These packages provide all utilities required for this application like database connection, HTML forms and validation, template rendering etc. For database, **SQLite** has been used. For admin analytics dashboard, **Chart JS** has been used as it easily integrated with HTML and enables creation of visually appealing graphs. For HTML frontend, **Bootstrap 5** has been used.

DB Schema Design



items
item_id (int) (PK)
item_name (text) (unique)
item_qty_avail (int)
item_price (int)
item_desc (text)
item_unit (text)
item_max_units_per_customer (int)

managers
mngr_id (int) (PK)
mngr_email (text) (unique)
mngr_password (text)
mngr_firstname (text)
mngr_midname (text)
mngr_lastname (text)
mngr_username (text) (unique)

unique_orders
order_id (int) (PK)
user_id (int)
order_total (int)
order_time (float)

item_types
cat_id (int) (PK) (FK)
item_id (int) (PK) (FK)

users
user_id (int) (PK)
user_email (text) (unique)
user_password (text)
user_firstname (text)
user_midname (text)
user_lastname (text)
user_username (text) (unique)
user_city (text)
user_pincode (int)
user_address (text)

orders
order_rowID (int) (PK)
order_id (int)
order_userID (int)
order_itemID (int)
order_qty (int)
order_price (qty)
order_time (float)
order_total (int)
order_unit (text)
order_itemName (text)

carts
cart_rowID (int) (PK)
cart_userID (int) (FK)
cart_itemID (int) (FK)
cart_qty (int)

latest_order
lo_id (int) (PK)
user_id (int)
order_id (int)

categories
cat_id (int) (PK)
cat_name (text)

The above database schema reduces the no. of columns per table to bare minimum and provides easy querying. A separate table is used to maintain each function of the app like cart, placing orders, product and category management, etc.

Architecture and Features

The app follows the MVC (Model Views Controller) paradigm. The models are in the **models.py** file, the controllers (basically the app URLs/routes and the functions therein) are in the **routes.py** and some forms used in routes in the **forms.py** folder. The part of the controller which runs the app and establishes configuration and connection with the database is in **app.py** with the database in the **appdb.db** file. Product images are stored in the "**/assets/uploads**" folder. All the Jinja2 templates are in the "**/templates**" folder.

Admin features:

1. Dashboard, which is the access point for all other features for the admin.
2. Category management:
    a. Create new category.
    b. Edit category – change name, add/delete products.
3. Product management:
    a. Create new product.
    b. Change product image
    c. Change product details
    d. Limit the quantity of a product which a single user can purchase.
4. **Inventory alert:** Shows out of stock products and low in stock products (those which have less inventory than the purchase limit imposed on the user).
5. **Analytics dashboard:** Shows top products and categories using visually stunning and easy to read charts.

User features:

1. Home page, which is the access point for all other user features.
2. View categories and products therein
3. Cart, where users can save multiple products across different categories and buy them together in a single order.
4. Order details where users can see the products ordered, their quantity and price, and the date and time when the order was placed.
5. Past orders, wherein the user can access all the orders made by him/her till date.
6. Out of stock feature, which automatically disables the buy option for a product which is out of stock. The user can still view the product though.

All features are implemented using separate URLs which have been created using Flask routing. To access the app, the user or the admin **must log in.** Separate login pages have been created for but are interlinked. A sign-up page is also there for new users.

Video

https://drive.google.com/file/d/1NNn6pMJX6ym8AhCIP03sgZnLnaTWEuyw/view?usp=sharing