DVP E-Commerce Project Set Up and Design Doc

Adeeb Khan, John Chandler, Jeff Chen, Tyler Chan

Set Up Optimal Git Ignore File (*for group members)

```
# Secrets
secrets.json

# Virtual environments
env_3.11.5
env_node_20.11.1

# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# Virtual environment (other potential names)
venv/
env/

# Django stuff:
*.log
*.pot
*.pyc
*.pyo
*.pyd
*.sqlite3

# Local development settings
local_settings.py

# IDEs and editors
.vscode/
.idea/

# Node
node_modules
```

Environment Set Up (MacOS)

*assuming pyenv versions are already installed
*** indicate steps that are NOT needed if pyenv already installed

curl https://pyenv.run | bash

export PATH="$HOME/.pyenv/bin:$PATH"
eval "$(pyenv init --path)"
eval "$(pyenv virtualenv-init -)"

pyenv update

pyenv install 3.11.5   ***
pyenv versions

pyenv local 3.11.5

python -m venv env_3.11.5

source env_3.11.5/bin/activate

pip install django
pip install psycopg2-binary
pip install jupyter ***done for testing graphs in notebooks
pip install pandas
pip install matplotlib
pip install nodeenv
pip install django_vite
pip install django_extensions
pip install requests #in case an error arises in a views.py about missing request

Django Set Up

mkdir requirements_env
touch main.in
touch dev.in

Inside dev.in:
-c main.txt

nodeenv
django-extensions
jupyter

jupyterlab

Inside main.in:
django
django_vite

Build packages by running:
pip install --upgrade pip-tools pip setuptools wheel
pip-compile --upgrade --generate-hashes --output-file requirements_env/main.txt requirements_env/main.in
pip-compile --upgrade --generate-hashes --output-file requirements_env/dev.txt requirements_env/dev.in

Install packages:
pip-sync requirements_env/main.txt requirements_env/dev.txt

Node Environment Set Up

1. Activate python virtual environment
2. Install nodejs 20.11.1 (the latest LTS(long-term) version at this time), anywhere you want.
   nodeenv --node=20.11.1 --prebuilt env_node_20.11.1
3. Deactivate your python => deactivate
4. Activate your node env
   source env_node_20.11.1/bin/activate

Design Doc

Alimama

Adeeb Khan, John Chandler, Tyler Chan, Jeff Chen

Project Structure (updated)

```
ecommerce (main Django folder)
|
├── ecommerce (Django project folder)
|   ├── settings.py
|   ├── urls.py
|   └── ...
|
├── core (Django app)
|   └── ...
|
├── shop (Django app)
|   ├── views.py
|   ├── urls.py
|   └── templates
|       └── shop
|           └── product_show.html
|
├── users (Django app)
|   └── ...
|
├── notebooks (Folder)
|   └── product_db.ipynb
|
└── vue_commerce (Vue.js project folder)
    ├── src
    |   ├── apps
    |   |   ├── product_show
    |   |   |   ├── product_show.js
    |   |   |   └── ProductShow.vue
    |   |   ├── favorite
    |   |   |   ├── favorites.js
    |   |   |   └── favorites.vue
    |   └── main.js
    └── vite.config.js
```

UX Overview

Component Interactions
On the Django end, there will be 3 apps: core, users, and shop
The core app will contain views and templates to properly configure the base of our project. This will include the home page, category page, and base.html.

The users app is in charge of providing a custom means of login and authentication for the project. Since in our project, users are being directly added to the database and are an integral part of providing an individualized past orders and fasvorites section, we are using Django's built in LoginView and CustomUserCreationForm to create abstract users in our models. All of this will be used to set up a log in, logout, and sign up features.

On the shop app, there will be the individual templates and views for the product detail, cart, checkout, favorite, and past order templates for our site. Essentially, a user will be able to navigate to a product, and choose to add some quantity of it to their cart and/or favorite the item to add it to their favorites list (remember each list varies per user). Once the user is satisfied with what they've added to their cart, they may go to the cart page to checkout an item which will then be added to their past orders list.

In the shop section, there will likely be Vue components implemented within the Django templates (e.g. forms for checkout, add to cart, etc.) These apps will be inside the vue_ecommerce/src/apps

How to run
WE ARE ASSUMING THE USER HAS PGADMIN AND IS WILLING TO LINK THEIR OWN DATABASE TO TEST

As such, we have put a secret_template.json file in the ecommerce inner project folder. Simply rename this file to be secrets.json and fill in the information for your own PostgreSQL database.

secrets.json Template

```
{
   "environment": "development",
   "ecommerce_url": "http://localhost:8000",
   "database_name": "YOURDATABASENAME",
   "database_user": "YOURDATABASEUSER",
   "database_pwd": "YOURDATABASEPW",
   "database_host": "localhost",
   "database_port": "5432",
   "vite_dev_server_port": "5173",
   "PAYPAL_CLIENT_ID": "YOURCLIENTID"
}
```

In order to make this website work as intended, you MUST use a client ID. It is unique for every user, and is automatically generated for each paypal account. In order to obtain your paypal client ID, navigate to the following link:
https://developer.paypal.com/dashboard/applications/sandbox
Once there, copy the Client ID and paste it inside of your secrets.json file.

In one terminal window, the user should navigate to the vue_ecommerce folder and run

In one terminal window, the user should navigate to the vue_ecommerce folder and run
npm install
npm run dev

In another terminal window, the user should navigate to the first ecommerce folder in their directory and run
python manage.py makemigrations
python manage.py migrate

In the folder notebooks, navigate to product_db.ipynb and run the notebook in order. If during the first cell any modules are missing when it is run, be sure to promptly pip install them into your python environment if you haven't already. Running this notebook will populate your Products database.

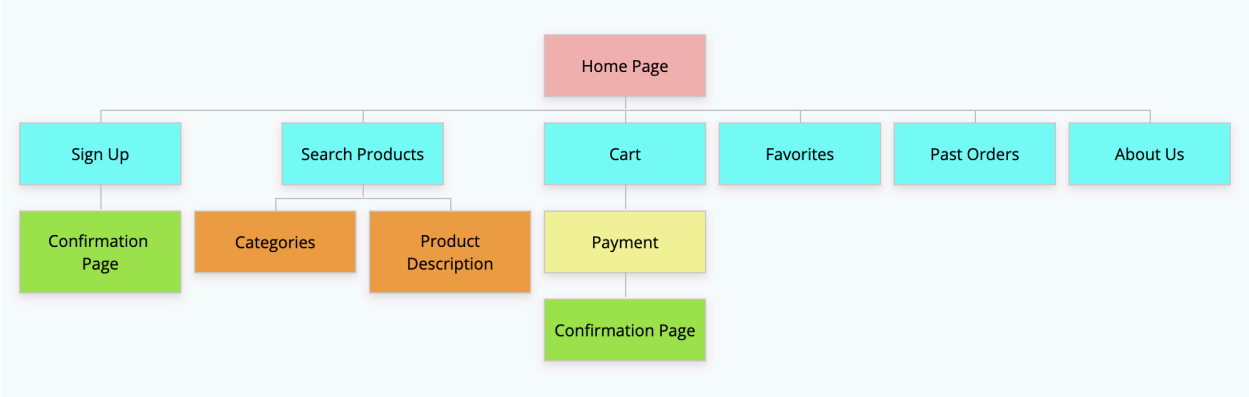
Now back in terminal
python manage.py runserver

The user should go to the URL prompted by this which is likely to look something like http://localhost:8000. Do note that this url is determined by what was put in the secrets.json file so if the user put a different specified url in secrets.json, the url will look different.

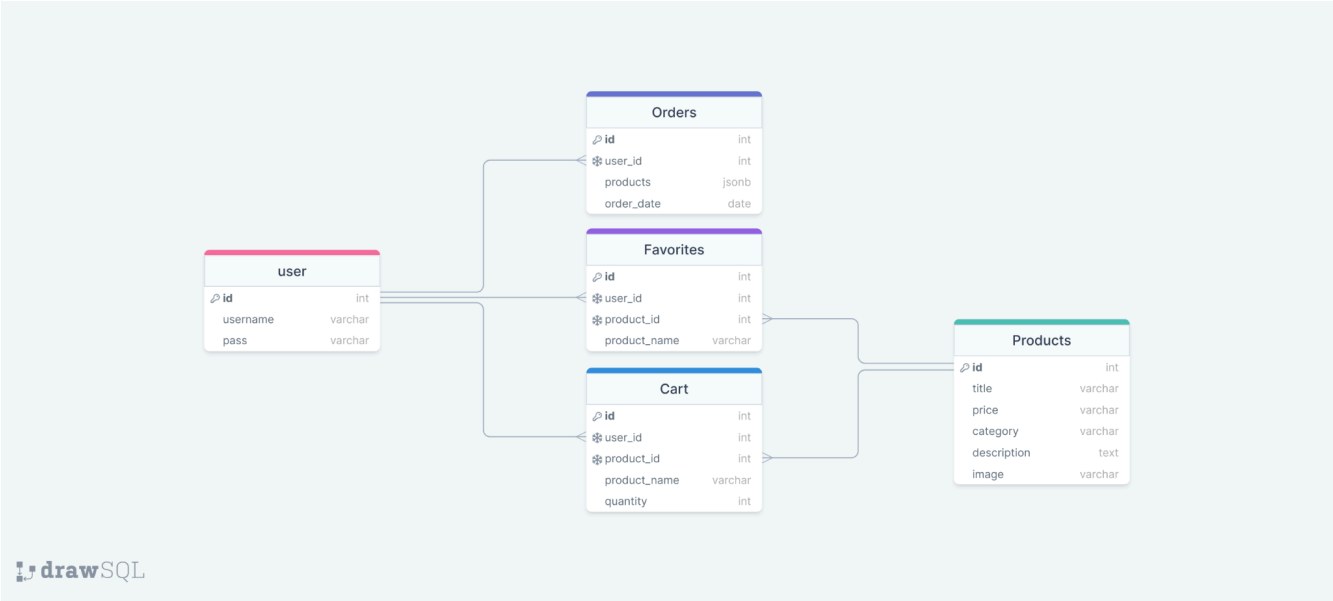
API in Use

https://fakestoreapi.com/docs
This is a public API and thus there will be no need to get a personal primary key. All fetch requests are done automatically with no need to configure anything on RapidAPI or make any payment plan. The notebook will however need to be configured.

Site Map



DB Diagram & Table Descriptions (revised)



1. **Users Table**

Fields
id (Primary Key, Integer, Auto-increment)
username (Unique, String, VARCHAR(255))
password (String, VARCHAR(255))

Relationships
One-to-Many with favorites
One-to-Many with orders
One-to-Many with cart_items

2. **Favorites Table**

Fields
id (Primary Key, Integer, Auto-increment)
user_id (Foreign Key, Integer, References users(id))
product_id (Foreign Key, Integer, References Product(id))
product_name (String, VARCHAR(255))

Relationships
Many-to-One with users
Many-to-One with Product

3. **Orders Table**

Fields
id (Primary Key, Integer, Auto-increment)
user_id (Foreign Key, Integer, References users(id))
order_date (Timestamp)
products (JSONB)

Relationships
Many-to-One with users

4. **Cart Table**

Fields
id (Primary Key, Integer, Auto-increment)
user_id (Foreign Key, Integer, References users(id))
product_id (Foreign Key, Integer, References Product(id))
product_name (String, VARCHAR(255))
quantity (Integer)

Relationships
Many-to-One with users
Many-to-One with Product

5. **Product Table**

Fields
id SERIAL PRIMARY KEY
title VARCHAR(255) NOT NULL
price VARCHAR(255) NOT NULL
category VARCHAR(255) NOT NULL
description TEXT
image: VARCHAR(255)

Relationships
One-to-Many with Cart
One-toMany with Favorites

Task Distributions

Adeeb Khan (Project Manager) - allocate tasks to group members; set up plan and structure for project; set up basic user authentication; set up Django and Vue layout; set up initial home and category pages

John Chandler - Assist with Django development for handling data in models, research API alternatives and data handling, design frontend components using Bootstrap and Vue

Tyler Chan - Configure Django and Vue; handle data transfer processes between Django and Vue; construct and implement Vue components into Django templates; design frontend with Vue

Jeff Chen - Analyze API data and configure Pandas and Matplotlib for data analysis; construct usable graphs that can be shown on UI for product detail pages; research alternative APIs for better data analysis

Basic Git (*for group members to avoid conflicts)

I have updated my own branch, how do I get main up to date with my own branch?

1. **Check out the main branch:**

git checkout main

2. **Pull the latest changes for the main branch:**

git pull origin main

3. **Merge the your-branch into the main branch:**

git merge your-branch

4. **Resolve any merge conflicts:**

If there are any merge conflicts, resolve them manually. After resolving the conflicts, you need to stage the changes and commit them:

git add .
git commit -m "Resolved merge conflicts while merging your-branch into main"

5. **Push the updated main branch to the remote repository:**

git push origin main

Main has been updated and I want to be up to date with main, how do I get my branch up to date with main?

1. **Check out to your branch:**

git checkout your-branch

2. **Pull the latest changes from the main branch into your branch:**

git pull origin main

3. **Push the updated main branch to the remote repository:**

git push origin your-branch