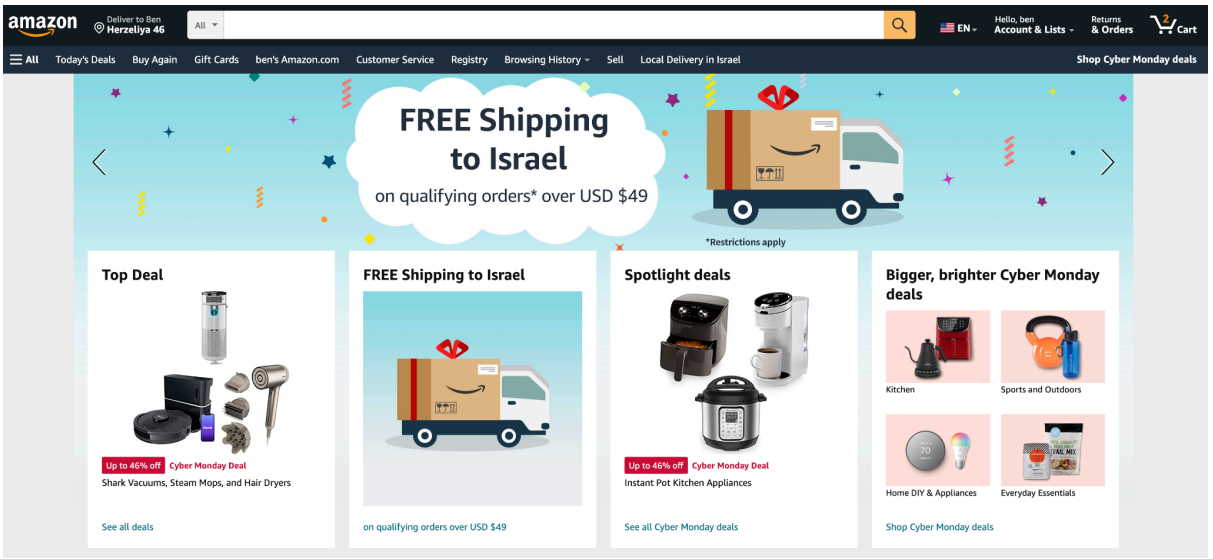
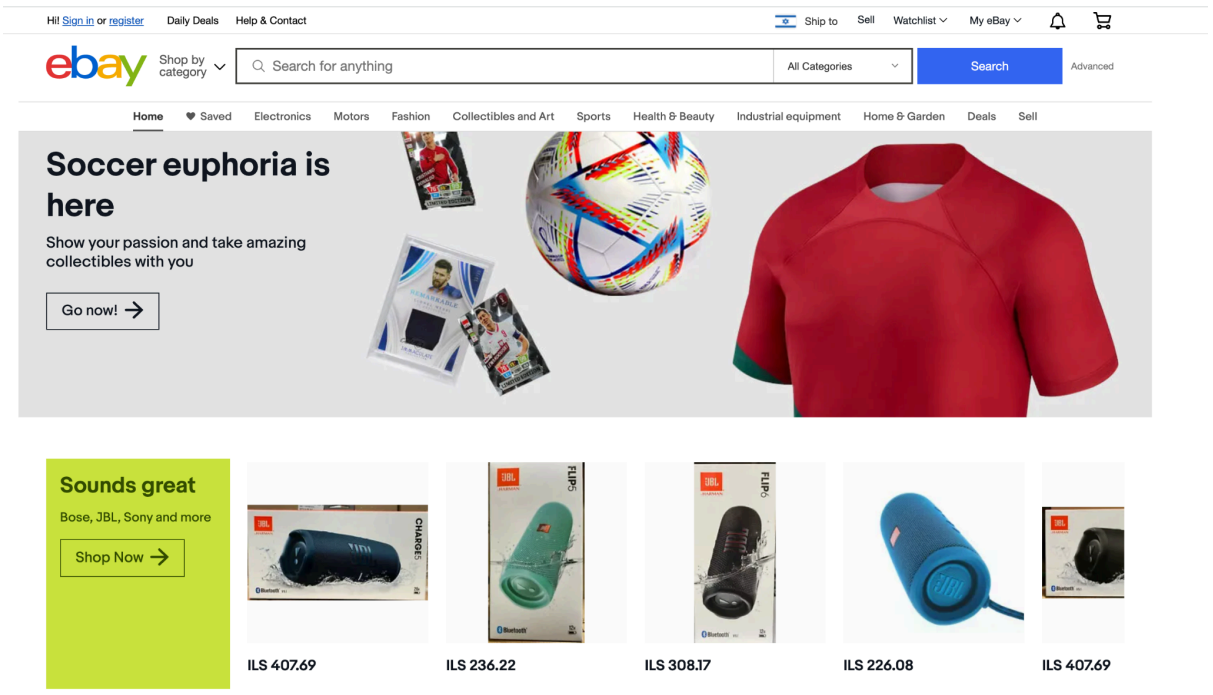
AI Final Project - Shopping Website

Instructions:

1. Your mission is to create a full end to end of a shopping website with AI capabilities.
   You can take examples from familiar websites such as Amazon, Ebay, Shufersal, Rami Levi, etc…
   Keep in mind that the project should focus on the AI capabilities and the backend side of the website. The UI is not the important part.





2. Your website should implement the following:
    a. Website pages:
        - Main page
        - Order page
        - Favorite items page
        - Chat assistant page

    b. **Main page** → The main page is the first page your users will see.
    This page should have the following ui components:
        - Main heading → Your shopping website heading
        - Searching area
        - Available items grid

    c. **Show available items list** →
    Your main page should support the ability to show all available items from your DB. The list should be shown as a data frame and should allow the user to search and filter specific items by name, amount in stock and price.

    Each item should be shown in the main
    screen and should have:
        - Item name
        - Price in USD
        - How much items available in stock

    You should have at least 10 different items on your website.

    d. **Show search bar** → The search bar should send an API call to search specific items by name, the search result should show all the items that have the requested name in their item name.
    For example → searching for the word “sun” can return sunglasses.

    Your search bar should allow multiple searching as well as single searching.

For example → searching the words “sun”, “table” should return all items with the word “sun” of “table” in their item name.

Search by amount in stock and price should be with range (< / > / =)
For example → amount > 10 should return all the items with more than 10 items in stock.

If your search didn’t find any item you should notify the user about it.
The search results should replace the dataframe that shows all available items data frame and should show the same info as the main data frame.

e. **Login process** → Each user should login to your website by simple authentication process. If a user doesn’t have an account yet, he should be able to create one with a simple create account form.
You should save the following data on each user account:
   - User ID
   - User first name
   - User last name
   - User email
   - User phone
   - User address (country & city)
   - Username (Unique)
   - User Password

   User password should be encrypted once saved in your DB.
   User should have the ability to logout from the website, once logout the user should see the page as a visitor.
   User should have the ability to delete himself from the website, if a user deleted his account you should also delete anything that was associated with that user.

f. **Favorite list page** → Each login user could create a favorite list.
   The user can add from the favorite item page any item he wants to his list and once the user goes to that list he could see all the favorite

items he added.
The user can add items and remove items from this list.
Each item should appear in this list only once.
The user should see the favorite items the same way they appear in the main page (via data frame).
Favorite list should be saved so even when the user logged out he should see his favorite list again when he logged back in.
Only when a user is logged in he should have the ability to access to favorite items list and add / remove items from this list.

g. **Chat assistant page** → The chat assistant is a ChatGPT chat base page that allows the user to ask questions in a chat form regarding the available items that the user can purchase on the website.
Example for questions can be “what is the average size of a basketball? (in case there is a basketball item available).
The page should allow the user to interact with the chat, see his prompt and the chat responses.
The assistant should be familiar with the products available in the store and also should be familiar with products that are out of stock.

User should be limited to only 5 prompts per session, meaning that once the user uses all his 5 prompts your system should block him from generate additional prompt to the assistant.

h. **Stock management** → For each item in your website your system should handle the stock. Each item should have available items in stock. If a user orders an item this number should decrease.
Users should not be able to order more items than what you have in stock.
If an item is not available in the stock anymore it should appear in the main page and in the favorite list page with 0 in stock.
If a user tries to add an item that is not in stock to his order your system should notify that this action is not possible because there are no available items in stock.

i. **Order list page** → Your system should have the ability to handle orders.

   For each order you should save:

   - Order id
   - Order user id (The user that created the order)
   - Order date (The date the temp order was create)
   - Order shipping address
   - Order total price
   - Order items
   - Item quantities
   - Order status (TEMP, CLOSE)

   There are two types of orders in your systems - TEMP and CLOSE.

   TEMP order → Order that is currently open and can be modified.
   If a user clicks on the order with the temp status he
   should see the order process form. In that form he could add / remove items from his temp order.
   In any time the user will be able to delete the order completely or purchase the order.
   Purchasing flow should update the stock quantities of the items purchased and closed the order.

   CLOSE order → Historical orders that the user previously purchased.
   Your system should show for each login user his order list. This list should show all the orders that this user created and closed by purchasing the order.
   If a user clicks on an order with a close status he should just see the order details such as list of the items in the order, price, address etc…)
   The user could not make any changes on close orders.

j. **Order process page** → Each login user can have the ability to add any item to his pending order. Once a user adds the first item to the order he is automatically creating a new order with pending status and he should see this order in the order list page.
   The pending order should always appear first in the order list and should be styled differently from the closed orders.
   Once a user clicks on the pending order in the order list he should see the order process form.
   This form should show a list of all items that already were added to his order. The user should have the ability to add / remove items from the order.
   The order page should show:
   - List of all items in the order (With title, price)
   - Order total price
   - User shipping address
   - Purchase button

   Once a user press on the purchase button the order should be saved again with CLOSE status and should be added to the close orders in the user order list. In that case the user could create a new pending order again.
   The pending order should be saved even after the user logged out.
   If the user removed all items from the list you should delete his pending order completely.
   Each user should have only 1 order in status TEMP.

3. For your implementation you should use the following stack:
   - **For the Backend side** → Python & FastAPI framework, MySQL DB & Docker.
     Notice that you are building this service as we learned in the course (MVC, Configuration, Request to Response, Enums, proper naming, etc …)

- **For the Frontend side** → Python Streamlit
- **For the chat assistant** → ChatGPT API

Bonus (10 points):
Generate a meaningful training supervised learning model of your choice.
Provide the model dataset and train it accordingly to help you generate meaningful insights about your website users.
For example → If a user will leave your website in the coming months,
Predict how much a specific user is going to spend on your website, etc..
Your system should use the trained model and by using a dedicated API should get the requested forecast from the model.
In your answer provide also the trained dataset that you used for training the model.

Before submission, make sure you test your system for edge cases or bugs.
Make sure your system notify the user on any outcome
(For example → order saved successfully, order not saved, chat assistant is not available, the requested item is not available in stock etc…)

The entire project should be uploaded to github as a single repository.
Your repository should have a professional read me file that describe the project, the project logic and the technology stack.

Good Luck