 **Run the triplechoice.sql/triple\_choice data**:

* Load the MySQL database with the triplechoice.sql file to set up the necessary tables and data.

 **Run the website locally (manage.py) on port 8000**:

* Start the Django web application locally by running:

bash

Copy code

python manage.py runserver 8000

 **Run DynamoDB Local database on port 8001**:

* Start DynamoDB Local on port 8001 with the following command:

bash

Copy code

java -jar "D:\TripleChoice\2024\Automation\dynamodb\_local\_latest\DynamoDBLocal.jar" -sharedDb -port 8001

 **Run extractor.py**:

* This script monitors the MySQL database for new requests. If a new request is added with the title "pump", it triggers PumpMatch.py, which searches through the DynamoDB database and runs logic to find and list matching pumps. If the part type is "compressor", it will eventually run CompressorMatch.py (currently focusing on pumps, but will add compressors later).

 **Interact with the website**:

* Go to the **TripleChoice** front page and use the dropdown to select a part.
  + For now, select **1\_Pump** from the dropdown.
  + Set the **Flow** to 30 gpm and **Head** to 85 ft.
  + Add comments and specify the quantity, then submit the form.

 **Monitor extractor.py in Visual Studio**:

* If you're running the code in Visual Studio, extractor.py will print the matched pump list based on the DynamoDB search.
* This list will contain attributes like pump model, manufacturer, and a link.

 **Display the matched pumps on the frontend**:

* The pump matches printed by extractor.py should be displayed on the same page where the user submitted the order.
* I should be able to change the result table attributes on the PumpMatch.py and display it on the front end, for example for a pump, includes
  + **Pump model**
  + **Manufacturer**
  + **Link** (clickable, redirecting to another website like Google Flights)

**For a compressor would be different attributes.**

Minor change: Also change the “Submit” button to “Search” button.

* Create a Django view and API endpoint to handle form submissions.
* Feed or run extractor.py, which monitors the MySQL database for new entries and triggers
* PumpMatch.py. It will CompressorMatch.py if the part is Compressor, etc..
* Ensure PumpMatch.py processes the pump data and stores the results.
* Use AJAX or JavaScript on the frontend to display the results dynamically