María José Castillo Gallo 0233815
Annia Navarro Medina 0236619
Fabián Ruíz Sánchez 0233807

DEVELOPMENT LOG - MIDTERM II

1. PROBLEMS & SOLUTIONS

While making the code for the Dashboard, we came across the following problems that, fortunately, we were able to resolve before turning it in:

a. Printing Data

Just as it happened to us in the previous midterm, we were having a lot of problems printing the required data and variables in the terminal to verify if we were calculating them correctly and to be able to visualize the data for the graphs we had to make.

Originally, we were saving the values of the variables in an *output* variable that was returned by the function we were using. In the end, we were able to solve this problem by using the *print()* function, rather than storing the values in a variable.

b. Saving variables in a database

After we were finally able to visualize all of the variables needed in the terminal of Visual Studio Code, it was time to pass that data to a new file where we would create the database for the charts and graphs. But, as always, we faced some problems in doing so.

While investigating, we found that we needed a .json file where we would be able to store all of the values of the production line. We had some troubles doing so but we were able to solve it after a few tries and some help from other classmates.

c. Reading mode for the .json file

Once we had the .json file where all of our variables were being exported, we needed to be able to visualize the data of several runs of the simulation, not just one.

We were stuck trying to figure out how to print all of the data in that file since the program printed a single run’s data and, when we did a second run, the data was replaced, rather than added after.

To solve this, we investigated and found out that .json files have different reading modes and we had to choose the one that fitted most. In the end, we choose the (A) reading mode to *append* the new information. With this, we were now able to have a complete report of multiple runs of the simulation.

Here's the link of the page where we found some useful information on the topic:
https://www.geeksforgeeks.org/read-json-file-using-python/

d. Organizing information in the database

Finally, we had a complete .json file with all the information needed to create the graphs with our results. Now we faced the challenge of organizing said information in a way that was easy for both us and the *React* code to visualize and read.

The problem was that while passing the information to the .json file, the program wanted to make new "blocks" of information - kind of like a second file within the existing one - instead of adding them to the existing dictionary.

We were able to fix this by using a *try-except* function, which works like and *if-else*. This way, the code had to write the information inside the existing file if it already had something written in it; if not, it would create a new one.

e. Initializing a React App

When we wanted to start designing our page, we decided to use *React* since we know there are easy ways to create graphs. Sadly, we did not remember how to start a react project, and we had to do different tests with different methods and different computers. None of them were working.

After trying to initialize the project on three different computers and none of them working, we decided to switch to HTML, which was the final solution to this problem.

f. Inserting more than one graph

While creating the visualization pages, we faced the problem where we couldn't insert more than one graph in the .js using the library *Chart Js.* This was a bit alarming at first since we needed six graphs in the page and we were only successful with one.

After a while investigating and reading some documentation on the topic, we were able to solve the problem, which turns out we had because of some errors while transcribing the code.

Here's the link to the documentation page we used to solve the problem:
https://www.chartjs.org/docs/latest/samples/information.html

g. Connecting the database

The final or our problems turned out to be the worst one and, sadly, the only one we weren't able to solve. The final step of the process was to connect the .json file that had our database to the .js file but we were never able to do so.

We started by using a *fetch* function to get the data from one file and insert it into the graph file. We researched a bit on the topic and even asked a few classmates for help but we weren't able to identify what we were doing wrong. In the end, we decided that, to be able to turn in the project with usable data, we would hardcode the data.

We ran the simulation to have enough data for two months of production and hardcoded the numbers in the .js file to generate the six graphs of information we had.