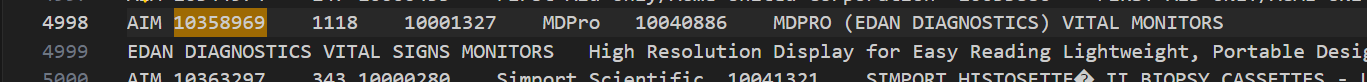
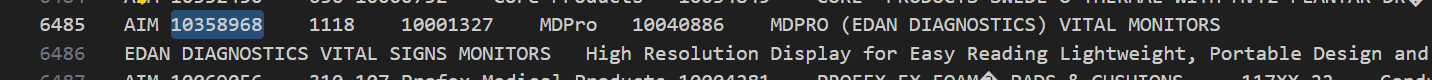
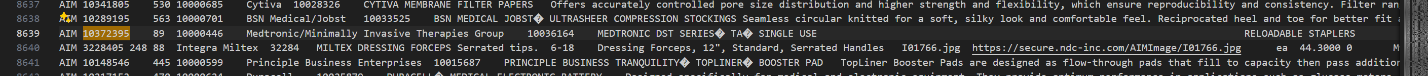
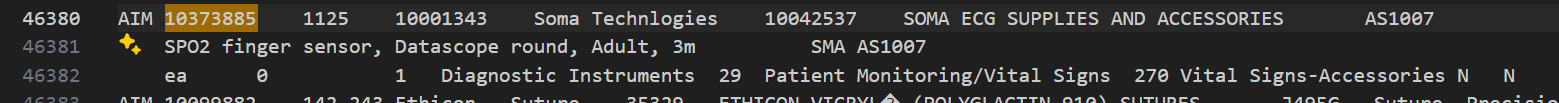
For time purposes, the following were omitted:

* since there’s no API, but scheduler:
  + Throttling omitted
  + Swagger documentation
  + CORS
  + Versioning
  + Validation Pipes
* Unit test can be implemented to check based on the file

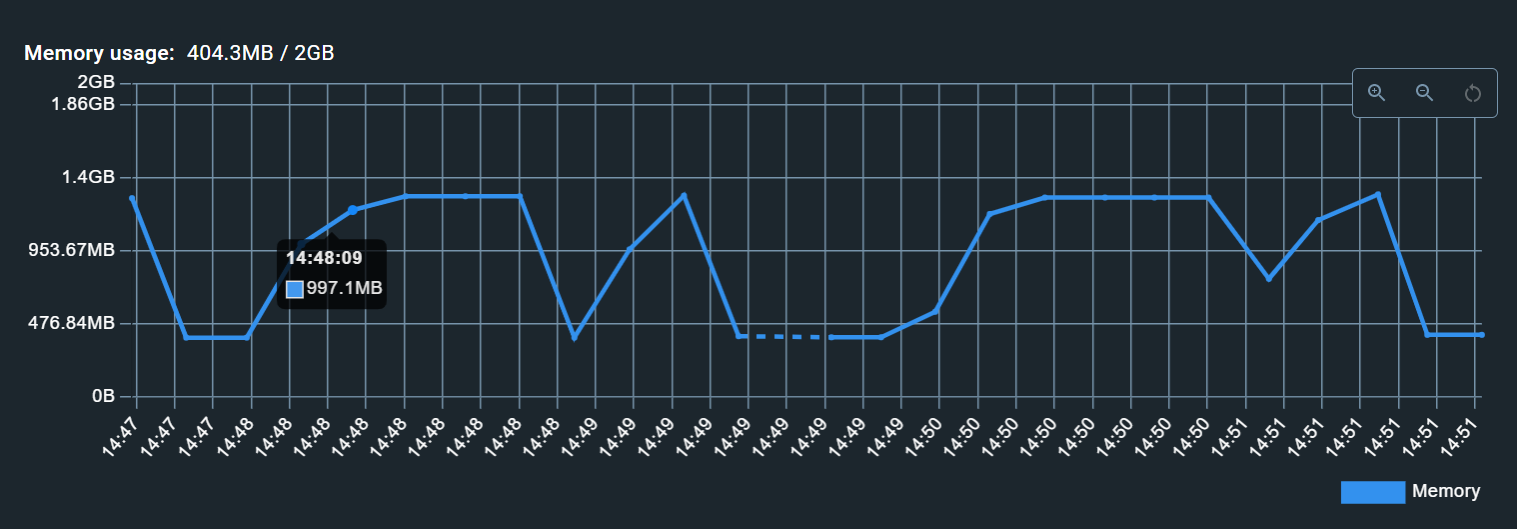
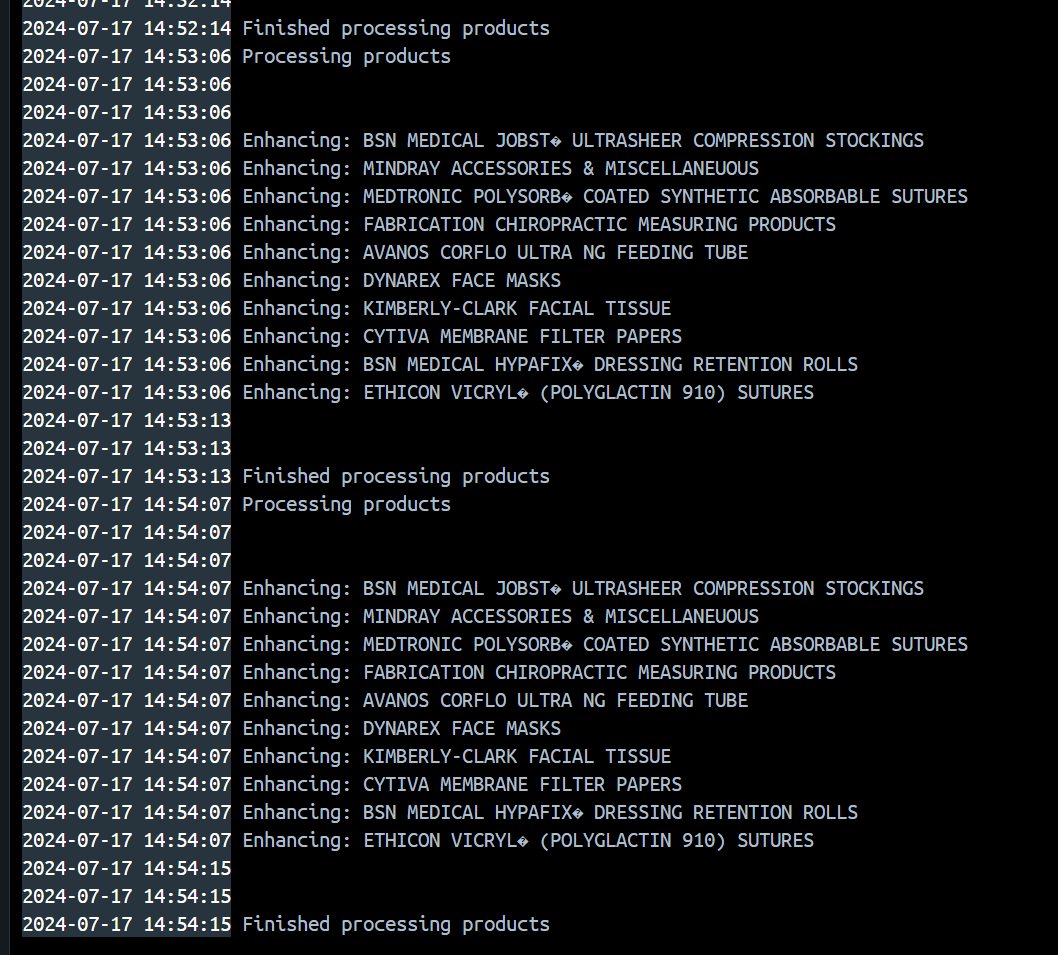
Notes:

* I used docker compose to limit the RAM of the container to 2 GB
* I did not fully ignore categories since the ChatGPT prompt asked for category name.
* Many registries on the CSV are badly inserted. The plausible cause comes from the generation. A lot of the registries that are badly inserted have in common that they have the word EDAN DIAGNOSTICS VITAL SIGNS MONITORS in its rows. For times sake, I will omit these incorrect registries, but I could do a logic to fix the CSV rows with the wrong split line before the ‘EDAN DIAGNOSTICS VITAL SIGNS MONITORS’.
  + Examples
    - ItemId - 10358969
    - ItemId – 10358968
    - ItemId – 10358970
    - ItemId – 10358972
    - ItemId – 10376007
    - ItemId – 10358971
    - ItemId – 10374965
    - ItemId – 10379977
    - ItemId – 10374964
    - ItemId – 10379978
    - ItemId - 10358973
* This has the same problem as EDAN DIAGNOSTICS, but with another name
  + Examples
    - ItemId – 10334756 - BOWMAN SEMI-RECESS HAND SANITIZER DISPENSER
* These registries have more columns that required when parsed. This may be due to the fact that its not separated correctly. Since the separator of the csv is \t, these registries have in them extra \t.
  + ItemId – 10372395
  + ItemId – 10314175
  + ItemId – 10313442
  + ItemId – 10313443
  + ItemId – 10353470
  + ItemId – 10322598
  + ItemId - 10345328
* This registry is split into 3 different rows
  + ItemId – 10373885

Without these items, the result is 13,252 products.

A screen shot of a computer

Description automatically generated

* Right now I am using the name of the product as index to figure out which item the product belongs to if the product exists in the database since ProductID is not is the database. We could create a property called VendorProductID and create a composite unique index VendorProductID -VendorID. That way we could have a direct reference for what we receive in the csv with what is in the database
* The RAM usage executing the parsing every 30 seconds is as follows  
  
* After the whole process, 13252 products were inserted/updated
* You can check that process is executing from the container logs. And check which products got the enhancement
* If docker compose main service build target is “prod” then the cron runs every day at 6 am, if it’s dev it runs every 30 seconds.