Gherkin

1. ***Write Gherkin tests for the program you wrote above. Use any Gherkin features or practices you want. Don't write step definitions (i.e., the tests don't have to be executable).***

Assumption: This set of Gherkin feature definitions only focuses on the Docker container side of the solution. Another set of Gherkin features can be developed to test the underlining Python scripts as well. Depending on how the organization is structured, the Python scripts BDD testing may be performed by the dev team or the QA team. It is intentionally left out here for brevity.

There are three features listed in this document for the Docker container solution. The first one is quite comprehensive while the other two only have a brief description of the feature themselves. No implementation is currently available to test those features as it is included here only for illustration purposes.

Feature 1: The Docker container for csv file content sorting with specific requirements for file names and locations.

The input file is always named as "`input.csv`" and output, "`output.csv`", and they are required to present in `./app` subdirectory of the current folder. The Docker container solution can generate the correct output given the proper input, and can process a variety of csv and non-csv inputs and input formats properly. The normal processing can take in a file with any number of lines of comma-delimited words and produce the correct ascendingly sorted comma-delimited line of words. The incorrect input scenarios will silently produce incorrect results for the time being.

Background:

Given a `README.MD` file and the highlighted shell scripts with comments

And a Dockerfile and the Python scripts for actually sorting and outputting the results

And the Dockerfile has a CMD configuration that always require the input file being named as "`input.csv`" and the output file being named "`output.csv`"

And the default well-known input csv file ("`input.csv`", already built into the image)

And the Docker command to create a container image

And the Docker command to create a container with the correct volume mapping

And a current folder with all the right structure and content for operations

And the shell scripts to automate all the steps from image creation and output inspection

Scenario: A simple, well-formatted one-liner csv file gets processed

Given the default "`input.csv`" with the following content:
`Copenhagen,Stockholm,Oslo`

When I run `./setup_runtest.sh` scripts

Then I see the "`output.csv` " is generated in the current folder

And the following content: `Stockholm,Oslo,Copenhagen`

Scenario: A well-formatted multi-line csv file gets processed

Given a multiple-line, csv formatted file

When I remove the existing "`input.csv`" if applicable

And rename the given file to "`input.csv`" and copy it to `./app` folder

And remove the existing "`output.csv` " file if applicable

And I invoke `./run_test.sh`

Then I see the "`output.csv` " is generated

And the correctly sorted content, corresponding to the input file content

Scenario: A simple, ill-formatted one-liner non-csv file gets processed

Given a one-line, non-csv formatted file

When I remove the existing "`input.csv`" if applicable

And rename the given file to "`input.csv`"

And remove the existing "`output.csv` " file if applicable

And I execute `./run_test.sh`

Then I see the "`output.csv` " is generated

And the content of "`output.csv`" exactly matches that of the "`input.csv`" file

Scenario: An ill-formatted multi-line non-cvs file gets processed

Given a multiple-line, non-cvs formatted file

When I remove the existing "`input.csv`" if applicable

And rename the given file to "`input.csv`"

And remove the existing "`output.csv` " file if applicable

And I invoke `./run_test.sh`

Then I see the "`output.csv` " is generated

And the content of "`output.csv`" exactly matches that of the "`input.csv`" file

Scenario: Process a non-existing "`input.csv`"

Given the Docker container already exists (I.e., "`./run_test.sh`" has been already executed)

And the "`output.csv`" file has been removed in the current folder

When I execute run `./run_test.sh`

Then I see errors and no "`output.csv`"

And the missing input file error message from `docker logs $(docker ps -lq)`

Feature 2: The Docker container for csv file content sorting with flexible requirements for file names and locations

With more flexible requirements for the file names and location the Docker container solution can perform all the operations outlined in Feature 1. The scenario definitions are left for future exercises.

Feature 3: The Docker container for any well-delimited file content sorting with either non-flexible and flexible requirements for file names and locations

The Docker container solution is further expanded to process all regularly delimited contents and with more flexible requirements for the file names and location. The expanded solution can perform all the operations outlined in Features 1 and 2. Similarly, the scenario definitions are left for future exercises.

2. Explain in detail why these tests might be helpful in the future.

I can see three benefits from the Gherkin feature and scenario development: a) newly emerged insight and/or feedback gained by the engineering team to the product management team; b) improvement and enhancements for the solution requirements, specifications, architecture and design; and c) better understanding of the boundaries of the solution, the new details for the existing key user cases and scenarios. There should be more if more time is available to continue digging into this aspect of the work.

One of the main advantages of using Gherkin feature definitions is to capture the initial mutual understanding between the product management and the engineering (Dev, QA, DevOps) teams. Additionally, the features may also provide feedback from the engineering side to the product management side when further details of a proposed solution are investigated and researched. In this particular exercise, I believe the two additional features popped into my mind while I was working on this. Some of might be new insights even to the product management team.

Another benefit is that while the engineering teams proceed to work on the details of the features and scenarios, either independently or collaboratively, insights may be gained to produce useful revisions of the specifications, architecture and design for a more robust, scalable and high-performing solutions. Alternatively, it may help the emergence of future solution release lineup or roadmap.

There are, of course, the normal benefits: The features help outline the boundaries of the project and focus the engineering team to develop the functionality or other operations that matters the most to the product management team or the end users.