First part of the assignment

Before answering the question, here is the thought process I had while analyzing the working process provided in the assignment, to help me choose the best answers.

Some research must have been done before working on the assignment. Research included:
What is an FEA model??
What does a “load” represent in this case??

At the end of the document, there is a list of sources I have used in my research.

Here is the list of tasks I made from the process, together with the automation potential and the explanation for it:

1. **Design Engineers create a mechanical design**

Automation potential is low, since designing requires more engineering judgement and creativity.

2. **Structural Analysis engineer tasks**
   2.1 Makes a selection of the most critical loads

Automation potential is very high, since the loads department probably provides a massive Excel file consisting of hundreds of load cases and the structural engineer must choose the load cases that matter the most, and the automation of this particular task would save a significant amount of time. Also, selection through filtering or thresholding is a great task to be automated since it’s mostly rule-based.

2.2 Build an FEA model

Automation potential is medium. Although there are tools to support the automation of different tasks, like applying loads or mesh generation, I think the automation of this task is only worth it if many similar models are created frequently.

2.3 Simplifications on the model

Automation potential here is also medium, since the typical manual tasks like editing the model or removing unnecessary features are rule-based, but it still requires human judgment to check if the model is not over- or undersimplified.

3. **The model is run on a Linux cluster**

In this case, automation potential is very high. Linux job submissions are great for automation, dealing with uploading output files, writing submission scripts, automatic job monitoring and auto-download upon completion.

4. **Results are retrieved and imported into the model**

Automation potential here is high since most FEA tools allow Python or command-line post-processing, which automates the typical tasks done, such as generating plots or exporting results into structured formats like Excel.

5. **Report creation and feedback to designers**

Automation potential is medium. Some tasks in this step can be automated, such as auto-inserting plots or auto-filling tables, but main tasks, such as writing feedback or recommendations for design changes, still need to be done by an engineer.

Very High Potential Tasks:
- Making a selection of the most critical loads
- Running the model on a Linux cluster

High Potential Tasks:
- Results retrieved and imported into the model

Medium Potential Tasks:
- Building the FEA model
- Simplifying the model
- Report creation and feedback to designers

Low Potential Tasks:
- Creating a mechanical design

1. Which part of the process would you try to automate first?

In the very high-potential tasks list, there are two tasks: making a selection and running the model. Both tasks are very appropriate for automation using Python scripts, for example but out of these two, I would choose to automate the selection of the most critical loads. The Structural Analysis engineers spend a lot of time choosing the appropriate load cases among hundreds or thousands. Automating this task, using ranking and thresholding, would save a substantial amount of time, massively reducing the time it would take for the whole working process to finish.

2. Describe in max 300 words how you would approach first

I've used ChatGPT to help me understand the criteria for choosing the appropriate loads for an FEA model, so the first step in approaching would be to understand the structure of these load cases and what exactly defines these loads, such as pressures, temperatures, forces, etc. Based on engineering principles, important load cases would probably be identified and ranked using criteria such as total force magnitude, maximum pressure or temperature values, etc.

I would develop a script, with independent and clear modules, that imports the Excel file, extracts and uses the values from the load cases in order to calculate resultant values such as the resultant force or the resultant moment. After discussions with the Structural Analysis engineers, I would apply the rules and criteria for choosing the loads. These could include choosing loads with the highest frequency or excluding loads with non-relevant conditions. The script would then rank these loads by severity or more specifically, the ones to produce the highest stress in the structure, and would output a list ready for FEA modelling. For flexibility reasons, the logic would be configured in a way that engineers can adjust thresholds or add new criteria without changing the whole code. That's why the need for independent modules exists.

In the end, the script can be integrated into an existing workflow where engineers can submit Excel files with hundreds of load cases, which they would normally rank themselves, and receive a clear and filtered output list, ready for the FEA model.

3. What are the steps in the automation of the task you have chosen?

a) Consult with the Structural Analysis engineers and define rules and criteria for load selections
b) Developing the automation code with these functionalities in mind:
    1. Importing the Excel file
    2. Extract the values from load cases and compute resultant forces
    3. Implementing the criteria and the rules, which would be the biggest module and would consist of many more detailed steps for fully implementing it. In this module, I would also include calculating the severity and the ranking of the load by severity.
    4. Making a clear list and sending it to the FEA model.

4. Which functions would you define?

- load_excel(file_path)
- calculate_resultant_force(loads_table)

- calculate_resultant_moment(loads_table)
- filter_load_cases(loads_table, criteria_table)
- calculate_severity(loads_table)
- rank_load_cases(loads_table)
- select_load_cases(loads_table)
- export_results(loads_table, path_model)

My thought process for these functions is that, initially, the loads_table holds the raw data, and each function would return a modified version of the loads_table. In the end, after the select_load_cases function, we would have the perfected table, with the most critical load cases organized and ready to send to the model.

5. What are the advantages and disadvantages of your approach?

There are multiple advantages to my approach. In general, the flow of the program is straightforward, easy to understand and predictable, as we know what happens at every stage of the process. The modular approach to this task allows for maximum flexibility and scalability, with each component being independent of the other and having only one functionality. This approach allows for the software to be easily extended if the need for it appears. Another major advantage is the low difficulty in testing and debugging in each module. Unit testing each function is much easier with this approach, since every function receives an input and produces an output and doesn't depend on outside state.

A disadvantage in my approach is the high memory usage, since after each function, a new table is created, while the old ones are still being held in memory. For large tables, this can be expensive, when it comes to memory. A mitigating action for this would be to work on the tables in order to reduce memory load and an example of this would be to replace or add only the necessary columns instead of copying the whole table. Besides that, there is also a problem of data validation. Each function implicitly assumes that the data is correct, e.g. the table has the right column names, right numeric types or not have missing required data. If not handled correctly, there is a chance for the software to run in errors or output bad results. To mitigate this, I would implement extra validation checks, like adding schema validation at the beginning of each function or validating data types, ensuring that numerical values are actually integers instead of strings, for example.

6. How will you make sure your code is robust?

- Separating software into multiple modules

With my approach, some initial software design choices already take into account robustness. Separating the whole functionality into multiple different, independent modules makes each function stateless, depending only on the input it receives and giving a predictable output. One function can not break another function, as they are not connected or dependent on one another. Also, with isolation, bugs are much easier to detect and avoid cascading failures.

- Strong input validation

An additional step towards greater robustness is strong input validation. Before any function does anything, the system must check that the required columns/rows exist or that the values are physically realistic (no negative values). Instead of wasting time and computational power on results that are bad from the very start, the system will check that the input file is actually usable.

- Robust test suite

Every module will have its own unit test suite, ensuring correct and errorless functionality. This allows erors to be caught before deployment. In addition, an end-to-end test will be implemented with a realistic Excel file to ensure that the program functions correctly from the first function to the last one. With this test, integration issues can be caught early.

- Graceful error handling and meaningful exceptions

If errors should happen, the system should gracefully capture them, log them and output clear and helpful explanations behind that error instead of relying on the errors raised by libraries or the programming language. Raising domain-specific errors helps engineers know what went wrong.

Bibliography:

https://www.autodesk.com/support/technical/article/caas/tsarticles/ts/68avt11P8h9hu5oxbqtQK0.html