Factory Twin Follow-Up Questions

1. What specific data sources will the AI chat agent have access to—what is the data retrieval mechanism for each?

The AI chat agent will have access to two data sources:

1. A MSSQL Server database containing up-to-date data from the factory according to the schema described in the mock data dictionary
2. An API is available for each visual in the tool which returns the underlying data. Each API expects calls in JSON format which specifies certain arguments and returns CSV data. For example, imagine a simple bar chart which displays revenue by month for a certain period of time, with no additional filtering or processing. That bar chart would have just two arguments, start\_date and end\_date. The JSON-formatted API call would look like:

{

“start\_date”: ‘2024-01-01”,

“end\_date”: “2025-01-01”

}

This API call would then be made over an HTTP GET request to actually receive the data. The data returned would have one value per month returned:

“1.2M,1.3M,1.4M,1.6M,0.9M,1.3M,1.5M,1.6M,1.0M,1.8M,1.8M,1.5M”

2. Are all APIs returning structured data (e.g., JSON, CSV)? If not, will there be a need for additional parsing or preprocessing?

You may assume that all APIs are returning CSV data.

3. How many users are expected to use the chat agent simultaneously?

The agent should be able to reliably handle up to 50 concurrent users per installation, although this is a maximum and should not be considered regular traffic.

4. What are the hardware specifications for the deployment environment? (important for optimizing the AI model selection.)

I can’t give specific specs, but a good general guideline would be that of a mid to high end laptop without a dedicated GPU for on-prem installation: e.g. some 8-core CPU, 16GB RAM, 512GB of disk. Obviously cloud installations have scalable compute power, but it should be approximately the same.

5. Should the AI agent be able to generate charts and visual summaries along with text-based responses?

   - If so, are there preferred visualization libraries or formatting standards?

The AI agent should be able to direct users to existing FactoryTwin visuals, but for the moment it is not expected to generate its own.

6. Does FactoryTwin have a user role hierarchy for data access?

You may assume a simple binary hierarchy in which a System Administrator is able to control data access for normal users, but normal users will have unrestricted access unless otherwise specified by the System Administrator.

7. Are there any restrictions on external library dependencies?

   - Should we avoid specific open-source frameworks due to licensing or security concerns?

There are no specific restrictions on open-source frameworks or libraries. The only restriction is that the deployment environment will not have access to the internet, so it must be compatible with such an environment.

8. What KPIs will determine the success of the AI chat agent?

This is ultimately up to you. I would suggest considering KPIs which cover:

1. Performance: Response time and throughput especially, but also memory usage, storage usage if applicable
2. Quality: Accuracy and relevance of responses as assessed by users. Consider both human-in-the-loop and what might be required to have automated metrics of evaluation. Can also include failure/crash rates.
3. Usage: User engagement metrics such as frequency of use and duration of sessions

9. Can we generate our own dataset?  
  - Will we be provided a mock structure of the data currently being used?

You can generate your own dataset if you wish and find it helpful, but I wouldn’t recommend it as it would be quite time consuming.

10. **Is there any available documentation or a list that outlines the metrics Factory Twin monitors, such as delivery times, costs, or supplier performance?**

FactoryTwin monitors and predicts all major shop KPIs: on-time delivery (both for the Factory and its suppliers), revenue, cash flow, profit, expenditures, inventory turns, and any variations thereof.

**11. What kinds of factory metrics does Factory Twin focus on?**

(e.g., On-time delivery, production delays, or inventory levels)

See answer to #10.

**12. How detailed is the data within Factory Twin? Does it include precise timestamps for every transactional step, such as when an item/part is ordered, shipped, or encounters delays?**

Yes to all of the above. You may assume that all data has a precise timestamp identifying when that datapoint was generated.

**13. Is the data fine-tuned enough to figure out why something specific, like a line item, was late?**

(e.g., Do we have enough detail to analyze delays without guessing?)

For most cases, yes, but the reasoning is not stored in the data itself, but rather derived from the data and available through various APIs. It may also be the case that for certain line items, the cause is not able to be derived.

14. What is the average size of the data uploaded onto the software?

You may assume the average volume of stored data is approximately 32GB.

15. Would it be possible to integrate paid external APIs into the model to enhance its predictive capabilities?

Yes, so long as they work in an offline environment; most paid APIs are usage-based, which tends to require an internet connection for tracking.

16. Could you provide insights into the data structure?

The mock data dictionary describes the database structure – if there are questions about the database structure or information not in the dictionary, I’d need a more specific question.

17. Would it be possible to share a sample dataset to better understand the nature of the captured data?

As of now we do not have any sample datasets which would be available to share. I would recommend referencing the data dictionary.

18. What does the current backend workflow of the software look like?

You may assume a relatively simple workflow in which front-end HTTP queries are sent to APIs hosted on a single server. Calling an API executes a parametrized query against the database and returns the result set in CSV format, with some possible pre- and post-processing.