AI Workflow Exercise: Generating Mocked Data with an LLM

In the “Using Playwright for End-to-End Testing” section, we suggest running most of your front-end tests via mocked responses that simulate data the back-end server would have sent. Therefore, you must mock out sufficient JSON data. In prior sprints, we asked you to create mock data in the form of TypeScript arrays or objects, but in actuality, the backend will send JSON strings that you will turn into TypeScript arrays or objects. Now that you know about promises and JSON parsing in Typescript, your mocked data can take the shape of JSON strings and thus will be more faithful to reality for testing purposes.

Navigate to an LLM of your choice (ChatGPT, Claude, etc.) and create an account if you do not already have one. Prompt the LLM to generate some JSON strings to use for mock data, giving it the appropriate context, which will ultimately allow you to be more efficient by automating the generation of mocked data. Please note that you may have to be intentional with specific edge cases to ensure that your test suite is robust; as always, don’t trust the LLM to be thorough, or even correct! Use it as an assistant, not as a replacement for your creativity and craft. Finally, answer the reflection questions corresponding to this exercise.

S_DIST and 1340: In the previous sprints, specifically Sprint 3.1 and Sprint 3.2, you created mock data that took the form of arrays of arrays. Try to leverage the LLM you chose to transform the arrays that you manually created into JSON strings for this sprint. Finally, answer the reflection questions corresponding to this exercise.

1. AI Workflow Exercise Follow-Ups

1.1) What concerns, if any, do you have about using an LLM in this way? Think from the perspective of an *engineer*: how might an LLM’s training data in this case impact the mocked data it is returning?

LLM’s are trained on large amounts of data, but we have no way of knowing how much or how robust the data is or where it’s coming from. This raises privacy concerns, as we could be including ‘mocked’ data that’s pulled from real sources. There are also concerns about edge cases, as the LLM may not be trained on malformed or empty datasets and therefore wouldn’t consider using them for our mocked data. On top of this, our LLM may generate a specific type of data from a specific demographic. Assuming we’re expecting accurate and true data and are using this for anything other than testing functionality, this could cause large issues with bias and only representing a certain subset of users.

1.2) What might “bias” mean in this context? I.e., what kinds of bias might you specifically want to mitigate when you use an LLM to generate test data?

In this context, bias could mean that the data generated by the LLM is different from the real data the application could encounter once it’s no longer mocking. For example, not using

malformed data, only including specific values (strings/ints), keeping the values consistent across rows and columns, only providing small or large datasets. These are all points of bias that we may want to mitigate when we use an LLM to generate test data.

1.3) How did you attempt to mitigate the above concerns, if any?
We attempted to mitigate these concerns by providing the LLM with more detailed instructions on the range of data we wanted it to produce for it. For example, if we want datasets between a range of x and y rows or x and z columns, or if we want the datasets to contain consistent/unique values across rows and columns, and if we want the datasets to include malformed data.

S_DIST and 1340
Was the LLM able to perform this data transformation properly? If not, why might this have happened?
The LLM was able to perform this data transformation properly. Issues just came from our prompt being unclear or us not mentioning data types or edge cases.