Kusho.AI – Import Postman Collection Use Case and Comparison Testing Session

I have a website, https://datamakerjs-f3b6b7d13de0.herokuapp.com/api, which implements a REST API for creating, viewing, and generating data examples based on custom defined schema. I used Kusho.AI to create test cases against that site based on a set of requests in a Postman collection. I also ran a comparison of what it would take to create the same cases in Postman.

Postman Collection

GET https://datamakerjs-f3b6b7d13de0.herokuapp.com/api/namespaces

GET http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadef?namespace=baseobjecttypes&schemaname=initialalpha1to8charstring

GET http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadefs?namespace=baseobjecttypes

GET http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadef/getrandomexample?namespace=baseobjecttypes&schemaname=initialalpha1to8charstring&count=1

POST http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadef

Post body:

```
{
    "SchemaName": "testingsequenceschemaobject",
    "RootSchemaObject": {
        "ObjectTypeName": "SequenceSchemaObject",
        "SequenceArray": [
            {
                "ObjectTypeName": "ReferenceSchemaObject",
                "Namespace": "baseobjecttypes",
                "SchemaName": "alphalower"
            },
            {
                "ObjectTypeName": "OptionalSchemaObject",
                "OptionalValue": {
                    "ObjectTypeName": "ReferenceSchemaObject",
                    "Namespace": "baseobjecttypes",
                    "SchemaName": "alphanumericchar"
                }
```

```
},
{
    "ObjectTypeName": "ReferenceSchemaObject",
    "Namespace": "baseobjecttypes",
    "SchemaName": "alphalower"
},
{
    "ObjectTypeName": "OptionalSchemaObject",
    "OptionalValue": {
        "ObjectTypeName": "ReferenceSchemaObject",
        "Namespace": "baseobjecttypes",
        "SchemaName": "alphanumericchar"
    }
},
{
    "ObjectTypeName": "ReferenceSchemaObject",
    "Namespace": "baseobjecttypes",
    "SchemaName": "alphalower"
},
{
    "ObjectTypeName": "OptionalSchemaObject",
    "OptionalValue": {
        "ObjectTypeName": "ReferenceSchemaObject",
        "Namespace": "baseobjecttypes",
        "SchemaName": "alphanumericchar"
    }
},
{
    "ObjectTypeName": "ReferenceSchemaObject",
    "Namespace": "baseobjecttypes",
    "SchemaName": "alphalower"
},
{
    "ObjectTypeName": "OptionalSchemaObject",
    "OptionalValue": {
        "ObjectTypeName": "ReferenceSchemaObject",
        "Namespace": "baseobjecttypes",
        "SchemaName": "alphanumericchar"
    }
},
{
    "ObjectTypeName": "ReferenceSchemaObject",
    "Namespace": "baseobjecttypes",
    "SchemaName": "alphalower"
},
{
    "ObjectTypeName": "OptionalSchemaObject",
    "OptionalValue": {
```

```
                    "ObjectTypeName": "ReferenceSchemaObject",
                    "Namespace": "baseobjecttypes",
                    "SchemaName": "alphanumericchar"
                }
            },
            {
                "ObjectTypeName": "ReferenceSchemaObject",
                "Namespace": "baseobjecttypes",
                "SchemaName": "alphalower"
            },
            {
                "ObjectTypeName": "OptionalSchemaObject",
                "OptionalValue": {
                    "ObjectTypeName": "ReferenceSchemaObject",
                    "Namespace": "baseobjecttypes",
                    "SchemaName": "alphanumericchar"
                }
            },
            {
                "ObjectTypeName": "ReferenceSchemaObject",
                "Namespace": "baseobjecttypes",
                "SchemaName": "alphalower"
            },
            {
                "ObjectTypeName": "OptionalSchemaObject",
                "OptionalValue": {
                    "ObjectTypeName": "ReferenceSchemaObject",
                    "Namespace": "baseobjecttypes",
                    "SchemaName": "alphanumericchar"
                }
            },
            {
                "ObjectTypeName": "ReferenceSchemaObject",
                "Namespace": "baseobjecttypes",
                "SchemaName": "alphalower"
            },
            {
                "ObjectTypeName": "OptionalSchemaObject",
                "OptionalValue": {
                    "ObjectTypeName": "ReferenceSchemaObject",
                    "Namespace": "baseobjecttypes",
                    "SchemaName": "alphanumericchar"
                }
            }
        ]
    },
    "Namespace": "testingnamespace"
}
```

Session Report

- IN Kusho.AI, I imported the collection (filename Datamaker collection.postman_collection.json)
- Kusho indicated I would get an email after done processing

    ISSUE: I never received an email indicating the imported collection was complete

    I waited several hours, checked my junk mail, all my folders, nothing.

    ISSUE: the imported collection only created 3 out of 5 test suites,

    Created: "Datamaker Collection: POST SchemaDef", "Datamacker Collection: GET getRandomExample", "Datamaker Collection: Get SchemaDefs"

    Missing: "Datamaker Collection: GET Namespaces", "Datamaker Collection: GET SchemaDef"

- Import started at 8:36 am, as of 8:44 am it indicated 3/5 of five processed – was reporting complete by 8:54 am

    ISSUE QUESTION: Completing collection processing seemed to take longer than expected

    I do not know if the above times are slow or not. It seems when I create an API specification in Kusho.AI it is much faster than that, about a minute or two before it is done with ~30 test cases. Taking something between 8-18 minutes to report complete in the UI seems far longer than the interactive experience.

- I clicked away from the status page to look at the test suites, I saw 2 there while 3/5 was processing, so that seemed to make sense
- I ran one of the tests, "Test with non-English characters count", which had a non-numeric value for a parameter that I believe should have been numeric, which makes for a good test case. My server responded without an error, and that might be a bug in my code, which is a good thing for Kusho.AI to expose, although it does raise an issue that I believe you have said others have raised:

    ISSUE: No way to indicated expected results from test case

    For example, "Test with non-English characters" in the suite I generated passed a character string to a parameter that I believe (?) should have been numeric, and the server it was testing should have returned an HTTP 400 and a meaningful error message. Instead the server returned HTTP 200 and an empty result set. I want to be able to tell Kusho.AI that this response was incorrect so it can mark the test as failed. The body of the request is below:

    ```
    {
      "method": "GET",
      "url": "http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadef/getrandomexample",
      "headers": {},
      "path_params": {},
      "query_params": {
        "count": "测试",
    ```

```
    "namespace": "baseobjecttypes",
    "schemaname": "initialalpha1to8charstring"
  },
  "json_body": {}
}
```

ISSUE: Test cases do not describe the field they changed or what they changed it to

This is very confusing. They will say something like “Test without optional fields” but not say which fields. On inspection, I could not see a difference from other cases, so I don’t know what the test changed. I believe Kusho needs to present a more thorough description of what it changed to distinguish the tests. It may be best if that was done in the title to make surveying the cases easier.

ISSUE: Status icon keeps the animated spin going even after results are complete

I click Run, and the case seems to be almost immediately complete. I can click the down arrow to expand results, and I see them in the UI. The status icon will remain spinning forever, as far as I can tell. I have never seen it stop.

- I examined the results for "Test with www in URL" and the results were empty. I tried the same thing in POSTMAN.

ISSUE: Kusho.ai is not reporting errors returned by the server, is not displaying http 500 status.

The following request in Kusho.ai returned a blank result:

```
{
  "method": "GET",
  "url": "http://www.datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadefs",
  "headers": {},
  "path_params": {},
  "query_params": {
    "namespace": "baseobjecttypes"
  },
  "json_body": {}
}
```

Testing the same request in Postman returned an HTTP 500 error, with this information:

```
GET http://www.datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadefs?namespace=baseobjecttypes

Error: getaddrinfo ENOTFOUND www.datamakerjs-f3b6b7d13de0.herokuapp.com

Request Headers

User-Agent: PostmanRuntime/7.36.1
```

Accept: */*

Cache-Control: no-cache

Postman-Token: 35fb6178-5524-4686-81a2-9dacf56b20ca

Host: www.datamakerjs-f3b6b7d13de0.herokuapp.com

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

The error is a bug in my server code. Kusho should present all the information about the request in the UI. Not seeing the HTTP 500 status, not seeing the actual error message returned makes it very difficult to analyze and investigate the problem.

I had a similar error with another request: "Test with all caps string for RootSchemaObject.SequenceArray.Namespace". The JSON for the request was:

```
{
  "method": "POST",
  "url": "http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadef",
  "headers": {},
  "path_params": {},
  "query_params": {},
  "json_body": {
    "Namespace": "testingnamespace",
    "SchemaName": "testingsequenceschemaobject",
    "RootSchemaObject": {
      "SequenceArray": [
        {
          "Namespace": "BASEOBJECTTYPES",
          "SchemaName": "alphalower",
          "ObjectTypeName": "ReferenceSchemaObject"
        }
      ],
      "ObjectTypeName": "SequenceSchemaObject"
    }
  }
}
```

Via POSTMAN, I got an http 400 with my error message:

```
{
    "error": "invalid schema: argument must be a string"
}
```

- I took one of the suites, “Datamaker Collection: Get SchemaDefs” and timed myself replicating the same tests in Postman. I created 10 tests (instead of the 15 that Kusho created, but 5 of them were essentially duplicates, so I skipped creating them). It took me 12 minutes. The other two suites, Kusho.ai created 33 and 31 test cases respectively. Assuming it took me the same time to create those cases in Postman (1.2 minute per case), the time it would have taken me to build the same suite in Postman would have been about 1.48 hours, just shy of an hour and a half. Meanwhile, creating cases in Kusho via PostMan collection upload is about a minute or so. I am not counting the time wait for server to process because I assume someone can spend their time doing something else.
- My conclusion is that the data entry savings for Kusho.ai’s automatic test case creation is a decent time saver, but I have not checked how well it scales. As the inputs get more complex, do the number of cases likewise scale up? Could this go to thousands of cases with an equal time savings on the data entry?

Issue: Several test cases seem like duplicates

Several of the test cases would have exactly the same JSON. The titles would be things like “Test with all valid inputs as specified in API information” and “Test with valid input for namespace field” but the actual test is identical. Sample offered below. This kind of redundancy puts a burden on anybody surveying the suite to determine its value

```
{
  "method": "GET",
  "url": "http://datamakerjs-f3b6b7d13de0.herokuapp.com/api/schemadefs",
  "headers": {},
  "path_params": {},
  "query_params": {
    "namespace": "baseobjecttypes"
  },
  "json_body": {}
}
```

ISSUE: user generated test cases like "test with 10 different valid values for namespace" all gave the same name "test with valid namespace value" - shouldn't it differentiate?

I tried a user prompt for “test with 10 different valid values for namespace” and it generated the same case over and over again. It seems I should have got 10 different cases.

ISSUE: "test with 10 different breeds of dog for namespace value" resulted in "breed1", "breed2", etc. but "test with 10 different breeds of bird for namespace value" actually resulted in bird names (e.g. "peacock", "pigeon", etc.)

As described in the title, I prompted “test with 10 different breeds of dog for namespace value” it the values were just “breed1”, “breed2”, etc. whereas “test with 10 different breeds of bird for namespace value” created “owl”, “peacock”, “dove”, etc. I saw a lot of inconsistency in how the user generated prompts behave. Sometimes I got just generic insertions (as in the dog case) sometimes it seemed to do what I asked (as in the bird case), sometimes it added no new cases. The feature is kind of cool, but it is difficult to know how to control it to get what you really need.

ISSUE: "test with two different derogatory terms for an african american person for namespace value" yielded "n**r" and "s***n", while "test with two different derogatory terms for a person with disabilties for namespace value" yielded "handicapped" and "cripple"**

This one is sensitive. I am not sure how to think about this one. There are legitimate reasons why someone would want offensive words when testing. For example, the system under test might be designed to flag words, block them, report them, etc.. But if the terms are going to be censored with "***" that impedes ability to use those terms. If we don't want the tool producing such terms, then should there be a statement indicating reason the terms were censored? Should there be setting, a warning?

Sensitive terminology with LLMs is a topic each business is going to need to take a stand on. How open do they leave the door? Shut it complete? Warn the user? The part I am concerned with is that the varying behavior seems to keep Kusho.ai from being able to control it. Even though it is probably just the variance in the LLM engine at play, different members of different marginalized communities might be offended if derogatory terms about them are handled differently than about another group.

Some thoughts on Kusho.ai overall:

- The test case creation capabilities seem useful, but still very limited. It feels like there are a lot more categories that could be explored, but also as if there is a lot to do when the suite is created to evaluate it, analyze it, etc.
- I found several bugs in my website REST API using Kusho.ai, so there is some utility in the test cases it creates.
- The lack of a way of editing the suites, the cases, indicating desired return value behavior makes the tool very limiting. Most of the time where I did find a bug, Kusho.ai indicated everything seemed fine, or flagged no error.
- I used Kusho.ai while simultaneously debugging on my web server. I also found myself switching to Postman to imitate what Kusho.ai was doing so I could see more information about the request. So long as this is the case, I believe most engineers will abandon Kusho.ai quickly for tools and methods that give them more information and more control.
- Kusho.ai should present more information about the test cases and suites in the UI so a person can understand what the cases are doing without having to read the JSON. The generated titles do not distinguish well and it is hard to know what they are doing.