The Automated tests were carried out on katalon studio and scripts were written with Groovy language.

This can be setup by downloading the Katalon studio software from the katalon website and making sure you have java 7 or later installed on the computer you are running the katalong on and your test environment is ready.

To setup Jmeter make sure you have Java installed on your system and download the Jmeter zip file from the Jmeter website and unzip into the C folder.

To test the API object repositories must be added to the Katalon studio from the API documentation, the objects are then used to create scripts.

To test the UI of the application the application is recorded and the objects of the application are stored in the object repository as well, then the test scripts are written using the object repository.

The test scripts and results are stored in this [**Github Repo**](https://github.com/PeterAbah1819/Bundle-QA-Challenge) for assessment, or just visit this link: <https://github.com/PeterAbah1819/Bundle-QA-Challenge>

The API was tested with test cases stated in the test case sheet.

The application UI was tested as well.

**Description of Automated Tests**

The UI of the application was tested with all text display, text boxes and application usability tests. I tested that the application can be interacted with ie, texts are visible to the user, the text box when clicked displays the keyboard and the text allows for paste option when long pressed.

The API was tested for the post request to make sure that data parsed were created and stored in the server for future recall.

Tested the API with different 64bit encoded payloads to see if the data was created on the server, called that data created using the invoice ID created and it worked.

After creating a 64bit encoded payload, I tested that posting an incorrect payload as well as an incomplete payload would not work and it passed as the data did not parse.

I tested that using any invoice ID created by the initial posts requests would pull out the corresponding company invoice details from the server and it worked hence passed.

I tested that using an incorrect invoice ID that wasn't created should not produce any company invoice details but it produced a company, however without details, so this was a failed test as the API should not be able to call any information not created.

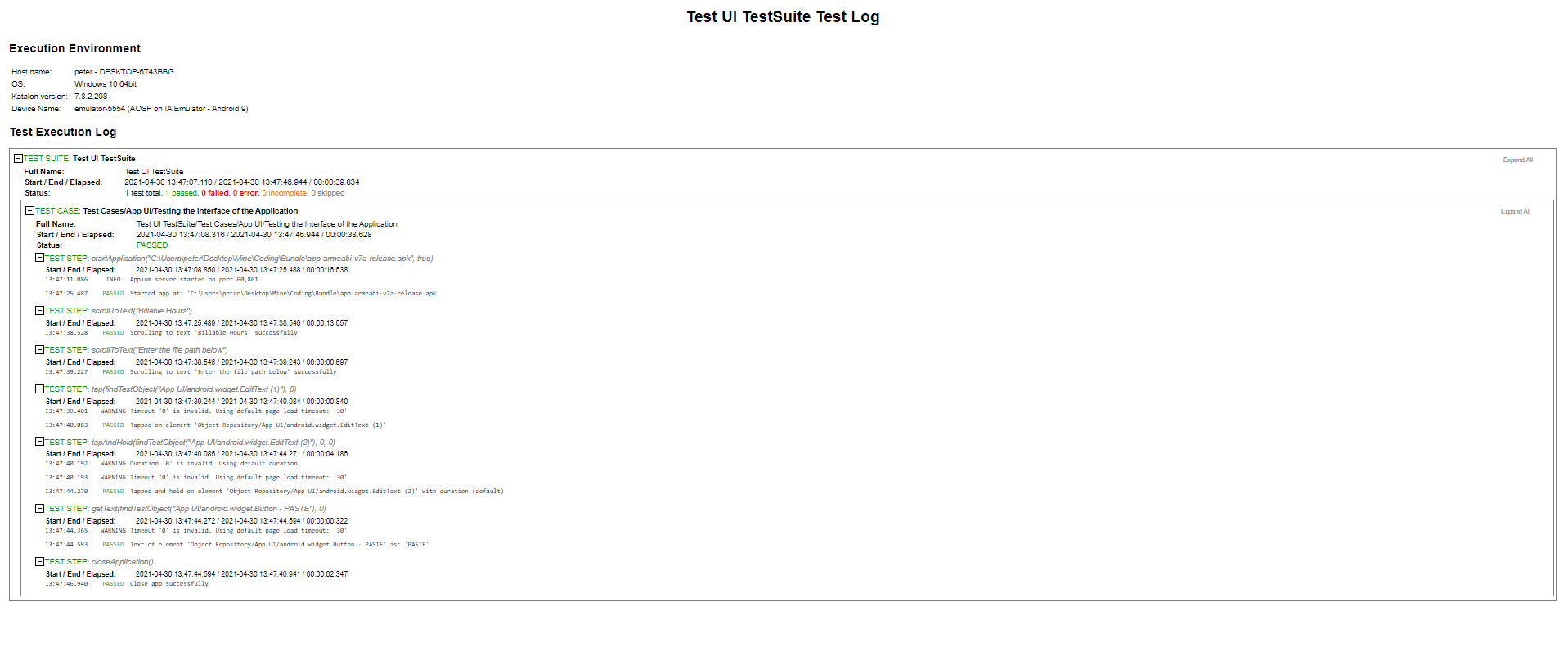
I tested that using an incomplete invoice ID of a company initially created would not get any company invoice details but it indeed got the invoice details of that company even though the invoice number was incomplete, tried this to the last character on the invoice uuid before the test finally failed.

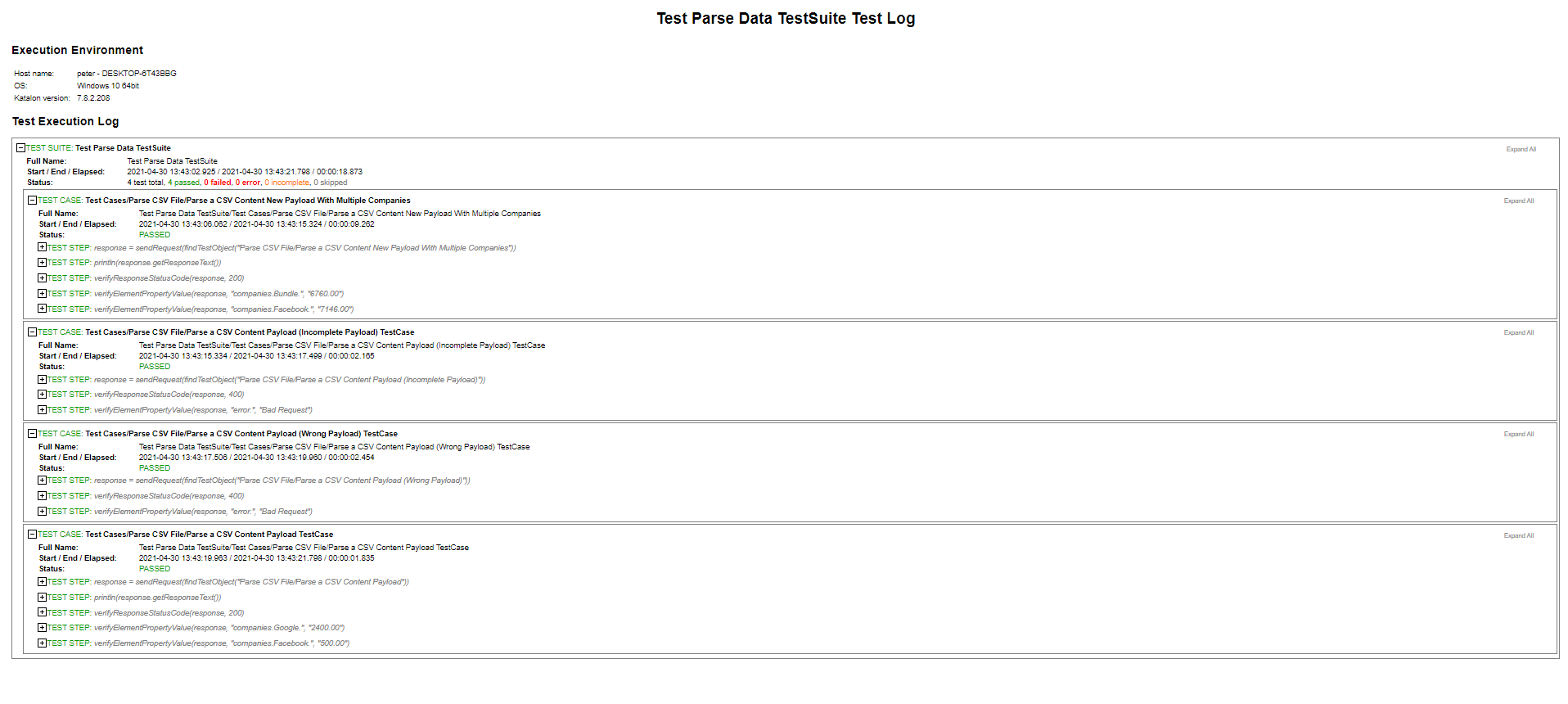
I tested that the API is capable of retrieving any company detail once the company name and invoice ID initially created is provided and this worked properly as expected.

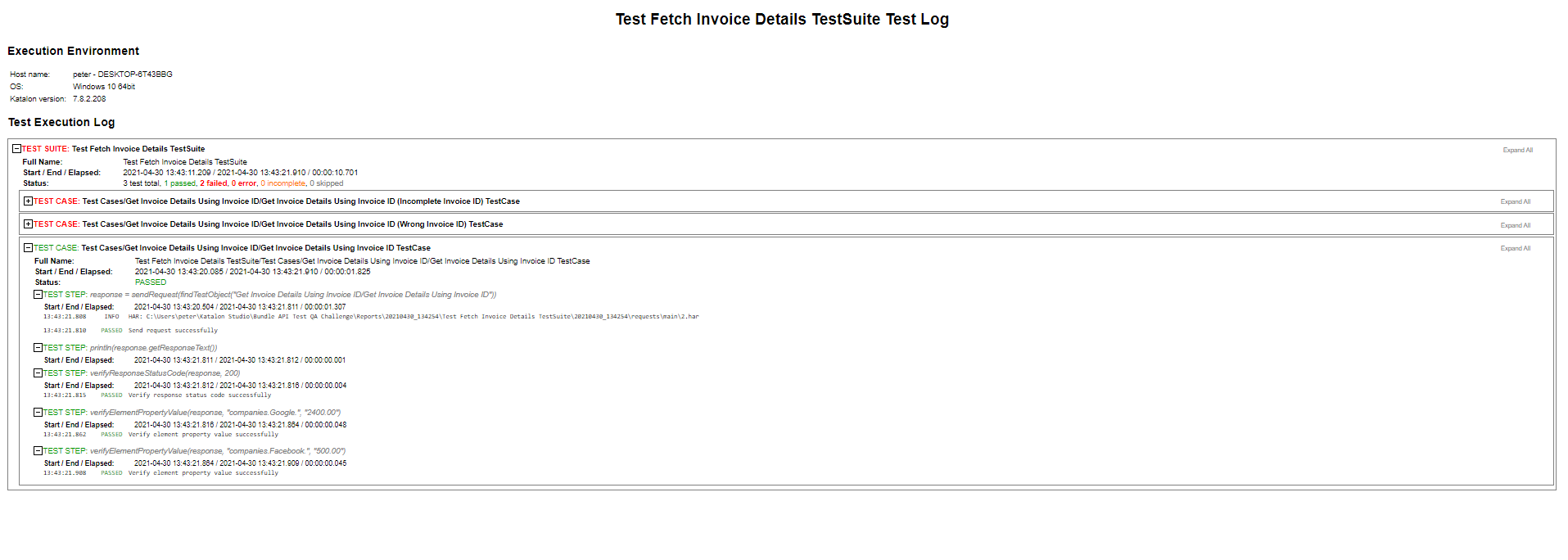
I tested that if an incorrect company name is provided with the get request via the API the response will not contain any company details and this worked properly as expected.

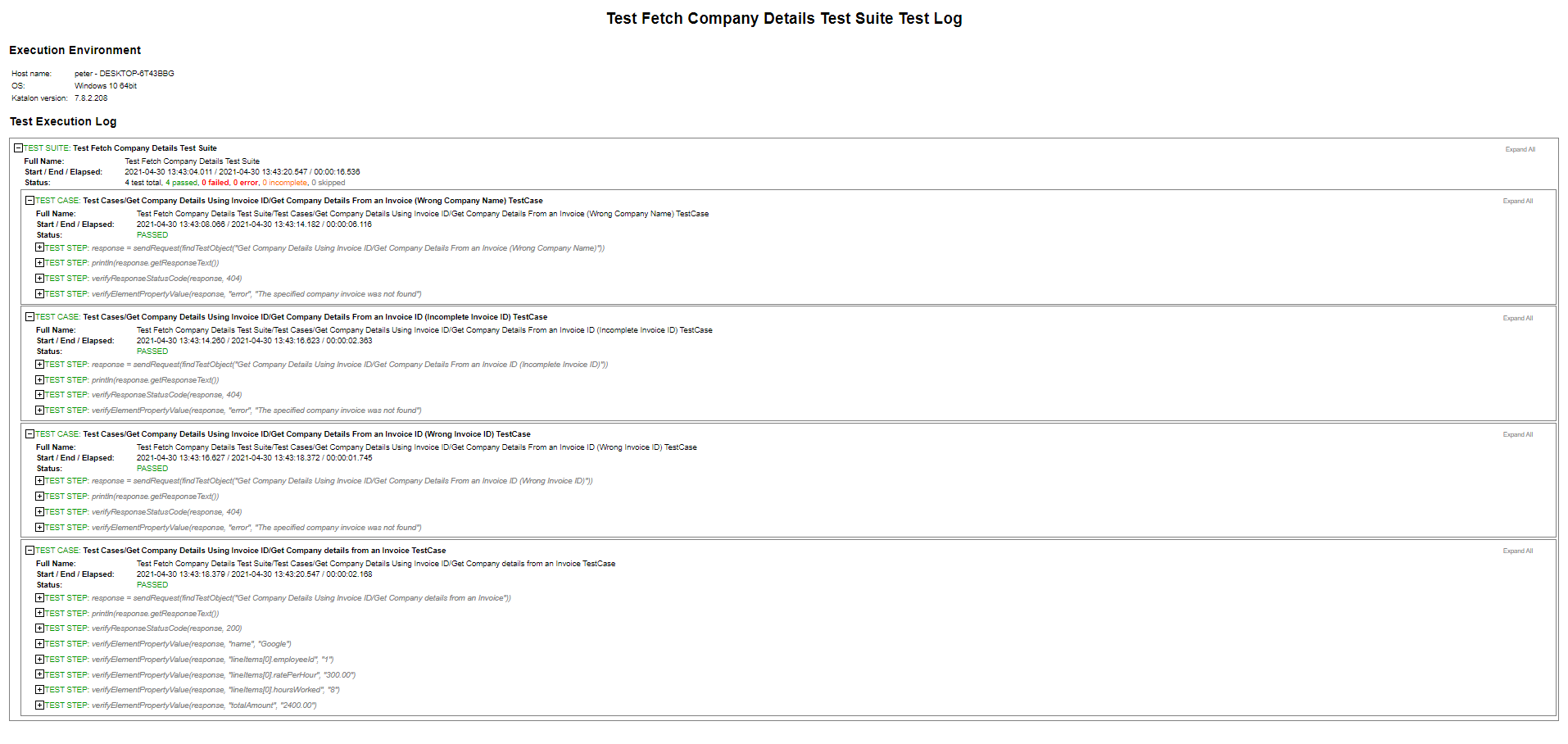
I finally tested the get company details request via the API using the correct name of the company initially created and stored but then I made the invoice wrong as well as incomplete and the test did not return any company details so this test passed, I believe that this method should also be applied to retrieving the invoice details.

Here are the results:









I load tested the API using Jmeter with 100 concurrent users sending requests and increasing every 5seconds in 5 loops and the server operated optimally with a throughput of about 2 requests per ms, the average time taken for all samples to execute the requests was 297ms this means that the server is operating optimally.

Here are the results:

