NED University of Engineering & Technology, Karachi

Department of Software Engineering

Software Quality Engineering (SE-309)
ASSIGMMENT

Professor: Dr.Kashif Mehboob Khan

NAME: KHAWAR KHAN
[redacted]

DATE: April 14, 2024

POSTMAN

Postman is a popular API development and testing tool that simplifies the process of working with APIs, empowering teams to build and maintain high-quality APIs more efficiently. Originally launched in 2012, it has become one of the go-to choices for developers, testers, and DevOps teams worldwide due to its intuitive interface and robust feature set.

Here are some key aspects and features of Postman:

1. **API Testing:** One of the primary features of Postman is its comprehensive API testing capabilities. It allows users to create and execute automated tests for REST, SOAP, and GraphQL APIs. Test scripts can be written in JavaScript, enabling advanced testing scenarios such as data-driven tests, assertions, and validations.
2. **Request Building:** With Postman, users can easily construct HTTP requests using a user-friendly interface. It supports various request types such as GET, POST, PUT, DELETE, etc., and allows for the customization of headers, parameters, body content, and authentication methods.
3. **Environment Variables:** This feature allows users to define variables once and use them dynamically in requests, making it easy to switch between configurations without modifying individual requests.
4. **Mock Servers:** Postman allows users to create mock servers for APIs, which can be used for testing purposes before the actual backend implementation is ready. We can develop or test an API before it's ready for production (or without utilizing production data) by using mock servers, which will return pre-defined data for API requests allowing for testing of the API with the frontend before production.
5. **API Documentation:** Postman provides tools for generating interactive API documentation from collections(contains number of requests). This documentation includes details such as request methods, request body, endpoints, authentication method, response schema, etc, providing it in human-readable for other developers to easily work with the APIs.
6. **Collaboration:** Postman offers collaboration features that facilitate teamwork. Users can share workspaces with other developers and can collaborate in real-time, also roles can be assigned to members you are sharing workspace with.
7. **Monitoring Performance:** Postman offers monitoring capabilities to track API performance and uptime. It provides detailed insights into response times, status codes, and error rates, helping teams identify and troubleshoot performance issues effectively.

Getting Started with Postman:

- Go to Postman website(https://www.postman.com/)
- Signup for free account.
- Then download postman application for desktop, for you version of Operating System.
- You can work start testing APIs directly at Postman website after login.

Starting Testing APIs:

- Now create a workspace for testing your APIs, the workspace contains all the APIs for one project.
- After creating a workspace, you can create different collections in the workspace for different endpoints of the APIs that you have defined. A Collection will contain the group of APIs made to similar endpoints.
- Now, in your collection, you can provide request method, request url(endpoint), request body parameters, authorization headers, and write test scripts in the IDE provided.
- Then you can run the request to execute all testcases.
- The API endpoint and the query parameters can also be stored in environment variables for their reusability.

Testing some Node APIs with Postman:

CODE:

index.js

```
const express = require('express');
const app = express();
const cors = require('cors');
const bodyParser = require('body-parser')
app.use(cors());
app.use(bodyParser.json());
const prodRouter = require('./product.router.js');
app.use("/product", prodRouter);

app.listen(8000, () => {
    console.log("App running on port : http://localhost:8000");
});
```

user.router.js

```
const Router = require('express').Router()
const { getAllProducts, getProductById, postAddProduct } =
require('./product.controller.js');

Router.get("/all", getAllProducts);
Router.get("/:id", getProductById);
Router.post("/add", postAddProduct);
module.exports = Router;
```

user.controller.js

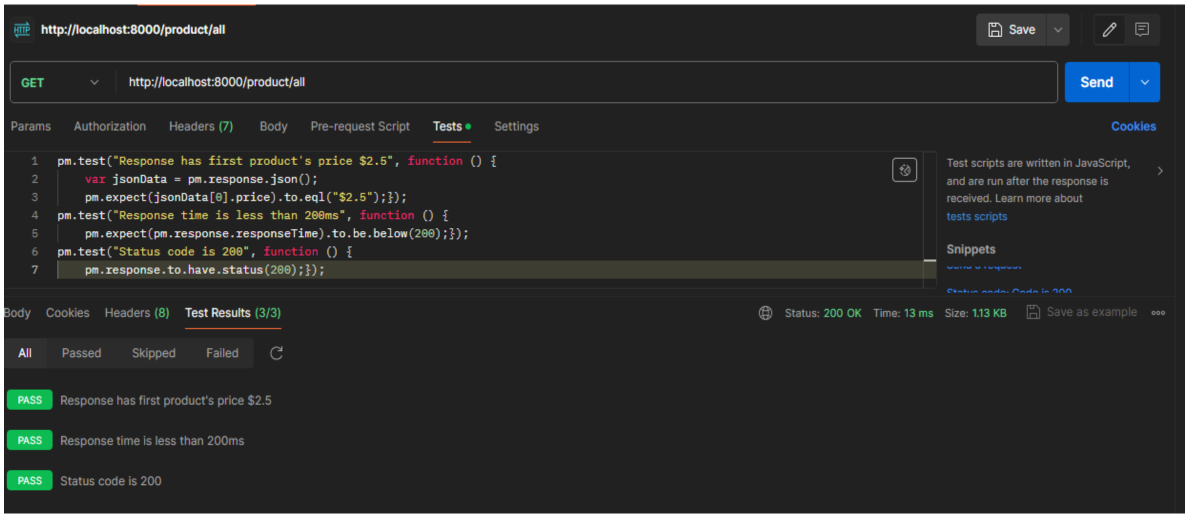
```
const productData = require('./product.data.json');
const getAllProducts = (req, res) => {
    res.send(productData);
};
const getProductById = (req, res) => {
    if(!(Number.isInteger(parseInt(req.params.id)))){
        res.status(400).json({
            message: "Id of the required product is not provided or
is not integer."
        });
    }else{
        let product = productData.filter(p =>
p.id==parseInt(req.params.id));
        if(product.length==0){
            res.status(400).json({
                message: "Provided id is not correct."
            });
        } else{
            res.status(200).json({
                data: product
            });
        };
    }
}


const postAddProduct = (req, res) => {
    const { id, name, description, price } = req.body;
    if(!(id&&name&&description&&price)){
        res.status(400).json({
            message: "Not enough data provided, provide all fields."
        });
    } else{
        productData.push({
            "id": id,
            "name": name,
            "description": description,
            "price": price
        });
        res.status(201).end();
    }
};

module.exports = {
    getAllProducts,
    getProductById,
    postAddProduct
}
```

product.data.json
contains a collection of objects

Repository URL: https://github.com/KhawarGit/APIs-for-SQE-assignment

TESTING Screenshots with Postman:

Hitting /product/all endpoint



```
GET  http://localhost:8000/product/all
```

```
pm.test("Response has first product's price $2.5", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[0].price).to.eql("$2.5");});
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);});
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);});
```

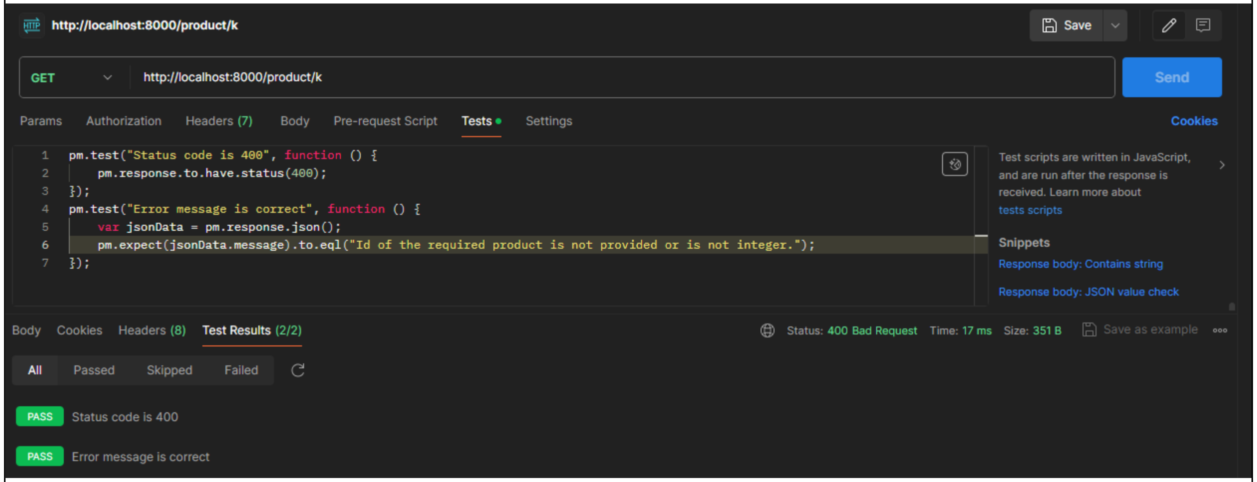
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Status: 200 OK Time: 13 ms Size: 1.13 KB

Test Results (3/3)

```
PASS  Response has first product's price $2.5
PASS  Response time is less than 200ms
PASS  Status code is 200
```

Hitting /product/:id endpoint with a non Numeric id of product.



GET http://localhost:8000/product/k

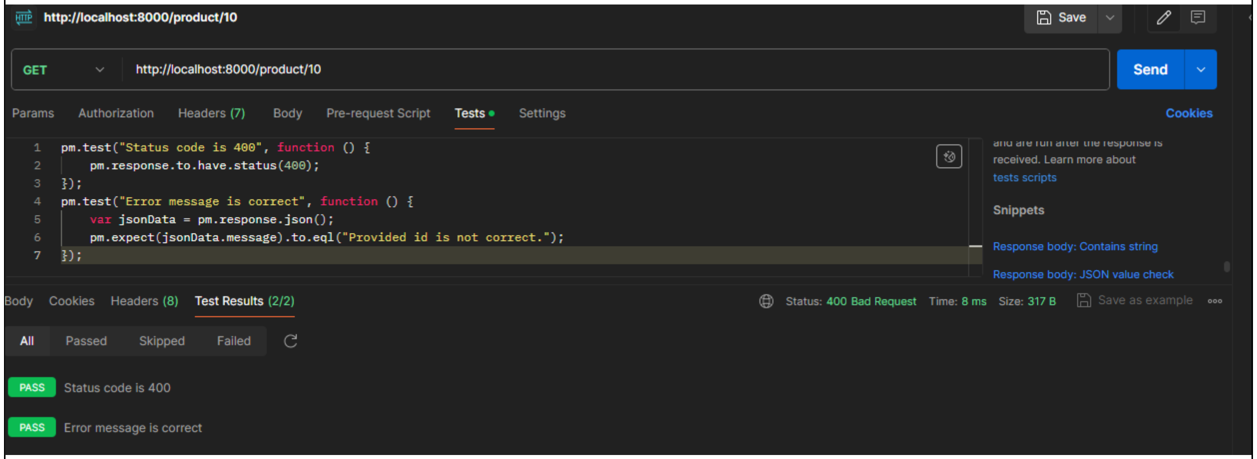
```
pm.test("Status code is 400", function () {
    pm.response.to.have.status(400);
});
pm.test("Error message is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql("Id of the required product is not provided or is not integer.");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Body Cookies Headers (8) Test Results (2/2) Status: 400 Bad Request Time: 17 ms Size: 351 B

```
PASS Status code is 400
PASS Error message is correct
```

Hitting /product/:id endpoint with an id which is not present in our product data.



GET http://localhost:8000/product/10

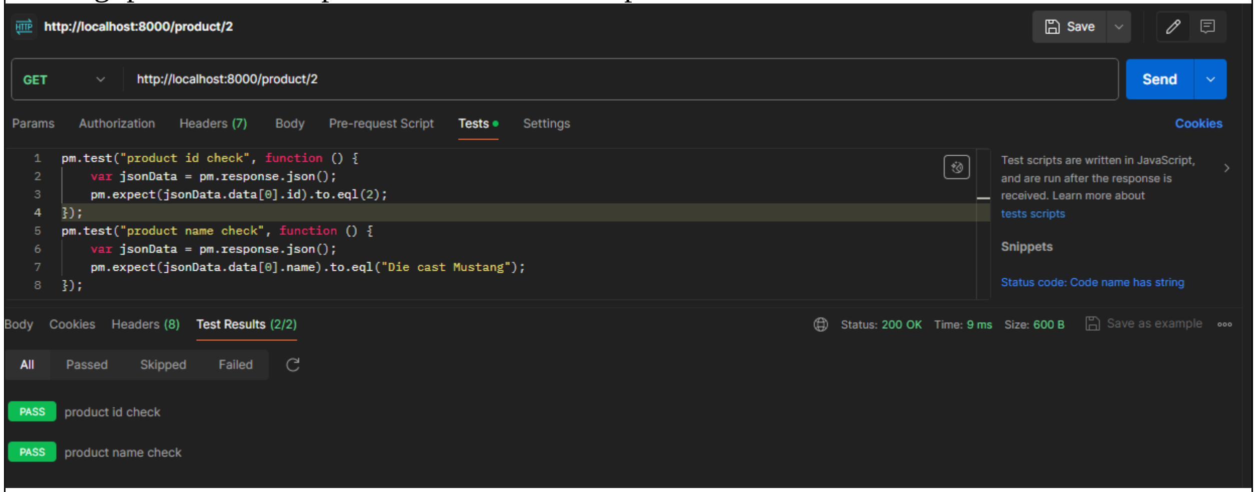
```
pm.test("Status code is 400", function () {
    pm.response.to.have.status(400);
});
pm.test("Error message is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql("Provided id is not correct.");
});
```

and are run after the response is received. Learn more about tests scripts

Body Cookies Headers (8) Test Results (2/2) Status: 400 Bad Request Time: 8 ms Size: 317 B

```
PASS Status code is 400
PASS Error message is correct
```

Hitting /product/:id endpoint with correct id of product.



GET http://localhost:8000/product/2

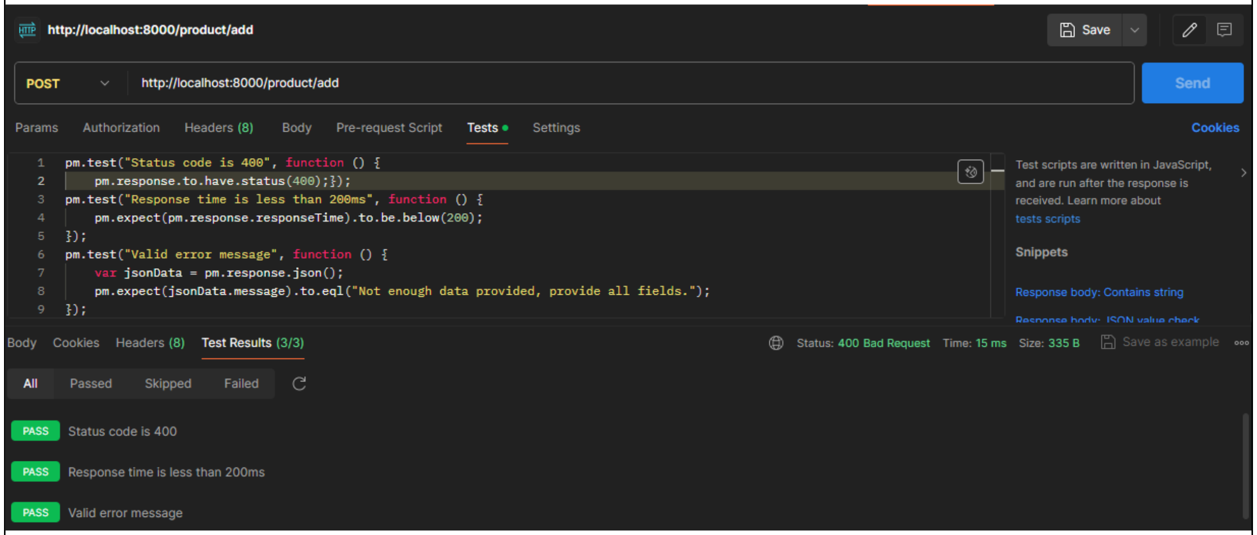
```
pm.test("product id check", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.data[0].id).to.eql(2);
});
pm.test("product name check", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.data[0].name).to.eql("Die cast Mustang");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Body Cookies Headers (8) Test Results (2/2) Status: 200 OK Time: 9 ms Size: 600 B

```
PASS product id check
PASS product name check
```

Hitting POST /product/add request without request body.



http://localhost:8000/product/add

POST http://localhost:8000/product/add

Tests

```
pm.test("Status code is 400", function () {
    pm.response.to.have.status(400);});
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});
pm.test("Valid error message", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql("Not enough data provided, provide all fields.");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Test Results (3/3)

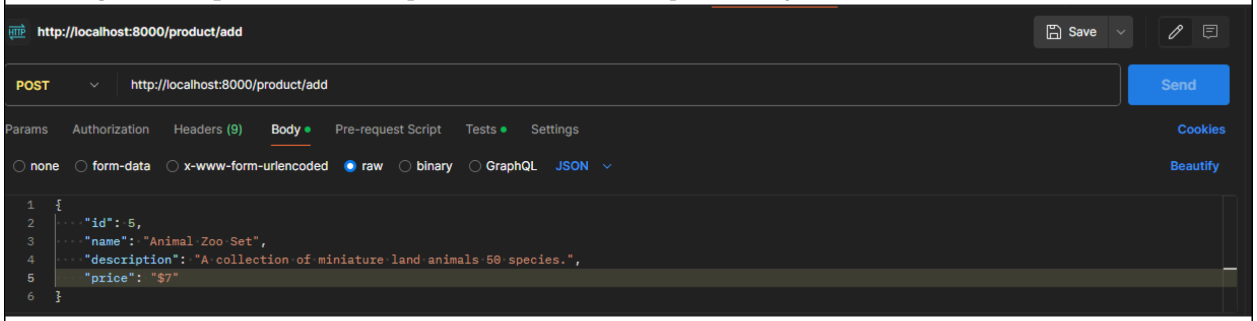
Status: 400 Bad Request Time: 15 ms Size: 335 B

PASS Status code is 400

PASS Response time is less than 200ms

PASS Valid error message

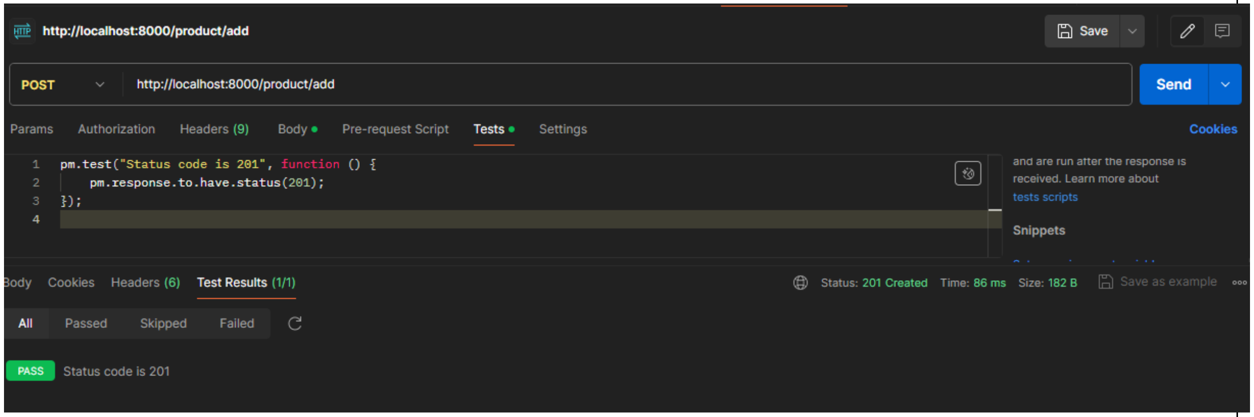
Hitting POST /product/add request with correct request body.



http://localhost:8000/product/add

POST http://localhost:8000/product/add

```
{
    "id": 5,
    "name": "Animal Zoo Set",
    "description": "A collection of miniature land animals 50 species.",
    "price": "$7"
}
```



http://localhost:8000/product/add

POST http://localhost:8000/product/add

Tests

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
```

Test Results (1/1)

Status: 201 Created Time: 86 ms Size: 182 B

PASS Status code is 201

2nd Tool: Katalon Studio

Katalon Studio is a free and robust automation solution for API, Web, Desktop, and Mobile Testing. It provides an easy start for beginners and also offers extensions and customization for experts. It provides native integration with popular CI tools. It is an integrated test automation platform designed to streamline the process of automated testing for web, API, mobile, and desktop applications. It offers a comprehensive suite of tools and features to help testers and developers create, execute, and manage automated tests efficiently.

Here are some key aspects and features of Katalon Studio:

1. **Cross-Platform Support:** Katalon Studio supports testing for a wide range of platforms, including web applications, APIs, mobile applications (iOS and Android), and desktop applications (Windows).
2. **Rich Set of Test Automation Capabilities:** Katalon Studio provides a rich set of built-in keywords and actions for test automation, enabling testers to create powerful and comprehensive test scripts without the need for extensive coding. It offers support for both keyword-driven testing and script-driven testing, catering to users with varying levels of technical expertise.
3. **Reporting and Analytics:** Test reports can be generated into different forms such as PDF, etc using Katalon Platform.
4. **Web Testing Feature:** Katalon Studio excels in web testing, offering a plethora of features to automate interactions with web elements, validate page content, and handle various scenarios such as pop-ups and frames. With support for popular web browsers like Chrome, Firefox, Safari, and Edge, testers can ensure their web applications function flawlessly across different browser environments.
5. **API Testing Feature:** Simplifying the intricacies of API testing, Katalon Studio provides intuitive tools for creating, executing, and validating API requests and responses. Testers can effortlessly construct API requests, specify parameters and headers, and validate response data using built-in assertions. With support for RESTful APIs, SOAP services, and GraphQL, Katalon Studio caters to a broad spectrum of API testing needs.
6. **Mobile Testing Feature:** Katalon Studio offers a comprehensive suite of features for both iOS and Android applications. Testers can automate interactions with mobile app elements, conduct functional testing, and ensure consistent behavior across various devices and screen sizes.
7. **Desktop Testing Feature:** For desktop application testing on the Windows platform, Katalon Studio provides robust capabilities to automate interactions with UI elements, perform functional testing, and validate application behavior across different Windows versions.

Getting Started with Katalon Studio:

- Visit the Katalon Studio website at https://katalon.com/download and download appropriate version of Katalon Studio for your operating system (Windows, macOS, Linux).

- Katalon studio offers both free version and a paid version, choose the one that best suits your needs.
- After downloading Katalon Studio, install from your downloads folder.
- Create a Katalon Account on the Katalon website and login your Katalon Studio.
- After successful login, you can explore Katalon Studio features.

Starting Testing APIs:

- Select "New Draft REST Request" option.
- Provide your API url, and other necessary data, query string and request body, and authorization headers.
- Then save this request in "Object Repository" folder.
- Then right click on "Test Cases" and select new Test Case.
- Provide name for the new Test Case.
- First select API request from "Object Repository" folder.
- then select "Add Web Service Keyword" option for inserting a test scenario or you can write directly in the script after select "Script" option.
- Then run this test case.
- You can add test cases in a test suite to form a collection of test cases.

Testing Some Node APIs with Katalon Studio:

We test the same APIs which code is provided above.

CODE:

index.js

```
const express = require('express');
const app = express();
const cors = require('cors');
const bodyParser = require('body-parser')
app.use(cors());
app.use(bodyParser.json());
const prodRouter = require('./product.router.js');
app.use("/product", prodRouter);

app.listen(8000, () => {
    console.log("App running on port : http://localhost:8000");
});
```

user.router.js

```
const Router = require('express').Router()
const { getAllProducts, getProductById, postAddProduct } =
require('./product.controller.js');
```

```
Router.get("/all", getAllProducts);
Router.get("/:id", getProductById);
Router.post("/add", postAddProduct);
module.exports = Router;
```

user.controller.js

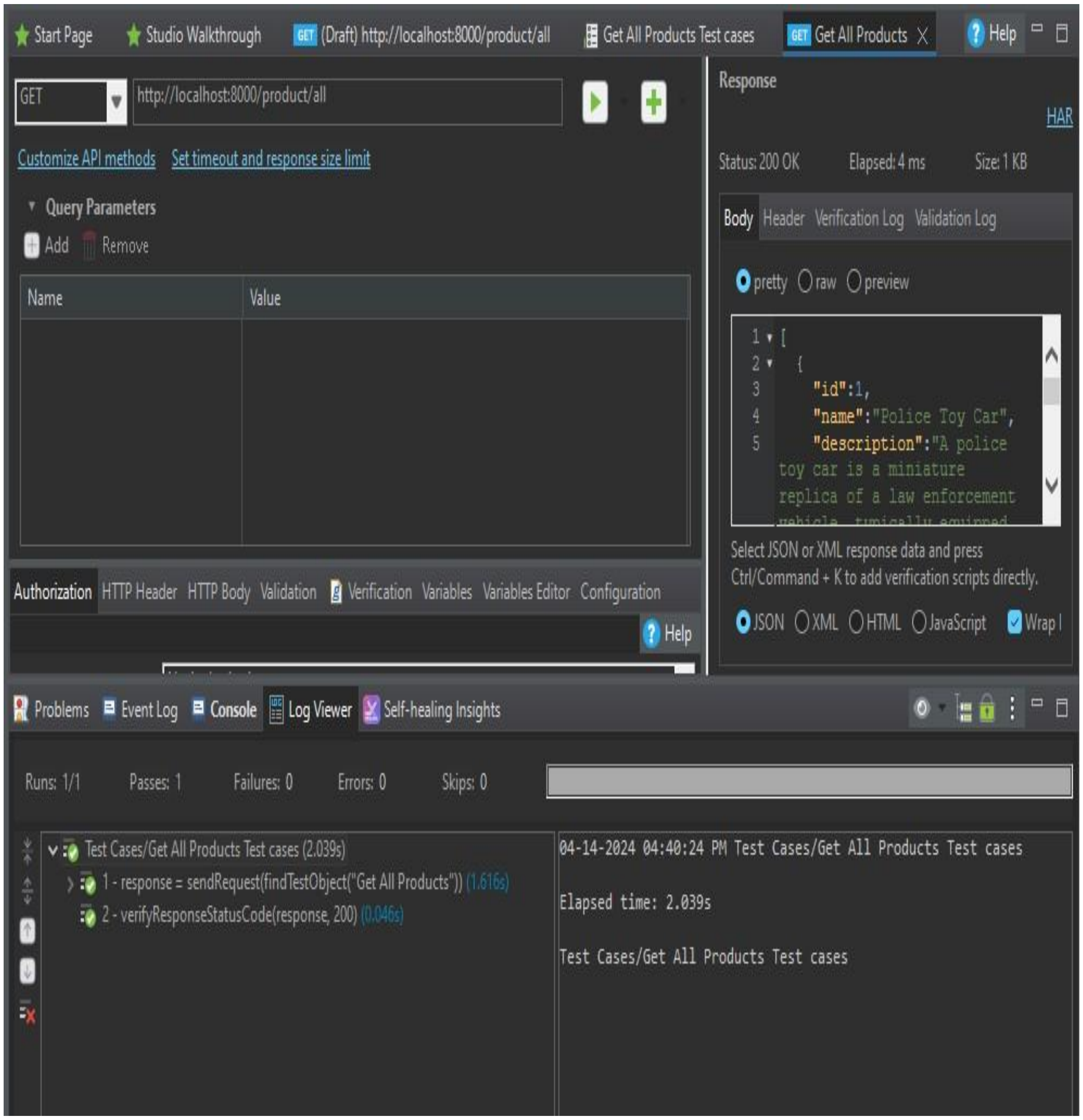
```
const productData = require('./product.data.json');

const getAllProducts = (req, res) => {
    res.send(productData);
};
const getProductById = (req, res) => {
    if(!(Number.isInteger(parseInt(req.params.id)))){
        res.status(400).json({
            message: "Id of the required product is not provided or
is not integer."
        });
    }else{
        let product = productData.filter(p =>
p.id==parseInt(req.params.id));
        if(product.length==0){
            res.status(400).json({
                message: "Provided id is not correct."
            });
        } else{
            res.status(200).json({
                data: product
            });
        };
    }
}


const postAddProduct = (req, res) => {
    const { id, name, description, price } = req.body;
    if(!(id&&name&&description&&price)){
        res.status(400).json({
            message: "Not enough data provided, provide all fields."
        });
    } else{
        productData.push({
            "id": id,
            "name": name,
            "description": description,
            "price": price
        });

        res.status(201).end();
    }
};

module.exports = {
    getAllProducts,
    getProductById,
    postAddProduct
}
```

product.data.json
contains a collection of objects

Repository URL: https://github.com/KhawarGit/APIs-for-SQE-assignment

Testing Screenshots with Katalon Studio:

GET /product/all



Status: 200 OK Elapsed: 4 ms Size: 1 KB

```
[
  {
    "id":1,
    "name":"Police Toy Car",
    "description":"A police
toy car is a miniature
replica of a law enforcement
```

Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly.

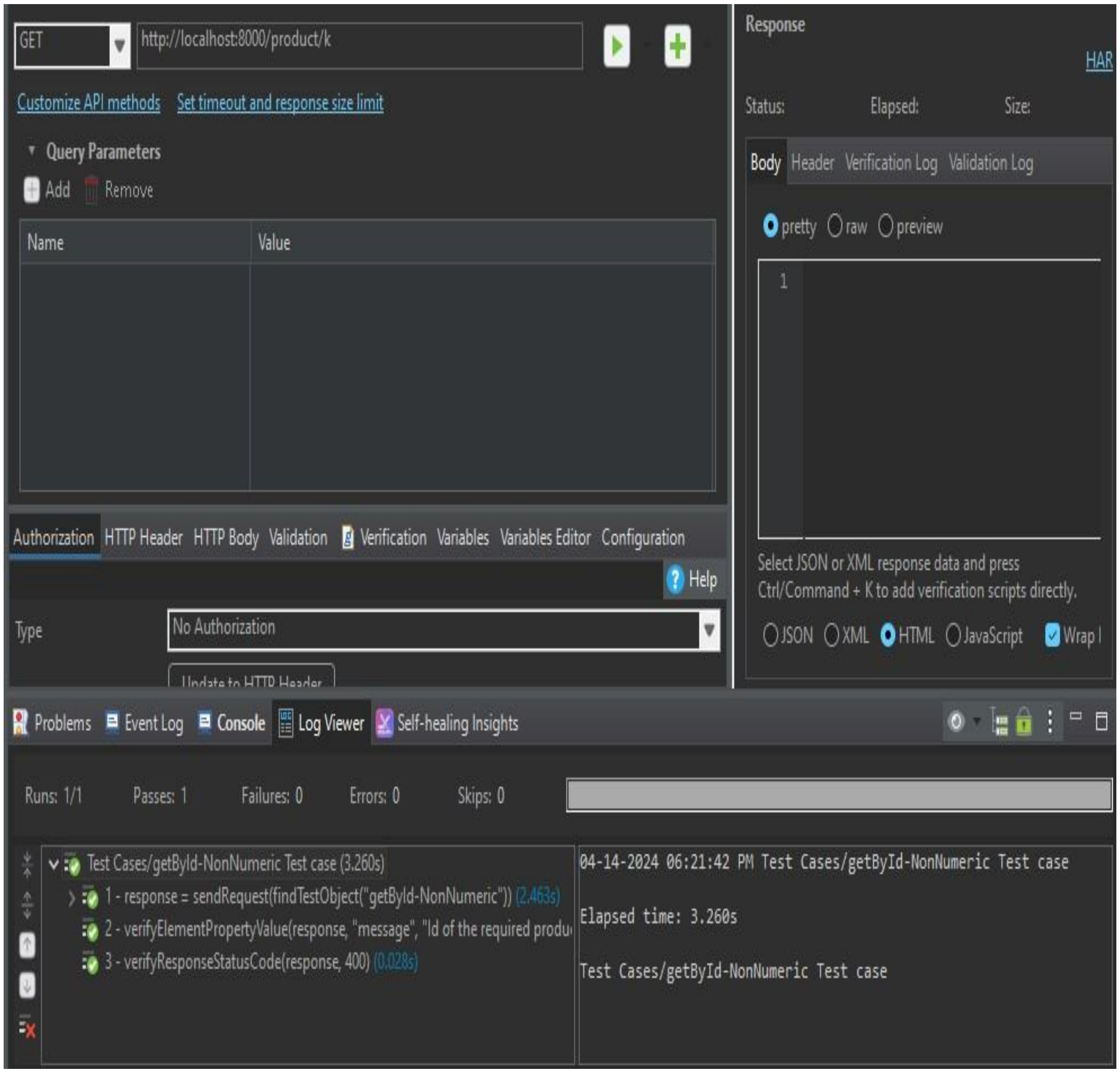
[✓ JSON] XML HTML JavaScript [✓ Wrap l]

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/Get All Products Test cases (2.039s)
  1 - response = sendRequest(findTestObject("Get All Products")) (1.616s)
  2 - verifyResponseStatusCode(response, 200) (0.046s)
```

```
04-14-2024 04:40:24 PM Test Cases/Get All Products Test cases

Elapsed time: 2.039s

Test Cases/Get All Products Test cases
```

GET /product/:id with invalid Non-numeric id.



GET http://localhost:8000/product/k

Customize API methods Set timeout and response size limit

Query Parameters

Name	Value

Authorization HTTP Header HTTP Body Validation Verification Variables Variables Editor Configuration

Type: No Authorization

Response

Status: Elapsed: Size:

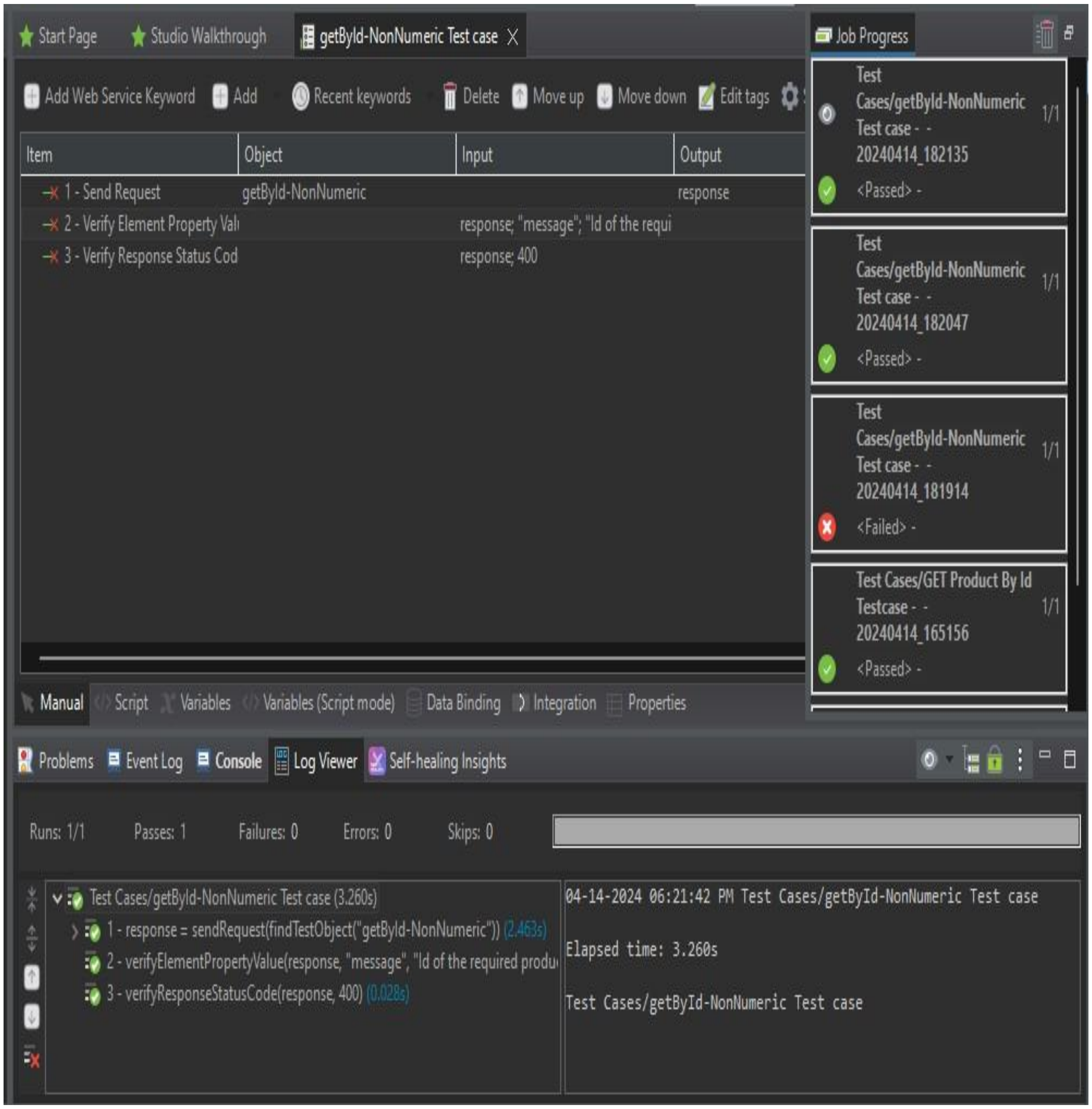
Body Header Verification Log Validation Log

Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly.

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/getById-NonNumeric Test case (3.260s)
  1 - response = sendRequest(findTestObject("getById-NonNumeric")) (2.463s)
  2 - verifyElementPropertyValue(response, "message", "Id of the required produ
  3 - verifyResponseStatusCode(response, 400) (0.028s)
```

```
04-14-2024 06:21:42 PM Test Cases/getById-NonNumeric Test case

Elapsed time: 3.260s

Test Cases/getById-NonNumeric Test case
```



Item	Object	Input	Output
1 - Send Request	getById-NonNumeric		response
2 - Verify Element Property Val		response; "message"; "Id of the requi	
3 - Verify Response Status Cod		response; 400	

Test Cases/getById-NonNumeric Test case - - 20240414_182135 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_182047 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_181914 1/1 <Failed> -

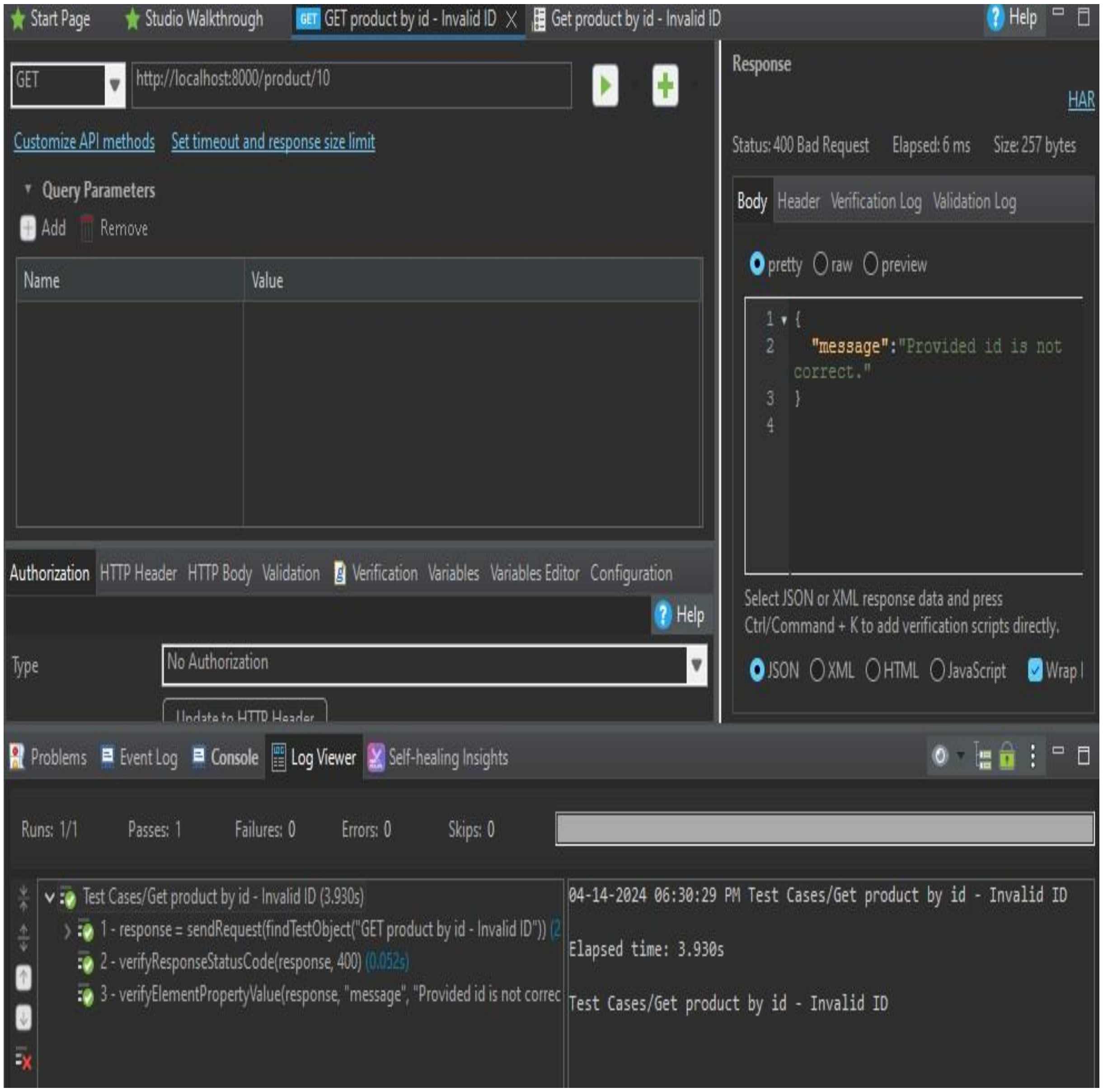
Test Cases/GET Product By Id Testcase - - 20240414_165156 1/1 <Passed> -

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/getById-NonNumeric Test case (3.260s)
  1 - response = sendRequest(findTestObject("getById-NonNumeric")) (2.463s)
  2 - verifyElementPropertyValue(response, "message", "Id of the required produ
  3 - verifyResponseStatusCode(response, 400) (0.028s)
```

```
04-14-2024 06:21:42 PM Test Cases/getById-NonNumeric Test case

Elapsed time: 3.260s

Test Cases/getById-NonNumeric Test case
```

GET product/:id with Invalid ID whose corresponding product is not present.



GET http://localhost:8000/product/10

Type: No Authorization

Status: 400 Bad Request Elapsed: 6 ms Size: 257 bytes

pretty raw preview

```
{
  "message":"Provided id is not correct."
}
```

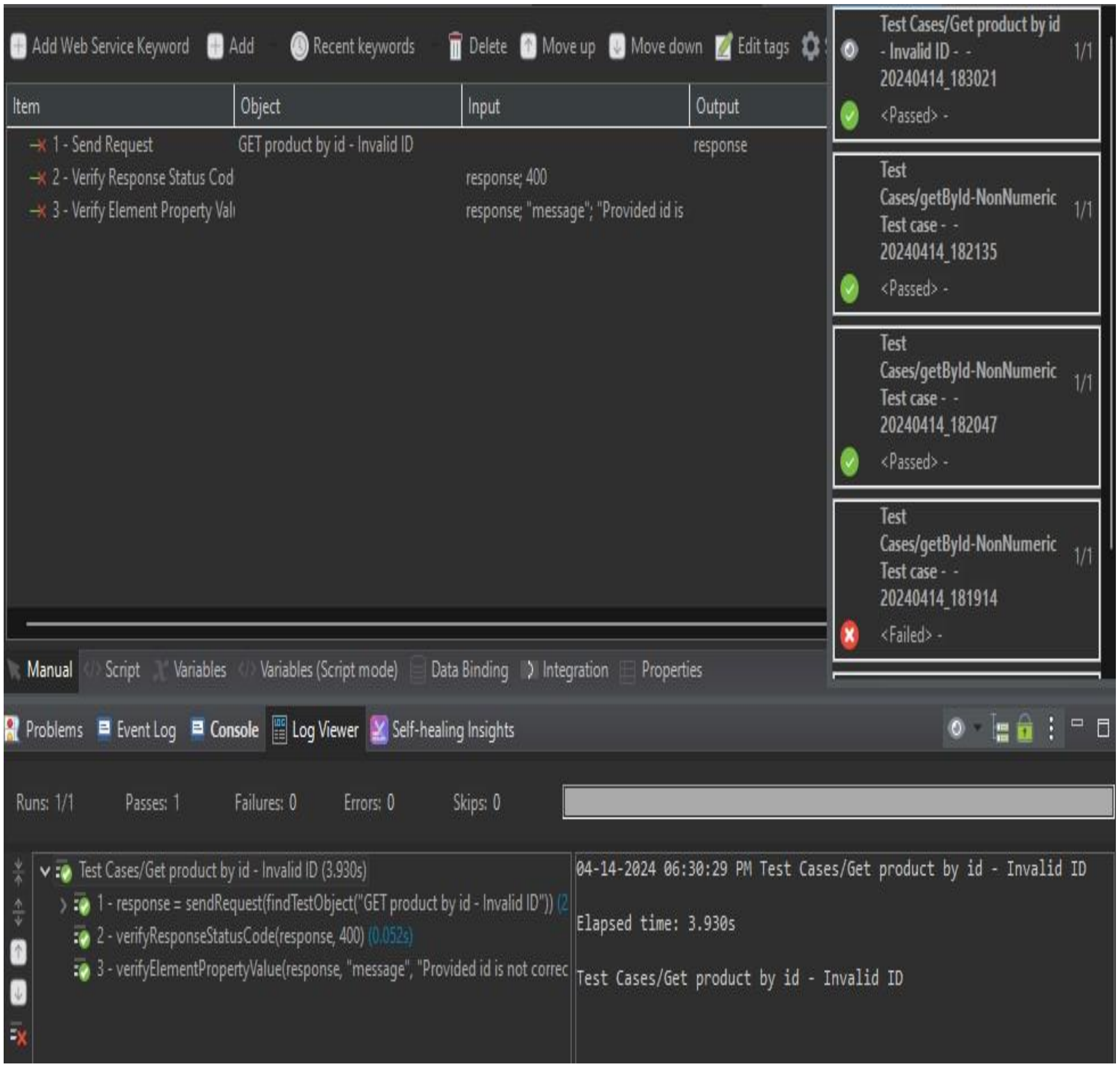
Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly.

JSON XML HTML JavaScript Wrap l

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/Get product by id - Invalid ID (3.930s)
  1 - response = sendRequest(findTestObject("GET product by id - Invalid ID")) (2
  2 - verifyResponseStatusCode(response, 400) (0.052s)
  3 - verifyElementPropertyValue(response, "message", "Provided id is not correc
```

```
04-14-2024 06:30:29 PM Test Cases/Get product by id - Invalid ID

Elapsed time: 3.930s

Test Cases/Get product by id - Invalid ID
```



Item	Object	Input	Output
1 - Send Request	GET product by id - Invalid ID		response
2 - Verify Response Status Cod		response; 400	
3 - Verify Element Property Val		response; "message"; "Provided id is	

Test Cases/Get product by id - Invalid ID - - 20240414_183021 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_182135 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_182047 1/1 <Passed> -

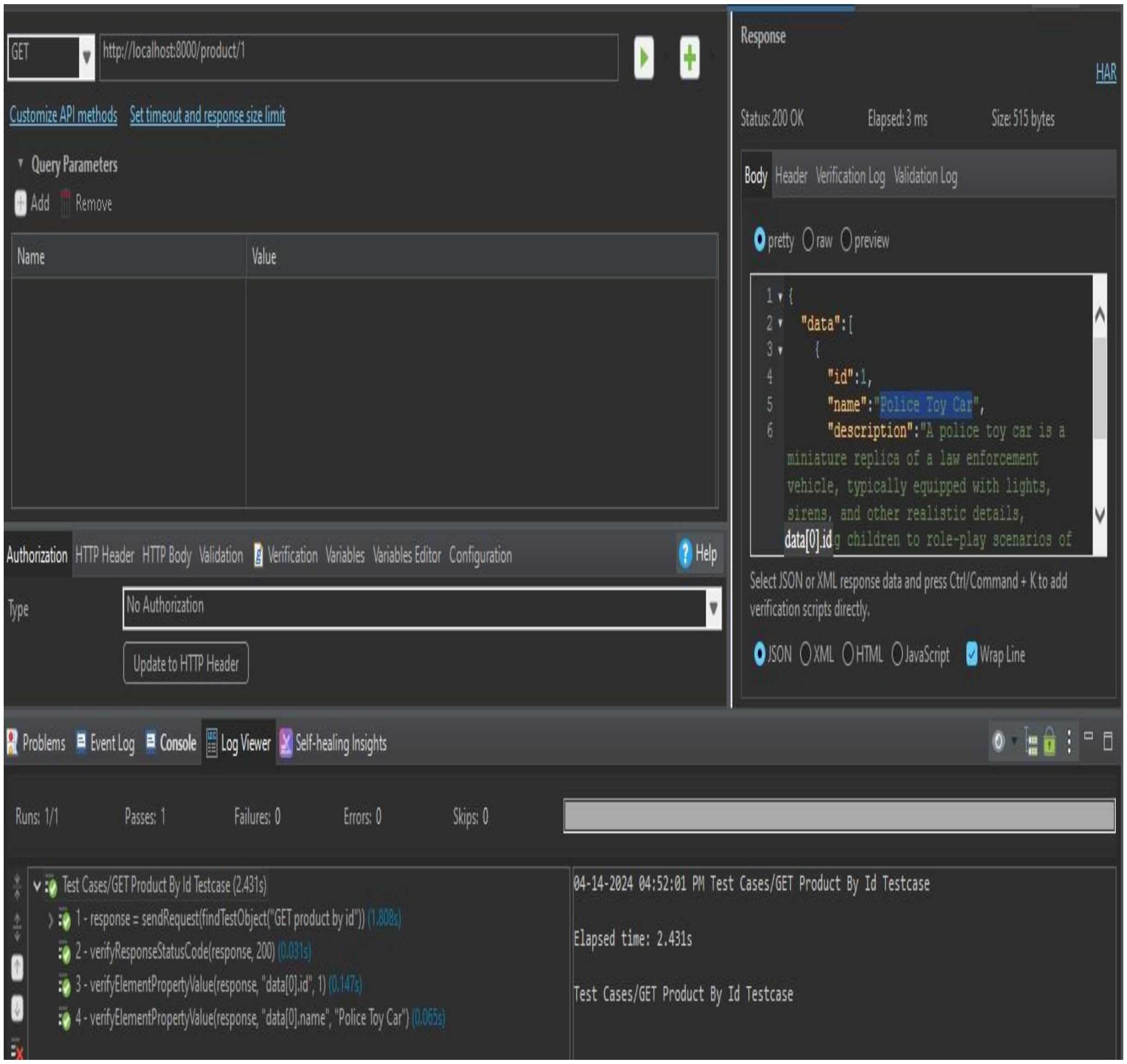
Test Cases/getById-NonNumeric Test case - - 20240414_181914 1/1 <Failed> -

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/Get product by id - Invalid ID (3.930s)
  1 - response = sendRequest(findTestObject("GET product by id - Invalid ID")) (2
  2 - verifyResponseStatusCode(response, 400) (0.052s)
  3 - verifyElementPropertyValue(response, "message", "Provided id is not correc
```

```
04-14-2024 06:30:29 PM Test Cases/Get product by id - Invalid ID

Elapsed time: 3.930s

Test Cases/Get product by id - Invalid ID
```

GET product/:id with correct id



GET http://localhost:8000/product/1

Status: 200 OK Elapsed: 3 ms Size: 515 bytes

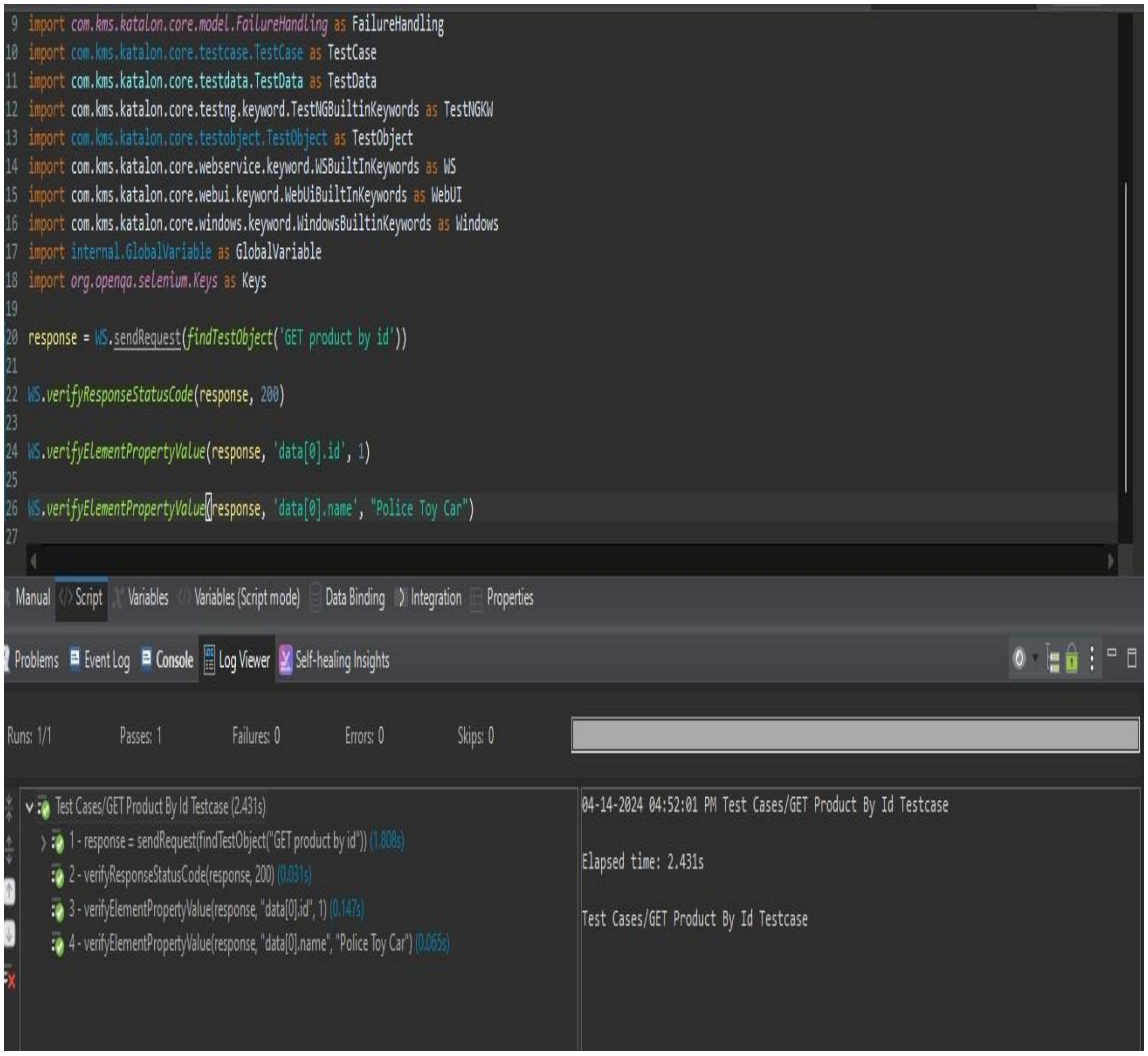
```
{
  "data":[
    {
      "id":1,
      "name":"Police Toy Car",
      "description":"A police toy car is a miniature replica of a law enforcement vehicle, typically equipped with lights, sirens, and other realistic details, ...g children to role-play scenarios of
```

Type: No Authorization

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/GET Product By Id Testcase (2.431s)
  1 - response = sendRequest(findTestObject("GET product by id")) (1.808s)
  2 - verifyResponseStatusCode(response, 200) (0.031s)
  3 - verifyElementPropertyValue(response, "data[0].id", 1) (0.147s)
  4 - verifyElementPropertyValue(response, "data[0].name", "Police Toy Car") (0.065s)
```

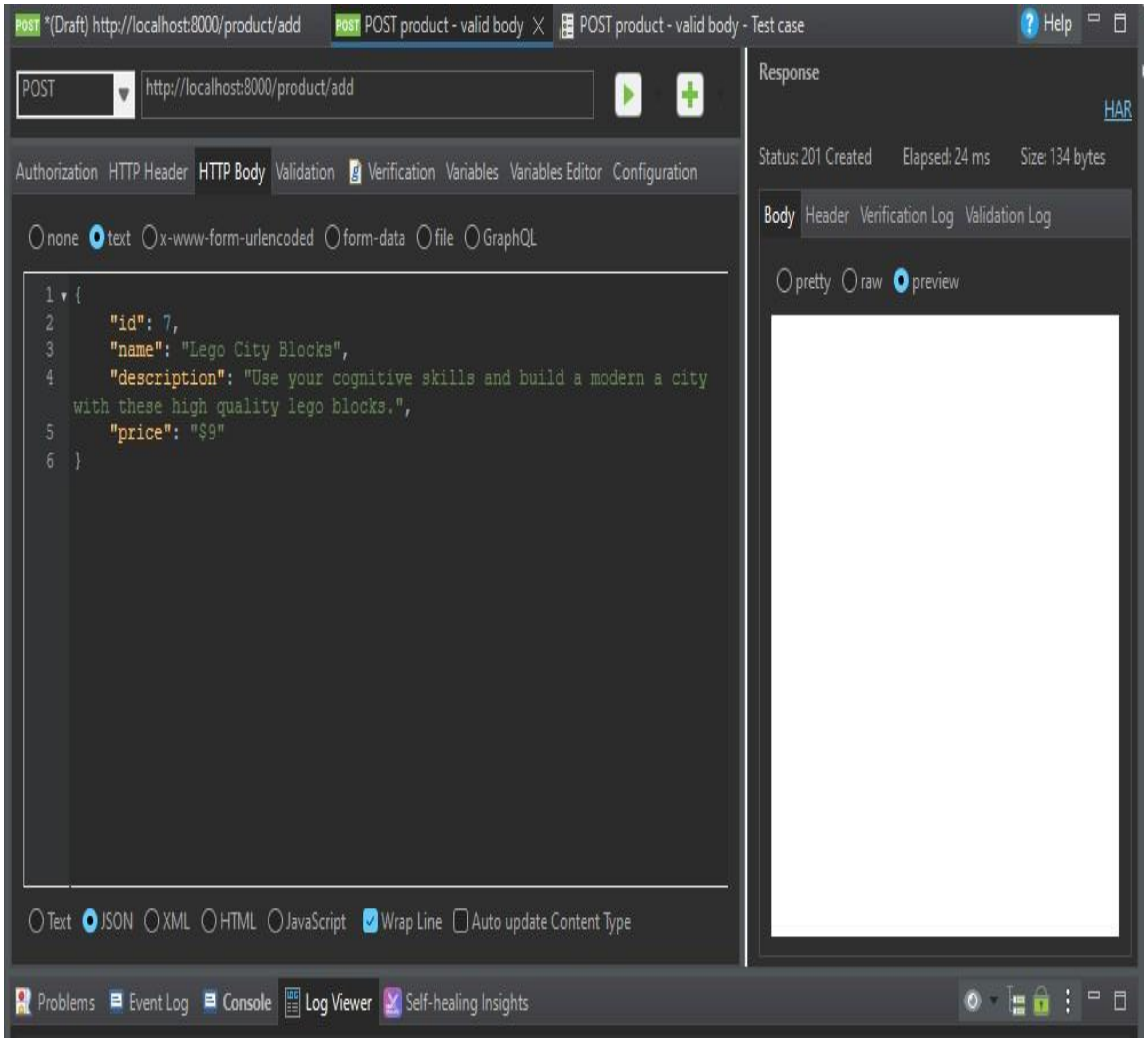
```
04-14-2024 04:52:01 PM Test Cases/GET Product By Id Testcase

Elapsed time: 2.431s

Test Cases/GET Product By Id Testcase
```



```
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.core.testdata.TestData as TestData
import com.kms.katalon.core.testng.keyword.TestNGBuiltinKeywords as TestNGKW
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import com.kms.katalon.core.windows.keyword.WindowsBuiltinKeywords as Windows
import internal.GlobalVariable as GlobalVariable
import org.openqa.selenium.Keys as Keys

response = WS.sendRequest(findTestObject('GET product by id'))

WS.verifyResponseStatusCode(response, 200)

WS.verifyElementPropertyValue(response, 'data[0].id', 1)

WS.verifyElementPropertyValue(response, 'data[0].name', "Police Toy Car")

```

```
Runs: 1/1    Passes: 1    Failures: 0    Errors: 0    Skips: 0

Test Cases/GET Product By Id Testcase (2.431s)
  1 - response = sendRequest(findTestObject("GET product by id")) (1.808s)
  2 - verifyResponseStatusCode(response, 200) (0.031s)
  3 - verifyElementPropertyValue(response, "data[0].id", 1) (0.147s)
  4 - verifyElementPropertyValue(response, "data[0].name", "Police Toy Car") (0.065s)

04-14-2024 04:52:01 PM Test Cases/GET Product By Id Testcase

Elapsed time: 2.431s

Test Cases/GET Product By Id Testcase
```

POST product/add with correct request body



none [✓ text] x-www-form-urlencoded form-data file GraphQL

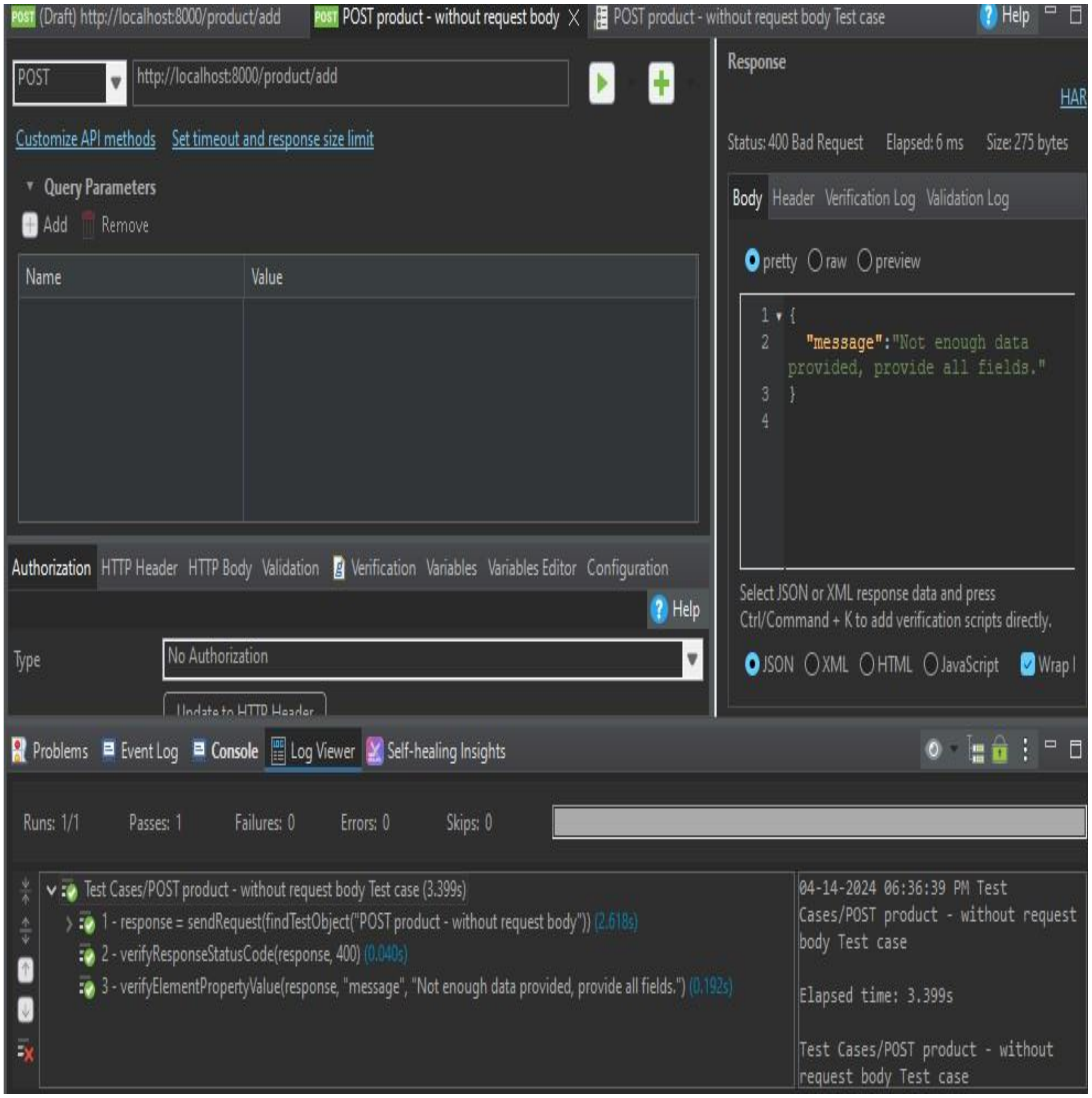
```
{
    "id": 7,
    "name": "Lego City Blocks",
    "description": "Use your cognitive skills and build a modern a city with these high quality lego blocks.",
    "price": "$9"
}
```

Text [✓ JSON] XML HTML JavaScript [✓ Wrap Line] Auto update Content Type

Status: 201 Created   Elapsed: 24 ms   Size: 134 bytes

pretty raw [✓ preview]

POST product/add without request body.



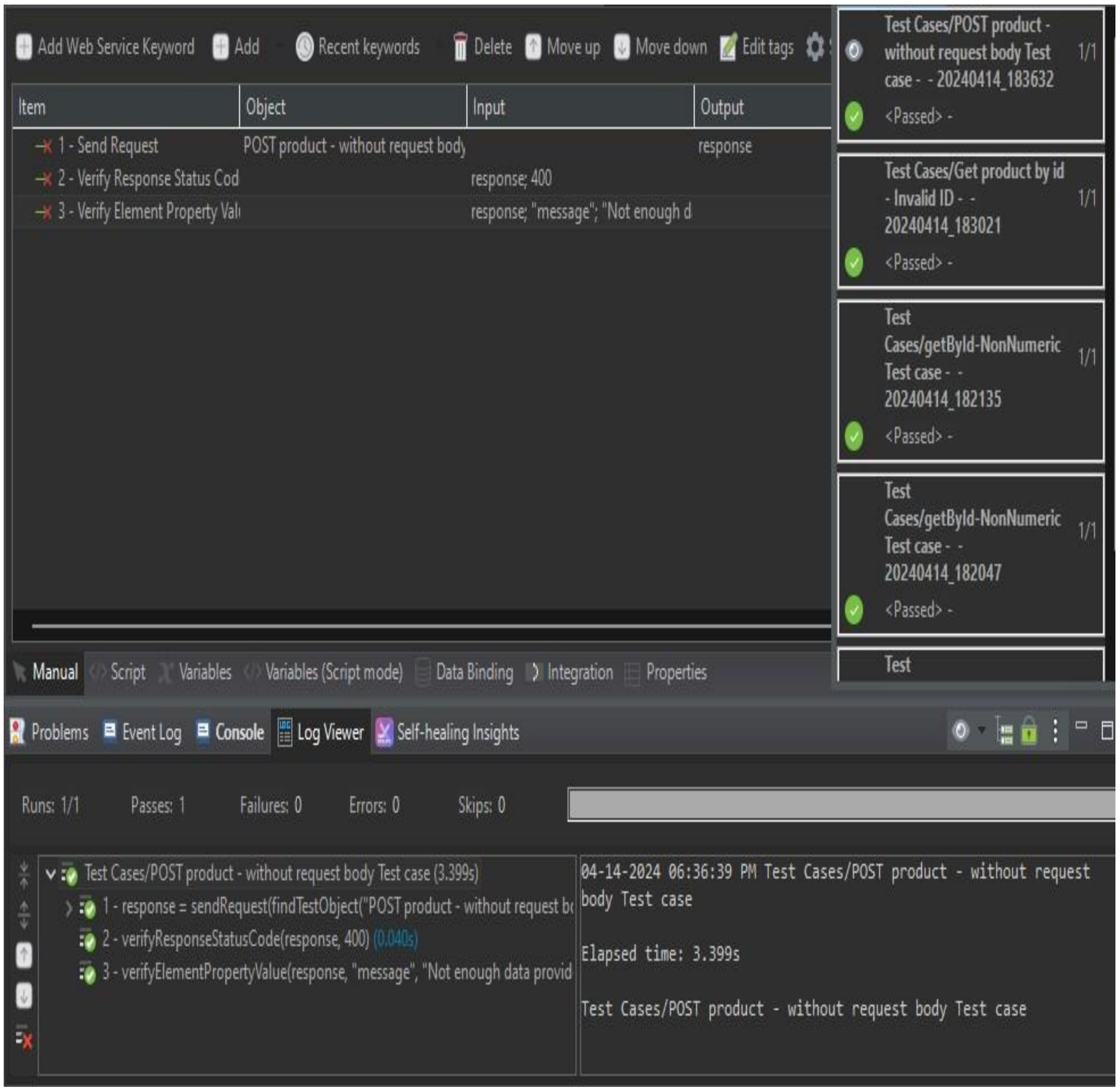
Status: 400 Bad Request   Elapsed: 6 ms   Size: 275 bytes

```
{
  "message":"Not enough data provided, provide all fields."
}
```

Runs: 1/1   Passes: 1   Failures: 0   Errors: 0   Skips: 0

```
Test Cases/POST product - without request body Test case (3.399s)
  1 - response = sendRequest(findTestObject("POST product - without request body")) (2.618s)
  2 - verifyResponseStatusCode(response, 400) (0.040s)
  3 - verifyElementPropertyValue(response, "message", "Not enough data provided, provide all fields.") (0.192s)
```

```
04-14-2024 06:36:39 PM Test Cases/POST product - without request body Test case

Elapsed time: 3.399s

Test Cases/POST product - without request body Test case
```



Item	Object	Input	Output
1 - Send Request	POST product - without request body		response
2 - Verify Response Status Cod		response; 400	
3 - Verify Element Property Val		response; "message"; "Not enough d	

Test Cases/POST product - without request body Test case - - 20240414_183632 1/1 <Passed> -

Test Cases/Get product by id - Invalid ID - - 20240414_183021 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_182135 1/1 <Passed> -

Test Cases/getById-NonNumeric Test case - - 20240414_182047 1/1 <Passed> -

Runs: 1/1 Passes: 1 Failures: 0 Errors: 0 Skips: 0

```
Test Cases/POST product - without request body Test case (3.399s)
  1 - response = sendRequest(findTestObject("POST product - without request bo
  2 - verifyResponseStatusCode(response, 400) (0.040s)
  3 - verifyElementPropertyValue(response, "message", "Not enough data provid
```

```
04-14-2024 06:36:39 PM Test Cases/POST product - without request
body Test case

Elapsed time: 3.399s

Test Cases/POST product - without request body Test case
```