**Test Summary Report**

# Contents

1. Purpose 2
2. Application Overview 2
3. Testing Scope 2
4. Metrics 3
5. Test Tools 5
6. Types of testing performed 6
7. Recommendations 6
8. Conclusion 7

## Purpose

This document explains the various activities performed as part of API Testing of restful services which serve their content via HTTP.

## Application Overview

Auto1.com Application is built to provide the best price to sell cars by choosing from provided interdependent dropdown options. There are several modules like Car Manufacture, Car Model & Car Registration Year which are integrated to fulfill the purpose.

## Testing Scope

* + 1. **In Scope**

API Testing of for the following modules are in Scope of Testing

● **GET** /v1/cartypes/manufacturer

○ locale ( accepts any locale)

○ wa\_key

● **GET** /v1/cartypes/maintypes

○ manufacturer (accepts manufacturer codes)

○ locale (accepts any locale)

○ wa\_key

● **GET** /v1/cartypes/builtdates

○ manufacturer (accepts manufacturer codes)

○ maintype(accepts maintype codes)

○ locale

* + 1. **Out of Scope**

Services also accept other input parameters like pagination or sorting but those are not in the scope of this task.

## Test Tools

**Test Tools:**

This automation framework has been developed using some of the best open source technologies which have large community based support available for continuous improvement.

**JDK 1.8 –** Utilized for Java Stream API for Bulk Data Operations on Collections and forEach() method in Iterable interface.

**Rest Assured –** Easy setup for HTTP connection, send a request, receive and parse a response.

**Maven –** For build, execution & version control

**Cucumber –** Implementation of BDD automation framework. Feature based Gherkin test scenarios are easy to understand for both technical & business team.

**TestNG –** produce HTML Reports, generate logs & Parallel testing capable.

**Jackson Core –** Utilized for POJO creation to handle Serialization & Deserialization of JSON objects.

**Log4J –** Utilized to log info, warning, errors for debugging purpose.

**Apache POI –** Utilized to use excel as input source to handle data variations in different environments due to implementation of new manufacturer, car models or registration year.

**Cucumber Basic Reports –** Comes by default with cucumber implementation.

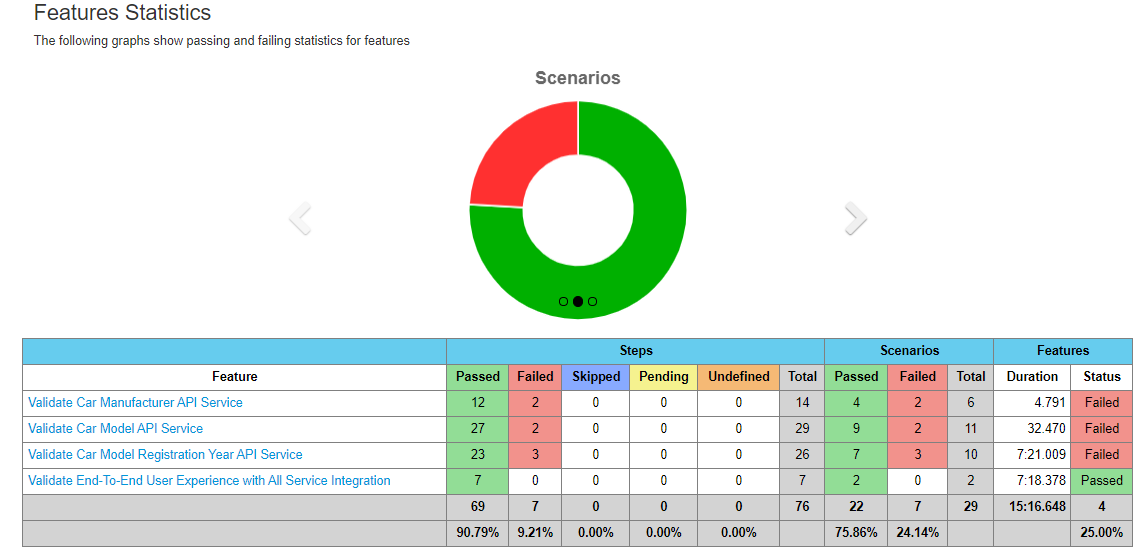
**Cucumber JVM Reports (Jenkins like) –** These reports can be utilized to share the automation status among the scrum team, IT leadership & business stakeholders.

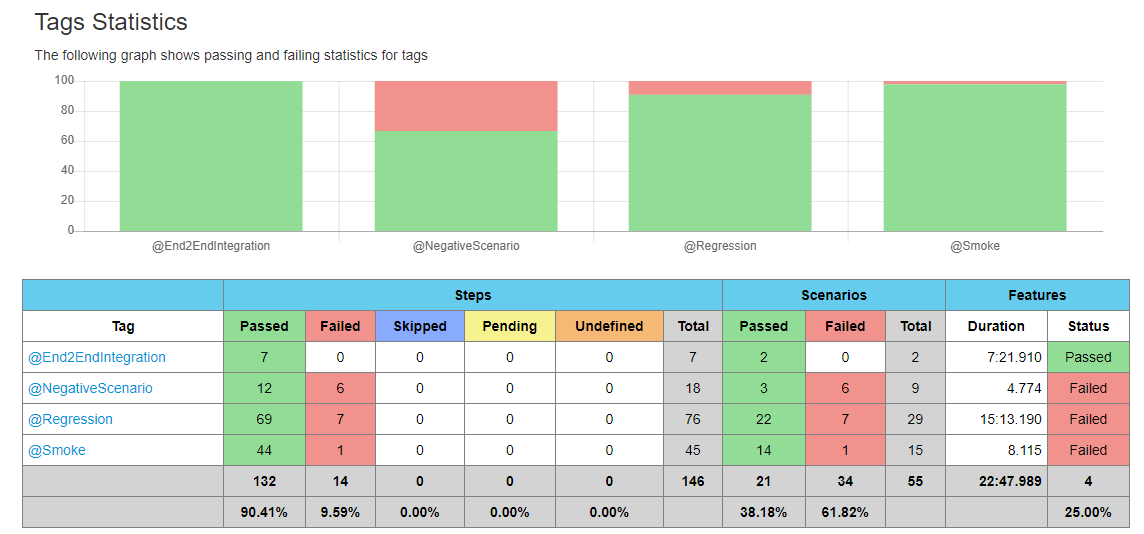
## Metrics

### No. of test cases planned vs executed

1. **No. of test cases passed/failed**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test cases**  **planned** | **Test cases**  **executed** | **TCs**  **Pass** | **TCs**  **Failed** |
| 29 | 29 | 22 | 7 |





1. **No of defects identified and their Status & Severity**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Critical** | **Major** | **Medium** | **Cosmetic** | **Total** |
| **Defects** | 3 | 4 | 0 | 1 | 8 |

|  |  |
| --- | --- |
| Defect ID | D01 |
| Defect Name | Accepting Incorrect Protocol for Manufacturer Service Endpoint |
| Defect Description | Service is returning valid response even after changing the protocol of endpoint from HTTP to HTTPS which is a big risk and can compromise the security of the service and also not matching as per the requirement shared. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/manufacturer>   1. Check the values returned in wkda object which represents the car Manufacturer name. 2. Now change the protocol to HTTPS for the endpoint   <https://api-aws-eu-qa-1.auto1-test.com/v1/car-types/manufacturer>   1. Check the values returned in wkda object which represents the car Manufacturer name. |
| Severity | **Critical** |
| Expected | Service should only return successful response when endpoint has HTTP protocol |
| Actual | Service is returning successful response when endpoint has both HTTP & HTTPS protocol |

|  |  |
| --- | --- |
| Defect ID | D02 |
| Defect Name | Accepting Incorrect Protocol for Car Model Service Endpoint |
| Defect Description | Service is returning valid response even after changing the protocol of endpoint from HTTP to HTTPS which is a big risk and can compromise the security of the service and also not matching as per the requirement shared. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/main-types?manufacturer=487>   1. Check the values returned in wkda object which represents the car Model name. 2. Now change the protocol to HTTPS for the endpoint   <https://api-aws-eu-qa-1.auto1-test.com/v1/car-types/main-types?manufacturer=487>   1. Check the values returned in wkda object which represents the car Model name. |
| Severity | **Critical** |
| Expected | Service should only return successful response when endpoint has HTTP protocol |
| Actual | Service is returning successful response when endpoint has both HTTP & HTTPS protocol |

|  |  |
| --- | --- |
| Defect ID | D03 |
| Defect Name | Accepting Incorrect Protocol for Registration Year Service Endpoint |
| Defect Description | Service is returning valid response even after changing the protocol of endpoint from HTTP to HTTPS which is a big risk and can compromise the security of the service and also not matching as per the requirement shared. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/built-dates?manufacturer=107&main-type=Azure>   1. Check the values returned in wkda object which represents the car registration year. 2. Now change the protocol to HTTPS for the endpoint   <https://api-aws-eu-qa-1.auto1-test.com/v1/car-types/built-dates?manufacturer=107&main-type=Azure>   1. Check the values returned in wkda object which represents the car registration year. |
| Severity | **Critical** |
| Expected | Service should only return successful response when endpoint has HTTP protocol |
| Actual | Service is returning successful response when endpoint has both HTTP & HTTPS protocol |

|  |  |
| --- | --- |
| Defect ID | D04 |
| Defect Name | Accepting Incorrect/Additional query parameters for Manufacturer Service Endpoint |
| Defect Description | The Service Endpoint is accepting Incorrect/Additional query parameters and still giving the valid response. This is not a critical issue as end user will not be able to pass any additional parameters from UI front. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/manufacturer>   1. Check the values returned in wkda object which represents the car Manufacturer name. 2. Now include additional/invalid query parameters to the endpoint   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/manufacturer?&test=test>   1. Check the values returned in wkda object which represents the car Manufacturer name. |
| Severity | **Major** |
| Expected | Service should only return successful response for valid endpoint with no additional/invalid query parameters |
| Actual | Service is returning successful response for valid endpoint with additional/invalid query parameters |

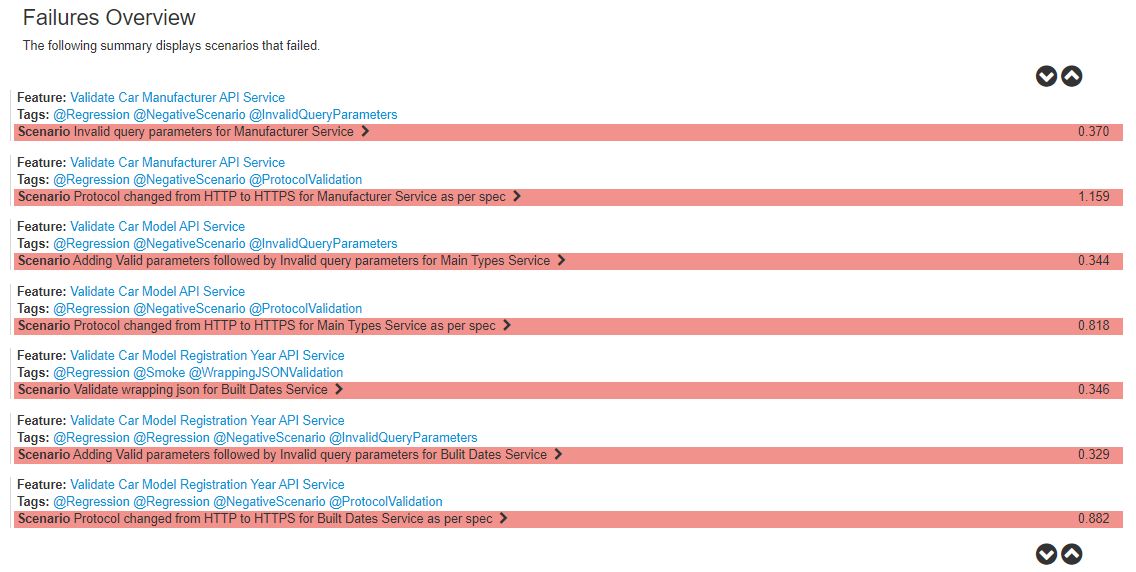
|  |  |
| --- | --- |
| Defect ID | D05 |
| Defect Name | Accepting Incorrect/Additional query parameters for Car Model Service Endpoint |
| Defect Description | The Service Endpoint is accepting Incorrect/Additional query parameters and still giving the valid response. This is not a critical issue as end user will not be able to pass any additional parameters from UI front. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/main-types?manufacturer=590>   1. Check the values returned in wkda object which represents the car model name. 2. Now include additional/invalid query parameters to the endpoint   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/main-types?manufacturer=590&test&test>   1. Check the values returned in wkda object which represents the car model name. |
| Severity | **Major** |
| Expected | Service should only return successful response for valid endpoint with no additional/invalid query parameters |
| Actual | Service is returning successful response for valid endpoint with additional/invalid query parameters |

|  |  |
| --- | --- |
| Defect ID | D06 |
| Defect Name | Accepting Incorrect/Additional query parameters for Registration Year Service Endpoint |
| Defect Description | The Service Endpoint is accepting Incorrect/Additional query parameters and still giving the valid response. This is not a critical issue as end user will not be able to pass any additional parameters from UI front. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/built-dates?manufacturer=190&main-type=LN>   1. Check the values returned in wkda object which represents the car registration year. 2. Now include additional/invalid query parameters to the endpoint   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/built-dates?manufacturer=190&main-type=LN&test=test>   1. Check the values returned in wkda object which represents the car registration year. |
| Severity | **Major** |
| Expected | Service should only return successful response for valid endpoint with no additional/invalid query parameters |
| Actual | Service is returning successful response for valid endpoint with additional/invalid query parameters |

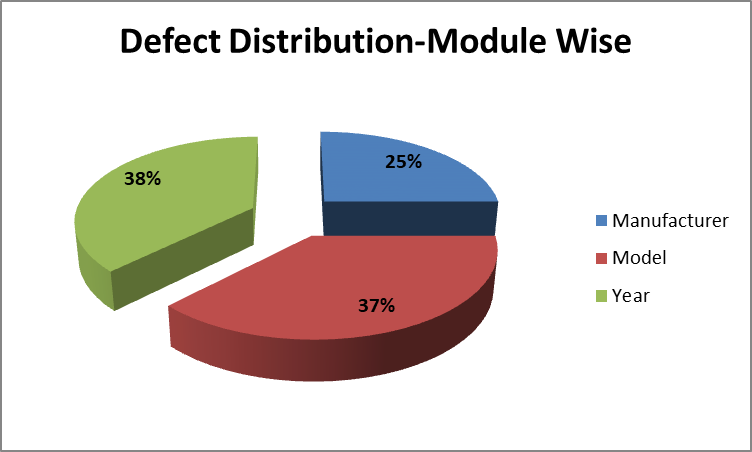
|  |  |
| --- | --- |
| Defect ID | D07 |
| Defect Name | Incorrect JSON wrapper for Registration Year Service Endpoint |
| Defect Description | The Service is not returning the expected JSON wrapper shared in requirement document where it should return keys like page, pageSize, totalPageCount & Json object wkda with its key-value pairs but we are not getting page, pageSize, totalPageCount in response. This is not critical as on now because per requirement pagination or sorting are out of scope where these values play a key role. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/built-dates?manufacturer=190&main-type=LN>   1. Check the the response is only returning wkda object and its key-value pairs. |
| Severity | **Major** |
| Expected | Service should return the exact wrapper shared in business requirement which includes page, pageSize, totalPageCount & Json object wkda with its key-value pairs in the response. |
| Actual | Service is only returing & Json object wkda with its key-value pairs in the response. |

**Note – Cosmetic defect was caught during part of exploratory testing and is not included in Automation report.**

|  |  |
| --- | --- |
| Defect ID | D08 |
| Defect Name | Car Model names spelled incorrectly |
| Defect Description | Car model name not correct for few cars. |
| Steps to reproduce | 1. Open below URL in postman.   <http://api-aws-eu-qa-1.auto1-test.com/v1/car-types/main-types?manufacturer=487>   1. Check the values returned in wkda object which represents the car model name. |
| Severity | **Low(Cosmetic)** |
| Expected | Car model name for Lexus brand should be "CT-Series", "GS-Series", "IS-Series", "LC-Series", etc |
| Actual | Car model names for Lexus brand is showing "CT-Serie", "GS-Serie", "IS-Serie", "LC-Serie", etc. |



1. **Defects distribution – module wise**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Manufacturer** | **Model** | **Year** | **Total** |
| **Critical** | 1 | 1 | 1 | **3** |
| **Major** | 1 | 1 | 2 | **4** |
| **Medium** | 0 | 0 | 0 | **0** |
| **Cosmetic** | 0 | 1 | 0 | **1** |
| **Total-->** | **2** | **3** | **3** | **8** |

## Types of testing performed

* + 1. **Smoke Testing**
* This testing will ensure that major functionalities are working fine.
* This testing covers getting the right success code for the services along and also ensures that we are getting atleast one key value pair for drop downs so that customer don’t get blank options which blocks further actions on application resulting in bad user experience.
* We are also ensuring that we get right structure of wrapping JSON for dependent functionalities like pagination or sorting works fine.

**Scenarios Covered:**

1. **GET operation Success Status for Manufacturer Service and no blank wkda object (to ensure customer not getting blank dropdown on UI)**
2. **Validate wrapping json for Manufacturer Service**
3. **GET operation Success Status for Main Types Service and no blank wkda object (to ensure customer not getting blank dropdown on UI)**
4. **Validate wrapping json for Main Types Service**
5. **Random Validation of Car models and Year of Manufacturer using Excel Input Sheet**
6. **GET operation Success Status for Built Dates Service and no blank wkda object (to ensure customer not getting blank dropdown on UI)**
7. **Validate wrapping json for Built Dates Service**
   * 1. **System Integration Testing**
        + The Integration Testing on the Application under test is to ensure that interdependent modules play well together as per the requirements.
        + Critical Business scenarios were tested to make sure important functionalities in the application works as intended End-To-End without any errors.

**Scenarios Covered:**

1. **GET operation for interdependency validation of Entire Flow**
2. **Random Validation of Car models and Year of Manufacturer using Excel Input Sheet**
   * 1. **Regression Testing**
        + Regression testing with below scenarios will cover both positive and negative scenarios to ensure the broader coverage for the rest services.
        + In the negative scenarios we are ensuring endpoints should not accept invalid query parameters and HTTP protocol & method works as per requirement which is important aspect of security testing.
        + This testing will also ensure that user see unique values in dropdown which will result in better user experience.

**Scenarios Covered:**

1. **All Scenarios mentioned under Smoke & Integration testing were covered.**
2. **POST operation Failure Status for Manufacturer Service**
3. **Validate list of Unique Manufacturer**
4. **Invalid query parameters for Manufacturer Service**
5. **Protocol changed from HTTP to HTTPS for Manufacturer Service as per spec**
6. **POST operation Failure Status for Manufacturer Service**
7. **Adding Valid parameters followed by Invalid query parameters for Main Types Service**
8. **Protocol changed from HTTP to HTTPS for Main Types Service as per spec**
9. **Validate list of Unique Car models for different manufacturer**
10. **POST operation Failure Status for Manufacturer Service**
11. **Adding Valid parameters followed by Invalid query parameters for Built Dates Service**
12. **Protocol changed from HTTP to HTTPS for Built Dates Service as per spec**
13. **Validate list of Unique Registration Year for all Car Models of Manufacture**

## Recommendations

* Noticed mis-spelled names for some car model which might be confusing for the end customers. Hence business analyst should properly document the naming convention on Manufacturer & models and same needs to be shared with testing team so that the service response can be validated against those values shared in requirement document.
* Subsequent API requests should use a token based authentication like JWT/oAuth instead of the key/value pair as query parameter to enhance security of services.
* Content Type can be set explicit to application/json. It doesn’t have any severe impact on the GET services as we are not passing any request body to it but as we are dealing with JSON response so from the standardization and code consistency perspective for RESTful services, it’s a good practice to follow.

## Conclusion

This automated testing framework developed by utilizing the open source applications and has potential to be enhanced continuously for future requirements. This framework will increase team’s test speed and efficiency, improve test accuracy, and will reduce test maintenance costs as well as lower risks. Few essential key reasons to utilize this framework are mentioned below:

* Improved test efficiency
* Lower maintenance costs
* Minimal manual intervention
* Maximum test coverage