Trello API Project Report

Introduction:

Trello is a project management tool that enables agile teams to implement the KANBAN approach. Teams create a board on which they create lists that indicate a task's lifecycle thus enabling teams to monitor their progress throughout a sprint. However as with the case of any software tool, Trello needs to be tested to validate that the tool's core functionalities are running as expected. The following will be discussed in this report:

- The approach taken to test Trello's core functionalities
- The test scenarios created to test and validate those functionalities
- The results attained after the test execution

Please bear in mind that we are going to test the following functionalities only through APIs as per requirements:

- Board Management
- Card Management
- List Management

Test Approach:

In order to test the tool's APIs, we used java's RESTAssured framework. We could have used the SHAFT-Engine's API testing framework however we chose the native RESTAssured framework in order to gain a thorough understanding of how RESTAssured handles an API's requests and responses.

In order to gain insight of Trello's API's, we went through the official Atlassian Developer's REST API documentation that provides all APIs used to communicate with Trello's server. From there, we identified the following core APIs:

- GET boards/{id}
- PUT boards/{id}
- DELETE boards/{id}
- POST boards
- GET lists/{id}
- PUT lists/{id}
- POST lists
- GET cards/{id}
- PUT cards/{id}
- POST cards

As a result, we identified that each element in Trello has its unique ID which is why it is used as path parameter for the GET, PUT, and DELETE methods. All other element attributes are inserted as query parameters.

Even though the API documentation provides extensive information about the API requests used to perform various operations on Trello, the documentation doesn't provide sufficient information about the responses to these operation requests. This is when Postman comes in handy and luckily though the documentation provides a Postman collection to be imported directly into Postman.

However in order for the server to authorize the requests, it requires an API key and Token which require going through this tutorial. After we got the API key and Token, we started taking note of the response bodies returned to the aforementioned requests so that we can create POJO models for each of the board, list, and card elements to facilitate extraction of and assertion on the response data.

Furthermore, we implemented the DSL design approach to decrease repetitive code, increase test case writing speed and efficiency, and provide a better understanding to anyone who is not a seasoned tester.

In addition, we used the Extent Reports reporting tool to log the test case events and generate an HTML file that provides an insight to the successful and failed test scenario and the test cases of each test scenario.

Since extensive testing is impossible and it is not feasible to test each and every element attribute, we only picked the critical attributes without which an element will be pretty useless.

The CLASS hierarchy used in our DSL design is as follows:

1. Board (represents a single board)
   Has attributes:
   - ID
   - Name

   1.1. Label (represents a single label)
   Has attributes:
   - Name
   - Color

   1.2. Background (represents the board's background)
   Has attributes:
   - ID
   - Background (can be a color or an ID to a picture)
2. API Manager (acts as the base test class which handles the report manager initialization, configuration data extraction, and test closure)
3. Boards Manager (handles the creation of board, list and card objects)
4. Report Manage (handles the logging test assertion and test progress and outputs the test report HTML file)

The board attributes are located in separate JSON files located in the resources folder.

The configuration file contains the base URL, base path, and API Key and Token. The file is critical as it provides a safe means to store the API key and token rather than using them directly in the test scenario files.

Finally, it is worth noting that all operations check on the status code to make it easier to debug each step in case of unexpected errors.

Test Scenarios:

The following test scenarios have been made to test the core operations on each of the board, list, and card elements:

1. Board (Create Retrieve Update Delete) CRUD:

Step	Test Data	Expected Response

Create a board	Name = RESTAssured Board 1
	Background = orange
		Status Code = 200
		Returns the board's attributes along with its ID

Change the board's green label name	Name = Done	Status Code = 200 Returns the board's attributes with the new label name
Change the board's red label name	Name = Blocked	Status Code = 200 Returns the board's attributes with the new label name
Change the board's background to blue	-	Status Code = 200 Returns the board's attributes along with the new background
Delete the board	-	Status Code = 200 Note: we checked this from the website only

2. List (Create Retrieve Update Delete) CRUD:

Step	Test Data	Expected Response

Create a board	Name = RESTAssured Board 1
	Background = orange
		Status Code = 200
		Returns the board's attributes along with its ID

Create a list	Name = To-Test	Status Code = 200 Returns the list's attributes along with its ID
Change the list's name	Name = Testing	Status Code = 200 Returns the list's attributes along with its new name
Get the list's board ID	-	Status Code = 200 Returns the list's attributes along with its board's ID
Archive the list	-	Status Code = 200 Returns the list's attributes along with its 'closed' attribute changed to true
Delete the board	-	Status Code = 200 Note: we checked this from the website only

3. Card (Create Retrieve Update Delete) CRUD:

Step	Test Data	Expected Response

Create a board	Name = RESTAssured Board 1
	Background = orange
		Status Code = 200
		Returns the board's attributes along with its ID

Create a list	Name = To-Test	Status Code = 200 Returns the list's attributes along with its ID
Create a card	Name = task1	Status Code = 200 Returns the card's attributes along with its ID
Change the card's name	Name = task2	Status Code = 200 Returns the card's attributes along with its new name
Get the card's board ID	-	Status Code = 200 Returns the card's attributes along with its board's ID
Archive the list	-	Status Code = 200 Returns the card's attributes along with its 'closed' attribute changed to true
Delete the board	-	Status Code = 200

4. Send a request with Invalid API token:

Step	Test Data	Expected Response

Create a board	Name = RESTAssured Board 1
	Background = orange
	API token = null
		Status Code = 401
		Returns "invalid key"

5. Send a request without a required query parameter:

Step	Test Data	Expected Response
Create a board	Background = orange API token = null	Status Code = 401 Returns "invalid value for name"

6. Move cards between lists:

Step	Test Data	Expected Response

Create a board	Name = RESTAssured Board 1
	Background = orange
		Status Code = 200
		Returns the board's attributes along with its ID

Create the first list	Name = To-Test	Status Code = 200 Returns the list's attributes along with its ID
Create the second list	Name = Tested	Status Code = 200 Returns the list's attributes along with its ID
Create a card in the first list	Name = task1	Status Code = 200 Returns the card's attributes along with its ID
Move card to the second list	idList = Second list ID	Status Code = 200 Returns the card's attributes along with the ID of the second list
Delete the board	-	Status Code = 200

7. Move lists between boards:

Step	Test Data	Expected Response

Create the first board	Name = RESTAssured Board 1
	Background = orange
		Status Code = 200
		Returns the board's attributes along with its ID

Create the second board	Name = RESTAssured Board 2
	Background = red
		Status Code = 200
		Returns the board's attributes along with its ID

Create a list in the first board	Name = To-Test	Status Code = 200 Returns the list's attributes along with its ID
Move the list to the second board	idBoard = Second board ID	Status Code = 200 Returns the list's attributes along with the ID of the second board
Delete the board	-	Status Code = 200

Test results:

As expected, all test scenarios mentioned before passed successfully. This due to the fact that Trello is already in the release stage and all the aforementioned functionalities have been tested before. However the aforementioned test scenarios are done only for the sake of the project and to show that we have gained extensive and valuable knowledge and experience working with RESTAssured, Postman, and the DSL design approach.