**Rest Assured Assignment Documentation**

**Introduction**

This documentation outlines the requirements and guidelines for completing the Rest Assured assignment. The assignment involves creating automated tests for a RESTful API using Rest Assured, a Java library designed specifically for API testing.

**Objective**

The main objective of this assignment is to demonstrate my understanding and proficiency in:

* Setting up automated API tests using Rest Assured.
* Writing test cases to validate various aspects of a RESTful API.
* Executing tests and analysing test results.

**Set-up**

To begin with the assignment, we need to have JDK and MAVEN. Once done, you can open the project in an IDE of your choice or run directly from the Command line.  
User can change values from the **config.properties** under the **src/test/resources** folder. The user can change the **token** and **key** values generated from **Trello** to run on their Trello workspace or can utilize my Credentials for the test. Other properties include names for creating cards lists and boards and the base URL for the test.

To run the project with the Command Line using maven command in the project directory.

>>> mvn clean test

The Command will clean the target folder creating a working field for the surefire plugin to test build and test the created test cases.  
OR  
If on IDE Click on the testng.xml file, right-click on the file and run it as a TestNg Suit.

**Project Description**

**POM FILE**

The pom file contains dependencies and their details which is read by Maven to install the required dependencies and plugins on build.

**Build Configuration:**

* **Maven Compiler Plugin**: Configures the Java compiler plugin to use Java 8 as the source and target compatibility.
* **Maven Surefire Plugin**: Configures the Surefire plugin to run testNG tests defined in the **testng.xml** file.

**Dependencies:**

* **TestNG**: Testing framework dependency.
* **Selenium Java**: Dependency for Selenium WebDriver automation.
* **Rest Assured**: Library for testing RESTful APIs.
* **JSON Schema Validator**: Dependency for JSON schema validation.
* **JSON Path**: Dependency for JSON parsing.
* **XML Path**: Dependency for XML parsing.
* **JSON**: Library for JSON manipulation.
* **Extent Reports**: Dependency for generating test reports.
* **Log4j Core**: Dependency for logging.

**TestNG.XML FILE**

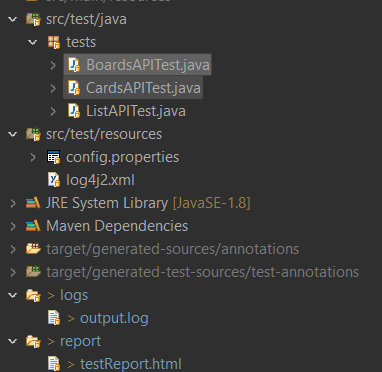
The **testng.xml** file serves as a configuration file for organizing and executing TestNG test cases within a test suite.

1. **Test Suite Declaration**:
   * The file begins with the declaration of a TestNG test suite named "TestSuite".
2. **Test Listeners**:
   * Within the suite, a listener class named **utils.TestNGListener** is specified. This listener class is responsible for capturing and handling test events during execution.
3. **Test Definitions**:
   * The suite comprises multiple test definitions, each representing a distinct testing scenario.
   * The first test is named "Individual Tests" and is configured to run with a thread count of 5. It includes tests from the class **tests.BoardsAPITest**.
   * Subsequent tests are named "Create flow" and "Delete flow". Each test is associated with specific test groups ("CardFlowCreate" and "CardFlowDelete" respectively), and includes tests from multiple classes (**BoardsAPITest**, **ListAPITest**, and **CardsAPITest**).
4. **Test Grouping**:
   * Tests are grouped based on their functionalities to facilitate modular and organized testing.
   * Grouping allows for selective execution of tests based on specific criteria, enhancing flexibility and efficiency in test management.
   * The TestGroups used in the Assignment are "CardFlowCreate", " CardFlowDelete" and “FailTest”.

**Project Structure**

**project structure 1
**

* The project uses a Page Object Model. Under the src/main/java folder we first see API package, This package contains all used api functionalities in their respective classes. Each API Class has its own static Logger and Endpoint that it will cater to.
* Followed by the api package we have the Base page which is extended by all the other API classes and tests for some basic methods and TestNG Suit and Event Listener Implementations. The BasePage has methods that need to be run before and after every suit, Test or method.
* The last Utils package contains classes that have various uses and each class focused on a single purpose.
* The File Reader Focuses on Retrieving Data from the config.properties file and make the data accessible to other classes throughout the project.
* The Helper class contains methods that are used multiple times like CRUD Operations to keep the code more readable and avoiding repetitiveness.
* PostPayload stores Maps that need to be added to the payload for post operations.
* Resources stores all the resource path that need to be covered while testing.
* Finally, the TestNGListener is the implementation of the TestNGListener interface to implement custom method body for logs.

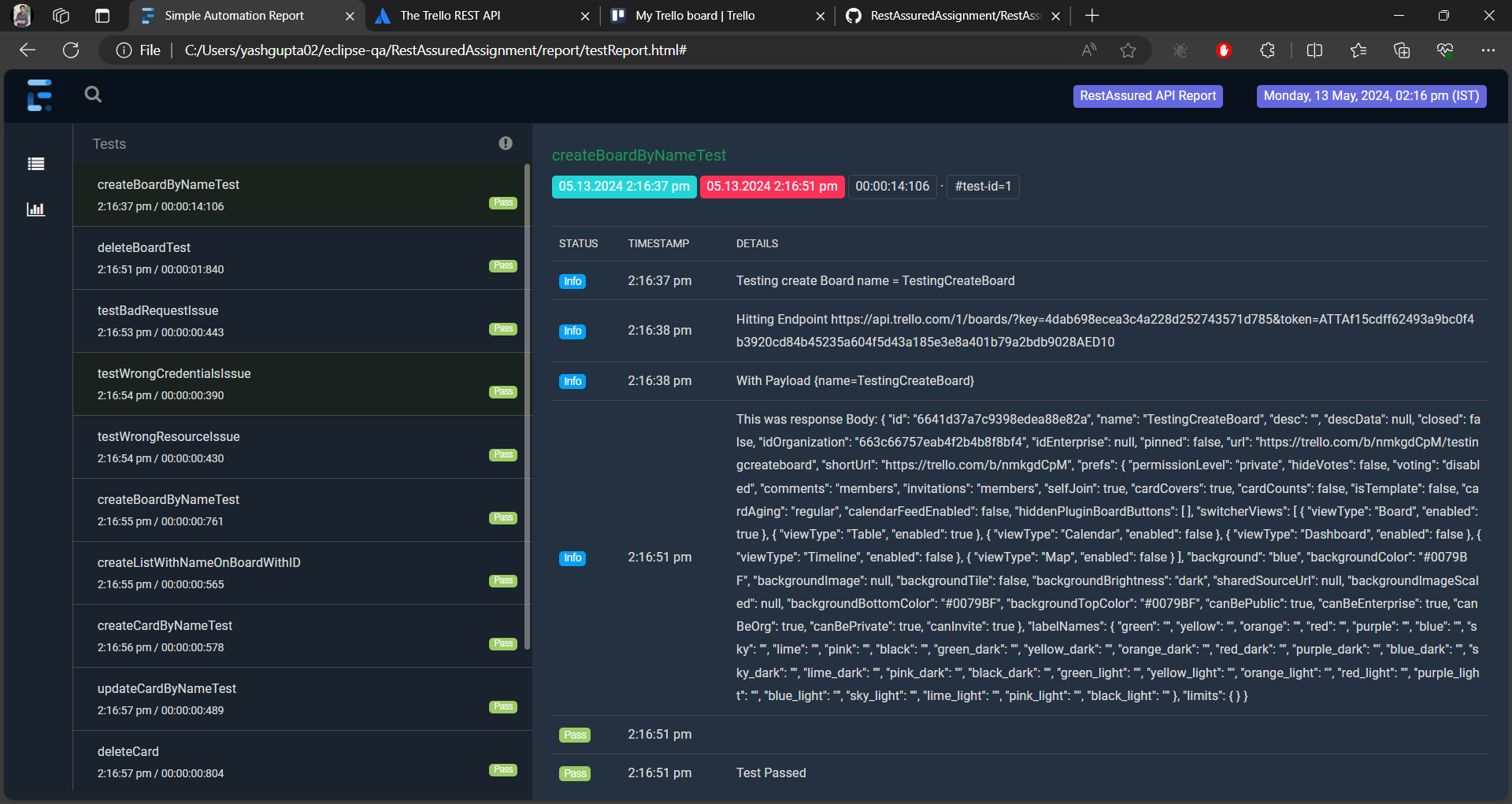


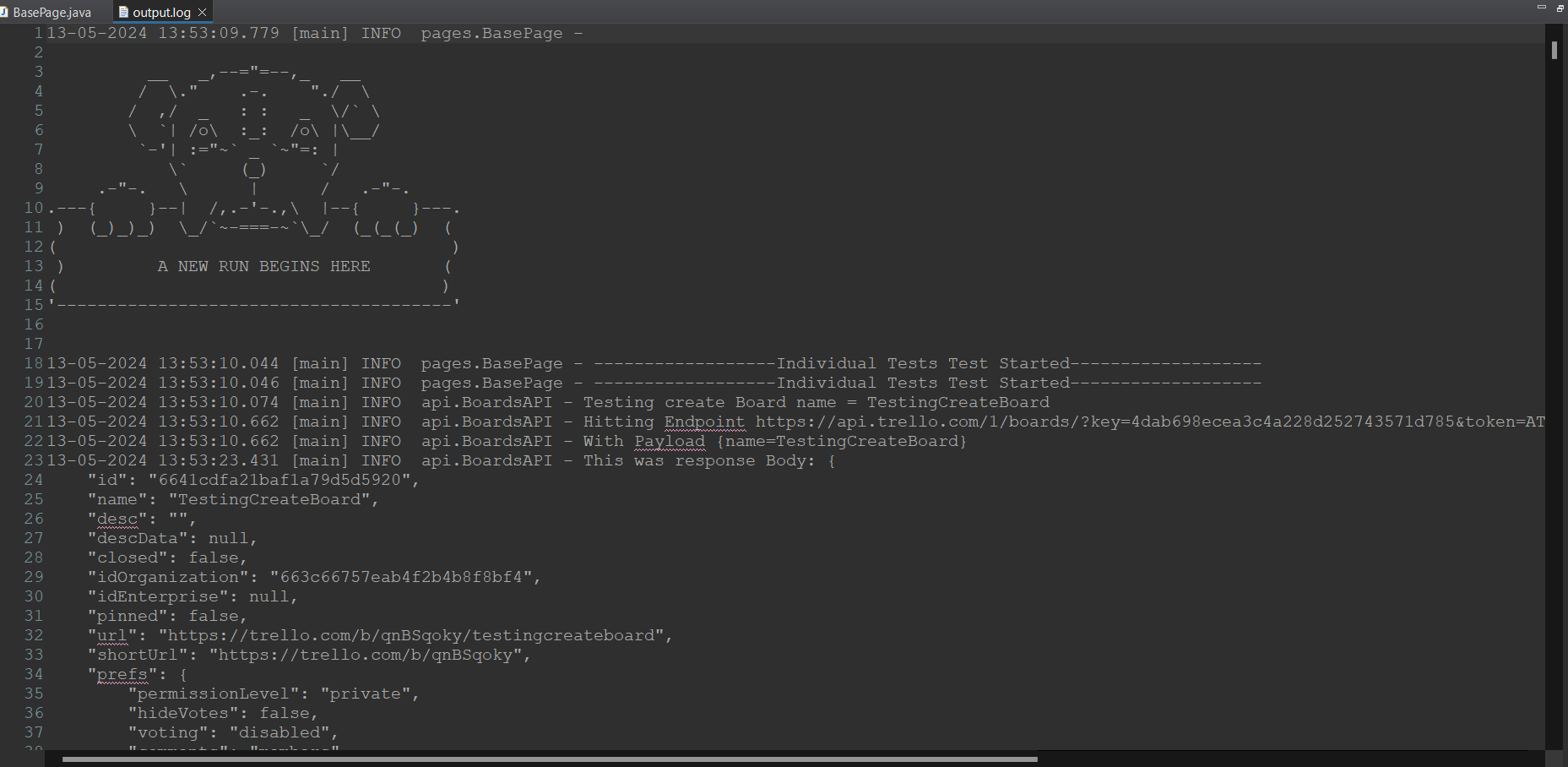
Under the src/test/java

* We have all our tests in the tests package, Each TestNG test class is responsible for tests regarding their api endpoint.
* The Resource packge under src/test stores the config properties and the log4j.xml   
  Config.properties stores data that is used in the project and log4j stores configuration regarding the Log4j Logger used in the assignment.
* The logs and report folders store the logs and extentReports generated after the test runs.

Extent Report:

Extent report Stats

Log File:   


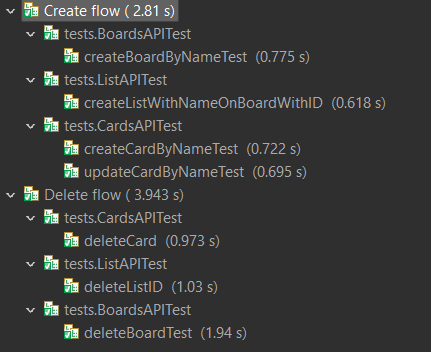
**Results**

**All Test pass:**

**All test pass
**

**Test run flow :**

**Create Board -> Create List -> Create Card-> Update Card Name -> Delete Card-> Archive list-> Delete board**

****

**Individual tests :**

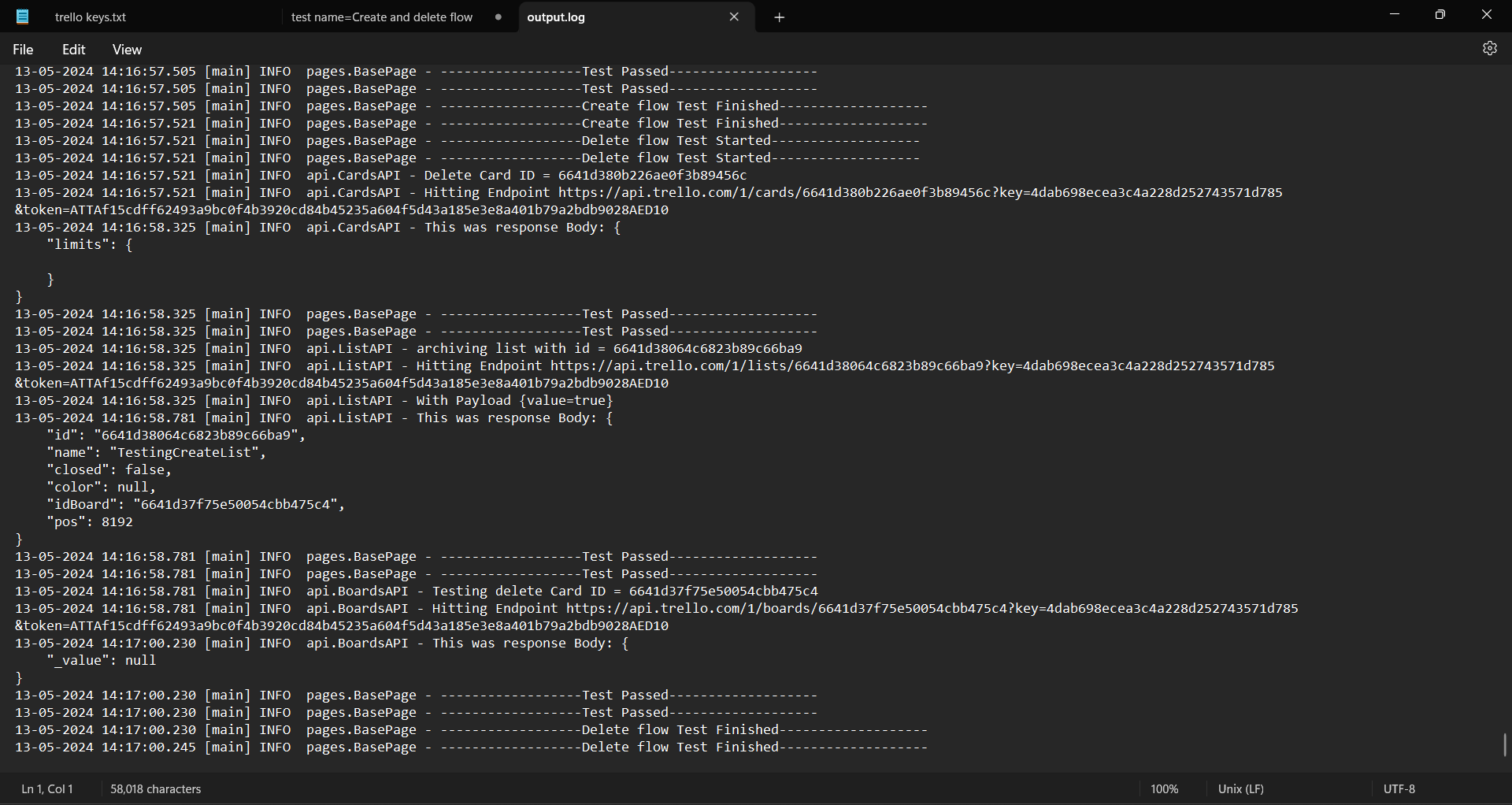
**A screenshot of a computer

Description automatically generated**

**Console output:**

**Test run success in CLI
**

**Final logs:**

****

**Complete project can be found at** [**Github - RestAssuredAssignment**](https://github.com/YashGupta1510/RestAssuredAssignment/tree/master) **along with Reports and Logs.**