# API Automation Framework Documentation

## 1. Overview

This repository is an API Automation Framework built using Java, Maven, and RestAssured. It is designed to validate APIs through schema validation (JSON/XML), execute automated functional tests, and integrate with CI/CD pipelines. The framework also supports logging, report generation, and TestRail integration for test management.

## 2. pom.xml

The pom.xml file is the heart of the Maven project. It defines dependencies, plugins, and build configurations needed to run the automation framework. Below are the key dependencies:

- RestAssured: For API testing.  
- Jackson Databind & Dataformat: For JSON/XML serialization & deserialization.  
- JSON Schema Validator: For validating JSON API responses against schema.  
- Apache POI: For working with Excel files (test data input/output).  
- Freemarker: For templating.  
- Vavr: For functional programming utilities.  
- Log4j2: For logging.  
- JUnit/TestNG (through test runners): For test execution.

Additionally, the pom.xml configures build plugins like Maven Compiler Plugin, sets the source/target Java version, and includes repositories for artifact resolution.

## 3. Project Structure

### src/test/java

- api/: Contains API request builder classes and methods to interact with endpoints.  
- cons/: Stores constants (e.g., endpoints, tokens, keys).  
- core/: Contains base classes, cache managers, retry mechanisms, cookie/session handlers, and core reusable utilities for the framework.  
- dataparser/: Includes classes for parsing XML/JSON test data into Java objects.  
- flow.rps/: Defines test flows and business scenarios combining multiple API calls.  
- model/: POJOs representing request and response models.  
- test/: Test definitions and step definitions (if Cucumber is used).  
- utils/: Helper utility classes (e.g., schema validator, date utils, file utils).  
- TestRunner.java: Main test runner class that triggers execution via TestNG or JUnit.

### src/test/resources

- autodata/: Contains JSON or Python-based test data inputs.  
- CISuite/: XML suite files defining groups of tests for CI/CD execution.  
- jsonSchema/: JSON schema files for validating API responses.  
- scripts/: Utility scripts for AWS operations, test setup, or DB loading.  
- templates/: Request/response templates used by tests.  
- TestCaseSqlQuery/: SQL queries for DB validations.  
- XMLSchema/: XML schema definitions (XSD files) for validating XML responses.  
- apis.json: Contains API definitions.  
- assembly.xml: Maven build configuration for packaging.  
- config.xml: General framework configuration (environment-specific).  
- datapointsCompared.json: Stores expected vs. actual comparison data.  
- log4j2.properties: Logging configuration.  
- TestRailSectionMapping.properties: Mapping file for TestRail integration.

## 4. Execution Flow

1. Test execution starts with TestRunner.java.  
2. TestRunner picks the test suite (e.g., from CISuite XML).  
3. API calls are executed via RestAssured (request built from api/ and model/).  
4. Test data comes from autodata/ or TestCaseSqlQuery/.  
5. Responses are validated against JSON schemas (jsonSchema/) or XML schemas (XMLSchema/).  
6. Results and logs are generated (log4j2.properties, test-output/).  
7. Reports are integrated into CI/CD pipelines and can be mapped to TestRail.

## 5. CI/CD Integration

The framework integrates with CI/CD pipelines via buildspec.yml (AWS CodeBuild) or Jenkins. It ensures that API regression tests are triggered automatically in pipelines and results are published for stakeholders.