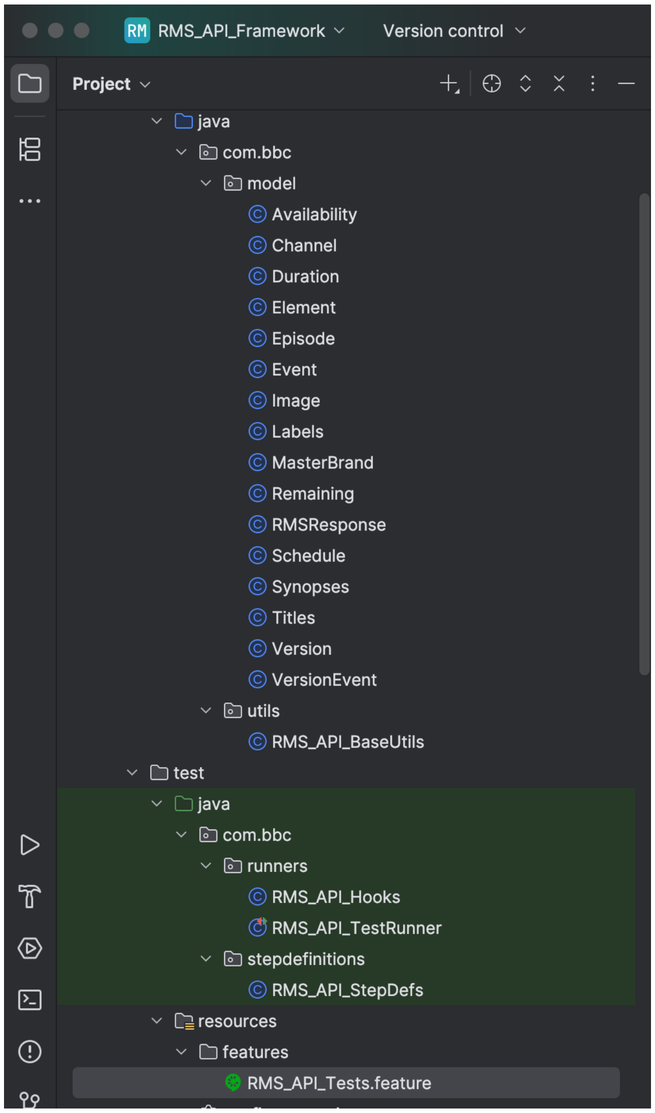
BBC Software Engineer in Test Take Home Test Solutions

Task 1: Automation Task Solutions

Step 1: Introduction

This documentation provides an overview of the BBC RMS API Automation Framework, a Java-based project designed to automate testing of the RMS (Radio and Music Services) API endpoint https://testapi.io/api/RMSTest/ibltest. The framework utilizes Cucumber with Gherkin for test scenario definition, RestAssured for API interactions, and Maven for build management. It addresses seven automated test scenarios and three manual test cases as outlined in the take-home test requirements, ensuring robust validation of API responses and metadata.

Step 2: Project Step-by-Step



```
java
  com.bbc
    model
      Availability
      Channel
      Duration
      Element
      Episode
      Event
      Image
      Labels
      MasterBrand
      Remaining
      RMSResponse
      Schedule
      Synopses
      Titles
      Version
      VersionEvent
    utils
      RMS_API_BaseUtils
test
  java
    com.bbc
      runners
        RMS_API_Hooks
        RMS_API_TestRunner
      stepdefinitions
        RMS_API_StepDefs
  resources
    features
      RMS_API_Tests.feature
```

1. **Environment Setup**

Set up the environment by installing Java 11 and Maven, making sure they work with the project's configuration.

2. **Project Initialization**

Started a new project using Maven with the group name as "org.example", the project name as "RMS_API_Framework", and the version as "1.0-SNAPSHOT".

3. **Dependency Addition**

Added necessary tools like Cucumber, JUnit, RestAssured, Jackson, Lombok and PrettyReports to the project configuration for testing and reporting.

4. **Model Class Definition**

Created simple data objects in a package called "com.bbc.model" to match the structure of the API response, such as RMSResponse and Element.

5. **Utility Implementation**

Built a utility class in a package called "com.bbc.utils" named RMS_API_BaseUtils to read the configuration file and build API URLs.

6. **Runner Configuration**

Set up helper classes in a package called "com.bbc.runners" named RMS_API_Hooks and RMS_API_TestRunner to prepare and run the tests.

7. **Feature File Creation**

Created test scenario files in a folder called "resources/features" named RMS_API_Tests.feature to outline the tests.

8. **Step Definition Writing**

Wrote step definitions in a package called "com.bbc.stepdefinitions" named RMS_API_StepDefs to manage the test steps and check results based on the test scenarios.

9. **Property Configuration**

Added settings like base API URL, endpoint, and response time limit to the configuration file.

10. **Build and Test Execution**

Compiled and ran the tests using Maven, producing reports to review the results.

Step 3: Folder Description

- **java/com.bbc/model**: Contains Java classes (e.g., Availability, Channel, Element, RMSResponse) representing API response structures.
- **java/com.bbc.utils**: RMS_API_BaseUtils for configuration loading and URL construction.
- **java/com.bbc.runners**: Includes RMS_API_Hooks for setup/teardown and RMS_API_TestRunner for Cucumber test execution.
- **java/com.bbc.stepdefinitions**: Contains RMS_API_StepDefs with logic for Gherkin step implementations.
- **src/test/resources**: Stores features (RMS_API_Tests.feature) and config.properties for test scenarios and configuration.
- **Pom**.xml

Library	**Artifact ID**	**Version**	**Usage / Purpose**
JUnit	junit	4.13.2	Unit testing framework; used to run and assert test outcomes.
Cucumber Java	cucumber-java	7.16.1	Provides Cucumber step definition support with @Given, @When, @Then.
Cucumber JUnit	cucumber-junit	7.16.1	Enables running Cucumber scenarios via the JUnit test runner.
Gherkin	gherkin	33.1.0	Parses .feature files written in Gherkin syntax.
Rest Assured	rest-assured	5.4.0	Makes HTTP calls to APIs and simplifies request/response validations.
Jackson Databind	jackson-databind	2.15.2	Converts JSON to Java POJOs and vice versa (used with ObjectMapper).
SLF4J Simple	slf4j-simple	2.0.9	Lightweight logging implementation used by RestAssured and other libraries.
Lombok	lombok	1.18.30	Auto-generates boilerplate code like getters/setters using annotations.
PrettyReports	reporting-plugin	5.0.0	Generates rich HTML Cucumber reports from JSON outputs.

How It Works

1. Feature File (Human-readable scenarios)

This is where we write test cases in plain English using **Given, When, Then.**

Example:

```
Scenario: Verify GET request returns 200 status and response time
  Given the RMS API is up and running
  When I send a GET request to the RMS API
  Then the response status code is 200
```

2. Step Definition File (RMS_API_StepDefs.java)

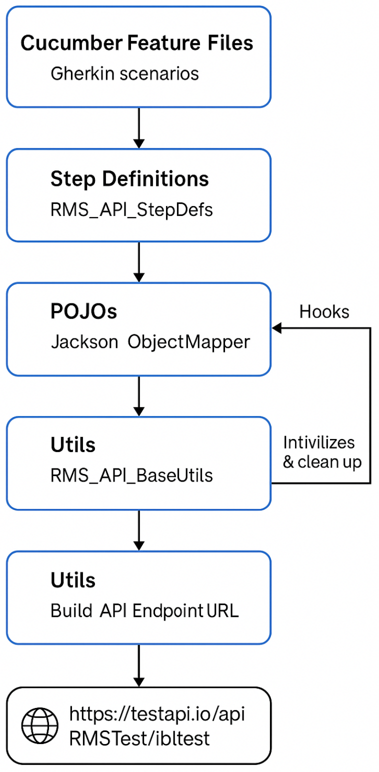
Each step above is mapped to a Java method like:

```
@Given("the RMS API is up and running")
public void theRmsApiIsUpAndRunning() {
    // Setup already handled in Hooks
}

@When("I send a GET request to the RMS API")
public void sendGetRequestToRmsApi() {
    apiResponse = given().when().get(apiEndpoint);
}

@Then("the response status code is {int}")
public void verifyStatusCode(int expectedCode) {
    assertEquals(expectedCode, apiResponse.getStatusCode());
}
```

---

3. Supporting Files

- **Hooks (RMS_API_Hooks.java)**: Runs setup and teardown logic before/after each scenario.
- **Utils (RMS_API_BaseUtils.java)**: Reads from config.properties, constructs URLs.
- **POJOs (RMSResponse, Element)**: Automatically map API JSON into Java objects using Jackson.
- **Config (config.properties)**: Central config for base URL, endpoints, timeouts.



---

4. Execution Flow

1. **Gherkin step** → triggers → **matching annotated method in StepDefs**
2. **Step method** → builds URL using BaseUtils → sends request using **RestAssured**
3. **Response** → validated using assertions → reports generated

---

Notes

- Keeps test cases readable for non-technical stakeholders (Followed Java Standards and OOPS)
- Separates **test logic (Java)** from **test scenarios (Gherkin)**.
- Allows easy maintenance and reuse of step definitions.

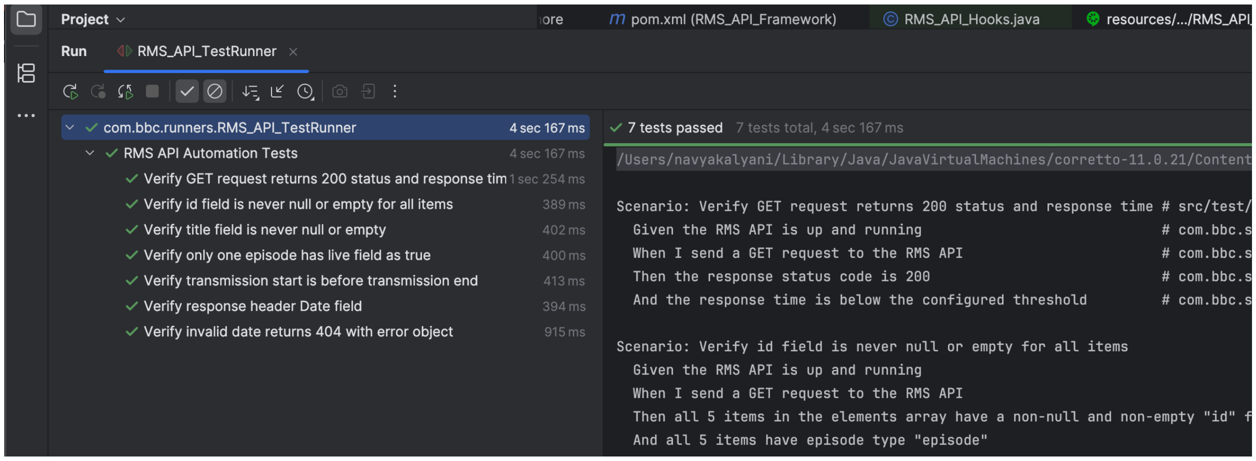
Step 4: Execution

Maven Test

1. **Navigate to Project Directory**: Open a terminal and cd to the project root where pom.xml resides.
2. **Clean Build**: Run mvn clean to remove old build artifacts.
3. **Compile and Test**: Execute mvn test to compile the project and run all tests.
4. **Verify Output**: Check the console for test results and ensure reports are generated in target/cucumber/ and target/cucumber-html-reports/.

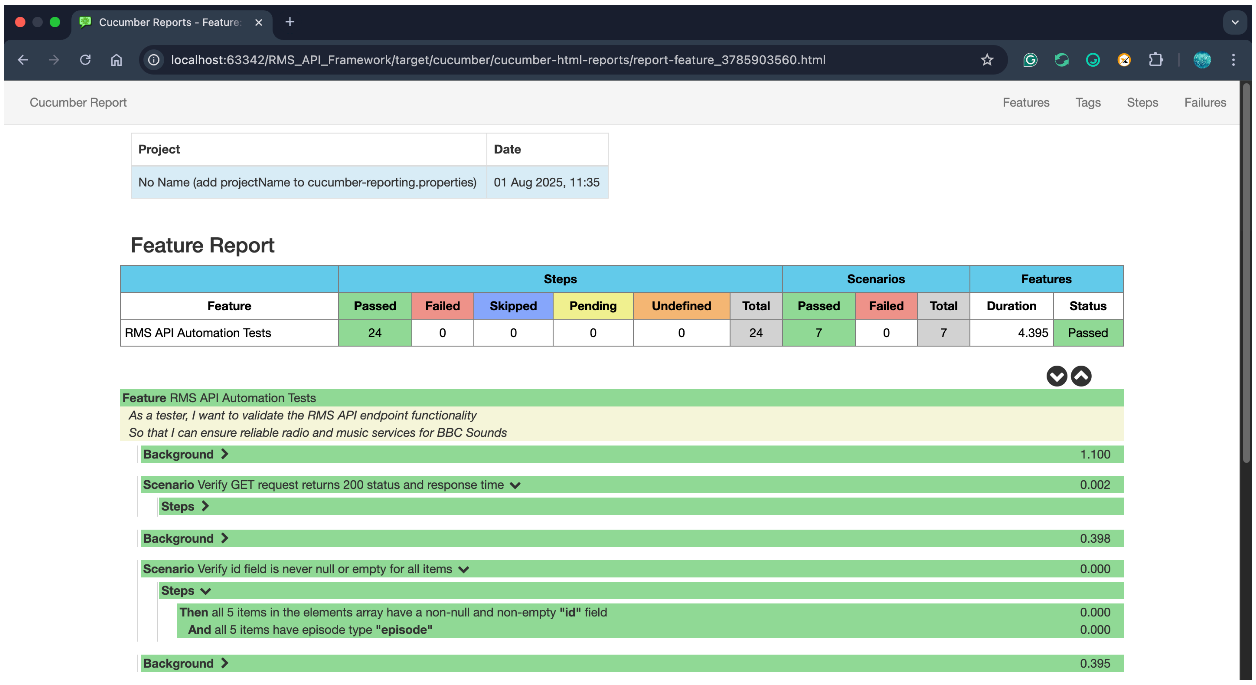
Cucumber Report

1. **Locate Reports**: After mvn test, find JSON reports in target/cucumber/cucumber.json and HTML reports in target/cucumber-html-reports.
2. **View HTML Report**: Open the index.html file in target/cucumber-html-reports in a web browser to review detailed test results, including pass/fail status and scenario coverage.
3. **Analyze Data**: Use the report to assess test execution time, failures, and overall API validation success.



```
com.bbc.runners.RMS_API_TestRunner                                   4 sec 167 ms
  RMS API Automation Tests                                           4 sec 167 ms
    Verify GET request returns 200 status and response tim           1 sec 254 ms
    Verify id field is never null or empty for all items                   389 ms
    Verify title field is never null or empty                              402 ms
    Verify only one episode has live field as true                         400 ms
    Verify transmission start is before transmission end                   413 ms
    Verify response header Date field                                      394 ms
    Verify invalid date returns 404 with error object                      915 ms
```

```
7 tests passed  7 tests total, 4 sec 167 ms
/Users/navyakalyani/Library/Java/JavaVirtualMachines/corretto-11.0.21/Content

Scenario: Verify GET request returns 200 status and response time # src/test/
  Given the RMS API is up and running                             # com.bbc.s
  When I send a GET request to the RMS API                        # com.bbc.s
  Then the response status code is 200                            # com.bbc.s
  And the response time is below the configured threshold         # com.bbc.s

Scenario: Verify id field is never null or empty for all items
  Given the RMS API is up and running
  When I send a GET request to the RMS API
  Then all 5 items in the elements array have a non-null and non-empty "id" f
  And all 5 items have episode type "episode"
```


Cucumber Reports - Feature
localhost:63342/RMS_API_Framework/target/cucumber/cucumber-html-reports/report-feature_3785903560.html
Cucumber Report
Features
Tags
Steps
Failures

Project	Date
No Name (add projectName to cucumber-reporting.properties)	01 Aug 2025, 11:35

Feature Report

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
RMS API Automation Tests	24	0	0	0	0	24	7	0	7	4.395	Passed

Feature RMS API Automation Tests
As a tester, I want to validate the RMS API endpoint functionality
So that I can ensure reliable radio and music services for BBC Sounds

Background 1.100

Scenario Verify GET request returns 200 status and response time 0.002
Steps

Background 0.398

Scenario Verify id field is never null or empty for all items 0.000
Steps
Then all 5 items in the elements array have a non-null and non-empty **"id"** field 0.000
And all 5 items have episode type **"episode"** 0.000

Background 0.395

Task 2: Manual Testing Scenarios

1. **Category Label Display**: Show category label only when episode categories exist.

Feature: Display category label only when categories are present

As a user,
I want to ensure that the category label is shown only if there are categories listed in each episode's metadata,
So that viewers see category information only when it's relevant.

Scenario: Category label visibility depends on categories list
Given I have a list of episodes provided by the RMS API
When I examine the metadata of each episode
Then if the "categories" list contains one or more items
The "labels" field should have a "category" value that is not empty
And if the "categories" list is empty or missing
The "labels.category" field should be either absent or present with an empty value

----

2. **Episode Availability Period:** Validate episode availability dates cover the current viewing period.

Feature: Verify episode version availability dates in RMS API

As a user,
I want to confirm that every episode version contains correct availability start and end dates,
So that I know viewers will only see currently available content.

Scenario: Check availability period for episode versions
Given I have access to the RMS API endpoint
When I send a GET request to the RMS API
Then the response should include at least one episode version with an "availability" object
And the "start" date in "availability" should be before today's date
And the "end" date in "availability" should be after today's date

----

3. **'Childrens' Field Validation : Verify every episode correctly includes a true/false 'childrens' field.**

Feature: Validate presence and type of 'childrens' field in all episodes

As a user,
I want to ensure every episode correctly identifies if it is children's content,
So that content classification remains accurate.

Scenario: Confirm 'childrens' field in every episode
Given I have access to the RMS API endpoint
When I send a GET request to the RMS API
Then the response should contain exactly 5 episodes
And each episode should include a "childrens" field
And the value of "childrens" should be either true or false