QA Automation Assesment

TABLE OF CONTENTS

Structure of Project - QA-AUTOMATION-ASSESSMENT

```
├── postman-collection/              # Task 1: Postman API Tests
│   ├── Restful_BA_Env.postman_environment.json   # Postman environment file
│   ├── Restful_Booker_API.postman_collection.json # Postman collection file
│   ├── Restful_Booker_API.postman_test_run.json   # Test run result
├── api-automation/                  # Task 2: API Automation Tests in .NET
│   ├── RestfulBookerTests/
│   │   ├── ApiClients/              # Contains API client wrapper class
│   │   ├── BookingTests.cs          # Test cases for API testing
│   │   ├── RestfulBookerTests.csproj  # Project file for .NET tests
│   │   ├── RestfulBookerTests.sln          # Solution file
├── ui-automation/                   # Task 3: UI Automation Tests in Playwright
│   ├── medirect-frontend-tests/
│   │   ├── node_modules/            # Node.js dependencies
│   │   ├── pages/             # Page Object Model (POM) files
│   │   ├── playwright-report/       # Playwright test reports
│   │   ├── screenshots/             # Screenshots captured during tests
│   │   ├── show-report/             # Report viewer
│   │   ├── test-report/             # Detailed test report (USE THIS)
│   │   ├── test-results/            # Raw test results
│   │   ├── tests/             # UI test scripts
│   │   ├── after-click.png          # UI capture after an interaction
│   │   ├── package-lock.json        # npm dependency lock file
│   │   ├── package.json             # npm dependencies file
│   │   ├── playwright.config.ts     # Playwright test configuration
│   │   ├── tsconfig.json            # TypeScript configuration
```

Restful Booker API – Postman Collection

Overview

This repository contains Postman collections and environment configurations to test the Restful Booker API. The collection is arranged to run CRUD requests in a logical sequence, supporting data reuse throughout the requests (Test Chained).

Files Included

- **Restful_Booker_API.postman_collection.json**
  Main Postman collection.
- **Restful_BA_Env.postman_environment.json**
  Environment file containing variables (like baseUrl, authToken, bookingId etc).
- **Restful_Booker_API.postman_test_run.json**
  JSON file with exported test run results from the Collection Runner.

How to Use the Postman Collection

1. Import the Collection and Environment

1. Download **Restful_Booker_API.postman_collection.json** and **Restful_BA_Env.postman_environment.json**.
2. Open Postman.
3. Click **Import** at the top left.
4. Select both downloaded files to import.

2. Configure the Environment

1. Go to **Environments** in Postman.
2. Select **Restful-BA_Env** as your active environment.
3. Ensure the variables (e.g., baseUrl, authToken, bookingId) are set correctly.

3. Run the Collection

1. Open the **Collection Runner** in Postman.

2. Select **Restful_Booker_API**.
3. Choose the **Restful-BA_Env** environment.
4. Click **Run Collection** and observe the test results.

Test Coverage

The collection covers the following CRUD API operations and utility endpoints:

- **Create**
    - **CreateToken**: Generates an authentication token.
    - **CreateBooking**: Creates a new booking.
- **Read**
    - **GetBookingIds**: Fetches all booking IDs.
    - **GetBooking**: Retrieves a specific booking by {bookingId}.
- **Update**
    - **UpdateBooking**: Updates an existing booking for {bookingId}.
    - **PartialUpdateBooking**: Partially updates a booking for {bookingId}
- **Delete**
    - **DeleteBooking**: Removes the booking associated with {bookingId}
- **Utility**
    - **HealthCheck**: Verifies if the API is running.

Assertions in Tests

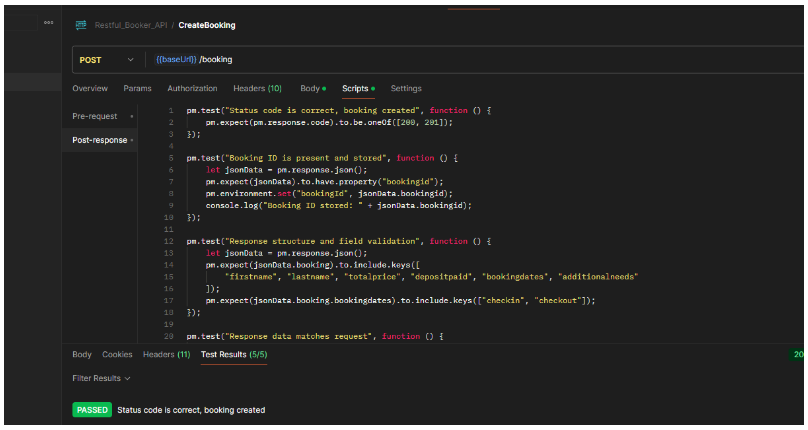
- **Status Codes** (e.g., 200, 201, 404).
- **Response Structure** (ensuring necessary fields exist).
- **Data Validation** (checking values match expected data).
- **CRUD Functionality** (confirming correct API behavior).

Note:

- Postman **Tests** tab contains post-response scripts for assertions.
- **Pre-request Scripts** manage environment variables and randomization.
- **Body** tab in Postman specifies the JSON payload for each request.

Test Run Screenshot

- Most test assertions were written in post-response tab of Scripts



Restful_Booker_API / CreateBooking

POST {{baseUrl}} /booking

Overview Params Authorization Headers (10) Body Scripts Settings

Pre-request
Post-response

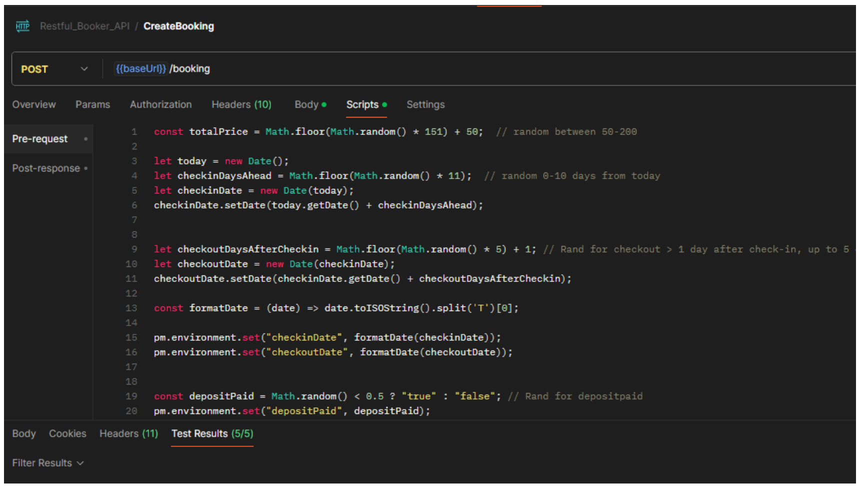
```javascript
pm.test("Status code is correct, booking created", function () {
    pm.expect(pm.response.code).to.be.oneOf([200, 201]);
});

pm.test("Booking ID is present and stored", function () {
    let jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("bookingid");
    pm.environment.set("bookingId", jsonData.bookingid);
    console.log("Booking ID stored: " + jsonData.bookingid);
});

pm.test("Response structure and field validation", function () {
    let jsonData = pm.response.json();
    pm.expect(jsonData.booking).to.include.keys([
        "firstname", "lastname", "totalprice", "depositpaid", "bookingdates", "additionalneeds"
    ]);
    pm.expect(jsonData.booking.bookingdates).to.include.keys(["checkin", "checkout"]);
});

pm.test("Response data matches request", function () {
```

Body Cookies Headers (11) Test Results (5/5)

Filter Results

PASSED Status code is correct, booking created

- Pre request was used for randomizing and updating the environment variables



Restful_Booker_API / CreateBooking

POST {{baseUrl}} /booking

Overview Params Authorization Headers (10) Body Scripts Settings

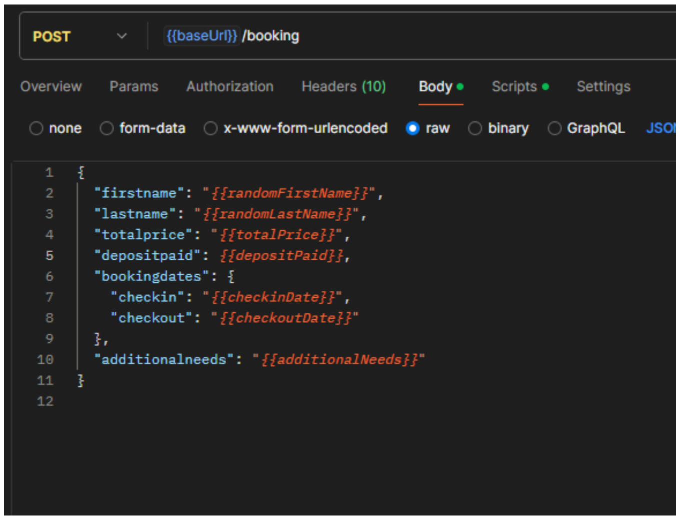
Pre-request
Post-response

```javascript
const totalPrice = Math.floor(Math.random() * 151) + 50;  // random between 50-200

let today = new Date();
let checkinDaysAhead = Math.floor(Math.random() * 11);  // random 0-10 days from today
let checkinDate = new Date(today);
checkinDate.setDate(today.getDate() + checkinDaysAhead);


let checkoutDaysAfterCheckin = Math.floor(Math.random() * 5) + 1; // Rand for checkout > 1 day after check-in, up to 5
let checkoutDate = new Date(checkinDate);
checkoutDate.setDate(checkinDate.getDate() + checkoutDaysAfterCheckin);

const formatDate = (date) => date.toISOString().split('T')[0];

pm.environment.set("checkinDate", formatDate(checkinDate));
pm.environment.set("checkoutDate", formatDate(checkoutDate));


const depositPaid = Math.random() < 0.5 ? "true" : "false"; // Rand for depositpaid
pm.environment.set("depositPaid", depositPaid);
```

Body Cookies Headers (11) Test Results (5/5)

Filter Results

Body was used to send request via json



POST {{baseUrl}} /booking

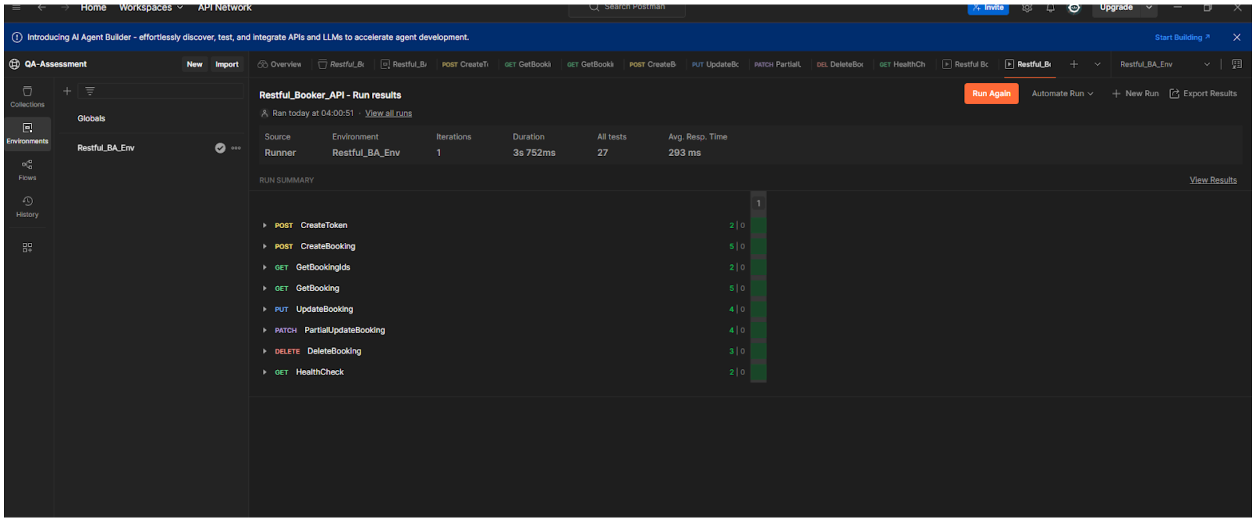
Overview Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSO

```json
{
  "firstname": "{{randomFirstName}}",
  "lastname": "{{randomLastName}}",
  "totalprice": "{{totalPrice}}",
  "depositpaid": {{depositPaid}},
  "bookingdates": {
    "checkin": "{{checkinDate}}",
    "checkout": "{{checkoutDate}}"
  },
  "additionalneeds": "{{additionalNeeds}}"
}
```

Test Run Screenshot

A test run was executed in Postman to validate the API behavior:



Restful_Booker_API - Run results

Ran today at 04:00:51 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Restful_BA_Env	1	3s 752ms	27	293 ms

RUN SUMMARY

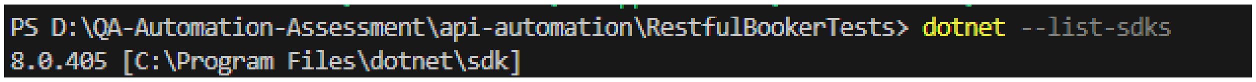
```
POST    CreateToken            2 | 0
POST    CreateBooking          5 | 0
GET     GetBookingIds          2 | 0
GET     GetBooking             5 | 0
PUT     UpdateBooking          4 | 0
PATCH   PartialUpdateBooking   4 | 0
DELETE  DeleteBooking          3 | 0
GET     HealthCheck            2 | 0
```

Restful Booker Integration Tests (C# / .NET)

In addition to the Postman collection, this repository has C# integration tests using **xUnit**, **FluentAssertions**, and **RestSharp**. These tests verify the same endpoints but from a programmatic perspective.

Prerequisites

- **.NET 8.0** (or newer).
  Confirm installation by running: dotnet --version



```
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> dotnet --list-sdks
8.0.405 [C:\Program Files\dotnet\sdk]
```

- A stable internet connection (since this is a public API).
- Ms VS code

Project Structure

RestfulBookerTests/

├── ApiClients/

│ └── RestfulBookerApi.cs

├── BookingTests.cs

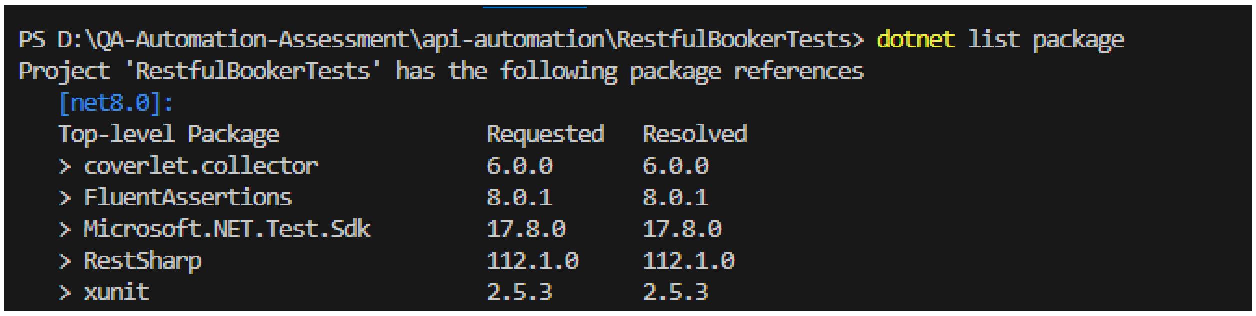
├── RestfulBookerTests.csproj

└── … other files

- **ApiClients/RestfulBookerApi.cs**
  Contains methods for all REST operations (create, get, update, partial update, delete, health check).
- **BookingTests.cs**
  Houses xUnit-based test classes that confirm each endpoint's functionality.

Installation & Setup

1. **Clone the repository**: git clone https://github.com/nazardeen/QA-Automation-Assessment.git
2. **Navigate into the project folder**: cd QA-Automation-Assessment/RestfulBookerTests
3. **Restore NuGet packages**: dot net restore (this will download all required dependencies, such as RestSharp, FluentAssertions, etc.):
4. Open in Visual Studio or VS Code to explore or modify the tests.



```
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> dotnet list package
Project 'RestfulBookerTests' has the following package references
   [net8.0]:
   Top-level Package               Requested   Resolved
   > coverlet.collector            6.0.0       6.0.0
   > FluentAssertions              8.0.1       8.0.1
   > Microsoft.NET.Test.Sdk        17.8.0      17.8.0
   > RestSharp                     112.1.0     112.1.0
   > xunit                         2.5.3       2.5.3
```

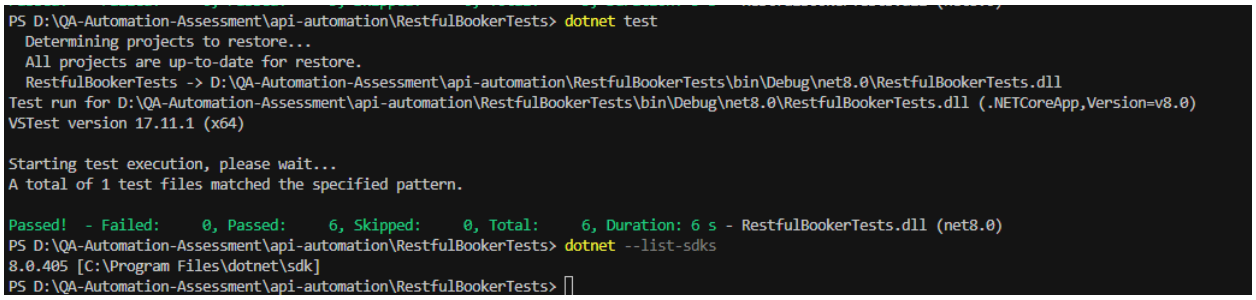
Running the Tests

Option 1: .VS code install C# dev kit

dotnet test

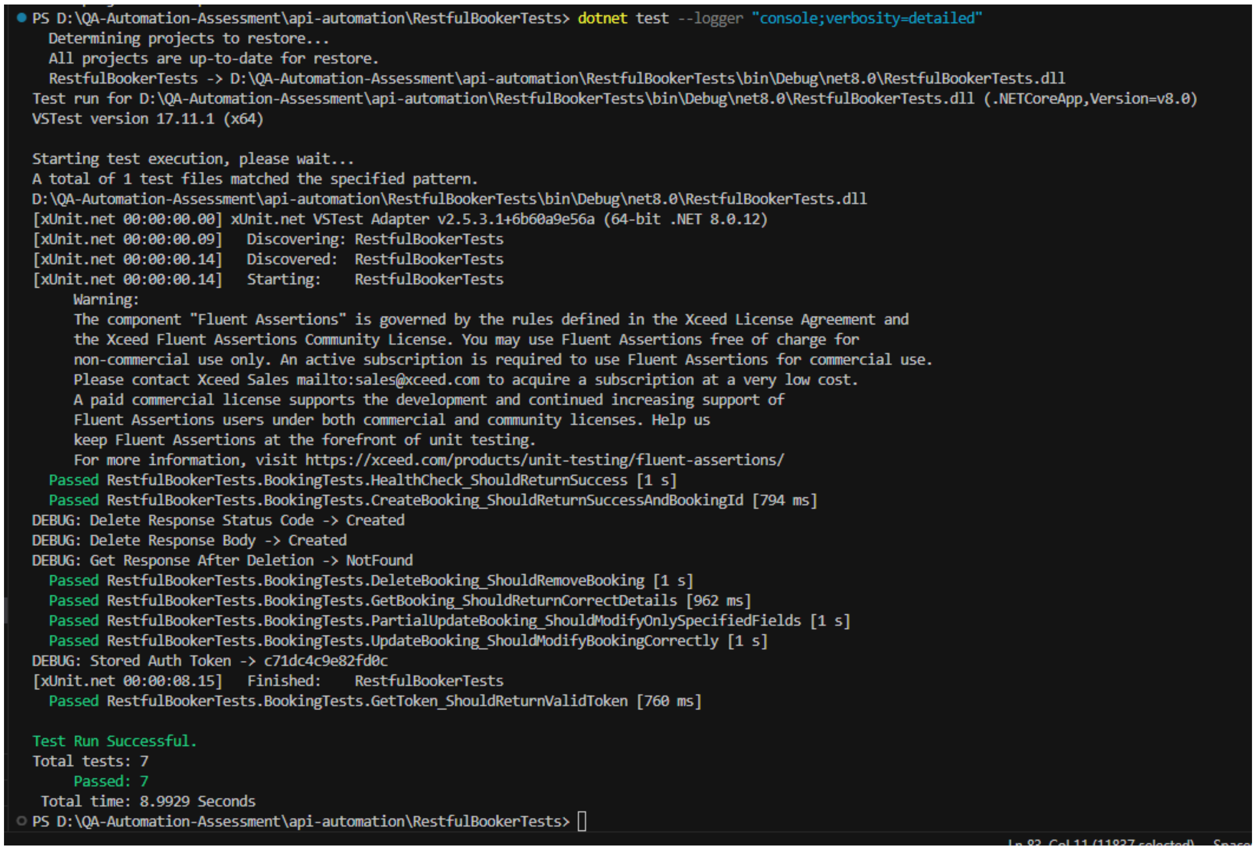
- Builds the project.
- Runs all xUnit tests.

- Prints a summary of the test results.



```
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> dotnet test
  Determining projects to restore...
  All projects are up-to-date for restore.
  RestfulBookerTests -> D:\QA-Automation-Assessment\api-automation\RestfulBookerTests\bin\Debug\net8.0\RestfulBookerTests.dll
Test run for D:\QA-Automation-Assessment\api-automation\RestfulBookerTests\bin\Debug\net8.0\RestfulBookerTests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.1 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:     6, Skipped:     0, Total:     6, Duration: 6 s - RestfulBookerTests.dll (net8.0)
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> dotnet --list-sdks
8.0.405 [C:\Program Files\dotnet\sdk]
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> 
```

Final Test Result



```
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> dotnet test --logger "console;verbosity=detailed"
  Determining projects to restore...
  All projects are up-to-date for restore.
  RestfulBookerTests -> D:\QA-Automation-Assessment\api-automation\RestfulBookerTests\bin\Debug\net8.0\RestfulBookerTests.dll
Test run for D:\QA-Automation-Assessment\api-automation\RestfulBookerTests\bin\Debug\net8.0\RestfulBookerTests.dll (.NETCoreApp,Version=v8.0)
VSTest version 17.11.1 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.
D:\QA-Automation-Assessment\api-automation\RestfulBookerTests\bin\Debug\net8.0\RestfulBookerTests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.5.3.1+6b60a9e56a (64-bit .NET 8.0.12)
[xUnit.net 00:00:00.09]   Discovering: RestfulBookerTests
[xUnit.net 00:00:00.14]   Discovered:  RestfulBookerTests
[xUnit.net 00:00:00.14]   Starting:    RestfulBookerTests
     Warning:
     The component "Fluent Assertions" is governed by the rules defined in the Xceed License Agreement and
     the Xceed Fluent Assertions Community License. You may use Fluent Assertions free of charge for
     non-commercial use only. An active subscription is required to use Fluent Assertions for commercial use.
     Please contact Xceed Sales mailto:sales@xceed.com to acquire a subscription at a very low cost.
     A paid commercial license supports the development and continued increasing support of
     Fluent Assertions users under both commercial and community licenses. Help us
     keep Fluent Assertions at the forefront of unit testing.
     For more information, visit https://xceed.com/products/unit-testing/fluent-assertions/
  Passed RestfulBookerTests.BookingTests.HealthCheck_ShouldReturnSuccess [1 s]
  Passed RestfulBookerTests.BookingTests.CreateBooking_ShouldReturnSuccessAndBookingId [794 ms]
DEBUG: Delete Response Status Code -> Created
DEBUG: Delete Response Body -> Created
DEBUG: Get Response After Deletion -> NotFound
  Passed RestfulBookerTests.BookingTests.DeleteBooking_ShouldRemoveBooking [1 s]
  Passed RestfulBookerTests.BookingTests.GetBooking_ShouldReturnCorrectDetails [962 ms]
  Passed RestfulBookerTests.BookingTests.PartialUpdateBooking_ShouldModifyOnlySpecifiedFields [1 s]
  Passed RestfulBookerTests.BookingTests.UpdateBooking_ShouldModifyBookingCorrectly [1 s]
DEBUG: Stored Auth Token -> c71dc4c9e82fd0c
[xUnit.net 00:00:08.15]   Finished:    RestfulBookerTests
  Passed RestfulBookerTests.BookingTests.GetToken_ShouldReturnValidToken [760 ms]

Test Run Successful.
Total tests: 7
     Passed: 7
 Total time: 8.9929 Seconds
PS D:\QA-Automation-Assessment\api-automation\RestfulBookerTests> 
```

Key Points

- **Test Framework**: xUnit
- **Assertion Library**: FluentAssertions
- **HTTP Client**: RestSharp
- **Endpoints Tested**:
  - HealthCheck (GET /ping)
  - CreateBooking (POST /booking)
  - GetBooking (GET /booking/{id})

    - UpdateBooking (PUT /booking/{id})
    - PartialUpdateBooking (PATCH /booking/{id})
    - DeleteBooking (DELETE /booking/{id})
    - GetValidToken
- **Authentication**:
  A token is generated through /auth for update, patch, and delete requests, then attached via a cookie in subsequent calls.

Troubleshooting

- **Unexpected Status Codes**
  The public API might have breaking changes. Verify the official documentation for updates.
- **Token Creation Failures**
  Check if the credentials (admin / password123) are valid. Refer to console logs if the token is not generated.
- **Timeout Errors**
  Network latency or API downtime can cause tests to hang. Retry later or confirm network availability.

UI Automation Tests for MeDirect Equities Search

This repository also includes UI tests for **MeDirect Equities Search**, implemented with **Playwright** (TypeScript). These tests validate:

- **Navigation**: Ensuring the equities, funds, ETFs, and bonds lists display.
- **Search**:
    - Looking up a popular equity (e.g., Apple).
    - Clicking “More Information” to confirm equity details.
- **Restricted Content**: Checking that some areas are off-limits for non-logged-in users.
- **Negative Tests**: Verifying no results for non-existent equities.
- **Partial Search**: Confirming expected results appear for partial queries.
- **Pagination**: Moving to page 2 and ensuring different results are shown.

Prerequisites

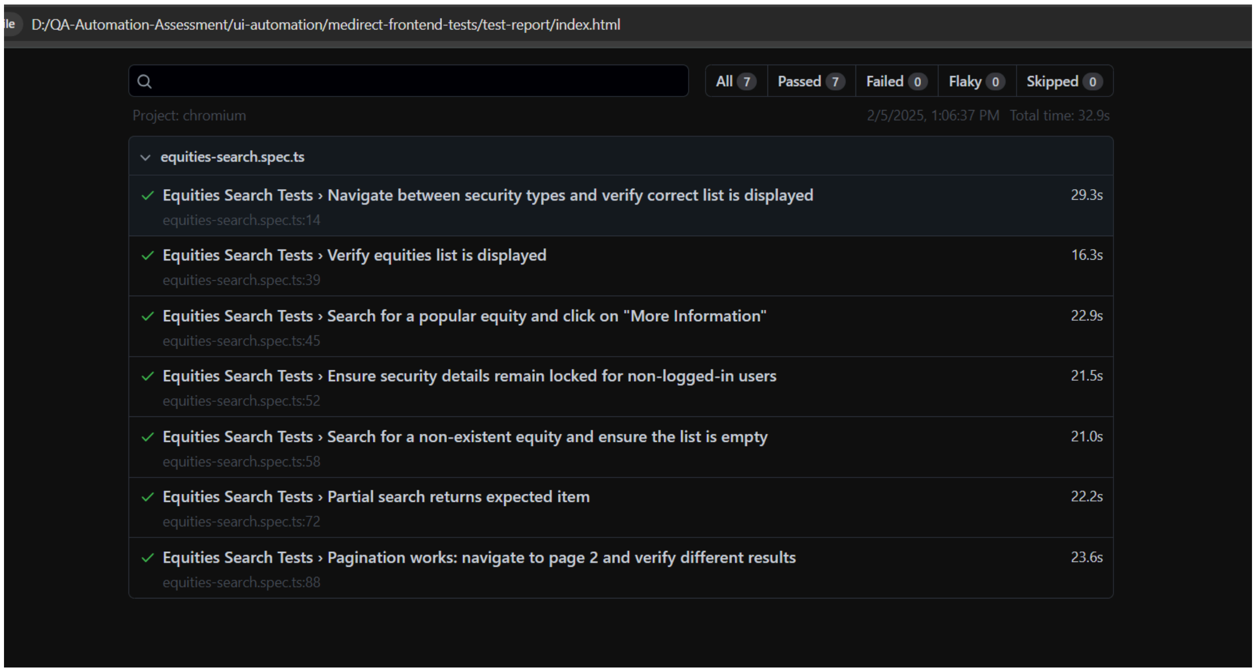
- **Node.js (v16 or higher)**
  Download from https://nodejs.org
- **npm** (bundled with Node.js).
- **Playwright** (installed as part of project dependencies 1.50.1).
- **TypeScript** (v5.7.3 or higher).

Setup Instructions

1. **Clone the Repository**: git clone https://github.com/nazardeen/QA-Automation-Assessment.git cd QA-Automation-Assessment
2. **Install Dependencies**: npm install
3. **Install Playwright Browsers**: npx playwright install

Running Tests

- **Run All Tests**: npx playwright test
- **Headed Mode**: npx playwright test --headed
- **Specific Test File**: npx playwright test tests/equities-search.spec.ts
- **Debug Mode**: npx playwright test --debug
- **View Test Reports**: npx playwright show-report


le D:/QA-Automation-Assessment/ui-automation/medirect-frontend-tests/test-report/index.html
All 7
Passed 7
Failed 0
Flaky 0
Skipped 0

```
Project: chromium                                                2/5/2025, 1:06:37 PM  Total time: 32.9s
equities-search.spec.ts
✓ Equities Search Tests › Navigate between security types and verify correct list is displayed   29.3s
  equities-search.spec.ts:14
✓ Equities Search Tests › Verify equities list is displayed                                      16.3s
  equities-search.spec.ts:39
✓ Equities Search Tests › Search for a popular equity and click on "More Information"            22.9s
  equities-search.spec.ts:45
✓ Equities Search Tests › Ensure security details remain locked for non-logged-in users          21.5s
  equities-search.spec.ts:52
✓ Equities Search Tests › Search for a non-existent equity and ensure the list is empty          21.0s
  equities-search.spec.ts:58
✓ Equities Search Tests › Partial search returns expected item                                   22.2s
  equities-search.spec.ts:72
✓ Equities Search Tests › Pagination works: navigate to page 2 and verify different results      23.6s
  equities-search.spec.ts:88
```

Project Structure

```
├── pages/
│ └── equities-search.page.ts (Page Object Model for the equities search)
├── tests/
│ └── equities-search.spec.ts (Test suite for searching & navigating equities)
├── playwright.config.ts (Playwright config)
├── package.json (Project dependencies)
```

Troubleshooting & Common Issues

1. **TimeoutError: page.waitForSelector(...) exceeded**
   - Increase wait times if the UI takes longer: await this.page.waitForSelector('.me-tbl tbody', { state: 'visible', timeout: 15000 });
2. **Tests Passing in Headed Mode, Failing in Headless**
   - Ensure all necessary waits are added: await page.waitForLoadState('networkidle');
3. **Locator or Element Not Found**
   - UI changes can break selectors. Use: npx playwright codegen https://www.medirect.com.mt/invest/equities/search
   - This helps generate or confirm accurate selectors.