
Software Development in Quality Assurance Skill Assessment

1 Assessment overview

One of the main purposes of a Software Developer in Test is to design and build tests required to make sure the products we develop are up to spec and of a high quality.

The objective of this assessment is to allow a prospective candidate to demonstrate basic development skills required to carry out the role.

2 Project delivery

Once you conclude all the tasks, please send us your artefacts. You can send us a link of the repository where your project is saved or send us the complete folder.

Include in the solution/s any documentation required, and the tests written for this solution.

3 Testing a RESTful API Service

The first part of the task focuses on consuming and testing out a RESTful API. Section 3.1 to 3.3 provide an overview of the API.

3.1 Overview of the API (RESTful) to be used for Testing

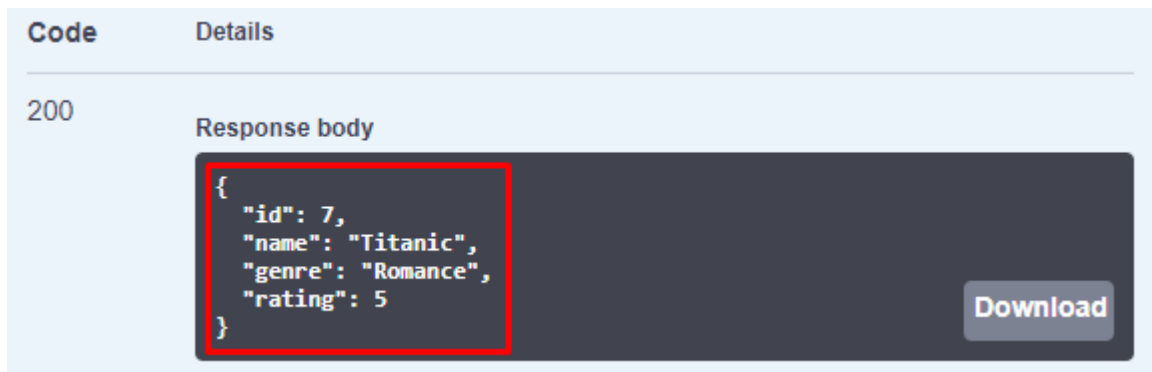
In order to carry out this test, you would need some basic knowledge on RESTful services. The Web App is making use of RESTful services.

A consumer that will consume the service, will have access to 20 sample movies. The data returned for each movie represents the following object:

Object returned	Description
ID	The ID of the movie in the Dataset
Name	The name of the movie
Genre	The genre of the movie (ex. Action, Comedy etc)
Rating	Rating of the movie

Of course, the data returned is just test data and does not reflect any actual data or movie information.

A sample response for movie with ID 7 is shown below.

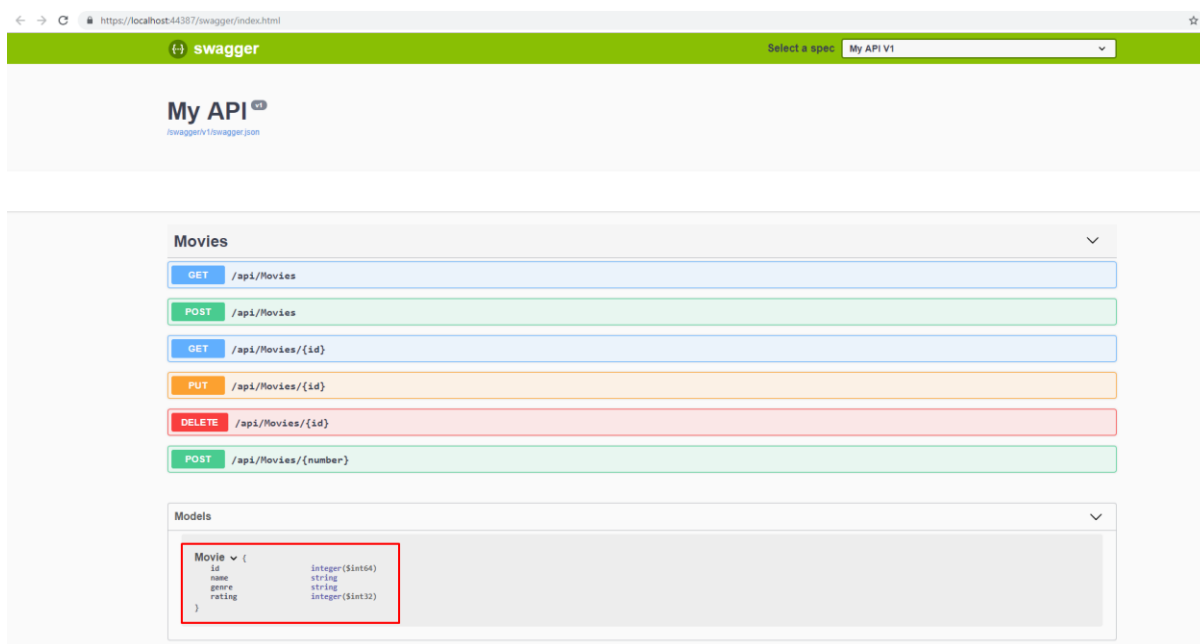


The screenshot displays an API response interface. It features a table with two columns: 'Code' and 'Details'. The 'Code' column shows '200'. The 'Details' column is titled 'Response body' and contains a JSON object: `{ "id": 7, "name": "Titanic", "genre": "Romance", "rating": 5 }`. A red rectangle highlights the JSON object. A 'Download' button is visible to the right of the JSON object.

Figure 1. Sample response when calling the API

The response in Figure 1 shows the response of obtained when getting the movie with ID 7.

You can view the model used for movie using swagger as shown in the screenshot below.



3.2 RESTful API description

The following table shows a list of verbs to be used to access the API available for you to consume when developing your tests.

Table 1. The table shows a description of the possible CRUD operations that can be performed on the REST service

API	Description
GET /api/movies	Retrieves all the available movies.
GET /api/movies/{id}	Retrieves the movie with the matching ID.
PUT /api/movies/{id}	Updates the details of the movie specified by the ID.
DELETE /api/movies/{id}	Deletes the movie specified by the ID.
POST /api/movies	Allows you to add a new movie.
POST /api/movies/{number}	Passing any number in the path, will make the service reload the data. This is only used to allow you to restore back the data when you test the DELETE verbose. You do not really need to test this.

As an example, if the service is hosted and has a domain name 'memoviestime' you would need to use the following URL to get all the movies:

<https://memoviestime/api/movies>

Or the following URL to get the movie with ID 7.

<https://memoviestime/api/movies/7>

3.3 Accessing/consuming the REST APIs

The link to access the environment is shown below:

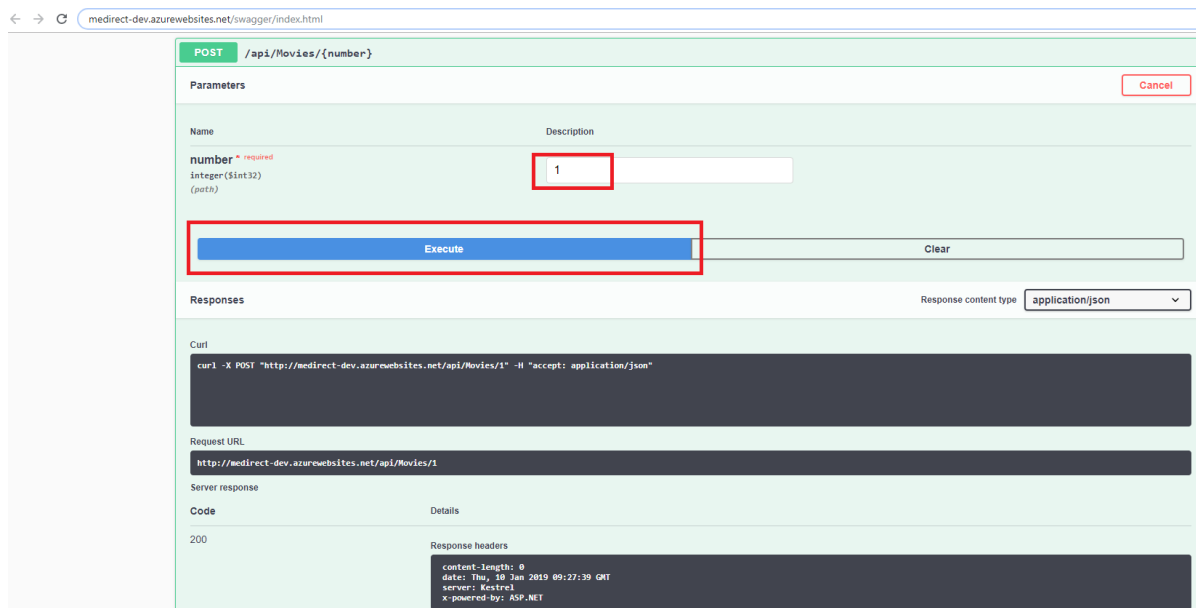
<http://medirect-dev.azurewebsites.net/swagger/index.html>

The following table shows a typical example of two HTTP GET requests.

URL	Description
http://medirect-dev.azurewebsites.net/api/Movies	Gets all the movies
http://medirect-dev.azurewebsites.net/api/Movies/4	Gets the movie with ID7

Table 2 Example of Typical HTTP GET requests.

Please note that you can refresh the movies list (used especially if you perform a delete) by calling the API using a POST request, as shown.



The screenshot shows the Swagger UI interface for the API. The top bar indicates the endpoint is `POST /api/Movies/{number}`. Below this, the 'Parameters' section shows a required parameter named 'number' of type 'integer(\$int32)' with a value of '1' entered in the input field. A red box highlights the 'Execute' button. The 'Responses' section shows the response content type is 'application/json'. The 'Curl' section displays the command: `curl -X POST "http://medirect-dev.azurewebsites.net/api/Movies/1" -H "accept: application/json"`. The 'Request URL' section shows the URL: `http://medirect-dev.azurewebsites.net/api/Movies/1`. The 'Server response' section shows the status code '200' and the response headers: `content-length: 0`, `date: Thu, 18 Jan 2019 09:27:39 GMT`, `server: Kestrel`, and `x-powered-by: ASP.NET`.

Figure 2. POST from Swagger to refresh the list of movies

Instead of swagger you can consume or access the API using [Postman](#).

3.4 Task Deliverables

In this task you are expected to develop end-to-end tests to test the Movies API as illustrated above. The following items need to be outlined:

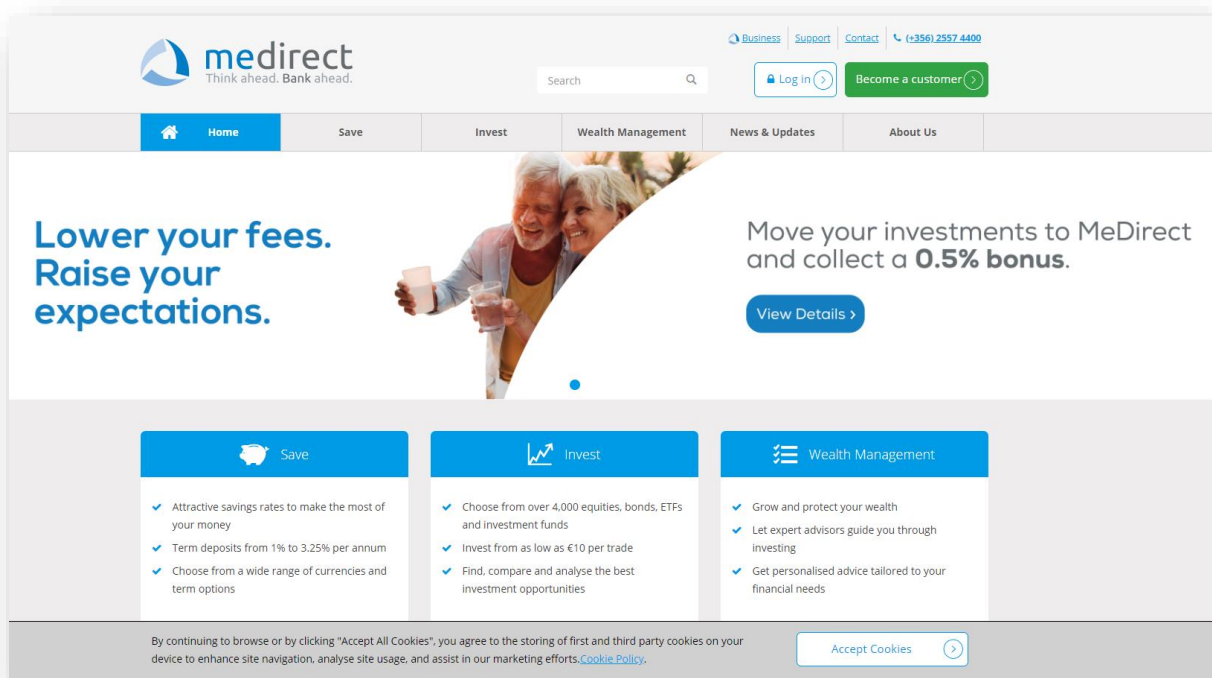
1. A Test Plan with Positive and Negative Test cases to test the Movies API.
2. A C# project consisting of the Integration Tests of all the tests outlined in point 1. All Tests must pass.
3. Optionally, you may want to create a Wrapper for the RESTful API, and create unit tests using a mocking service of your choice, to test your Wrapper.

4 Automation Testing for Front-end components

The second part of the assessment will focus on developing tests for existing front-end components using Selenium Web Driver. Optionally, you may include the use of Specflow to test the web pages.

4.1 Case Introduction

The MeDirect public website is updated regularly as part of our development cycle. In addition, Business uses can modify and update the content via our CMS. As a result, the web site needs constant grooming and monitoring to make sure that any changes (business, or development) don't break the overall customer experience.



The screenshot shows the MeDirect website homepage. At the top, there is a navigation bar with the MeDirect logo, a search bar, and links for Business, Support, Contact, and a phone number (+356) 2557 4400. Below the navigation bar is a main banner area with a large image of an elderly couple smiling. The banner contains the text "Lower your fees. Raise your expectations." on the left and "Move your investments to MeDirect and collect a **0.5% bonus.**" on the right, with a "View Details >" button. Below the banner are three columns of services: "Save", "Invest", and "Wealth Management". Each column has a list of benefits. At the bottom, there is a footer with a cookie consent message and an "Accept Cookies" button.

medirect
Think ahead. Bank ahead.

Business Support Contact (+356) 2557 4400

Search

Log in Become a customer

Home Save Invest Wealth Management News & Updates About Us

Lower your fees. Raise your expectations.

Move your investments to MeDirect and collect a **0.5% bonus.**

View Details >

Save

- ✓ Attractive savings rates to make the most of your money
- ✓ Term deposits from 1% to 3.25% per annum
- ✓ Choose from a wide range of currencies and term options

Invest

- ✓ Choose from over 4,000 equities, bonds, ETFs and investment funds
- ✓ Invest from as low as €10 per trade
- ✓ Find, compare and analyse the best investment opportunities

Wealth Management

- ✓ Grow and protect your wealth
- ✓ Let expert advisors guide you through investing
- ✓ Get personalised advice tailored to your financial needs

By continuing to browse or by clicking "Accept All Cookies", you agree to the storing of first and third party cookies on your device to enhance site navigation, analyse site usage, and assist in our marketing efforts. [Cookie Policy](#)

Accept Cookies

4.2 Objective

The QA team need to design and develop an automated test to regress the current website and functionality. The focus of the deliverable should be test code for one (or more) of the following pages:

- A. <https://www.medirect.com.mt/contact> OR
- B. <https://www.medirect.com.mt/save> OR
- C. <https://www.medirect.be/mutual-funds/selection-of-top-funds/mixed-funds>

Expected deliverables for this task:

1. A test plan of the tests you intend to execute against the pages you choose to test.
2. Automated tests in C# code to regress the content of the page between runs and provide a report when differences are found.
3. The use of the Page Object Model when testing the Web Pages is recommended.
4. Although not mandatory, you may want to use Specflow and Gherkin syntax to drive your test scenarios.