**Problem Statement**

- Book Management API:

- Adding a Book with valid and invalid inputs.

- Retrieving a Book with valid and invalid book IDs.

- Updating Book Details with valid and invalid data.

- Deleting a Book with valid and invalid book IDs.

**Solution**

**Setting Up the Project**

**Create a new test project in Visual Studio:**

1.Open Visual Studio.

Create a new project -> Select "NUnit Test Project".

Name your project and click "Create".

2. Install RestSharp NuGet package:

Right-click on your project in the Solution Explorer.

Select "Manage NuGet Packages".

Search for "RestSharp" and install it.

**Sample Code for Adding a Book with Valid and Invalid Inputs**

using NUnit.Framework;

using RestSharp;

using System.Net;

public class BookManagementTests

{

private RestClient client;

[SetUp]

public void Setup()

{

// Initialize RestClient with the base URL of the API

client = new RestClient("http://bookstoreapi.com");

}

[Test]

public void AddBook\_ValidInput()

{

// Create a new POST request to the addBook endpoint

var request = new RestRequest("/addBook", Method.POST);

// Add a JSON body with valid book details

request.AddJsonBody(new { title = "Valid Book", author = "Author Name", isbn = "1234567890", publishedDate = "2023-07-15" });

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 200 (OK)

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

// Assert that the response body contains a success message

var content = response.Content;

Assert.IsTrue(content.Contains("Book added successfully"));

}

[Test]

public void AddBook\_InvalidInput()

{

// Create a new POST request to the addBook endpoint

var request = new RestRequest("/addBook", Method.POST);

// Add a JSON body with invalid book details

request.AddJsonBody(new { title = "", author = "", isbn = "invalidisbn", publishedDate = "invaliddate" });

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 400 (Bad Request)

Assert.AreEqual(HttpStatusCode.BadRequest, response.StatusCode);

// Assert that the response body contains a validation error message

var content = response.Content;

Assert.IsTrue(content.Contains("Validation failed"));

}

}

}

**Sample code for Retrieving a Book with Valid and Invalid Book IDs**

[Test]

public void RetrieveBook\_ValidBookId()

{

// Create a new GET request to the getBook endpoint with a valid book ID

var request = new RestRequest("/getBook/{bookId}", Method.GET);

request.AddUrlSegment("bookId", 1);

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 200 (OK)

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

// Assert that the response body contains the book details

var content = response.Content;

Assert.IsTrue(content.Contains("\"id\": 1"));

}

[Test]

public void RetrieveBook\_InvalidBookId()

{

// Create a new GET request to the getBook endpoint with an invalid book ID

var request = new RestRequest("/getBook/{bookId}", Method.GET);

request.AddUrlSegment("bookId", 999);

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 404 (Not Found)

Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);

// Assert that the response body contains an error message

var content = response.Content;

Assert.IsTrue(content.Contains("Book not found"));

}

**Sample code for Updating Book Details with Valid and Invalid Data**

[Test]

public void UpdateBook\_ValidData()

{

// Create a new PUT request to the updateBook endpoint with a valid book ID

var request = new RestRequest("/updateBook/{bookId}", Method.PUT);

request.AddUrlSegment("bookId", 1);

// Add a JSON body with valid updated book details

request.AddJsonBody(new { title = "Updated Book", author = "Updated Author", isbn = "0987654321", publishedDate = "2024-12-25" });

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 200 (OK)

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

// Assert that the response body contains a success message

var content = response.Content;

Assert.IsTrue(content.Contains("Book updated successfully"));

}

[Test]

public void UpdateBook\_InvalidData()

{

// Create a new PUT request to the updateBook endpoint with a valid book ID

var request = new RestRequest("/updateBook/{bookId}", Method.PUT);

request.AddUrlSegment("bookId", 1);

// Add a JSON body with invalid updated book details

request.AddJsonBody(new { title = "", author = "", isbn = "invalidisbn", publishedDate = "invaliddate" });

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 400 (Bad Request)

Assert.AreEqual(HttpStatusCode.BadRequest, response.StatusCode);

// Assert that the response body contains a validation error message

var content = response.Content;

Assert.IsTrue(content.Contains("Validation failed"));

}

**Sample code for Deleting a Book with Valid and Invalid Book IDs**

[Test]

public void DeleteBook\_ValidBookId()

{

// Create a new DELETE request to the deleteBook endpoint with a valid book ID

var request = new RestRequest("/deleteBook/{bookId}", Method.DELETE);

request.AddUrlSegment("bookId", 1);

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 200 (OK)

Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);

// Assert that the response body contains a success message

var content = response.Content;

Assert.IsTrue(content.Contains("Book deleted successfully"));

}

[Test]

public void DeleteBook\_InvalidBookId()

{

// Create a new DELETE request to the deleteBook endpoint with an invalid book ID

var request = new RestRequest("/deleteBook/{bookId}", Method.DELETE);

request.AddUrlSegment("bookId", 999);

// Execute the request and store the response

var response = client.Execute(request);

// Assert that the response status code is 404 (Not Found)

Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);

// Assert that the response body contains an error message

var content = response.Content;

Assert.IsTrue(content.Contains("Book not found"));

}

### **Running the Tests**

1. **Build the project**: Ensure there are no errors in the code.
2. **Run the tests**: Use the Test Explorer in Visual Studio to run the tests. Ensure the tests pass and verify that the API handles each scenario as expected.