RSM - Automation QA tasks

Technical Requirements

- Java 17 or newer
- Maven - for dependency management
- Git - Push your project to a Git repository
- JUnit / TestNG for test execution
- RestAssured for API testing
- Selenium - latest version for UI testing.

Submission Guidelines

1. Push your project to a Git repository and send the link to it back to the interviewers.
2. Provide a README.md file with instructions on how to run the tests and any relevant information about the project.
3. Ensure your tests are executable with *mvn test.*

FooBarUtil Tests

Your task is writing tests for a given Java utility class FooBarUtil using JUnit or TestNG. The goal is to evaluate your ability to write effective, well-structured, and meaningful test cases.

You should focus on:

- Test coverage - ensuring all methods are tested
- Edge cases - handling unexpected or boundary inputs
- Code quality - writing clear, maintainable tests
- Bug detection - identifying potential issues in the implementation

```
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.stream.Collectors;

public class FooBarUtil {
    public static String fooBar(String input) {
        try {
            Set<Integer> duplicatesContainer = new HashSet<>();
            String fooBarsList = Arrays.stream(input.split(","))
                    .map(Integer::parseInt)
                    .map(number -> checkForFooBar(number) + (duplicatesContainer.add(number) ?
"" : "-copy"))
                    .collect(Collectors.joining(","));
            System.out.println(fooBarsList);
            return fooBarsList;
        } catch (Exception exception) {
```

```
            String errorMessage = "[Error] Invalid input.";
            System.out.println(errorMessage);
            return errorMessage;
        }
    }

    private static String checkForFooBar(int number) {
        if (number == 0 || number % 3 != 0 && number % 5 != 0) return String.valueOf(number);
        return (number % 3 == 0 ? "foo" : "") + (number % 5 == 0 ? "bar" : "");
    }
}
```

Explanation of FooBarUtil Code

The FooBarUtil class defines a single public static method, fooBar(String input), which processes a comma-separated string of numbers and prints out a modified version based on the following rules:

- Numbers divisible by 3 are replaced with "foo".
- Numbers divisible by 5 are replaced with "bar".
- Numbers divisible by both 3 and 5 are replaced with "foobar".
- Numbers that are neither divisible by 3 nor 5 remain unchanged.
- If a number appears more than once in the input, the second and subsequent occurrences have "-copy" appended.
- If the input is invalid (not a list of numbers), an error message is printed.

Example Inputs and Outputs

Input	Output
3,5,15,7	foo,bar,foobar,7
3,3,5,5,15,15,7,7	foo,foo-copy,bar,bar-copy,foobar,foobar-copy,7,7-copy
3,a,15,10	[Error] Invalid input.

Bookstore API Testing

Your task is to test an endpoint from the Books API available at **DemoQA Swagger API**. The goal is to evaluate your ability to implement clean and maintainable API tests, structure a project professionally, and your knowledge of best practices and programming principles.

Setup

1. Sign up: https://demoqa.com/register
2. Go to https://demoqa.com/swagger
3. Go to */Account/v1/GenerateToken* endpoint and try it out using your username and password. Copy the generated token.
4. Go to the Authorize button in the upper right corner
5. In the *Available Authorizations* dialog set up the Bearer (apiKey) authentication.
6. Now you can try out the endpoints with the JWT authentication.

Task Description

1. Authentication Setup:
    a. Implement one-time authentication at the beginning of the test run.
    b. Ensure the authentication token is reused for all API requests.
2. Choose an API Endpoint to Test:
    a. Select one of the endpoints from the Books API section of the DemoQA Swagger API.
    b. Implement a few tests - positive and negative
    c. Implement proper verifications for the API response

You should focus on:

- Code quality
- Project structure
- Best practices and design patterns

Create the project and write the code with the mindset of a new long-term project.

Web Tables UI Testing

Your task is to automate Web Table functionality testing on the DemoQA Web Tables page (https://demoqa.com/webtables) using Selenium, Java, and JUnit/TestNG. This task will evaluate your ability to:

- Implement UI automation effectively
- Use best practices (clean code, maintainability)
- Implement proper verifications
- Use Page Object Model (POM)

Task Description

Automate the Following Web Table Tests:

- Add a New Member
- Edit an Existing Member
- Delete a Member

What other automated tests can be added here to fully test the page functionalities? Please add them as automated tests accordingly.