**Back-End Test Automation – Regular Exam**



Submit your work as a **single zip / rar / 7z archive** holding your solutions for each problem at SoftUni Website.

Please refer to the end of this document for **instructions on how to submit your work**.

## The "Story Spoiler" System

**"Story Spoiler"** is an interactive web application for **sharing and managing story spoilers**. It is accessible through a dedicated URL; the platform is designed for story enthusiasts to connect and share. It offers a seamless experience with features like **story spoiler** **creation** and **story spoiler management**. Story Spoiler is also referred as Story.

**Your task** is focused on **using Postman**, **Newman and RestSharp** to conduct **API tests**, ensuring the application's functionality works as expected.  
You can find the Web App here: [**https://d24hkho2ozf732.cloudfront.net/**](https://d24hkho2ozf732.cloudfront.net/)

### API Endpoints

**"Story Spoil"** exposes a **RESTful API**, available at**:**   
[**https://d5wfqm7y6yb3q.cloudfront.net/api**](https://d5wfqm7y6yb3q.cloudfront.net/api)

The **supported API endpoints** and **the interactive documentation** can be found at:

[**https://d5wfqm7y6yb3q.cloudfront.net/swagger/index.html**](https://d5wfqm7y6yb3q.cloudfront.net/swagger/index.html)

For your convenience, here is a **brief overview of the most important endpoints** below as well:

### 1. User

* POST /api/User/Create - create a new user. Post a JSON object in the request body:  
  {  
  "userName": "string",  
  "firstName": "string",   
  "midName": "string",

"lastName": "string",

"email": "user@example.com",   
"password": "string",   
"rePassword": "string"   
}

* POST /api/User/Authentication - log in an existing user. Post a JSON object in the request body:  
  {  
  "userName": "string",   
  "password": "string"  
  }

### 2. Access Token

* When a user logs in, the response format is JSON object:  
  **{**"userName": "string",   
  **"password": "1234567",   
  "accessToken": "eyJhbGciOiJ…"  
  }**

**NB! Access token is needed for all story spoiler requests. It should be placed under the Authorization tab, Bearer Token option.**

### 3. Story Spoiler

All of the **following requests require Authorization**!

* **GET /api/Story/All** – list all story spoilers (empty request body).
* **GET /api/****Story/Search** – search spoilers by their name.  
  Requires **query parameter: ?keyword=storyTitle**
  + **POST** **/api/Story/Create** – create a new story spoiler.  
    Include a JSON object in the request body (title and description are mandatory, url is optional):

**{  
"title": "string",   
"description": "string",**  **"url": ""**

**}**

* PUT /api/Story/Edit/storyId – replace the existing story spoiler with a new one.  
  Include a JSON object in the request body (title and description are mandatory, url is optional):   
  **{  
  "title": "string",   
  "description": "string",   
  "url": ""**

**}**

* DELETE **/api/Story/Delete/storyId** – delete existing story spoiler.

## RESTful API: Postman API Tests (35 points)

Your task is to write **API tests** with Postman for certain **RESTful API endpoints**. Organize your tests within a collection, **use collection variables** and **pre-request scripts** to **guarantee successful execution on every run**. **It's important to use collection variables**, **NOT ENVIRONMENT VARIABLES**, to maintain the integrity and portability of the test suite.

### Prerequisites

First you need to **register a new user**. **Registration** of a **new user** is **a mandatory step** that you must complete prior to conducting your API tests. You have the **flexibility to register** either through the [**web UI**](https://d24hkho2ozf732.cloudfront.net/) or **by making a request via Postman**. Please note that this **initial registration process is not included in the scope of your assignment and will not contribute to your final score**. However, it is essential as you will **need an active user account** for all subsequent API requests that form the core of your test cases.   
**If you decide to register via Postman**, **remove this request from your collection.**

### Base Setup

* Add the base URL [**https://d5wfqm7y6yb3q.cloudfront.net**](https://d5wfqm7y6yb3q.cloudfront.net) as a collection variable **{baseURL}**.
* Ensure all requests use this **{baseURL}**.

### Login and Authentication

* Send a **POST request** for **user authentication**.
* **Assert a 200 status** code for success.
* **Assert** that the **response body includes** the attributes **username**, **password**, and **accessToken**. The objective is not to confirm the specific content of these fields but to ensure that they are present in the response.
* Save the **accessToken** as a **collection variable** **{{token}}** for **Bearer Token authorization in subsequent requests**.

### Create a New Story Spoiler

* Use a **pre-request script** to **generate a random title** (**a word followed by up to three digits**).
* Store this title as a **{{randomTitle}}** collection variable.
* **Send a POST request** with **{{randomTitle}}** and a **description** (description can be added manually).
* **Assert a 201 status** code.
* **Assert the "Successfully created!" message** in the response body.
* Extract **storyId** from the response body as a collection variable **{{storyId}}.**

### Search for the Created Story Spoiler by Title

* **Send a GET request** using **{{randomTitle}}** collection variable as the **search keyword** (query parameter).
* **Assert a 200 status** code.
* **Confirm the response** is a **non-empty array**.
* **Confirm that the response** contains the story with the **matching title**.

### Edit the Created Story Spoiler

* **Send a PUT request** to **modify the Story Spoiler** identified by **{{storyId}}. Change its title** (you can do this manually, no need for scripting).
* **Assert a 200 status** code.
* **Assert** the **"Successfully edited"** message.

### Delete the Edited Story Spoiler

* Send a **DELETE request to delete the edited Story Spoiler** identified by **{{storyId}}.**
* **Assert a 200 status** code.
* **Assert** the **"Deleted successfully!"** message.

### Final Steps

1. Make sure that your collection contains all the requests needed:

* **Login**
* **Create New Story Spoiler**
* **Search for the Created Story Spoiler**
* **Edit the Created Story Spoiler**
* **Delete the Created Story Spoiler**

1. Make sure that the collection can be executed successfully on each run.
2. Export and save your collection in a single file.

## Newman with htmlextra Reporter (15 points)

* **Run** the exported **collection** that you created via Postman in **Newman**.
* Use **htmlextra as a reporter.**
* Add the **generated html report** to the archive with your other tasks.

## RESTful API: RestSharp API Tests (50 points)

**In this task**, you will demonstrate your ability to interact with a **RESTful API** using **RestSharp** within a **.NET test project**. Your primary goal is to create a set of **automated tests from scratch** that validate the key functionalities of the **StorySpoil API**. You will be **assessed** on your ability to configure a **test project**, **utilize RestSharp** to **make API requests**, and **assert** the expected **responses using NUnit**.

### 3.0. Prerequisites

First, you are required to **set up a new NUnit Test Project** in your Visual Studio. Ensure you **install all necessary packages**, including **RestSharp**, to create a functional API testing suite. This project will serve as the foundation for your subsequent testing tasks.

### 3.1. Base Setup

* **Initialize a RestClient** with the **base URL of the API**.
* Since you already have an account**, authenticate** with your credentials, and store the received JWT token.
* Configure the **RestClient with an Authenticator using the stored JWT token**.

### 3.2. Data Transfer Objects (DTOs)

**Before you begin writing your tests**, it's important to **create Data Transfer Objects (DTOs).** Given that you are **familiar** with the **structure of both the requests and responses**, you have the flexibility to **create as many DTOs as you need**. However, these **two DTOs should be sufficient** for the scope of your task:

* **ApiResponseDTO** - his DTO will be used to parse common response structures from the API. It should include the following properties:
* **Msg** of **type string** to capture response messages.
* **StoryId** of **type string** to capture the unique identifier of a story. This field may be null for responses that do not include a story ID.
* **StoryDTO** - representing the structure of a story for creation and editing purposes. It should include the following properties:
  + **Title** of **type string** for the story's title.
  + **Description** of **type string** for the story's description.
  + An **optional Url** of **type string** representing a link to the story's picture, if applicable.

### 3.3. Create a New Story Spoiler with the Required Fields

* Create a test to send a **POST request** to **add a new story**.
* **Assert** that the response status code is Created (201).
* **Assert** that the **StoryId** is **returned in the response**.
* **Assert** that the **response message** indicates the story was **"Successfully created!".**
* **Store the StoryId** as a static member of the test class for use in **subsequent tests**.

### 3.4. Edit the Story Spoiler that you Created

* Create a test that **sends a PUT request** to edit the story using the **StoryId** from the **story creation test as a path variable**.
* **Assert** that the **response status code is OK (200).**
* **Assert** that the **response message** indicates the story was **"Successfully edited".**

### 3.5. Delete a Story

* Create test that sends a **DELETE request** using the **StoryId** from the created story.
* **Assert** that the response status code is OK (200).
* **Assert** that the response message **is "Deleted successfully!".**

### 3.6. Try to Create a Story Spoiler without the Required Fields

* Write a test that attempts to **create a story with missing required fields** (Title, Description).
* Send the **POST reques**t with **the incomplete data**.
* **Assert** that the response status code is **BadRequest (400).**

### 3.7. Edit a Non-existing Story

* Write a test to **send a PUT request to edit a story with a StoryId that does not exist**.
* **Assert** that the response status code is **NotFound (404).**
* **Assert** that the response message indicates **"No spoilers...".**

### 3.9. Delete a Non-existing Story

* Write a test to **send a DELETE request to edit a story with a StoryId that does not exist**.
* **Assert** that the response status code is **Bad request (400).**
* **Assert** that the response message indicates **"Unable to delete this story spoiler!".**

### 3.10. Final Steps

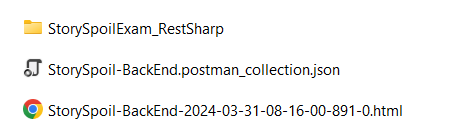
* Ensure that each test is correctly **ordered to maintain the required sequence of actions. Use [Order( )]**
* Verify that tests are designed to **run successfully in on each run.**
* **Delete bin and obj folders** from your solution folder.

## How to submit your exam You should have a single zip / rar / 7z archive containing all of your tasks

Upload your archive at SoftUni website, into [Regular Exam section](https://softuni.bg/trainings/4399/back-end-test-automation-february-2024#lesson-65570).

* The Postman collection should be exported in a single **JSON** file.
* You also need to export the html file **obtained from the htmlextra reporter in Newman.**
* Your **RestSharp API Test** project should be **in a folder**.

At the end, the content of your archive should look similar:



Before archiving, please make sure that you **deleted all bin and obj folders.**