**Back-End Test Automation - Exam Prep I**



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

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

# The "Idea Center" System

**"Idea Center"** is an **interactive platform** designed for the creation and **sharing of innovative ideas**. It's a space for users to engage, share, and **manage ideas** across various fields, enhancing collaboration and innovation. The platform, includes key features like user registration, idea submission, and management.

Your task is to conduct **API tests using Postman, Newman, and RestSharp**, ensuring the application's functionalities perform as expected.

**Access "Idea Center"** **Web App** through its dedicated URL:

[**http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:83**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:83)

### API Endpoints

**"Idea Center"** exposes a **RESTful API**, available at**:**   
[**http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84/api**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84/api)Keep in mind that the API is not directly available trough browser.You can see all the **supported methods** on the **following URL**:

[**http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84/api/Info/Methods**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84/api/Info/Methods)

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

**<http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84/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",   
  "email": "user@example.com",   
  "password": "string",   
  "rePassword": "string",   
  "acceptedAgreement": true   
  }
* POST /api/User/Authentication - log in an existing user. Post a JSON object in the request body:  
  {  
  "email": "user@example.com",  
  "password": "string"  
  }

### 2. Access Token

* When a user logs in, the response format is JSON object:   
  **{  
  "email": "test@gmail.com",   
  "password": "1234567",   
  "accessToken": "eyJhbGciOiJ…"  
  }**

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

### 3. Idea

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

* **GET /api/Idea/All** – list all ideas (empty request body)
* **POST** **/api/Idea/Create** – create a new idea.   
  Include a JSON object in the request body (title and description are mandatory, url is optional):  
  **{  
  "title": "string",  
  "url": "",  
  "description": "string"  
  }**
* PUT /api/Idea/Edit – replace the existing idea with the new one.  
  Include a JSON object in the request body (title and description are mandatory, url is optional):  
  **{  
  "title": "string",   
  "url": " ",   
  "description": "string"  
  }**Requires a **query parameter: ?ideaId={id}**
* DELETE /api/Idea/Delete – delete existing idea.  
  Requires a **query parameter: ?ideaId={id}**

## 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**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:83) 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 [**http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:84) 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 **email**, **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 Idea

* 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 200 status** code.
* **Assert the "Successfully created!" message** in the response body.
* **Assert** that the **title** and the **description** of the **created idea** in the response matches the expected title and the expected description.

### List all Ideas

* **Send a GET request to receive a list of all created ideas**.
* **Assert a 200 status** code.
* **Assert** that the **response is an array** and that it contains **at least one item**.
* Extract the **id of the last created idea** from the response body and **store it as a collection variable** **{{lastIdeaId}}.**

### Edit the Last Idea

* **Send a PUT request** to **modify the Idea** identified by **{{lastIdeaId}}.** **Change its title** (you can do this manually, no need for scripting).
* **Assert a 200 status** code.
* **Assert** the **"Edited successfully"** message.
* **Assert** that the **title in the response matches the new title you provided.**

### Delete the Edited Idea

* Send a **DELETE request to delete the edited Idea** identified by **{{lastIdeaId}}.**
* **Assert a 200 status** code.
* **Assert** that the **type of the response body is a string.**
* **Assert** that the **string equals "The idea is deleted!"**.  
  **\*Keep in mind that the response is not a JSON object, but a string.**

### Final Steps

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

* **Login**
* **Create New Idea**
* **List All Ideas**
* **Edit the Last Created Idea**
* **Delete the Edited Idea**

1. Make sure that the **collection** can be **executed successfully on each run**.

Export and save your collection in a **single JSON 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 **IdeaCenter 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've 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.
* **IdeaId** of **type string** to capture the unique identifier of a idea. This field may be null for responses that do not include a idea ID.
* **IdeaDTO** - representing the structure of a idea for creation and editing purposes. It should include the following properties:
  + **Title** of **type string** for the idea's title.
  + **Description** of **type string** for the idea's description.
  + An **optional Url** of **type string** representing a link to the idea, if applicable.

### 3.3. Create a New Idea with the Required Fields

* **Create a test** to send a **POST request** to **add a new idea**.
* **Assert** that the response **status code is OK (200).**
* **Assert** that the **response message** indicates the idea was **"Successfully created!".**

### 3.4. Get All Ideas

* **Create a test to send a GET request to list all ideas.**
* **Assert that** the response **status code is OK (200).**
* **Assert that** the response contains a **non-empty list of ideas.**
* **Store** the **id** of the **last created idea** in a **static member of the test class to maintain its value between test runs**.

### 3.5. Edit the Last Idea that you Created

* Create a test that **sends a PUT request** to edit the idea.
* Use the **id** that you **stored in the previous request as a query parameter**.
* **Assert** that the **response status code is OK (200).**
* **Assert** that the **response message** indicates the idea was **"Successfully edited".**

### 3.6. Delete the Idea that you Edited

* Create a test that **sends a DELETE request**.
* Use the **id** that you **stored in the "Get All Ideas" request as a query parameter.**
* **Assert that** the response **status code is OK (200).**
* **Confirm** that the response **contains** **"The idea is deleted!".**

**\* Keep in mind that the response in not a json object, but a string!**

### 3.7. Try to Create an Idea without the Required Fields

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

### 3.8. Try to Edit a Non-existing Idea

* Write a test to **send a PUT request to edit an Idea with a ideaId that does not exist**.
* **Assert** that the response status code is **BadRequest (400).**
* **Assert** that the response contains **"There is no such idea!".**

**\* Keep in mind that the response in not a json object, but a string!**

### 3.9. Try to Delete a Non-existing Idea

* Write a test to **send a DELETE request to edit an Idea with a ideaId that does not exist**.
* **Assert** that the response status code is **BadRequest (400).**
* **Assert** that the response contains **"There is no such idea!".**

**\* Keep in mind that the response in not a json object, but a string!**

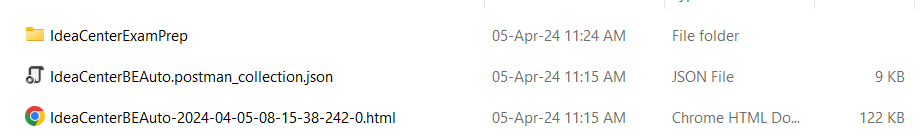
### 3.10. Final Steps

* 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 attach a single zip / rar / 7z archive containing all of your tasks.

* 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 from your RestSharp Test project.**