Assignment 1 (Unauthenticated Web APIs)

Published | Assign to | Edit

In this assignment, we want to develop a set of APIs that support the operations of a New Zealand sign language support group. A template project has been created for you. You can download the template project [here (https://canvas.auckland.ac.nz/courses/104278/files/13431419/download)](https://canvas.auckland.ac.nz/courses/104278/files/13431419/download). Your assignment MUST run on .NET 8.

Important Note:

This assignment will be marked using a program. Therefore, please follow the instructions below carefully.

- You MUST use the template project and the files in the project.
- You MUST NOT change the name of any file/folder or move any file to a different folder. You can change the contents of the files.
- You MUST NOT create any other file/folder apart from the Migrations folder.
  - In fact, the database has been given to you. You can change the data stored in the database, but you should not modify the structures of the tables in the database.
  - You can use the code-first method to generate the database and compare the structure of the generated database with the provided database to verify that your model classes and DbContext class have been implemented correctly.
- Your data MUST be stored in the database named "A1Database.sqlite".
- You MUST submit all the files as indicated in section "**Submission**".
- You MUST use the test program to make sure that your program works with the marking program BEFORE submission. The details on how to use the test program are given in section "**Checking Your Submission**".
- The prefix of the URLs of your endpoints MUST be http://localhost:8080/webapi/
- You MUST NOT use any absolute path, e.g., C:\Users\jbon007\335, in your code as the path on marker's machine would be different from your machine.
- Apart from the standard C# packages and the three packages for Entity Framework discussed in our lectures, you MUST NOT use any other packets.

Overview

You are required to implement several endpoints as described below. We have set up a server to demonstrate how each of the endpoints works. The data returned by the server may differ from the data provided for this assignment, but it showcases the general data format returned by the endpoints.

We recommend using Firefox to access the endpoints, as it displays the data in a human-friendly JSON format. The Swagger UI for the example server can be found at **https://cws.auckland.ac.nz/335P12024/swagger/index.html** **(https://cws.auckland.ac.nz/335P12024/swagger/index.html)** . The page lists multiple APIs, but for this assignment, you only need to implement a subset of those APIs. The remaining APIs will be implemented in the next assignment. Please ensure that the URIs of your endpoints match the corresponding ones shown at **https://cws.auckland.ac.nz/335P12024/swagger/index.html** **(https://cws.auckland.ac.nz/335P12024/swagger/index.html)** .

In the template project, the "Logos" directory stores the group's logo, while the "SignsImages" directory contains photos of the products sold by the company.

Database

A database is used to store information related to the operations of the group and the web server. The database consists of multiple tables, but for this assignment, we will focus on the "Signs" and "Comments" tables.

The "Signs" table contains the descriptions of various signs used in sign language. It has two columns: Id and Description. They both have string data type. "Id" is the key of the table. The "Description" column contains the descriptions of the signs.

The "Comments" table consists of five columns: Id, UserComment, Name, Time, and IP. The type of Id is int. The type of the other columns is string. Here's a breakdown of the column meanings:

- Id: This column serves as the unique identifier assigned to each comment. It is automatically assigned by the database when a comment is inserted. The Id column acts as the primary key for the table.
- Time: This column records the timestamp when a comment is inserted into the database. It is assigned by the server when the comment is written to the database. The format of the time should follow the pattern "yyyyMMddTHHmmssZ", where "yyyy" represents the year, "MM" represents the month, "dd" represents the date, "T" separates the date and time components, "HH" represents the hours in 24-hour format, "mm" represents the minutes, "ss" represents the seconds, and "Z" indicates Coordinated Universal Time (UTC).
- UserComment: This column stores the comment entered by the user.
- Name: This column stores the name of the user who provided the comment. The user provides this information when entering the comment.
- IP: This column stores the IP address of the user who wrote the comment. The server code obtains the user's IP address from the Request.HttpContext.Connection.RemoteIpAddress property.

Note that none of the columns in the "Comments" table are allowed to have null values.

Endpoint 1: GET THE VERSION OF THE WEB API (1 mark)

- This endpoint returns a string “1.0.0 (Ngāruawāhia) by xxx” where "xxx" must be your UPI, e.g., jbon007.
- Name this API as GetVersion.
- Example: **https://cws.auckland.ac.nz/335P12024/webapi/GetVersion** **(https://cws.auckland.ac.nz/335P12024/webapi/GetVersion)**
- Note: Apart from the UPI, the returned string must be exactly the same as the example implementation. That is, no extra spaces are allowed. The letters in the UPI must be lowercase letters.

Endpoint 2: GET THE LOGO OF THE GROUP (1 mark)

- This endpoint returns the logo of the group as an image file.
- Name this API as Logo
- Example: **https://cws.auckland.ac.nz/335P12024/webapi/Logo** **(https://cws.auckland.ac.nz/335P12024/webapi/Logo)**
- Note: The “Content-Type” header in the response must match the type of the returned image.

Endpoint 3: LIST THE SIGNS PROVIDED BY THE GROUP (1 mark)

- This endpoint returns the information of all the signs provided by the group. The information of the signs is stored in the "Signs" table of the "A1Database.sqlite" database.
- The data should be returned as a list of JSON objects. Each JSON object should have keys that correspond to the columns in the "Signs" table of the database. The JSON objects should follow the structure defined in the "Sign.cs" file, which defines the class for the Sign object.
- The response should include the “Content-Type” header.
- Name this API as AllSigns.
- Examples: **https://cws.auckland.ac.nz/335P12024/webapi/Signs/re** **(https://cws.auckland.ac.nz/335P12024/webapi/Signs/re)**
- Note: The order in which the signs appear in the message is unimportant.

Endpoint 4: LIST SIGNS/SIGN PROVIDED BY THE GROUP (1 mark)

- This endpoint returns information about the signs provided by the group whose descriptions match a search term. The information includes the descriptions of the signs and their IDs.

- The data should be returned as a list of JSON objects. Each JSON object should have keys that correspond to the columns in the "Signs" table of the database. Each JSON object should follow the structure defined by the "Sign.cs" file, which defines the class for the Sign object.
- The response should include the “Content-Type” header.
- The endpoint is called with a search term, which can be a word or a partial word. The API returns information about the sign(s) whose descriptions contain words or partial words that match the search term.
  - The search is case-insensitive. For example, the search term "go" would match words like "Google" and "good".
  - If the search term does not match the descriptions of any sign, an empty list is returned.
- Example: **https://cws.auckland.ac.nz/335P12024/webapi/Signs/re** (https://cws.auckland.ac.nz/335P12024/webapi/Signs/re)
- Name this API as Signs.

Endpoint 5: GET THE IMAGE OF A SIGN (1 mark)

- This endpoint returns the image of a sign with a given ID.
  - The endpoint is called with the ID of a sign.
  - The image of a sign is stored in a file. The file name consists of the ID of the sign, as well as a suffix reflecting the type of the image. For example, for a sign with ID 'black-3201', the name of the image file could be 'black-3201.png' if the image is in PNG format.
- Name this API as SignImage.
- If the image cannot be found, the API should return the image in file "default.png".
- Examples
  - The image of a valid sign **https://cws.auckland.ac.nz/335P12024/webapi/SignImage/black-3201** (https://cws.auckland.ac.nz/335P12024/webapi/SignImage/black-3201)
  - The image of a sign that does not exist **https://cws.auckland.ac.nz/335P12024/webapi/SignImage/black-320** (https://cws.auckland.ac.nz/335P12024/webapi/SignImage/black-320)
- Note: The “Content-Type” header in the response must match the type of the returned image.

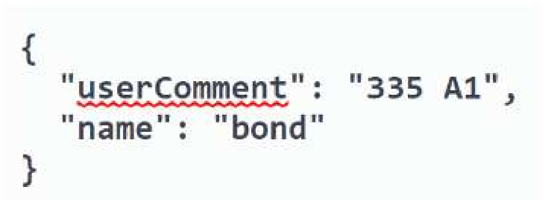
Endpoint 6: GET A COMMENT WITH A GIVEN ID (1 mark)

- This endpoint allows the user to retrieve a comment with a given ID.
- The endpoint takes a parameter, which is the ID of the comment to be retrieved from the "Comments" table of the database.
- The comment should be returned as a JSON object. The keys in the JSON object should have the same names as their corresponding columns in the "Comments" table. Each JSON object is an instance of the Comment class defined in the Comment.cs file.

- If the comment with the given ID does not exist, the message "Comment xxx does not exist." (quotation marks are not part of the response) should be returned (without the quotation marks), where "xxx" is the ID given by the user. The response code should be set to 400 Bad Request.
- Examples
  - Retrieve a comment that exists: **https://cws.auckland.ac.nz/335P12024/webapi/GetComment/3** **(https://cws.auckland.ac.nz/335P12024/webapi/GetComment/3)**
  - Retrieve a comment that does not exist: **https://cws.auckland.ac.nz/335P12024/webapi/GetComment/333** **(https://cws.auckland.ac.nz/335P12024/webapi/GetComment/333)**
- Name this method as GetComment.
- Hint: The BadRequest method can generate the 400 Bad Request response code.

Endpoint 7: WRITE COMMENT (1 mark)

- This endpoint allows users to write comments.
- When calling this API, users are required to provide values for the UserComment and Name fields. Both fields must have values provided. The values should be sent to the server as a JSON object of type CommentInput defined in the CommentInput.cs class.
- The “location” header should be set to the URL for retrieving the inserted comment as discussed in our lecture.
- The endpoint should be called using the POST method.
- The endpoint returns the comment that has been inserted into the "Comments" table. The comment should be returned as a JSON object. The keys in the JSON object should have the same names as their corresponding columns in the "Comments" table. Each JSON object is an instance of the Comment class defined in the Comment.cs file.
- Name this method as WriteComment
- Example: enter the request in Swagger (https://cws.auckland.ac.nz/335P12024/webapi/WriteComment) as shown below.



```
{
  "userComment": "335 A1",
  "name": "bond"
}
```

  Hint: You can obtain the current UTC time through the `UtcNow` property of the DateTime class of C#. You can format the time using the ToString method on the property.

Endpoint 8: DISPLAY COMMENTS (1 mark)

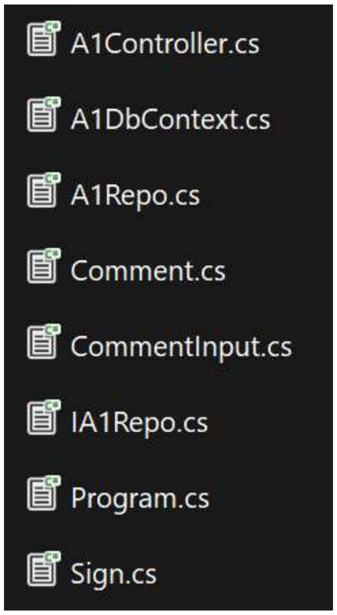
- This endpoint allows users to display comments retrieved from the "Comments" table of the database.
- When calling this API, users can provide a value indicating the number of comments to be displayed. If the user does not provide this value, the latest five comments entered by the clients will be displayed. The latest comment should be shown first.
- If the number of comments in the "Comments" table is less than the number given by the client, all the comments in the "Comments" table should be displayed. Similarly, if the client does not provide a number and there are fewer than five comments in the "Comments" table, all the comments in the table should be displayed.
- The data should be returned as a list of JSON objects. The keys in the JSON object should have the same names as their corresponding columns in the "Comments" table. Each JSON object represents an instance of the Comment class, which is defined in the Comment.cs file.
- The endpoint should be called using the GET method.
- Name this method as Comments.
- Example: **https://cws.auckland.ac.nz/335P12024/webapi/Comments/3**
  (https://cws.auckland.ac.nz/335P12024/webapi/Comments/3)
  (https://cws.auckland.ac.nz/335P12023/webapi/Comments/5)

Hint:

To retrieve the latest 5 comments, you can first convert all the comments in the database to a list of comments. Then, you can retrieve the last 5 elements from the list.

Submission

You must submit the 8 files below:


A1Controller.cs
A1DbContext.cs
A1Repo.cs
Comment.cs
CommentInput.cs
IA1Repo.cs
Program.cs
Sign.cs

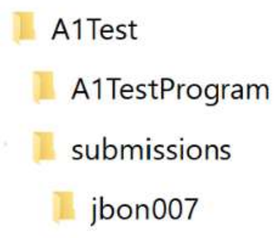
You MUST submit the files ONE BY ONE (i.e., do NOT pack the files into a single file). Submit the files through **adb.auckland.ac.nz. (https://adb.auckland.ac.nz/)**

Checking Your Submission

Create a folder, say A1Test.

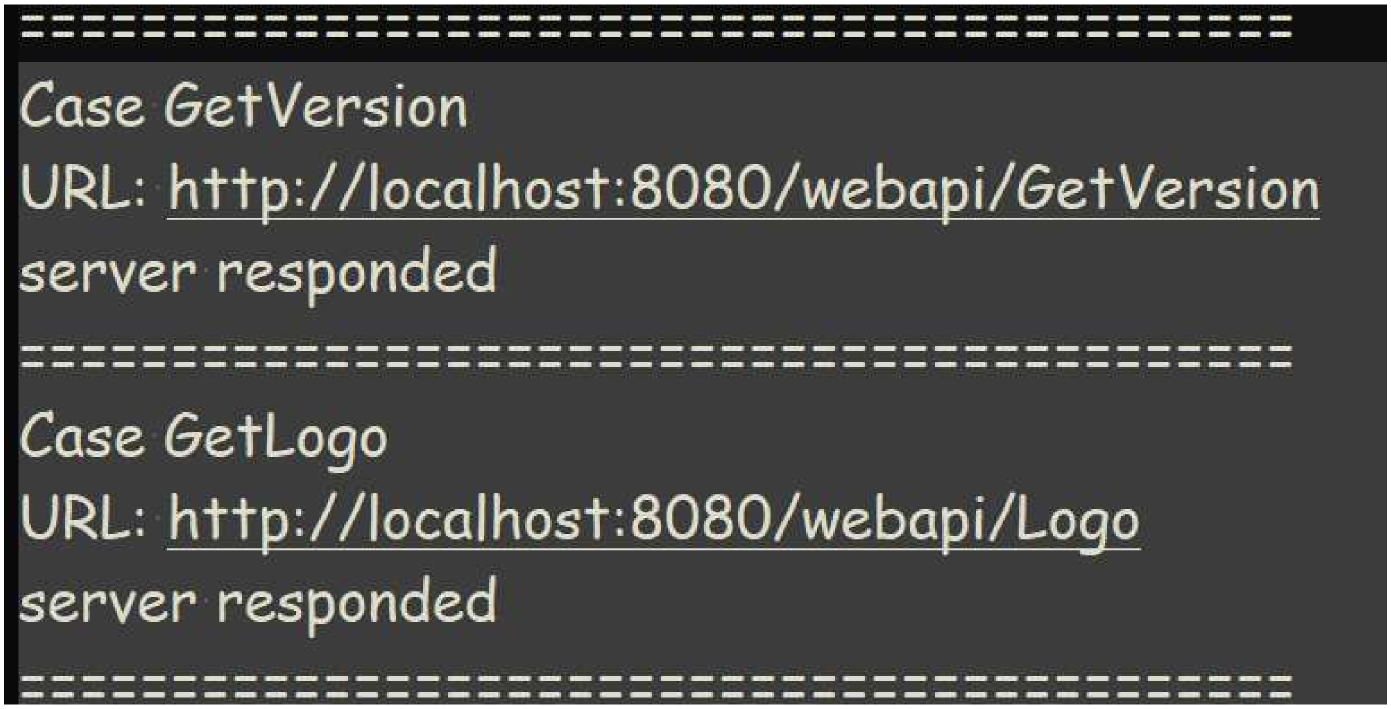
Download the test program **here (https://canvas.auckland.ac.nz/courses/104278/files/13427241/download)** to folder "A1Test". After unpacking it, you should have a folder “A1TestProgram”. I am afraid the program only works on Windows 11 platform as I have no access to any other platform. If you cannot run it on your machine, please use the lab machine to carry out the test.

Under folder “A1Test”, create a folder “submissions”. Create a folder in folder “submissions”. Use your UPI as the name of the new folder. For example, if your UPI is jbon007, your directory’s structure should be as below (other irrelevant folders are not shown):



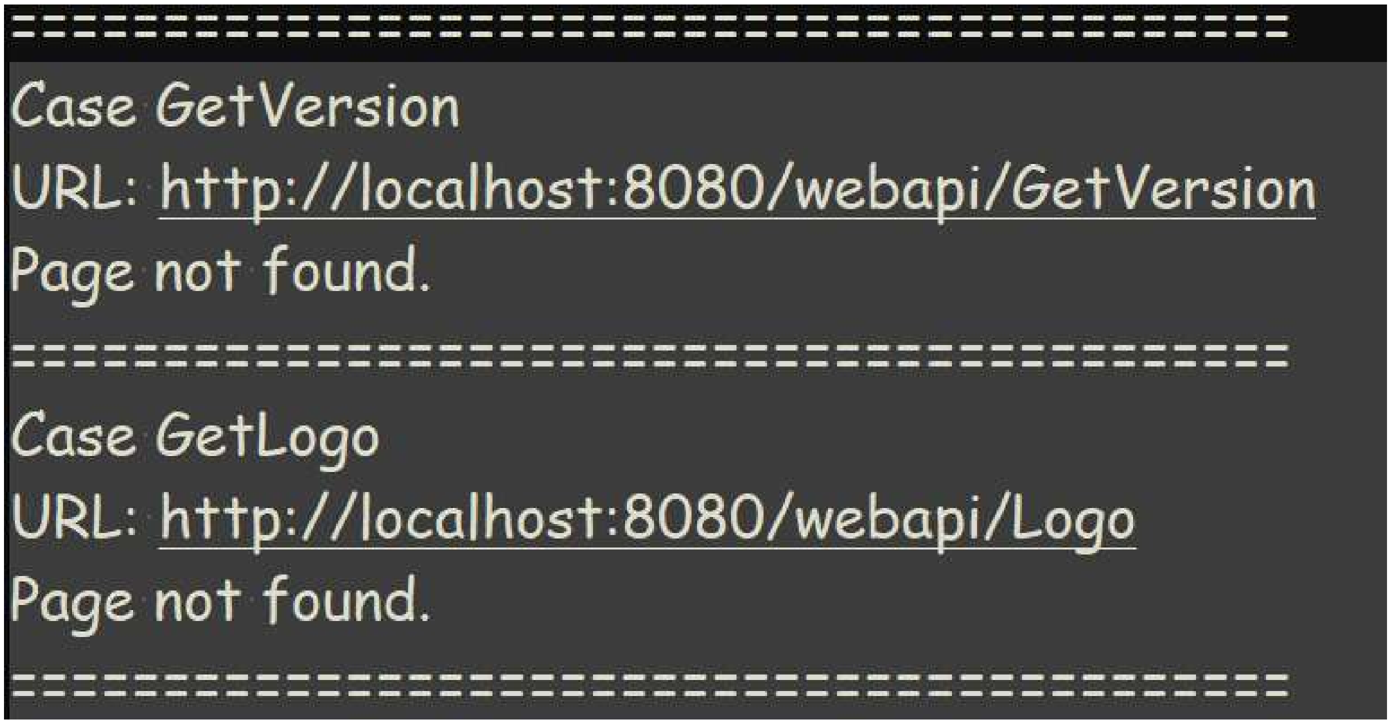
To ensure that your programs work with the marking program, do the following:

- Place the 8 files that you need to submit in folder jbon007.
- Open a command window. Use the “cd” command to change your current folder to folder “A1TestProgram”.
- Run the test program with the command below:
  - dotnet A1Test.dll
- The test program writes the result to a file under the folder “results” which is under folder “A1Test”. Open the file under folder “results”.
  - If the contents of the file look like below, it means your programs work with the marking program.

- 

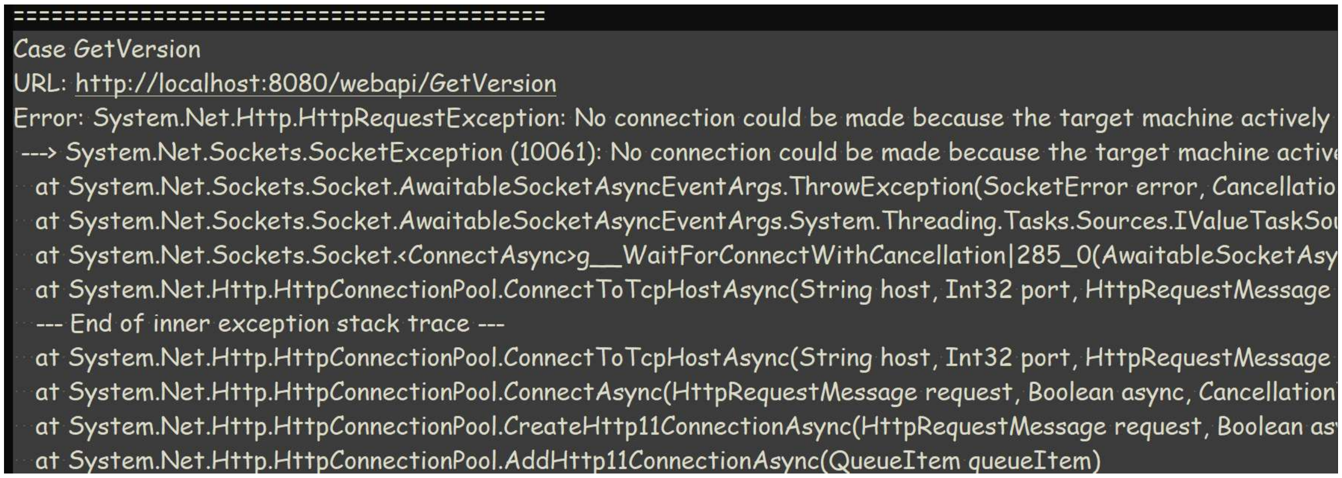
```
========================================
Case GetVersion
URL: http://localhost:8080/webapi/GetVersion
server responded
========================================
Case GetLogo
URL: http://localhost:8080/webapi/Logo
server responded
========================================
```

    - NOTE: The test program only checks whether your program responds to the calls of the marking program. It does not check whether your program gives correct response. You need to thoroughly test your programs using your own data.
- If you miss some of the files in your submission, the information in the file tells you which programs are missing.
- If the contents look like below, you should check whether you have set URI correctly in your code.



```
========================================
Case GetVersion
URL: http://localhost:8080/webapi/GetVersion
Page not found.
========================================
Case GetLogo
URL: http://localhost:8080/webapi/Logo
Page not found.
========================================
```

- If the contents look like below, the likely cause is your programs have compile error.

- 

```
=========================================
Case GetVersion
URL: http://localhost:8080/webapi/GetVersion
Error: System.Net.Http.HttpRequestException: No connection could be made because the target machine actively
 ---> System.Net.Sockets.SocketException (10061): No connection could be made because the target machine activ
   at System.Net.Sockets.Socket.AwaitableSocketAsyncEventArgs.ThrowException(SocketError error, Cancellatio
   at System.Net.Sockets.Socket.AwaitableSocketAsyncEventArgs.System.Threading.Tasks.Sources.IValueTaskSo
   at System.Net.Sockets.Socket.<ConnectAsync>g__WaitForConnectWithCancellation|285_0(AwaitableSocketAsy
   at System.Net.Http.HttpConnectionPool.ConnectToTcpHostAsync(String host, Int32 port, HttpRequestMessage
   --- End of inner exception stack trace ---
   at System.Net.Http.HttpConnectionPool.ConnectToTcpHostAsync(String host, Int32 port, HttpRequestMessage
   at System.Net.Http.HttpConnectionPool.ConnectAsync(HttpRequestMessage request, Boolean async, Cancellation
   at System.Net.Http.HttpConnectionPool.CreateHttp11ConnectionAsync(HttpRequestMessage request, Boolean as
   at System.Net.Http.HttpConnectionPool.AddHttp11ConnectionAsync(QueueItem queueItem)
```

- After running the test program, a "tmp" folder is created under folder "A1Test". This is the project created using your submitted files. To see how your submitted files work, you should run the web application in this folder.

Academic Honesty

Do NOT copy other people's code (this includes the code that you find on the Internet) and DO NOT give your code to people (this includes making your code available in a public repository).

We will use a sophisticated TurnItIn like tool to detect code plagiarism. Last year, more than 100 students were caught by the tool; and, they were dealt with according to the rules at https://www.auckland.ac.nz/en/about/learning-and-teaching/policies-guidelines-and-procedures/academic-integrity-info-for-students.html (https://academicintegrity.cs.auckland.ac.nz/)

Points	8
Submitting	Nothing

Due	For	Available from	Until
16 Aug 2024 at 18:00	Everyone	2 Aug 2024 at 0:00	-

+ Rubric