Assignment 2 (Authenticated 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/13596858/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.
- Your data MUST be stored in the database named “A2Database.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.
- YOU WILL NOT GET ANY MARK IF YOUR PROGRAMS DO NOT WORK WITH THE MARKING PROGRAM. So, please check your submission using the test program.

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 the ones that we have not implemented in A1. 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)** .

Database

A database stores information related to the group's operations and the web server. It has four tables: "Signs", "Users," "Events," and "Organizers."

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 "Users" table records information about registered users. It has three columns: UserName, Password, and Address. All columns are of type string, and none of them can have a null value. UserName serves as the key of the table.

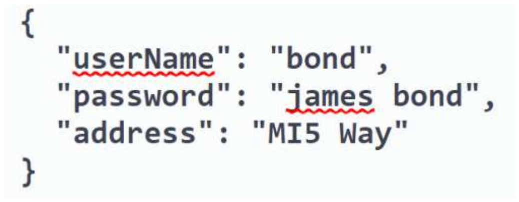
Table “Events” records the information of the events organised by the group. It has six columns as explained below. Id is of type int. All other columns are of type string. None of the columns can have null value.

- Id: This is the id number assigned to an event. It is assigned by the DB when the event is inserted into the DB. This is the key of the table.
- Start: This is the start time of the event. The format of the time should be yyyyMMddTHHmmssZ. Here, yyyy means year, MM means month, dd means date, T separates the date and the time. HH means hours (00 to 23), mm means minutes, ss means seconds and Z means Coordinated Universal Time (UTC).
- End: This is the end time of the event. The format of the time should be yyyyMMddTHHmmssZ.
- Summary: It is a summary of the event.
- Description: It is a detailed description of the event.
- Location: It is the location of the event.

Table "Organizers" records information about the organizers of the events. It consists of two columns: Name and Password. Name serves as the key of the table. Both columns are of type string, and neither of them can have a null value.

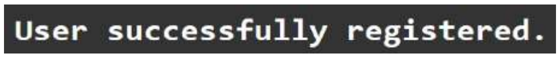
4. **The Endpoints**

Endpoint 1: USER REGISTRATION (1.5 marks)

- This endpoint registers a new user. The information of a registered user should be stored in DB table Users.
- The username of different users should be unique.
- It should be implemented using the POST method.
- The details of the new user must be sent to the server as a JSON object. The keys of the JSON object correspond to the columns in the "Users" table of the database. The JSON object is an instance of the "User" class defined in the "User.cs" file.
- Name this API as Register
- Example 1: This is the case that the UserName has not been registered by another user previously. Enter the request in Swagger (https://cws.auckland.ac.nz/335P12024/webapi/Register) as shown below.
  - 

```
{
  "userName": "bond",
  "password": "james bond",
  "address": "MI5 Way"
}
```

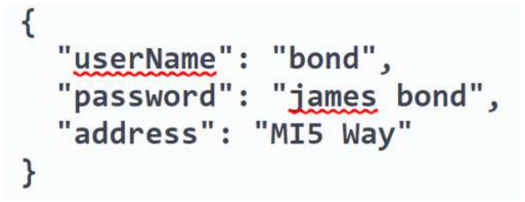
The response is as below.



```
User successfully registered.
```

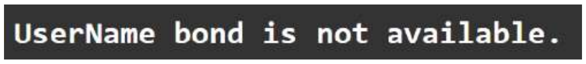
The information of a successfully registered user should be stored in the Users table of the DB.

- Example 2: This is the case that the UserName has already been registered. Enter the request in Swagger (https://cws.auckland.ac.nz/335P12024/webapi/Register) as shown below.



```
{
  "userName": "bond",
  "password": "james bond",
  "address": "MI5 Way"
}
```

The response is as below.



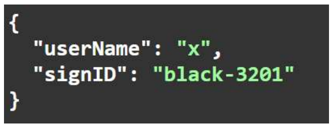
```
UserName bond is not available.
```

In this case, the HTTP response code is 200 as the server has handled the request successfully. The response message is “UserName xxx is not available.” (the quotation mark is not part of the response) where "xxx" is the username entered by the user. This means that the registration is not successful as the UserName “bond” has already existed in the Users table, i.e., another user has already registered with UserName “bond”.

- **NOTE** : The string in the response body must be exactly the same as the string given by the example implementation.
- **Hint**: The “Ok” method will set the status code of the HTTP response to 200. Look up the API document for method Ok to find out how to pass a return message back to the client.

Endpoint 2 PURCHASE A SIGN (1.5 marks)

- This endpoint allows a registered user to purchase a sign available in the "Signs" table. A registered user is a user whose username is stored in the "Users" table of the database.
- Only registered users can access this endpoint. Authentication is performed using Basic Authentication.
- When calling this endpoint, the user needs to specify the ID of the ordered sign.
- If the product with the given ID cannot be found in the "Signs" table, the response should be "Sign xxx not found", where xxx is the ID given by the client. The response code should be 400 (method BadRequest() sets the response code to 400).
- If the sign is found in the "Signs" table, the response should be a JSON object with two keys: "UserName" and "SignID". The values of these keys are the client's username and the ID of the sign, respectively. The JSON object is an instance of the "PurchaseOutput" class defined in the "PurchaseOutput.cs" file.
- Name this API PurchaseSign
- Example: Here is the body of the response to a valid user ordering an existing sign.



```
{
  "userName": "x",
  "signID": "black-3201"
}
```

- Note 1: If a user does not provide any credentials (i.e., UserName and Password) when calling the API, the response header 'www-authenticate' should be set.
- Note 2: For a client who is not a registered user but has valid credentials (i.e., the client is an organizer in table “Organizers”), the response status code should be set to 403.

Endpoint 3 ADD AN EVENT (1.5 marks)

- This endpoint allows an organizer to add an event to the "Events" table.
- Only organizers, whose names are recorded in the "Organizers" table, can access this endpoint. Authentication is required using Basic Authentication.
- The details of the event must be sent to the server as a JSON object. The keys of the JSON object are "Start", "End", "Summary", "Description", and "Location". These keys correspond to the columns in the "Events" table of the database. The JSON object is an instance of the "EventInput" class defined in the "EventInput.cs" file.
    - Note: The values of none of these keys can be null.
- The values of "Start" and "End" must adhere to the format "yyyyMMddTHHmmssZ" as explained in A1.
    - If both "Start" and "End" do not conform to the format, the HTTP response code should be 400, and the response message should be "The format of Start and End should be

    yyyyMMddTHHmmssZ."
    - If only "Start" does not conform to the format, the HTTP response code should be 400, and the response message should be "The format of Start should be yyyyMMddTHHmmssZ."
    - If only "End" does not conform to the format, the HTTP response code should be 400, and the response message should be "The format of End should be yyyyMMddTHHmmssZ."
- If the event is successfully inserted into the "Events" table, the response should be "Success".
- Note 1: If a user does not provide any credentials (i.e., UserName and Password) when calling the API, the response header 'www-authenticate' should be set.
- Note 2: For a client who is not an organizer but has valid credentials (i.e., the client is a registered user), the response status code should be set to 403.
- Note 3: The Name and Password of the organizer in the 'Organizers' table of the example program are 'o1' and 'o1pass', respectively.

Endpoint 4 COUNT THE NUMBER OF EVENTS (1.5 marks)

- This endpoint allows registered users and event organizers to count the number of events available in the “Events” table.
- Only registered users and event organizers can access this endpoint. Authentication is required using Basic Authentication.
- Name this endpoint as EventCount
- Note: If a user does not provide any credentials (i.e., UserName and Password) when calling the API, the response header 'www-authenticate' should be set.

Endpoint 5 LIST THE DETAILS OF AN EVENT (1 mark)

iCalendar is a media type that allows users to store and exchange calendar information. The specification of the iCalendar format is in RFC 5545 (https://www.ietf.org/rfc/rfc5545.txt). A file conforming to the specification is a text file with one or multiple event components. Each event component describes the properties of an event. For example, an event could be the descriptions of the time, the location and other information about the event.

- This endpoint displays the details of an event with a given ID.
- Only registered users and event organizers can access this endpoint. Authentication is required using Basic Authentication.
- If an event with the specified ID exists, the information of the event is retrieved from the database and sent back to the client in iCalendar format.

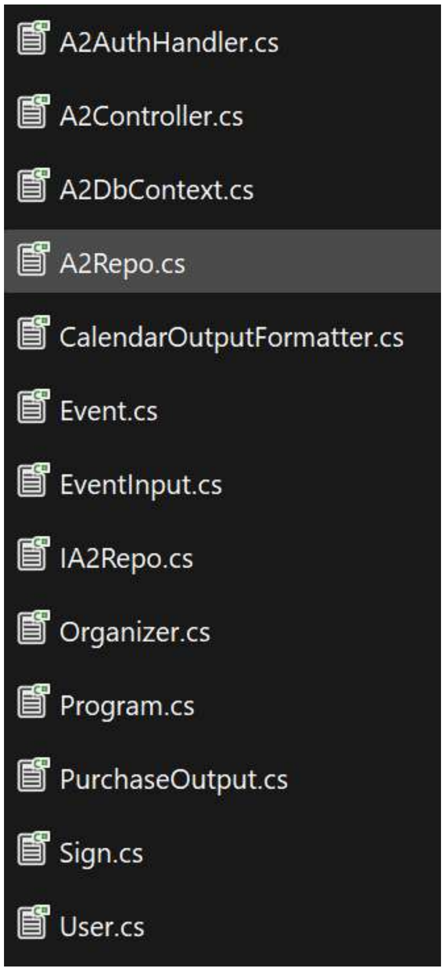
- If an event with the specified ID does not exist, the response message should be "Event xxx does not exist." where xxx is the ID given by the client (quotation mark is not part of the response). The HTTP response code should be set to 400.
- This endpoint should be implemented using the GET method.
- Name this endpoint "Event."
- Note: If a user does not provide any credentials (i.e., UserName and Password) when calling the API, the response header 'www-authenticate' should be set.

NOTE:

- You are expected to find out the required components and the properties that make up the iCalendar formatted data from RFC 5545 (**https://www.ietf.org/rfc/rfc5545.txt** **(https://www.ietf.org/rfc/rfc5545.txt)**).
- The Content-Type header of the HTTP response message that contains iCalendar data should be set to “text/calendar; charset=utf-8”.
- For each event in the iCalendar, it should contain properties: UID, DTSTAMP, DTSTART, DTEND, SUMMARY, DESCRIPTION and LOCATION.
- The value of property PRODID must be set to your UPI, e.g., abc012.
- The value of property DTSTAMP should be set to the time that the HTTP response message is generated.
- The value of the UID should be the value of the Id of the event.
- You should use the validator at **https://icalendar.org/validator.html** **(https://icalendar.org/validator.html)** to check whether the HTTP response conforms to the iCalendar format.
  - Copy the body of the HTTP response message and paste the body to the validator.
  - The validator only checks the syntax, i.e., the format of the data.

Submission

You must submit the 13 files below:



A2AuthHandler.cs
A2Controller.cs
A2DbContext.cs
A2Repo.cs
CalendarOutputFormatter.cs
Event.cs
EventInput.cs
IA2Repo.cs
Organizer.cs
Program.cs
PurchaseOutput.cs
Sign.cs
User.cs

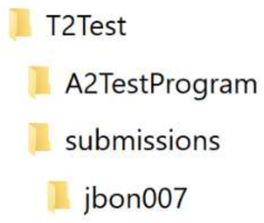
You MUST submit the files ONE BY ONE (i.e., do NOT pack the files into a single file). Submit the files through **adb.auckland.ac.nz. (https://adb.auckland.ac.nz/)**

Checking Your Submission

Create a folder, say A2Test.

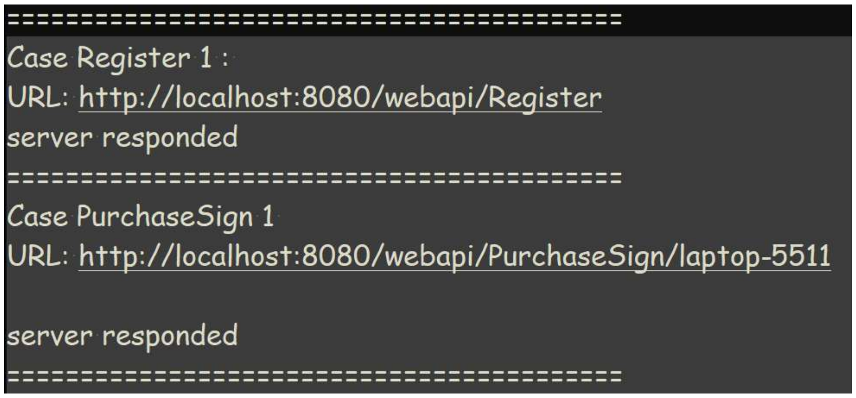
Download the test program **here (https://canvas.auckland.ac.nz/courses/104278/files/13692372/download)** to folder "A2Test". After unpacking it, you should have a folder “A2TestProgram”. I am afraid the program only works on Windows 11 platform as I have no access to any other platform. If you cannot run it on your machine, please use the lab machine to carry out the test.

Under folder “A2Test”, 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):

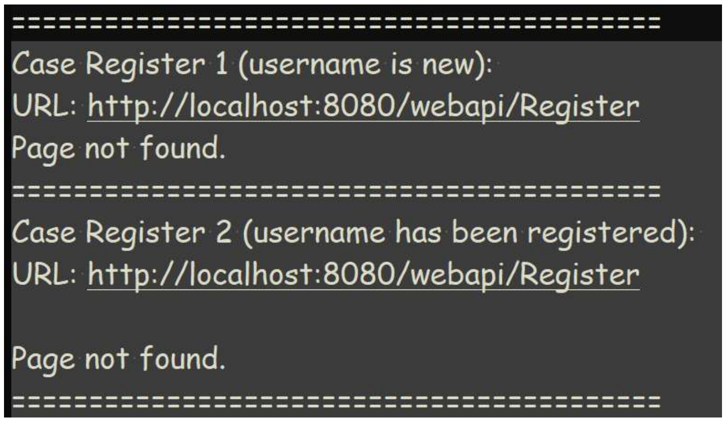


```
T2Test
  A2TestProgram
  submissions
    jbon007
```

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

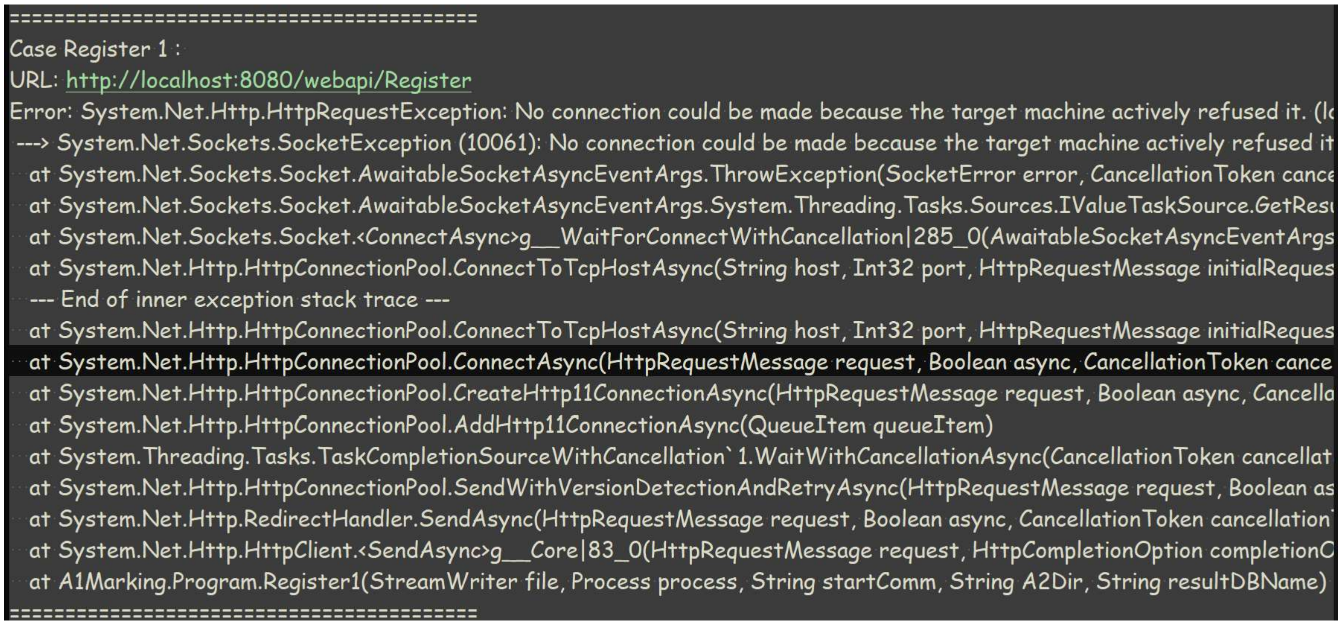
- Place the 13 files that you need to submit in folder jbon007.
- Open a command window. Use the “cd” command to change your current folder to folder “A2TestProgram”.
- Run the test program with the command below:
    - dotnet A2TestProgram.dll
- The test program writes the result to a file under the folder “results” which is under folder “A2Test”. 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 Register 1 :
URL: http://localhost:8080/webapi/Register
server responded
==========================================
Case PurchaseSign 1
URL: http://localhost:8080/webapi/PurchaseSign/laptop-5511

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 Register 1 (username is new):
URL: http://localhost:8080/webapi/Register
Page not found.
============================================
Case Register 2 (username has been registered):
URL: http://localhost:8080/webapi/Register

Page not found.
============================================
```

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

- 

```
========================================
Case Register 1 :
URL: http://localhost:8080/webapi/Register
Error: System.Net.Http.HttpRequestException: No connection could be made because the target machine actively refused it. (lo
 ---> System.Net.Sockets.SocketException (10061): No connection could be made because the target machine actively refused it
   at System.Net.Sockets.Socket.AwaitableSocketAsyncEventArgs.ThrowException(SocketError error, CancellationToken cance
   at System.Net.Sockets.Socket.AwaitableSocketAsyncEventArgs.System.Threading.Tasks.Sources.IValueTaskSource.GetResu
   at System.Net.Sockets.Socket.<ConnectAsync>g__WaitForConnectWithCancellation|285_0(AwaitableSocketAsyncEventArgs
   at System.Net.Http.HttpConnectionPool.ConnectToTcpHostAsync(String host, Int32 port, HttpRequestMessage initialReques
   --- End of inner exception stack trace ---
   at System.Net.Http.HttpConnectionPool.ConnectToTcpHostAsync(String host, Int32 port, HttpRequestMessage initialReques
   at System.Net.Http.HttpConnectionPool.ConnectAsync(HttpRequestMessage request, Boolean async, CancellationToken cance
   at System.Net.Http.HttpConnectionPool.CreateHttp11ConnectionAsync(HttpRequestMessage request, Boolean async, Cancella
   at System.Net.Http.HttpConnectionPool.AddHttp11ConnectionAsync(QueueItem queueItem)
   at System.Threading.Tasks.TaskCompletionSourceWithCancellation`1.WaitWithCancellationAsync(CancellationToken cancellat
   at System.Net.Http.HttpConnectionPool.SendWithVersionDetectionAndRetryAsync(HttpRequestMessage request, Boolean as
   at System.Net.Http.RedirectHandler.SendAsync(HttpRequestMessage request, Boolean async, CancellationToken cancellation
   at System.Net.Http.HttpClient.<SendAsync>g__Core|83_0(HttpRequestMessage request, HttpCompletionOption completionC
   at A1Marking.Program.Register1(StreamWriter file, Process process, String startComm, String A2Dir, String resultDBName)
========================================
```

- After running the test program, a "tmp" folder is created under folder "A2Test". This is the project created using your submitted files.

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. Many students have been 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	7
Submitting	Nothing

Due	For	Available from	Until
3 Sep 2024	Everyone	12 Aug 2024 at 8:00	-

+ Rubric