**Assignment 2 – Server-Side Scripting**

This assignment requires you to create a new ASP.NET MVC Web Application and integrate it with a SQL Server database to perform CRUD operations. Your mark counts for 25% of your final grade.

Due Dates:

Part 1: **Thursday, Nov 15 @ 5:00 pm**

Part 2: **Thursday, Nov 29 @ 5:00 pm**

<https://github.com/Honey23/AssignmentPart2>

<https://part1assignment.azurewebsites.net>

Part 3: **Thursday, Dec 6 @ 5:00 pm**

<https://github.com/Honey23/Part3>

<https://part3.azurewebsites.net>

**Submission Requirements via Blackboard**

**Part 1:**

* Link to your private Web Site repository (invite me – username **ifotn** - as a collaborator). This can be the same repository you used for Assignment 1.
* Link to your Assignment site running on Azure (or alternative server that supports .NET)

**Part 2:**

* Resubmit the same repository and Azure links you submitted in Part 1 as these should now be updated with the additions for Part 2.

**Part 3:**

* Link to a new private repository for your .NET Core Web API (invite me – username **ifotn** - as a collaborator).
* Link to your Web API running on Azure (or alternative server that supports .NET). This must be a different URL from your Web Site.

**All work must be your own.**  Failure to submit an independent assignment will result in a grade of zero. If your database tables are exactly the same (or very close to the same) as another student’s, it will potentially be considered as Academic Misconduct. Code will be checked carefully with code comparison software. **You are not permitted to download the MVC Music Store site or API, simply modify the words then submit these back to me as your own work.**

**All submissions are subject to a Code Review where you may be asked to walk through and explain your work.**

# Description: Dynamic ASP.NET MVC Site – Version 2

You will enhance the application you built for Assignment 1 by adding:

* Authentication – both Local and Social
* Unit Testing
* .NET Core Web API (in a separate project, repository, and live url)

**Application Requirements:**

**Part 1 (15 marks + 2 bonus)**

1. Ensure your connectionstring in web.config is called “DefaultConnection”
2. Add Register and Login links to your navbar that link to the auto-generated Account/Register and Account/Login views (unless these are already there)
3. Ensure ASP.NET Identity is working by using the Register link to create an account for yourself. This will create the 5 ASPNET tables in your database.
4. Once Authentication is working, add code to your navbar so that:
   1. Register and Login show when the user is anonymous
   2. Register and Login are replaced in the header by the username and Logout when the user is authenticated. You can use the code from our in-class MVC Music Store to enable this.
   3. Please create an account for me with these credentials:
      1. [rich@gc.ca](mailto:rich@gc.ca) / Abcd123$
5. Modify the site in the following ways:
   1. Make all Views where users can add, edit, or delete data PRIVATE, so only authenticated users can access them
   2. On your Index views, anonymous users can view the list of data but cannot see the Create, Edit, or Delete links
6. Enable Social Authentication with Google. Create new keys for your assignment rather than using your existing Music Store keys. Store all API Keys in your web.config file rather than inside your C# code.
7. Use GitHub to make regular commits (minimum 4) and create a README.md file that explains the purpose of your application. Use .gitignore so the Packages folder does not get included in your online repository.
8. Publish to the site to Azure or any other web server that supports ASP.NET and SQL Server. Include this link in your README.md file.
9. Bonus #1: Implement an additional Social login provider. (Part 1)

**Part 2 (11 marks)**

1. Create a Test Class for ONE of your Controllers and install Moq in your Unit Testing project.
2. Write Unit Tests for each method in ONE Controller to get as close as possible to 100% Code Coverage in this controller (roughly 23 tests), using Moq to create mock data. Use interfaces with your mock data and modify your controller so they use the mock data when testing but the live database in production.
3. Use GitHub to make regular commits (minimum 4) and create a README.md file that explains the purpose of your application. Use .gitignore so the   
   Packages folder does not get included in your online repository.
4. Publish to the site to Azure or any other web server that supports ASP.NET and SQL Server. Include this link in your README.md file.

**Part 3 (11 marks + 2 bonus)**

1. Create a new .NET Core Web API Solution
2. Create scaffolded Web API controllers with full Create-Read-Update-Delete functionality for both of the tables in your database
3. Use GitHub to make regular commits (minimum 4) and create a README.md file that explains the purpose of your API. Use .gitignore so the   
   Packages folder does not get included in your online repository.
4. Publish to the site to Azure or any other web server that supports ASP.NET and SQL Server. Include this link in your README.md file.
5. Bonus #2: Add an advanced .NET Core feature that we have not yet learned in class. You MUST document this feature in the README.md so I know what to look for. (Part 2)

**Evaluation Method**

Work is evaluated based on how your application performs on the following items:

# Evaluation Criteria

Part 1: 15 marks (2 bonus marks available)

Part 2: 11 marks

Part 3: 11 marks (2 bonus marks available)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **0-2** | **3-4** | **5-6** | **7-8** | **Marks** |
| **Authentication**  (Part 1) | No Authentication | Some implementation | Only minor omissions | Full, functional Auth Model with Working Local Registration & Login; Create, Edit, Delete for authenticated users only | **8** |
| Social Login (Part 1) | No Social Login | Some correct code but incomplete | 1 working provider | Google login fully functional | **4** |
| Unit Testing (Part 2) | No unit tests | Incomplete unit test implementation | Only minor changes outstanding | Functional Pass / Fail Tests for all Methods in **ONE** Controller using Mock Data | **8** |
| Web API (Part 3) | No API Project | API partly working | API working with minor errors or omissions | Complete Web API with working CRUD operations for both of your Controllers | **8** |
| Version Control (2 marks for each Part) | None | < 4 commits, incomplete README.md |  | Min 4 descriptively-named commits, plus clear README.md | **6** |
| Server Deployment (1 mark for each Part) | No deployment | Deployed with errors |  | Full deployed online. Link in README.md | **3** |
| **\*\*\* Bonus 1 \*\*\***  (Part 1) |  |  |  | Additional Social Login. Documented in README.md | **2** |
| **\*\*\* Bonus 2\*\*\***  (Part 2) |  |  |  | Up to 2 bonus marks for any extra working functionality of your choosing. Documented in README.md | **2** |