Project Brief

Before you begin, you must download the zipped file: **`CourseworkTwo.zip`** from Aula. Unzip this file into your local working directory.

You should have:

- A folder name **`CourseworkTwo`**

And inside this folder, you should have:

- A code workspace file named **`CourseworkTwo.code-workspace`**
- A copy of this project brief named **`CourseworkTwo.doc with a`** marking schedule.
- An SQLite Database named **`chinook.db`**
- An SQL file named **`chinook.sql`** – this file is for reference and contains the SQL commands used to generate the University database
- A database diagram named **`chinook.pdf`**
- A sub-folder named **`Project`**

Inside the **`project`** folder, develop a database driven web application that will allow a Chinook employee to manage the **Albums** the company has for sale. In particular, the Chinook employee should be able to view, insert, update and/or delete **Albums**.

While developing this application, you MUST limit yourself to the following technologies and frameworks: C#; and .net core 8. You may use Razor or Blazor. No credit will be given for submissions that make use of other technologies and frameworks.

This is to ensure that you use the .NET framework and NOT other server technologies such as PHP. You will still use web technologies HTML and others like CSS and JavaScript.

In addition to the web app, write a user manual for end-users with a non-technical background. The best user manuals illustrate the adage, *"show, don't tell!"* Place your user manual insider the **`project`** folder.

Place your entire solution into a zipped folder called your CourseworkTwoBannerID.

Before the submission deadline, zip up your entire workspace, and upload the resultant zipped file to the submission point on Aula.

Please carefully read the marking scheme which has a full breakdown of marks.

Marking Scheme

Web Development Skills (5 marks)
- *Is the HTML and CSS coding clean, neat, and tidy?*
- *Does the HTML and CSS coding pass validation?*

Scripting Skills (5 marks)
- *Is the coding clean, neat, and tidy?*
- *Is the coding error free?*
- *Is the coding well-documented?*

Functionality (40 marks)
- *Read*
  - *Can end-users view all the **albums**?*
  - *Can end-users order, filter, sort, and/or search the **albums**?*
  - *Can end users also view associated information such as **artists** and/or **tracks**?*
- *Create*
  - *Can end-users add new **albums**?*
  - *Is user input stringently validated?*
  - *Can end users also add associated information such as **artists** and/or **tracks**?*
- *Update*
  - *Can end-users update existing **albums**?*
  - *Is user input stringently validated?*
  - *Can end users also update associated information such as **artists** and/or **tracks**?*
- *Delete*
  - *Can end-users delete an **album**?*
  - *Is confirmation required before deletion?*
  - *Can end users also delete associated information such as the **artist** and/or **tracks**?*

Meeting the Brief (5 marks)
- *Is the database driven app fully functional?*
- *Is the database driven app simple and intuitive to use?*

Documentation (5 marks)
- *Does the user manual meet the needs of end-users?*
- *Does the user manual quickly show how the database driven app works?*