Tutorial 7 (Part A of Assignment 2)
Marks: 10
Due date: 15 October Sunday, 2023 23.59 Perth time

Title: Building a Bank Data Web Service in ASP.NET MVC Core with SQLite.
You are open to using any framework, i.e., ADO.NET or Entity Framework.

This is an individual/group assignment (a group of two or three members). If you decide to work in a group, the same group must work on part B of assignment 2.

Part C and Part D of Assignment 2 are strictly individual submissions.

Assignment Overview: In this assignment, you will be tasked with creating a Bank Data Web Service using ASP.NET MVC Core, utilising SQLite as the database for storing and managing banking-related information. Additionally, you will implement user profile management and populate the database with random data. This assignment will comprehensively understand web services, database integration, user management, and data seeding.

Task Description: You are required to build a Bank Data Web Service with the following functionalities:

1. **Account Management: [1]**
   - Create a new bank account.
   - Retrieve account details by account number.
   - Update account details (e.g., balance, account holder information).
   - Delete an existing bank account.
2. **Transaction Management: [1]**
   - Implement deposit and withdrawal operations.
   - Ensure that account balances are updated correctly after each transaction.
   - Store transaction history (for auditing purposes)
3. **User Profile Management: [1]**
   - Create user profiles with information such as name, email, address, phone, picture and password.
   - Retrieve user profiles by username or email.
   - Update user profiles.
   - Delete user profiles.
4. **Database Integration: [1]**
   - Use SQLite as the database for storing account information, transactions and user profiles.
   - Design an appropriate database schema for accounts, transactions, and user profiles. **Make sure they maintain relations and integrity**.
5. **Data Seeding: [4]**
   - Implement a data seeding script (automatic) to populate the database with random accounts, transaction data and user profiles for testing and development.
6. **API Design: [1]**
   - Design a RESTful API with appropriate endpoints and HTTP methods.
   - Test the API using tools like Postman to ensure all endpoints work correctly.

7. **Exception Handling: [1]**
    - Implement proper error handling and validation for API requests.
    - Return meaningful error responses.

Evaluation Criteria: The maximum mark is placed right after each requirement. In the marking criteria, there are six levels of implementation, i.e., zero attempts (0% -10% complete), novice (efforts shown – around 10%-30% are completed), promising (around 30%-50% are completed), intermediate (about 50%-70% of the requirements are achieved), competent (about 70%-90% of the requirements are achieved, and proficient (90% or above level are achieved).

Submission Guidelines:
Submit your assignment electronically, via Blackboard, before the deadline. **If you submit in a group, all the members must submit the same files.**
To submit, do the following:

1. Fill out and sign a declaration of originality. A photo, scan or electronically-filled
out form is fine. Whatever you do, ensure the form is complete and readable!
Place it (as a .pdf, .jpg or .png) inside your project directory.
2. Zip your entire project directory. Leave nothing out.
3. Submit your zip/tar.gz file to the assignment area on Blackboard.
4. Re-download, open, and run your submitted work to ensure it has been submitted correctly.

You are responsible for ensuring that your submission is correct and not corrupted. You may make multiple submissions, but only your newest submission will be marked. The late submission policy (see the Unit Outline) will be strictly enforced. Please note:

- DO NOT use WinRar.
- DO NOT have nested zip/tar.gz files. One is enough!
- DO NOT try to email your submission as an attachment. Curtin's email filters are configured to discard emails with potentially executable attachments silently. In an emergency, if you cannot upload your work to Blackboard, please instead upload it to Google Drive, a private GitHub repository, or another online service that preserves immutable timestamps and is not publicly accessible.

Marking Demonstration: You must demonstrate and discuss your application with a marker in a one-to-one online/in-person session. Most of the marks for your assignment will be derived from this demonstration. The demonstrator will ask you to rebuild and run your application (provided by the demonstrator) and demonstrate its major features. They may ask you about any aspect of your submission. **You will be asked about your contribution percentages during the demo. Your mark from the group submission will be adjusted based on your contribution percentage. You must specify your contribution in the individual report (check part D of the assignment).**

The demonstration schedule and policy will be published later (Check blackboard announcements). We may also cancel the demonstration-based marking due to unavoidable circumstances. In that case, it will be a full inspection-based marking.

Academic Integrity: This is an assessable task. If you use someone else's work or obtain someone else's assistance to help complete part of the assignment that is intended for you to complete yourself, you will have compromised the assessment. You will not receive marks for any parts of your submission that are not your original work.

Further, if you do not reference any external sources, you are committing plagiarism and collusion, and penalties for Academic Misconduct may apply. Please see Curtin's Academic Integrity website for information on academic misconduct (which includes plagiarism and collusion).

The unit coordinator may require you to provide an oral justification of or to answer questions about any piece of written work submitted in this unit. Your response(s) may be referred to as evidence in an Academic Misconduct inquiry.

End of PART A of Assignment 2