Tutorial 8 (Part B of Assignment 2)
Marks: 10
Due date: 15 October Sunday 2023, 23.59 Perth time

Title: Building a Bank Data Web Application using ASP.NET Core

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 A of assignment 2.

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

Project architecture: You will use the data web service built in tutorial 7. Create a separate ASP.NET core Web app (Model-view-controller) and call the data services using Restsharp or other libraries.

In this project, you are planning to separate it into two main layers: a business layer that consists of web services accessible through the "/api/" endpoint and a presentation layer or webpage accessible through the root ("/") endpoint. This separation follows the common architectural pattern known as a two-tier architecture.

Here's a more detailed explanation of how these layers will work together:

1. **Business Layer (Web Services):**
    - This layer handles all the business logic, data processing, and interaction with the database.
    - The "localhost:port/api/" endpoint will be the entry point for accessing these services.
    - The web services will expose various APIs (HTTP endpoints) that can perform actions like retrieving user account information, processing money transfers, fetching transaction history, and more.
    - These APIs will typically be implemented as RESTful services using technologies like ASP.NET Web API, allowing them to respond to HTTP requests with JSON or XML data.

2. **Presentation Layer (Webpage):**
    - The presentation layer is the user-facing part of your application, and users interact with it directly through their web browsers.
    - The root URL "localhost:port/" will be the entry point for users to access the application's user interface.
    - The webpage is typically built using HTML, CSS, and JavaScript. It represents the graphical user interface (GUI) through which users can perform actions and view data.
    - The HTML file (usually an index.html or similar) will be served when a user accesses "localhost:port/." This file will contain the webpage's structure, including placeholders for dynamic content.
    - The JavaScript files loaded within the webpage will play a crucial role. They will make asynchronous HTTP requests (using technologies like Fetch API or

Axios) to the business layer's API endpoints, sending and receiving data in JSON format.
    - These JavaScript files will handle user interactions, validate input, and dynamically update the web page's content based on the responses from the business layer. For example, when a user requests their transaction history, the JavaScript code will make an API call to fetch the data and then display it on the webpage without requiring a full-page refresh.
3. **Home Controller (Integration Point):**
    - The Home Controller in your ASP.NET Core application will play a key role in integrating these two layers.
    - It will render the initial webpage when a user accesses "localhost:port/." This webpage will include the necessary HTML structure and reference the JavaScript files.
    - The JavaScript code loaded by the webpage will communicate with the business layer's API endpoints through AJAX requests or similar techniques. These requests will be made to URLs like "localhost:port/api/endpoint-name" to access the relevant business functionalities.

The web application has two dashboards:

User dashboard:
The user dashboard in the bank data web application serves as the central hub where registered users can access and manage their financial information and perform various banking operations. It provides a user-friendly and secure interface to interact with their accounts. Here's a description of the typical components and functionalities:

1. **User Profile Information: [1]**
    - The user's name and profile picture (if available).
    - Contact information (email, phone number).
    - Options to update personal details such as password or contact information.
2. **Account Summary: [1]**
    - A summary of the user's account(s), including account number and current balance.
    - Quick access to view transaction history for each account.
3. **Transaction History: [1]**
    - A list of recent transactions for each account, including details like transaction date, description, and amount.
    - Filtering options to view older transactions using date range.
4. **Money Transfer: [2]**
    - A section to initiate money transfers between the user's own accounts or to other users.
    - Input fields for recipient account details (account number, amount, description).
    - Validation and error messages to ensure the accuracy of transfer requests.
5. **Security Features: [1]**
    - Log in and Log out button for securely ending the session.
    - The user can not access other user’s data.

Admin dashboard:

The admin dashboard in a bank data web application is a powerful interface designed specifically for bank administrators and staff responsible for overseeing and managing the system, user accounts, and financial transactions. It provides tools and insights to efficiently manage the bank's operations. Here's a description of the typical components and functionalities in an admin dashboard:

1. **Admin Profile Information: [1]**
    - The admin's name and profile picture (if available).
    - Contact information (email, phone number).
    - Options to update personal details such as password or contact information.
2. **User Management: [1]**
    - A section to manage user accounts, including creating, editing, or deactivating user accounts, resetting passwords.
    - The ability to search for users by name, account number, or other criteria.
3. **Transaction Management: [1]**
    - Access to view transaction records for all users, with options to search, filter, and sort transactions.
4. **Security and Access Control: [1]**
    - Logs and audit trails for tracking admin activities and system changes.

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 B of Assignment 2