**INSTRUCTIONS**

**Please read the instructions carefully before doing the questions.**

- You can use materials in your computer, notebook and text book.

- You are **NOT allowed** to use any device to share data with others.

Beside the above conditions, students must follow the following requirements:

**1. The work must complete by using Visual Studio 2022++**

**2. The Framework must be .NET 8.0**

**3. THIS PART IS VERY IMPORTANT, PLEASE READ IT CAREFULLY AND FOLLOW THE INSTRUCTIONS.**

- You are given a database script (.sql file) in Zip file. Execute the script before doing questions.
- **You must use the given solution.**
- **You are not allowed to add any more libraries via NuGet Package Manager into given solution.**
- **Submission Guideline:**

  Submit your work for each question separately. For each question, please:

  - Publish your project using the command:

    ```
    dotnet publish -c Release -o ./[QuestionNumber_StudentAccount]
    ```

    Example:

    ```
    dotnet publish -c Release -o ./Q1_trungnthe123432
    ```

  - Submit the root folder of the project into the PEA_Client application.
  - If the root folder of the project is too large, you may delete the following subfolders to reduce its size before submitting: /bin, /obj

**Just one of above requirements is violated, your work will be considered as invalid.**

**Question 1:**

## QUESTION 1

You are provided with a console application **BookTrackingServer** in given materials, acting as a TCP server, listening for connections at **127.0.0.1:4000**. The server stores information about readers, books, and the relationships between readers and books (borrow records). The server processes client requests as follows:

- The client sends an integer (Book ID) to the server.
- The server responds with a JSON string containing a list of readers who have borrowed the book with the given ID, including: Reader ID, Full Name, Email, Borrow Date, Return Date, and Status.

- If the Book ID does not exist in the books list, the server returns a special JSON response indicating "Book not found".
- If the Book ID exists but has no borrow records, the server returns a JSON containing an empty list.

**Your task** is to develop a **.NET 8 console application** that **acts as a client**, connects to the server, and implements the functionality described below.

The server manages the following data structures:

```
Readers Table:
- ReaderID (int) - Primary Key
- FullName (string)
- Email (string)
- PhoneNumber (string)
- RegistrationDate (DateTime)

Books Table:
- BookID (int) - Primary Key
- Title (string)
- Author (string)
- ISBN (string)
- PublishYear (int)
- Category (string)

BorrowRecords Table:
- RecordID (int) - Primary Key
- ReaderID (int) - Foreign Key
- BookID (int) - Foreign Key
- BorrowDate (DateTime)
- ReturnDate (DateTime?) - Nullable
- Status (string) - "Borrowed", "Returned", "Overdue"
```

**Note:** The server will be provided and running on port 4000. You only need to focus on developing the client application.

The client application must:

## 1. Wait for User Input:

- Continuously prompt the user to enter an integer (Book ID) via the console.
- If the user presses Enter (empty input), the program terminates.

## 2. Validate Input:

- Check if the user input is a valid integer greater than 0.
- If the input is not a valid integer or not greater than 0, display an error message and prompt the user to try again.

**Examples of invalid input:**

```
Enter Book ID (or press Enter to exit): abc
Invalid input! Please enter a valid Book ID (positive integer).
```

```
Enter Book ID (or press Enter to exit): -10
Invalid input! Please enter a valid Book ID (positive integer).

Enter Book ID (or press Enter to exit): 0
Invalid input! Please enter a valid Book ID (positive integer).
```

## 3. Connect to the Server:

- Establish a TCP connection to the server at **127.0.0.1:4000**.
- Send the integer (Book ID) entered by the user to the server.
- If the server is not running or the connection fails, display an error message (e.g., "Book tracking server is not running. Please try again later.") and return to waiting for new user input.

**Example connection failure:**

```
Enter Book ID (or press Enter to exit): 1001
Book tracking server is not running. Please try again later.
```

## 4. Process Server Response:

- Receive the JSON string from the server and deserialize it.
- The server can return three types of responses:
    1. **Book not found**: Display "Book with ID {ID} does not exist."
    2. **Book exists with borrower records**: Display each borrower's information in a clear format
    3. **Book exists with no borrower records**: Display "No borrower records found for Book ID {ID}."

**Example - Book does not exist:**

```
Enter Book ID (or press Enter to exit): 9999
Book with ID 9999 does not exist.
```

**Example - Book exists but no borrower records:**

```
Enter Book ID (or press Enter to exit): 1005
No borrower records found for Book ID 1005.
```

**Example - Book exists with borrower records:**

```
Enter Book ID (or press Enter to exit): 1001
=== Borrower History for Book ID: 1001
Reader ID: 101
Full Name: John Smith
Email: john@email.com
Borrow Date: 2024-01-15
Return Date: 2024-02-15
Status: Returned
---
Reader ID: 103
Full Name: Bob Johnson
```

```
Email: bob@email.com
Borrow Date: 2024-04-01
Return Date: Not returned yet
Status: Borrowed
---
Reader ID: 102
Full Name: Jane Doe
Email: jane@email.com
Borrow Date: 2024-02-10
Return Date: 2024-03-15
Status: Overdue
```

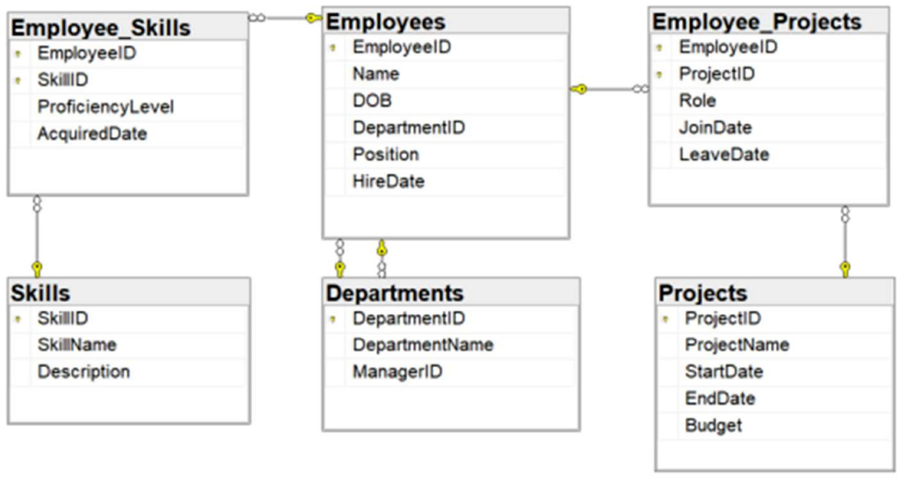
- After processing the response, return to waiting for new user input unless the user has entered an empty string.

**Example program termination:**

```
Enter Book ID (or press Enter to exit):
Goodbye! Book tracking client is shutting down.
```

## Question 2:

In this question, you are asked to write MVC/ Razor Pages model, using the database that has following schema:

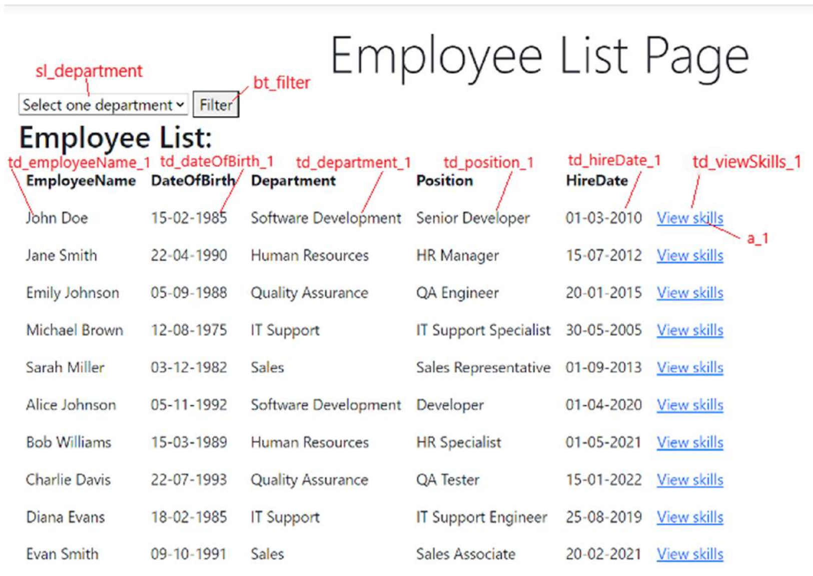


1. **Note:**

   - It is required to assign the URL to access the web page as given. Wrong URL will result in the examiner cannot access your developed features. You will receive ZERO for any WRONG URL.

   - Zero marks will be awarded if the database connection string is not stored in the file **appsettings.json**.

- All input and output elements in the HTML source **must** have an **'id'** attribute to ensure accessibility and traceability (Students can refer to the **sample2.html** file in given materials, which provides code snippets for the assignment).



Employee List Page

Employee List:

| EmployeeName | DateOfBirth | Department | Position | HireDate | |
|---|---|---|---|---|---|
| John Doe | 15-02-1985 | Software Development | Senior Developer | 01-03-2010 | View skills |
| Jane Smith | 22-04-1990 | Human Resources | HR Manager | 15-07-2012 | View skills |
| Emily Johnson | 05-09-1988 | Quality Assurance | QA Engineer | 20-01-2015 | View skills |
| Michael Brown | 12-08-1975 | IT Support | IT Support Specialist | 30-05-2005 | View skills |
| Sarah Miller | 03-12-1982 | Sales | Sales Representative | 01-09-2013 | View skills |
| Alice Johnson | 05-11-1992 | Software Development | Developer | 01-04-2020 | View skills |
| Bob Williams | 15-03-1989 | Human Resources | HR Specialist | 01-05-2021 | View skills |
| Charlie Davis | 22-07-1993 | Quality Assurance | QA Tester | 15-01-2022 | View skills |
| Diana Evans | 18-02-1985 | IT Support | IT Support Engineer | 25-08-2019 | View skills |
| Evan Smith | 09-10-1991 | Sales | Sales Associate | 20-02-2021 | View skills |

*Figure 1 – Required page*

## 2. Requirements

- The web application has one page at url **/Employee/List** and is designed as shown in Figure 1.

- Use display format **"dd/MM/yyyy"** for all date information.

**2.1.** When user access this page for the first time, list all departments from the database into the Dropdownlist (like Figure 2) and all employees (like Figure 1).

- Each **<td>** tag in table that list employees will have an id in the form **td_{columnName}_{employeeId}** (where **employeeId** is the id in database of employee and **columnName** write with camel case format). (see in Figure 1). Example: **td_employeeName_1**

- The links "View skills" use tag **<a>** with id **a_{employeeId}**. Each tag **<a>** is placed in tag **<td>** with id **td_{viewSkills}_{employeeId}**

- The Dropdownlist using **<select>** tag that has id **sl_department**.

- The option "Select one department" is set by default, using **<option>** tag and has id is "**op_0**"

- Each option tag in Dropdownlist have an id in the form **op_{departmentId}** (where **departmentId** is the id in database of department). Example: **op_1**

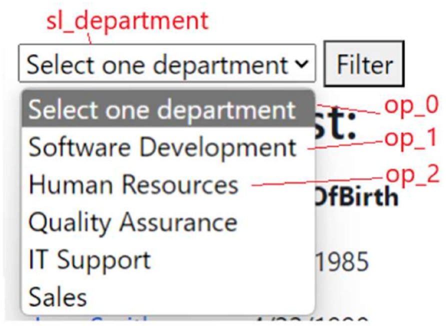
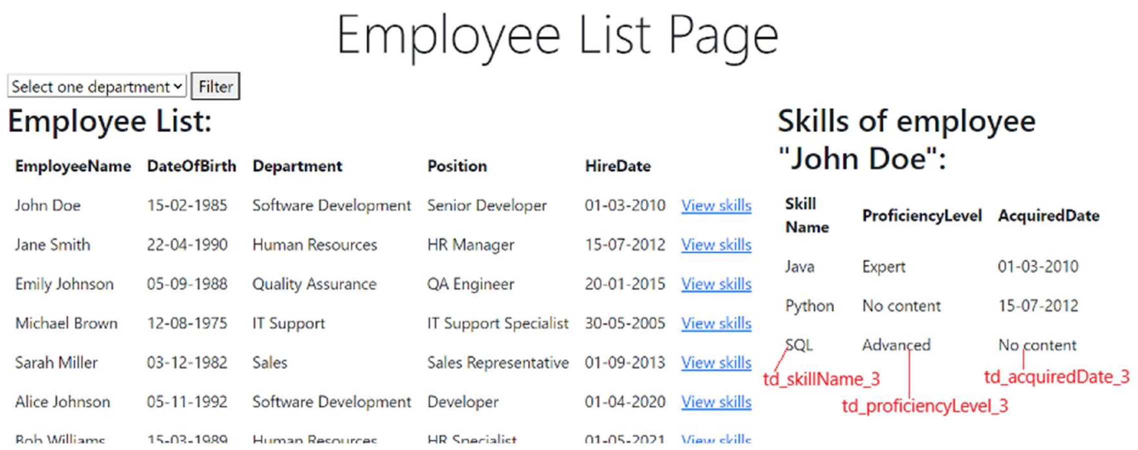


Figure 2 – The Dropdownlist [Departments].

**2.2.** User can use the Dropdownlist and [Filter] button to filter employees by department. The option "Select one department" represents all departments. The [Filter] button using **<input>** tag with id **bt_filter**.

**2.3.** When the user clicks on the link "View skills" of an employee, please display information about all the skills of this employee, on the left side of the current webpage, like Figure 3 below. Note that, if ProficiencyLevel or AccquiredDate is null, show the message "No content".



Employee List Page

Employee List:

| EmployeeName | DateOfBirth | Department | Position | HireDate | |
|---|---|---|---|---|---|
| John Doe | 15-02-1985 | Software Development | Senior Developer | 01-03-2010 | View skills |
| Jane Smith | 22-04-1990 | Human Resources | HR Manager | 15-07-2012 | View skills |
| Emily Johnson | 05-09-1988 | Quality Assurance | QA Engineer | 20-01-2015 | View skills |
| Michael Brown | 12-08-1975 | IT Support | IT Support Specialist | 30-05-2005 | View skills |
| Sarah Miller | 03-12-1982 | Sales | Sales Representative | 01-09-2013 | View skills |
| Alice Johnson | 05-11-1992 | Software Development | Developer | 01-04-2020 | View skills |
| Rob Williams | 15-03-1989 | Human Resources | HR Specialist | 01-05-2021 | View skills |

Skills of employee "John Doe":

| Skill Name | ProficiencyLevel | AcquiredDate |
|---|---|---|
| Java | Expert | 01-03-2010 |
| Python | No content | 15-07-2012 |
| SQL | Advanced | No content |

Figure 3 – The page after user clicks on "View skills" of employee "John Doe".

- Each **<td>** tag in table that list skills will have an id in the form **td_{columnName}_{skillId}** (where **skillId** is the id in database of skill and **columnName** write with camel case format). (see in Figure 3).

## 3. Summary of HTML Elements ID

The HTML id requirements are summarized in the table below.

| **Element** | **Element Tag** | **Id** |
|---|---|---|
| Each cell in the table that list employees | <td> | td_{columnName}_{employeeId} |

| | | |
|---|---|---|
| Each "View Skills" link | <a> | a_{employeeId} |
| Each cell in the table that list skills | <td> | td_{columnName}_{skillId} |
| DropdownList Department | <select> | sl_department |
| Option "Select one department" in DropdownList | <option> | op_0 |
| Each option in the DropdownList | <op> | op_{departmentId} |
| Button Filter | <input> | bt_filter |