**Assessment 1 – Computer Base Activities**

***Instruction:***

Trainer may decide whether this will be an individual or group assessment.

You will be developing different object oriented designs/diagrams with ASP.NET.

Source code with screen shots and other required documents specified in the “Tasks” section below need to be submitted to the Trainer.

***Software Requirements:***

|  |  |
| --- | --- |
| Operating System: | Windows 7/XP/Vista |
| Front End Software: | Microsoft Visual studio 2010 |
| Database: | Microsoft SQL Server 2010 or MS. Access |

***Hardware Requirements:***

|  |  |  |
| --- | --- | --- |
| Processor: |  | Pentium 4.0(1.6 GHz) and Higher |
| Memory: |  | 512 MB |
| Hard Disk: |  | 10 GB |

You must successfully complete all the tasks. If you are not sure about any aspect of this assessment, please ask your Trainer for further clarification.

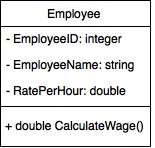
***Duration:***

Trainer will set the duration of the assessment.

**Your tasks:**

# Task 1

Use the following class diagram to develop Employee class



# Task 2

Use class develop from task 1 to implement:

1. Create a web page using ASP.NET that take 2 input, Employee ID and number of hour working per week, and then calculate and display the wages they earned for that week.

# Task 3

Add a function that calculate employee tax payable at 19.20% and display the result of tax payable in a label.

# Task 4

Expand your implementation in task 1 as below:

1. Create new class call Technician, which inherit from Employee. Technicians have their own ACS membership registration number and it has to record when it’ll expire.
2. Create new class call Accountant, which inherit from Employee. Accountants have their own lifetime CPA registration number.
3. Each class has a function to display their unique information as following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Rate per hour** | **Registration No.** | **Expire Date** |
| 1 | John | 39.0 | 311175 | 1/4/2016 |
| 2 | Tom | 45.5 | AC0001 | None |

# Task 5-Case Study

WIC is a popular school in **NSW**. It has got branches in **five different suburbs of Sydney** and planning to establish its branches in other states of Australia. The school is planning to have head office in Sydney City Branch which is one among its branch in NSW. All the other branches will be under the control of this **Sydney City Branch**.

The management decides to have the control of **database in the head office**, after a board meeting with all the stakeholders of the school they are deciding to connect the entire schools database through **server and create a networking that connects** the **entire schools** administration department.

Before they create a network to link the entire schools database, they have to start with building application that can interact. To achieve this goal they are looking for an IT expert/company that can help them achieve their goal.

**Summary of System Requirement Specification (SRS):**

The WIC database server is designed to run on the departmental server and to allow staffs create a new database entry, update an existing database entry and other related entries. The data will be held in an **SQL database on the departmental server**.

With summary of SRS given above create a detailed SRS and describe the methods to validate the system as well.

Your SRS should be clear, short (approx. 500 words) and easy to understand. The SRS may contain:

* Introduction

This report stated the summary of system requirement specification on the SQL database which soon will be used in WIC. The goal of the database is to centralize the data entries of five WIC branches located in NSW. The resulted database should allow staff to create, update, read, delete and search through database entries.

* Scope

The database will be held in the department server inSydney city branch while allowing access from other branches based in NSW. For the demonstration purpose, a demo of the database will be hosted online giving a better picture of the UI element design and how the interaction will look like.

* System Environment

The project would be accessible on any operation system since it was hosted online and runs as a web application. It will require a stable internet connection and modern web browser with JavaScript enabled to handling data exchange. (More details in Glossary section)

* Functional requirement

The project would allow staff to:

1. Create new employee details
2. Read employees details
3. Update employee details
4. Delete employee details
5. Search through database by employee’s name or employee ID
6. Calculate employee wage by entering number of hours that employee had worked for the week
7. Display tax payable in the calculated result

* Non-functional requirement

1. Provide a user interface design
2. Build a prototype based on the interface designed in point 1
3. User interface should have icons to indicate the interaction elements in the application

* Glossary

In the hope of reducing server payload and saving time for staff from navigating to different routes to perform data entry, it will be a **single page application** using **web API** handling data exchange.

The project was written in **C#**. An open source and cross-platform web development framework, **ASP.NET Core Blazor**, was chosen for the task because it provides a fast, lean and modular environment which fit the needs of the project.

**Entity Framework Core** was used to setup SQL database and configure database access.

Data validation for both frontend and backend were achieved by **DataAnnotation**.

In the future, an authentication method should be implemented in the project to prevent unauthorized access. It could be a simple login with username and password. Therefore, only staff selected can interact with the database, protecting company confidential information and maintain data integrity.

# Task 6-Prototype

Proof of concept by using Microsoft Word Shape or create your diagram prototype in draw.io Your prototype should demonstrate the following element:

* Database relationship (ER-Diagram)

A screenshot of a cell phone

Description automatically generated

* User Interface design

A screenshot of a cell phone

Description automatically generated

* Document the risk associated with the system

The system is lack of security feature that prevent unauthorized access. There would be chance of random visitor gaining access to the form and post invalid data to the database. The solution would be implementing a kind of authentication allowing only certain members of the company to have access of the form and the web API. It could be an IP whitelisting and/or login system.

***Task 7-Test Plan and Script***

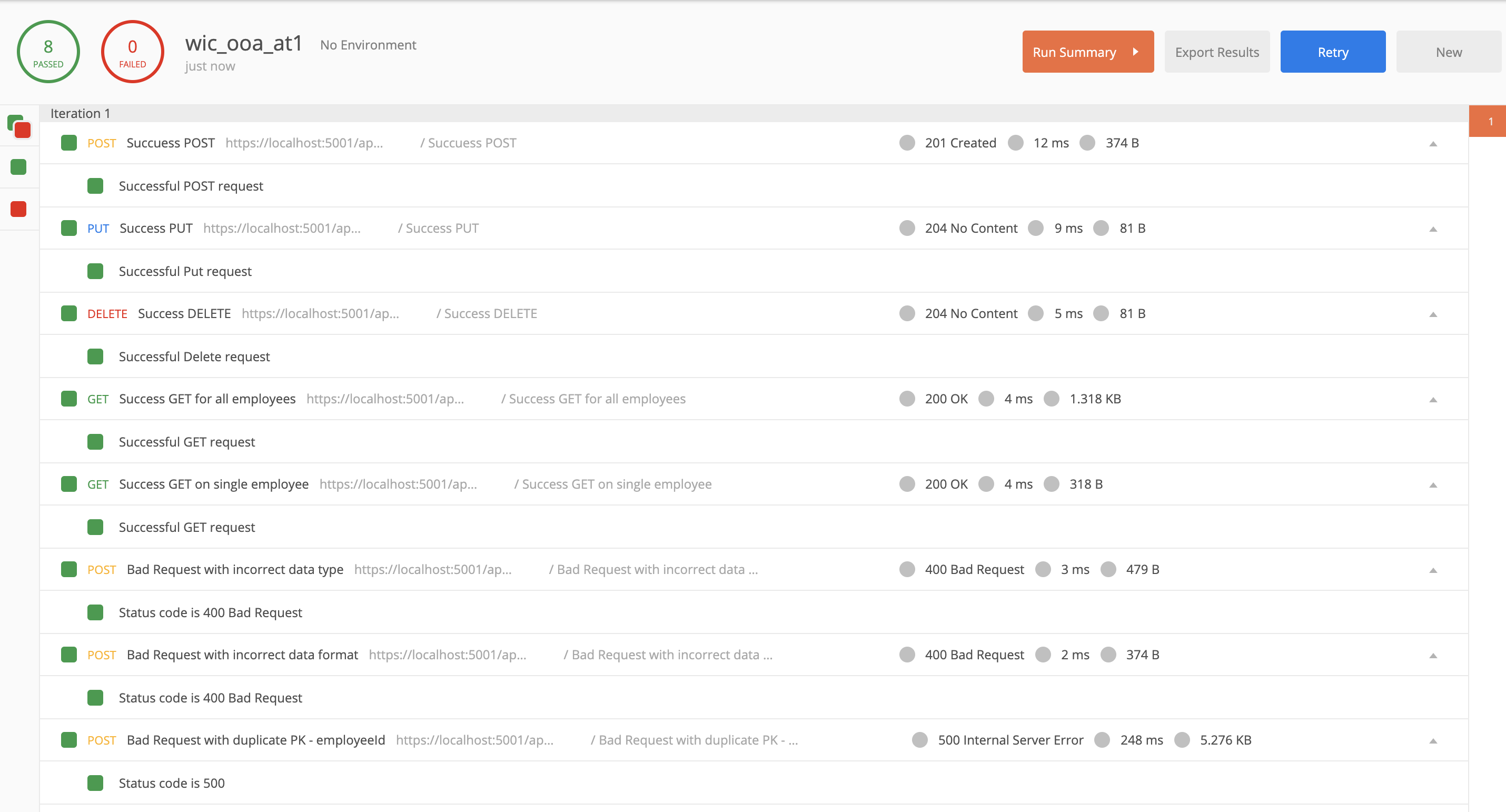
Create test plan and test script to validate your system

# Task 8

* Using MS SQL Server or MS Access create table as designed in Task 6
* Create Graphic User Interface (GUI) as designed in Task 6. Your GUI should be able to perform the following function: o Search membership record o Edit employee membership record o Delete employee record when they resign o Create new employee record
* Using dummy data, test your application against the test script was created in Task 7

Postman collection was included in Documentation folder.

* Submit your screen shot of debugging
* Submit your screen shot of test result



# Task 9

Now that you are ready with your software, in the Software design document suggest the requirements for installation of your software for both client and server systems. You have to develop software installation plan with the outline below:

* Installation plan

The project will be deployed online and provide access on people who know the website address. Domain name will be later added to the site.

* Back out plan

Please check the source of GitHub folder and contact technician team in WIC. Hosted platform should automatically try to re-deploy the project once the server is ready.

* Fall back plan

The project using a version control, git, to provide a fall-back safety net. Git records changes that developer commit. If some of the newly implemented feature conflict with the old code. It provides a safe way to revert back to the older version of the old and fix the conflicted part. You can also create different branch that is stand alone. Making sure it is safe to deploy after testing. Then merge it to the master branch to deploy online.