Portfolio of Evidence (PoE) — Background

Welcome to the Portfolio of Evidence (PoE) PROG6212, where you will embark on a transformative journey of developing a practical .NET web-based application known as the **Contract Monthly Claim System (CMCS)**. This system serves as a crucial tool for streamlining the often-complex process of submitting and approving monthly claims for **Independent Contractor (IC)** lecturers, offering a glimpse into real-world scenarios encountered in professional settings. As a student in this module, you will delve into the development of .NET GUI applications, leveraging the power of C# to create an interactive user interface and enhance the overall user experience for ICs. Throughout your learning journey, you will be guided step-by-step in designing and implementing the Monthly Claim System, honing your skills through hands-on practice and theoretical understanding.

Within the Contract Monthly Claim System, the role of a lecturer extends beyond merely submitting claims; it involves complex calculations based on hours worked and corresponding hourly rates. These claims undergo thorough scrutiny by both the Programme Coordinator and the Academic Manager, highlighting the importance of accuracy and accountability in administrative processes. Furthermore, the system's integration of features will go beyond basic claim submissions, providing a seamless platform for uploading essential supporting documents. By facilitating these functionalities, the system aims not only to increase efficiency but also to enhance user satisfaction and mitigate potential errors.

As you progress through this module, you will delve into the different aspects of .NET GUI development, from designing visually appealing interfaces to implementing robust functionality. Each Task or Assessment part will serve as a pivotal milestone, offering opportunities to apply theoretical knowledge to practical scenarios. Through iterative learning and hands-on projects, you will gradually master the art of GUI development using C# .NET Core, gaining valuable insights into industry best practices and methodologies.

The Contract Monthly Claim System stands as a testament to innovation in administrative processes, offering a glimpse into the future of streamlined claim management. With its user-centric design and seamless integration of features, the system aims to revolutionise the way claims are processed and approved. Automating repetitive tasks and providing intuitive interfaces empower both lecturers and administrators to focus on more strategic initiatives, ultimately enhancing organisational efficiency and productivity.

In conclusion, this POE not only equips you with the technical skills needed for GUI development but also instils a deeper understanding of the underlying principles driving modern software applications. Through hands-on experience and guided instruction, you will emerge as a proficient C# developer, ready to tackle real-world challenges in the dynamic landscape of software development.

Portfolio Of Evidence (POE) Objective:

The objective of this Portfolio of Evidence (POE) is to assess your understanding and practical application of C# GUI development in a real-world scenario. You will be developing a .NET web-based application called the Contract Monthly Claim System (CMCS), which is designed to streamline the process of submitting and approving monthly claims for independent contractor lecturers. This POE is divided into three parts, each focusing on different aspects of the system development.

Introduction

Complete the parts below to provide all the information and the prototype required for the POE.

Tip: Read the rubrics at the end of this document for details on how your work will be evaluated.

Part 1 — Project Planning and Prototype Development (Marks: 100)

In this part, you are required to design a prototype of the Contract Monthly Claim System. Your prototype should include a **Unified Modelling Language (UML) class diagram** for databases, a **project plan**, and a Windows Presentation Foundation (WPF) or Model-View-Controller (MVC) using .NET Core for the graphical user interface (GUI). ***Please note that the application should not be functional at this stage***.

1. Documentation:

- Provide a detailed explanation of your design choices, the structure of your database, and the layout of your GUI.
- Include any assumptions or constraints you have considered.

This will help us understand your thought process and the rationale behind your design decisions.

2. UML Class Diagram for Databases:

- Design a UML class diagram that accurately represents the data requirements of the Contract Monthly Claim System. Your diagram should include all necessary classes, attributes, and relationships and show how they are represented in a database.

3. Project Plan:

- Develop a project plan that outlines the tasks, dependencies, and timeline for developing the prototype. Your plan should be realistic and achievable.

4. GUI/UI:

- Design the user interface for the Contract Monthly Claim System using either MVC or WPF (.NET Core). Your design should be user-friendly and intuitive.

The GUI at this stage should only be a front-end prototype with the following options:

- Lecturers can submit their claims at any time with a click of a button.
- Programme Coordinators and Academic Managers can easily verify and approve the claims.
- Lecturers can upload supporting documents for their claims. The claim status can be tracked transparently until it is settled.
- The system always provides consistent and reliable information.

5. Version Control: Regularly commit and push changes to the GitHub repository (5 Times) with clear and descriptive commit messages.

Remember, the GUI at this stage should not be functional. It should only provide a visual representation of the proposed system. The functionality will be added in the subsequent parts of the POE.

Submission Guidelines:

- Submit a report that includes all your documentation, the UML class diagram, the project plan, and the GUI design. The report should be 400 to 500 words long, well-structured, clear, and concise.
- Format your report as a Microsoft Word document.
- Version Control: Push your source code and your Documentation to GitHub. Repository to be provided.

Part 2 — Implement a Prototype Web Application (Marks: 100)

Instructions

Building on the prototype from Part 1, you will now **add functionalities** to the GUI UI .NET Core web application. The application should be able to perform the following features:

1. Lecturers can submit their claims at any time with a click of a button:

- Implement this feature in your application.
  - Consider the layout, colour scheme, and user flow to make this process as straightforward as possible.
- You should design a simple and intuitive form for lecturers to input their claims.
- The form should include fields for the hours worked, hourly rate, and any additional notes.
- The 'Submit' button should be prominently displayed and easy to click.

2. Programme Coordinators and Academic Managers can easily verify and approve the claims:

- Design a separate view for coordinators and managers.
  - This view should display all pending claims and provide options to verify or reject them.
  - Each claim should be displayed in a clear and organised manner, showing all the necessary details for verification.
  - There should be 'Approve' and 'Reject' buttons for each claim.

3. Lecturers can upload supporting documents for their claims:

- Add a feature that allows lecturers to upload documents.
  - Ensure that the uploaded files are securely stored and linked to the corresponding claim.
  - You should provide an 'Upload' button in the claim submission form.
  - Once a file is uploaded, its name should be displayed on the form.
  - Consider implementing a file size limit and restricting the file types to common formats like .pdf, .docx, and .xlsx.

4. The claim status can be tracked transparently until it is settled:

- Implement a tracking system that updates the status of each claim as it moves through the approval process.
- You could represent the status as a simple text label (e.g., 'Pending', 'Approved', 'Rejected') or as a progress bar.
- The status should be updated in real-time whenever a coordinator or manager approves or rejects a claim.

5. The system always provides consistent and reliable information:

- Unit Testing: Write unit tests for the code. These tests should cover all the key functionalities of the system.
- Ensure that your application handles errors gracefully and displays accurate information. Implement error handling mechanisms to catch and handle exceptions. Display meaningful error messages to the user when an error occurs.

6. Version Control: Regularly commit and push changes to the GitHub repository (5 Times) with clear and descriptive commit messages.

Remember, the goal of Part 2 is to demonstrate your ability to add functionality to a GUI application. Focus on implementing the features as described, but also feel free to add any additional features that you think would improve the application.

Submission Guidelines:

- Add Lecturer Feedback in a Word document and show how you implemented the recommendations.
- Version Control: Push your source code and your Documentation to GitHub. Repository to be provided.

POE — Automation of Web Application (Marks: 100)

For the final part of the POE, you will enhance the functionality of the application developed in Part 2 and prepare a PowerPoint presentation to showcase your work. This presentation should provide a comprehensive overview of the Contract Monthly Claim System, highlighting its features, functionality, and benefits.

1. Application Enhancement (Automation): Implement additional features or improvements to enhance the overall functionality and user experience of the system.

Automation of Features:

Lecturer view: Automate the claim submission process, allowing lecturers to easily input their hours worked and hourly rate, and submit claims.

- Automation: Implement an auto-calculation feature to compute the final payment based on the hours worked and the hourly rate input by the lecturer. Additionally, integrate validation checks to ensure accurate data entry.
- Tools in C# ASP.NET: Build the web application using ASP.NET MVC or ASP.NET Core MVC. Leverage JavaScript libraries like jQuery for client-side calculations and validations. The Entity Framework can be used to interact with the database to store and retrieve claim data.

Programme Coordinator and Academic Manager view: Automate claim verification and approval processes, enabling efficient review and processing of submitted claims.

- Automation: Develop an automated system to check submitted claims against predefined criteria such as hours worked, hourly rates, and any other relevant policies. Implement approval workflows to streamline the verification and approval process.
- Tools in C# ASP.NET: Use ASP.NET Identity for user authentication and authorisation. Implement ASP.NET Web API to handle communication between the front-end and back-end systems. Entity Framework can be utilised to query and manipulate data in the database. Consider using workflow management tools, such as Windows Workflow Foundation, or third-party libraries like FluentValidation, to define and execute approval workflows.

HR view: Automate claim processing and lecturer data management tasks, streamlining administrative processes and improving overall efficiency.

- Automation: Develop functionality to automatically generate invoices or reports summarising approved claims for payment processing. Implement features for managing lecturer data, such as updating personal information or contact details.
- Tools in C# ASP.NET: Utilise ASP.NET Web Forms or ASP.NET Core Razor Pages for building the HR interface. Integrate reporting libraries like Crystal Reports, SQL Server Reporting Services (SSRS), or LINQ to generate invoices or reports. Entity Framework can be used for data access operations, while ASP.NET Identity can handle user authentication and authorisation.

2. PowerPoint Presentation: Create a visually appealing and informative presentation to showcase your application. Ensure that all key aspects of the Contract Monthly Claim System are covered and that its value is effectively communicated.

3. Version Control: Regularly commit and push changes to the GitHub repository (10 Times) with clear and descriptive commit messages.

4. Submission Guidelines:

- Add Lecturer Feedback in a Word document and show how you implemented the recommendations.
- PowerPoint Presentation to showcase your application.
- Version Control: Push your source code and your Documentation to GitHub. Repository to be provided.