MODULE NAME:	MODULE CODE:
PROGRAMMING 2B	PROG6212

ASSESSMENT TYPE: POE (PAPER)

TOTAL MARK ALLOCATION: 300 MARKS

TOTAL HOURS: A minimum of 45 HOURS is suggested to complete this assessment

By submitting this assignment, you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity and Property Rights Policy (IIE023), as well as any rules and regulations published in the student portal.

INSTRUCTIONS:

1. ***No material may be copied from original sources, even if referenced correctly, unless it is a direct quote indicated with quotation marks. No more than 10% of the assignment may consist of direct quotes.***
2. ***Please ensure that you submit your assignment through SafeAssign. Please make sure you attach a similarity report to your POE if you are required to submit a hard-copy of your PoE.***
3. ***Make a copy of your assignment before handing it in.***
4. *Assignments must be typed unless otherwise specified.*
5. *Begin each section on a new page.*
6. *Follow all instructions on the PoE cover sheet.*
7. *This is an individual assignment.*

Referencing Rubric

Providing evidence based on valid and referenced academic sources is a fundamental educational principle and the cornerstone of high-quality academic work. Hence, The IIE considers it essential to develop the referencing skills of our students in our commitment to achieve high academic standards. Part of achieving these high standards is referencing in a way that is consistent, technically correct and congruent. This is not plagiarism, which is handled differently.

Poor quality formatting in your referencing will result in a penalty **of** according to the following guidelines **a maximum of ten percent being deducted from the overall percentage**. Please note, however, that **evidence of plagiarism in the form of copied or uncited work (not referenced), absent reference lists, or exceptionally poor referencing, may result in action being taken in accordance with The IIE's Intellectual Integrity Policy (0023)**.

Markers are required to provide feedback to students by indicating **(circling/underlining) the information that best describes the student's work.**

Minor technical referencing errors: 5% deduction from the overall percentage. – the student's work contains **five or more errors** listed in the minor errors column in the table below.

Major technical referencing errors: 10% deduction from the overall percentage. – the student's work contains **five or more errors** listed in the major errors column in the table below**.**

If both minor and major errors are indicated, **then 10% only** (and not 5% or 15%) is deducted from the overall percentage. The examples provided below are not exhaustive but are provided to illustrate the error.

Required: **Technically correct referencing style**	**Minor errors in technical correctness of referencing style** **Deduct 5% from overall percentage.** **Example: if the response receives 70%, deduct 5%. The final mark is 65%.**	**Major errors in technical correctness of referencing style** **Deduct 10% from the overall percentage.** **Example: if the response receives 70%, deduct 10%. The final mark is 60%.**
Consistency • The same referencing format has been used for all in-text references and in the bibliography/reference list.	**Minor inconsistencies.** • The referencing style is generally consistent, but there are one or two changes in the format of in-text referencing and/or in the bibliography. • For example, page numbers for direct quotes (in-text) have been provided for one source, but not in another instance. Two book chapters (bibliography) have been referenced in the bibliography in two different formats.	**Major inconsistencies.** • Poor and inconsistent referencing style used in-text and/or in the bibliography/ reference list. • Multiple formats for the same type of referencing have been used. • For example, the format for direct quotes (in-text) and/or book chapters (bibliography/ reference list) is different across multiple instances.
Technical correctness • Referencing format is technically correct throughout the submission. • The correct referencing format for the **discipline** has been used, i.e., either **APA,** OR **Harvard** OR **Law** • Position of the reference: a reference is directly associated with every concept or idea. • For example, quotation marks, page numbers, years, etc. are applied correctly, sources in the bibliography/reference list are correctly presented.	**Generally, technically correct with some minor errors.** • The correct referencing format has been consistently used, but there are one or two errors. • Concepts and ideas are typically referenced, but a reference is missing from one small section of the work. • Position of the references: references are only given at the beginning or end of every paragraph. • For example, the student has incorrectly presented direct quotes (in-text) and/or book chapters (bibliography/reference list).	**Technically incorrect.** • The referencing format is incorrect. • Concepts and ideas are typically referenced, but a reference is missing from small sections of the work. • Position of the references: references are only given at the beginning or end of large sections of work. • For example, incorrect author information is provided, no year of publication is provided, quotation marks and/or page numbers for direct quotes missing, page numbers are provided for paraphrased material, the incorrect punctuation is used (in-text); the bibliography/reference list is not in alphabetical order, the incorrect format for a book chapter/journal article is used, information is missing e.g. no place of publication had been provided (bibliography); repeated sources on the reference list.
Congruence between in-text referencing and bibliography/ reference list • All sources are accurately reflected and are all accurately included in the bibliography/ reference list.	**Generally, congruence between the in-text referencing and the bibliography/ reference list with one or two errors.** • There is largely a match between the sources presented in-text and the bibliography. • For example, a source appears in the text, but not in the bibliography/ reference list or vice versa.	**A lack of congruence between the in-text referencing and the bibliography.** • No relationship/several incongruencies between the in-text referencing and the bibliography/reference list. • For example, sources are included in-text, but not in the bibliography and vice versa, a link, rather than the actual reference is provided in the bibliography.
In summary: the recording of references is accurate and complete.	In summary, at least **80%** of the sources are correctly reflected and included in a reference list.	In summary, at least **60%** of the sources are incorrectly reflected and/or not included in reference list.

Overall Feedback about the consistency, technical correctness and congruence between in-text referencing and bibliography:

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 dive deep into the development of .NET GUI development, leveraging the power of C# to develop 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 such functionalities, the system aims to not only increase efficiency but also 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 repetitious 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 the details of how your work will be marked.

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

At the end of this specific part, students should be able to:

- LU1: Advanced C# Programming
- LU2: Programming with the .NET Assemblies

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 Form or Model View Controller (MVC) using .NET Core for GUI User Interface (UI). ***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 IU:

- Design the user interface for the Contract Monthly Claim System using either MVC or Windows presentation Forms (.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)

At the end of this specific part, students should be able to:

- LU1: Advanced C# Programming
- LU2: Programming with the .NET Assemblies
- LU3: Files and Data
- LU4: Windows Presentation Foundation

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 something goes wrong.

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

Remember, the goal of this 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)

At the end of this specific part, students should be able to:

- LU1: Advanced C# Programming
- LU2: Programming with the .NET Assemblies
- LU3: Files and Data
- LU4: Windows Presentation Foundation
- LU5: ASP.NET Core Web Development

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 input their hours worked and hourly rate and submit claims easily.

- Automation: Implement an auto-calculation feature to compute the final payment based on the hours worked and hourly rate inputted 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 like Windows Workflow Foundation or third-party libraries such as 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: Utilize ASP.NET Web Forms or ASP.NET Core Razor Pages for building the HR interface. Integrate reporting libraries like Crystal Reports or SQL Server Reporting Services (SSRS) 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.

Appendix A - PoE Marking Rubrics

Assessment Sheet (Marking Rubric)

Please note: Tear off this section and **attach** it to your work when you submit it/ If this is an online submission, then this information needs to be included in the online submission.

MODULE NAME:	**MODULE CODE:**
PROGRAMMING 2B	**PROG6212**

STUDENT NAME:
STUDENT NUMBER:

PART 1					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Documentation: **Design Choices and Structure** **[15 Marks]**	• The explanation of design choices, database structure, and GUI layout lacks clarity and depth. • The rationale behind design decisions is unclear or poorly justified.	• The explanation of design choices, database structure, and GUI layout is clear and adequately detailed. • The rationale behind design decisions is reasonable but may lack some depth or coherence.	• The explanation of design choices, database structure, and GUI layout demonstrates clarity, depth, and coherence. • The rationale behind design decisions is well-developed and logically presented.	• The explanation of design choices, database structure, and GUI layout is exceptionally clear, detailed, and coherent. • The rationale behind design decisions is comprehensive and effectively justifies all aspects of the design.	

	0 – 7 Marks	8 – 10 Marks	11 – 12 Marks	13 – 15 Marks	
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Documentation: **Assumptions and Constraints** **[5 Marks]**	• Assumptions or constraints are not provided or are irrelevant to the project requirements.	• Relevant assumptions or constraints are provided but lack detail or clarity.	• Relevant assumptions or constraints are clearly stated and aligned with the project requirements.	• Comprehensive and well-explained assumptions or constraints are provided, demonstrating a thorough understanding of project requirements.	
	0 – 1 Marks	**2 Marks**	**3 – 4 Marks**	**5 Marks**	

PART 1					
UML Class Diagram for Databases: **Accuracy and Completeness** **[20 Marks]**	• The class diagram is inaccurate or incomplete, failing to represent the data requirements effectively. **0 – 9 Marks**	• The class diagram is mostly accurate and complete, representing most data requirements but with some inaccuracies or omissions. **10 - 14 Marks**	• The class diagram is accurate and complete, effectively representing the data requirements. **15 - 17 Marks**	• The class diagram is highly accurate and complete, providing a comprehensive representation of all data requirements. **18-20 Marks**	
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Project Plan: **Realism and Achievability** **[25 Marks]**	• The project plan is unrealistic or lacks detail, with unclear tasks, dependencies, or timelines. **0 – 12 Marks**	• The project plan is somewhat realistic and achievable, outlining tasks, dependencies, and timeline with some clarity but lacking detail. **13 - 18 Marks**	• The project plan is realistic and achievable, providing clear tasks, dependencies, and timeline with sufficient detail. **19 - 22 Marks**	• The project plan is highly realistic and achievable, presenting clear, detailed tasks, dependencies, and timeline, demonstrating excellent planning skills. **23 - 25 Marks**	

PART 1					
GUI UI: **Design and User-Friendliness** **[25 Marks]**	• The GUI design lacks user-friendliness and intuitiveness, with poor layout and usability. **0 – 12 Marks**	• The GUI design is somewhat user-friendly and intuitive, with adequate layout and usability but room for improvement. **13 - 18 Marks**	• The GUI design is user-friendly and intuitive, with good layout and usability. **19 - 22 Marks**	• The GUI design is highly user-friendly and intuitive, with excellent layout and usability, exceeding expectations. **23 - 25 Marks**	
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Version Control: **Commit Frequency and Descriptive Messages** **[10 Marks]**	• 1 Commit is infrequent, and commit messages lack clarity or description of changes. **0 – 4 Marks**	• 2 Commits are somewhat frequent, but commit messages may lack clarity or detail. **5 - 7 Marks**	• 3 Commits are reasonably frequent and commit messages to provide clarity and detail regarding changes. **8 - 9 Marks**	• 5 Commits are frequent, and commit messages are clear, descriptive, and informative, demonstrating excellent version control practices. **10 Marks**	
Total					

MODULE NAME:	MODULE CODE:
PROGRAMMING 2B	PROG6212

STUDENT NAME:
STUDENT NUMBER:

PART 2					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Lecturers' Claim Submission: **Implementation of Feature** **[20 Marks]**	• The feature is not implemented or does not function as expected, lacking essential functionality.	• The feature is implemented but with some flaws or missing elements, impacting usability or functionality.	• The feature is implemented effectively, meeting basic requirements and functioning adequately.	• The feature is implemented exceptionally well, exceeding basic requirements and enhancing usability or functionality significantly.	
	0 – 9 Marks	**10 – 14 Marks**	**15 – 17 Marks**	**18 – 20 Marks**	

PART 2					
Programme Coordinators and Managers' View: **Design of View** **[20 Marks]**	• The design of the view for coordinators and managers is unclear or disorganised, making it difficult to verify claims.	• The design of the view is somewhat clear but lacks organisation or user-friendly features.	• The design of the view is clear and organised, facilitating easy verification of claims.	• The design of the view is highly intuitive and well-structured, enhancing the verification process significantly.	
	0 – 9 Marks	**10 - 14 Marks**	**15 - 17 Marks**	**18-20 Marks**	
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Lecturers' Document Upload: **Feature Implementation** **[20 Marks]**	• The document upload feature is missing or does not work properly, failing to allow lecturers to upload supporting documents.	• The document upload feature is partially implemented or has some functionality issues.	• The document upload feature is implemented effectively, allowing lecturers to upload documents with ease.	• The document upload feature is implemented exceptionally well, providing a seamless experience for lecturers and ensuring secure storage of uploaded documents.	
	0 – 9 Marks	**10 - 14 Marks**	**15 - 17 Marks**	**18 - 20 Marks**	

PART 2					
Lecturers' Document Upload: **Error Handling and Display** **[10 Marks]**	• Error handling is nonexistent or ineffective, leading to frequent crashes or incorrect information display.	• Error handling is rudimentary, with limited effectiveness in catching and handling exceptions.	• Error handling is implemented effectively, catching most exceptions and displaying meaningful error messages.	• Error handling is implemented exceptionally well, ensuring the application remains stable and responsive even in the face of errors or exceptions.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	
PART 2					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Claim Status Tracking: **Implementation of Tracking System** **[10 Marks]**	• The tracking system for claim status is not implemented or does not update accurately, leading to inconsistencies in status representation.	• The tracking system is partially implemented, with some inaccuracies or delays in status updates.	• The tracking system is implemented effectively, updating claim status reasonably accurately and promptly.	• The tracking system is implemented exceptionally well, providing precision and reliability and real-time and accurate updates on claim status.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	

PART 2					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Consistency and Reliability: **Unit Testing and Error Handling** **[10 Marks]**	• Unit testing is not conducted, or error handling mechanisms are insufficient, leading to inconsistent or unreliable application behaviour.	• Unit testing is conducted to some extent, but error-handling mechanisms are limited in effectiveness.	• Unit testing is conducted effectively, covering key functionalities, and error handling mechanisms are adequate.	• Unit testing is conducted comprehensively, covering all critical functionalities, and error handling mechanisms are robust, ensuring consistent and reliable application behaviour.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	
Version Control: **Commit Frequency and Descriptive Messages** **[10 Marks]**	• 2 Commits are infrequent, and commit messages lack clarity or description of changes.	• 5 Commits are somewhat frequent, but commit messages may lack clarity or detail.	• 7 Commits are reasonably frequent and commit messages to provide clarity and detail regarding changes.	• 10 Commits are frequent, and commit messages are clear, descriptive, and informative, demonstrating excellent version control practices.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	
				Total	

MODULE NAME:	MODULE CODE:
PROGRAMMING 2B	PROG6212

STUDENT NAME:
STUDENT NUMBER:

POE					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Application Enhancement (Automation): **Lecturer View Automation** **[20 Marks]**	• The auto-calculation feature is not implemented, or validation checks are missing, leading to inaccurate or incomplete claim submissions.	• The auto-calculation feature and validation checks are partially implemented but may have some issues or limitations.	• The auto-calculation feature and validation checks are implemented effectively, improving the accuracy and completeness of claim submissions.	• The auto-calculation feature and validation checks are implemented exceptionally well, ensuring accurate and comprehensive claim submissions.	
	0 – 9 Marks	**10 – 14 Marks**	**15 – 17 Marks**	**18 – 20 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
Application Enhancement (Automation): **Coordinator and Manager View Automation** **[20 Marks]**	• The automated verification and approval processes are not implemented, or workflows lack efficiency, causing delays or errors in claim processing.	• The automated verification and approval processes are partially implemented but may have some inefficiencies or shortcomings.	• The automated verification and approval processes are implemented effectively, enhancing the efficiency and accuracy of claim processing.	• The automated verification and approval processes are implemented exceptionally well, ensuring streamlined and error-free claim processing.	
	0 – 9 Marks	**10 – 14 Marks**	**15 – 17 Marks**	**18 – 20 Marks**	
PART 3					
Application Enhancement (Automation): **HR View Automation** **[20 Marks]**	• The automation of claim processing and lecturer data management tasks is incomplete or ineffective, leading to manual intervention and inefficiencies.	• The automation of claim processing and lecturer data management tasks is partially implemented but may lack some essential features or functionalities.	• The automation of claim processing and lecturer data management tasks is implemented effectively, reducing manual effort and improving administrative efficiency.	• The automation of claim processing and lecturer data management tasks is implemented exceptionally well, significantly streamlining administrative processes.	
	0 – 9 Marks	**10 - 14 Marks**	**15 - 17 Marks**	**18-20 Marks**	

PART 3					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
PowerPoint Presentation: **Coverage and Presentation Quality** **[20 Marks]**	• The presentation lacks coverage of key aspects of the Contract Monthly Claim System, and the quality of presentation slides is poor or inconsistent.	• The presentation covers essential aspects of the system but may lack depth or visual appeal in some areas.	• The presentation provides a comprehensive overview of the system with visually appealing slides and clear communication of value.	• The presentation is exceptionally well-structured, visually appealing, and effectively communicates the value of the Contract Monthly Claim System.	
	0 – 9 Marks	**10 - 14 Marks**	**15 - 17 Marks**	**18-20 Marks**	
PART 3					
Design and User-Friendliness **[10 Marks]**	• The GUI design lacks user-friendliness and intuitiveness, with poor layout and usability.	• The GUI design is somewhat user-friendly and intuitive, with adequate layout and usability but room for improvement.	• The GUI design is user-friendly and intuitive, with good layout and usability.	• The GUI design is highly user-friendly and intuitive, with excellent layout and usability, exceeding expectations.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	

PART 3					
Marking Criteria	**Does not meet the required standard**	**Meets the required standard**	**Partially exceeds the required standard**	**Greatly exceeds the required standard**	**Feedback**
Version Control: **Commit Frequency and Descriptive Messages** **[10 Marks]**	• 2 Commits are infrequent, and commit messages lack clarity or description of changes.	• 5 Commits are somewhat frequent, but commit messages may lack clarity or detail.	• 7 Commits are reasonably frequent and commit messages to provide clarity and detail regarding changes.	• 10 Commits are frequent, and commit messages are clear, descriptive, and informative, demonstrating excellent version control practices.	
	0 – 4 Marks	**5 - 7 Marks**	**8 - 9 Marks**	**10 Marks**	
Total					