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 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. ***Make a copy of your assignment before handing it in.***
3. *Assignments must be typed unless otherwise specified.*
4. *Begin each section on a new page.*
5. *Follow all instructions on the PoE cover sheet.*
6. *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

It was that time of the semester when the first project submissions were due. The computer labs were busy all the way until closing time, with lots of students working on projects.

On the Monday evening, Sipho managed to finish one of his projects and hand it in a whole day early. 15 minutes before closing time, he waved goodbye to Lerato who was still furiously working. On Tuesday midday, Sipho went to the lab to read his emails and saw Lerato in the same spot. Still working hard and looking more determined than ever, she gave him a brave smile as he walked past.

On Wednesday morning, when a well-rested Sipho had a class scheduled in that computer lab, Lerato was still in the same spot. And by now, she was looking distinctly frazzled. No, she can't have been there the whole time. The labs definitely do close at night. But it sure looked like she had been working for two days straight without sleep.

Sipho felt sorry for Lerato. It was difficult to have so much work to do all at once. Not sleeping takes its toll. He had once been in the same position, burning the midnight oil and falling behind. He realised that it would help tremendously if he didn't leave things until the last minute. Now his life was so much more organised. He had time for things like sleep throughout the whole semester.

Right there and then, Sipho decided that a cool computer program could help Lerato to organise her life better. So, he would try it out for himself first and then give Lerato a copy as a surprise present. And maybe, just maybe, he could get to see his friend smile again.

The program will have to be able to store which modules a student is doing. Every module is worth a specific number of credits and that number multiplied by 10 will be the number of hours spent on it throughout the semester. For example, PROG6212 is 15 credits, so 150 hours should be spent on it. Some of that will be in class, and the rest will have to be distributed throughout the weeks.

Instructions

This portfolio of evidence (POE) consists of three parts – two parts submitted during the semester and a final submission at the end of the semester. The parts build on one another, so make sure that you **keep a copy of your work in a safe place**.

The requirements of real software projects frequently change, often in quite unexpected ways. Here you have the benefit of knowing what all the requirements will be in advance. So, make use of the opportunity. **Reading all three parts** before starting with the first one will minimise any reworking for later parts.

The **rubrics** that will be used to mark your submissions appear at the end of this document. Please pay attention to the weighting of items in the rubrics.

Note that marks will be awarded for **running functional software,** not just source code. So, ensure that your source code **compiles** and that the **readme** file contains enough information about running the software.

Important: This POE is **NOT** identical in terms of requirements to last year's one. Read carefully.

Part 1 — Basic Application (Marks: 100)

Using **C#** and **Windows Presentation Foundation** (**WPF**), design and implement a standalone desktop time management application that fulfils the following requirements:

1. The user must be able to add **multiple modules** for the semester. The following data must be stored for each module:
    a. **Code**, for example, PROG6212
    b. **Name**, for example, Programming 2B
    c. **Number of credits**, for example, 15
    d. **Class hours per week**, for example, 5
2. The user must be able to enter the **number of weeks** in the semester.
3. The user must be able to enter a **start date** for the first week of the semester.
4. The software shall display a **list** of the **modules** with the number of hours of self-study that is required for each module per week. The number shall be calculated as follows:

$$\text{self-study hours per week} = \frac{\text{number of credits} \times 10}{\text{number of weeks}} - \text{class hours per week}$$

5. The user must be able to record the **number of hours** they spend working on a **specific module** on a **certain date**.
6. The software shall display **how many hours of self-study remain** for each **module** for the **current week**. This should be calculated based on the number of hours already recorded on days during the current week.
7. The software shall **not persist** the user data between runs. The data shall only be stored in memory while the software is running.

Non-functional requirements:

1. You are required to use internationally acceptable **coding standards**. Include comprehensive comments explaining variable names, methods, and the logic of programming code.
2. You must make use of **LINQ** to manipulate the data.
3. You are required to create a **custom class library** that contains the classes related to the data and calculations. The WPF application project should make use of the custom class library.

Submit the following items for this part:

1. **Source code** including both the class library and Windows Presentation Foundation application.
2. Unified Modelling Language (UML) **class diagram** showing the classes in both the class library and WinForms application. You may use any software of your choosing to create the diagram, but the file that you submit must be a **.PDF export** of your diagram.
3. A **readme file** with instructions for how to compile and run the software.

Part 2 — Persisting the Data (Marks: 100)

The application developed in Part 1 is already useful in terms of functionality, but it has a severe usability flaw: the data is not persisted, forcing the user to capture all the data from scratch if the application is executed again.

For this part, you will continue working on the application you developed in Part 1. Remember to implement any feedback provided by your lecturer on Part 1 before working on Part 2. Marks will be awarded for this (see the rubric for details).

All the requirements from Part 1 must still be met by the program, with the following changes and additions:

1. The software **shall persist** the data in a **SQL database**.
2. The user shall be able to **register** with a username and password.
3. The software shall store only the **hash** of the **password** in the database.
4. The user shall be able to **log into** the software with their username and password.
5. The user shall only be able to **see their own data** and never that of other users.

Non-functional requirements:

1. The application should use the **custom class library** developed in **Part 1**. You may update the class library as necessary for the new functionality.
2. You can choose to **access the database** using the ADO.NET connected layer or Entity Framework Core.
3. Regardless of database access technology, the application should use **multi-threading** to ensure that the user interface never becomes unresponsive while retrieving or storing information.

Submit the following items for this part:

1. **Source code** including both the class library and WPF application.
2. Unified Modelling Language (UML) **class diagram** showing the classes in both the class library and application. Indicate which **changes** you had to make to the class library. You may use any software you choose to create the diagram, but the file you submit must be a **.PDF export** of your diagram.
3. Any **additional artefacts** that are required to run the application, for example, a SQL script to create tables, if required.
4. A **change log file** that lists the changes implemented after feedback on Part 1.
5. A **readme file** with instructions for how to compile and run the software. Remember to include all instructions related to the database!

POE — Web Application (Marks: 100)

The desktop application is now quite useful with the data being stored between runs. But having to always use a desktop computer to run the program is maybe not the most flexible user experience. Instead, create a web application that will allow the user to access their data from any device that has a browser.

You are required to develop an ASP.NET Core web application for this part. It should have all the same functionality as the application in Part 2, with your **choice** of **one** of the following features added:

1. The software shall display in the format of a graph over time the number of hours spent on a module per week. The ideal calculated number of hours should also be displayed on the graph.

OR

2. The user shall be able to set aside a specific day of the week for each module. When the website is accessed, a reminder must be shown that tells the user which module is planned for the day.

Non-functional requirements:

1. You must again **reuse** the **custom class library** implemented in Part 2.
2. You must make use of **ASP.NET Core**.

Submit the following items for this part:

1. **Source code** including both the class library and web application.
2. Unified Modelling Language (UML) **class diagram** showing the classes in both the class library and web application. Indicate which **changes** you had to make to the class library. You may use any software you choose to create the diagram, but the file you submit must be a **.PDF export** of your diagram.
3. Any **additional artefacts** that are required to run the application, for example, a SQL script to create tables, if required.
4. A **readme file** with instructions for how to compile and run the software. Remember to include all instructions related to the database!
5. A short **user manual** (1 200 to 1 500 words), including **screenshots**, that explains how to use the website. You may use any application of your choice to create the user manual, but the file you submit must be a **.PDF export** of the document.

Note: The manual will be marked only up to 1 500 words. Any extra words over the limit will be disregarded.

[TOTAL MARKS: 300]

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 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App Functionality: The user can add multiple modules for the semester with all the required data. The data is stored in memory.	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
[10 Marks]	**0 – 4 Marks**	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PART 1 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The user can enter the number of weeks and start date for the first week, and this is stored in memory. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The list of modules is displayed to the user. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PART 1 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The hours per week is correctly calculated and displayed to the user. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The user can record hours spent on a module and this is stored in memory. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PART 1 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The remaining hours for the week are correctly calculated and displayed to the user. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Usability: The user interface is easy to use. **[10 Marks]**	• The user interface is completely confused and illogical.	• The user interface can be used but is not very logical.	• The user interface is well implemented with a few small useability problems.	• The user interface is excellently implemented and very easy to use.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Application Structure: The application makes use of LINQ. **[5 Marks]**	• LINQ is not used at all in the application. • LINQ is used, but it does not work as expected.	• LINQ is used with some issues in the implementation.	• LINQ is used with one or two small mistakes in the implementation.	• The application makes excellent use of LINQ to manipulate data.	
	0 – 1 Mark	**2 – 3 Marks**	**3 – 4 Marks**	**5 Marks**	

PART 1 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
Application Structure: A custom class library was created. **[5 Marks]**	• No custom class library was created. • A custom class library was created, but it doesn't contain much code or is not used.	• A custom class library with some classes was created and is used by the application, but there are some issues.	• A custom class library with some classes was created and is used by the application.	• A custom class library was created that handles all the data and logic.	
	0 – 1 Mark	**2 – 3 Marks**	**3 – 4 Marks**	**5 Marks**	
Coding Standards: The code is well structured and documented. **[5 Marks]**	• The code is poorly structured, with no naming convention used and no comments included. • The code is not well structured but somewhat readable, and very few comments are included.	• Code structure can be somewhat improved or too few comments included.	• The code is mostly well structured, with some comments included.	• Code is excellently structured, easy to read, and with sufficient detail in the comments.	
	0 – 1 Mark	**2 – 3 Marks**	**3 – 4 Marks**	**5 Marks**	

PART 1 – BASIC APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
Documentation: UML diagram accurately reflects the class structure. **[10 Marks]**	• No diagram is included, or the diagram doesn't reflect the application at all. • A partial diagram is included with most of the classes and methods missing.	• A partial diagram includes at least half of the classes and methods.	• A good diagram with only one or two mistakes.	• An excellent diagram that accurately reflects the design of the application.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Documentation: The readme file provides enough information to run the app. **[5 Marks]**	• No readme file is included, or the readme file doesn't provide any useful information about running the application. • The readme file contains information, but it is hard to understand or doesn't work.	• The readme file presents some information about running the app but is missing some crucial steps.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	
	0 – 1 Mark	**2 – 3 Marks**	**4 Marks**	**5 Marks**	

PART 2 – PERSISTING THE DATA					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
Updates: All the feedback provided on Part 1 has been implemented. **[10 Marks]**	• Little or no feedback was implemented.	• Around half of the feedback was implemented.	• Most feedback was implemented.	• Excellent implementation of all feedback provided.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The data is saved to a SQL database and loaded again when the application is restarted. **[15 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 6 Marks	**7 – 9 Marks**	**10 – 11 Marks**	**12 – 15 Marks**	
App functionality: The user can register using a username and password. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PART 2 – PERSISTING THE DATA					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The user stores a hash of the password and uses that to allow the user to log into the app. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The user can only see their own data, and never that of other users. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Usability: The user interface is easy to use. **[10 Marks]**	• The user interface is completely confused and illogical.	• The user interface can be used but is not very logical.	• The user interface is well implemented with a few small useability problems.	• The user interface is excellently implemented and very easy to use.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PART 2 – PERSISTING THE DATA					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
Application Structure: The application uses ADO.NET connected layer or Entity Framework. **[5 Marks]**	• No database access technology is used. • A database access technology is used, but the implementation doesn't work well.	• A database access technology is implemented with some errors.	• A database access technology is implemented with only minor errors.	• A database access technology is consistently implemented and works correctly.	
	0 – 1 Mark	**2 – 3 Marks**	**4 Marks**	**5 Marks**	
Application Structure: The application uses multi-threading. **[5 Marks]**	• No multi-threading is implemented.	• Multi-threading is only partially implemented or only working under certain circumstances.	• Multi-threading is implemented in most places, with some exceptions.	• Multi-threading is correctly implemented throughout the application.	
	0 – 1 Mark	**2 – 3 Marks**	**4 Marks**	**5 Marks**	
Coding Standards: The code is well structured and documented. **[5 Marks]**	• The code is poorly structured, no naming convention was used, and comments are not included. • The code is not well structured but somewhat readable,	• Code structure can be somewhat improved, or too few comments included.	• The code is mostly well structured, with some comments included.	• Code is excellently structured, easy to read, and with sufficient detail in the comments.	

PART 2 – PERSISTING THE DATA					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
	and very few comments are included.				
	0 – 1 Mark	**2 – 3 Marks**	**4 Marks**	**5 Marks**	
Documentation: UML diagram accurately reflects the class structure. **[10 Marks]**	• No diagram is included, or the diagram doesn't reflect the application at all. • A partial diagram is included with most of the classes and methods missing.	• A partial diagram includes at least half of the classes and methods.	• A good diagram with only one or two mistakes.	• An excellent diagram that accurately reflects the design of the application.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Documentation: The readme file provides enough information to run the app. **[10 Marks]**	• No readme file is included, or the readme file doesn't provide any useful information about running the application. • The readme file contains information, but it is	• The readme file presents some information about running the app but is missing some crucial steps.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	

PART 2 – PERSISTING THE DATA					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
	hard to understand or doesn't work.				
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PORTFOLIO OF EVIDENCE – WEB APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The user can register and log into the web app. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The user can add multiple modules for a semester and record the number of weeks and start date of the semester. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs.	• The feature is implemented with some bugs.	• The feature is well implemented with only one or two bugs.	• The feature works perfectly without any errors.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

PORTFOLIO OF EVIDENCE – WEB APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The list of modules together with the correctly calculated number of hours per week, is displayed. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs. **0 – 4 Marks**	• The feature is implemented with some bugs. **5 Marks**	• The feature is well implemented with only one or two bugs. **6 – 7 Marks**	• The feature works perfectly without any errors. **8 – 10 Marks**	
App functionality: The hours spent on a module can be captured, and the number of self-study hours remaining is correctly calculated and displayed. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs. **0 – 4 Marks**	• The feature is implemented with some bugs. **5 Marks**	• The feature is well implemented with only one or two bugs. **6 – 7 Marks**	• The feature works perfectly without any errors. **8 – 10 Marks**	

PORTFOLIO OF EVIDENCE – WEB APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
App functionality: The data is persisted in the database and loaded again when the user logs in the next time. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs. **0 – 4 Marks**	• The feature is implemented with some bugs. **5 Marks**	• The feature is well implemented with only one or two bugs. **6 – 7 Marks**	• The feature works perfectly without any errors. **8 – 10 Marks**	
App feature: New feature (graph or setting aside a day for a module) works correctly. **[10 Marks]**	• The feature is not implemented or does not work at all. • The feature is implemented, but there are lots of bugs. **0 – 4 Marks**	• The feature is implemented with some bugs. **5 Marks**	• The feature is well implemented with only one or two bugs. **6 – 7 Marks**	• The feature works perfectly without any errors. **8 – 10 Marks**	
Coding Standards: The code is well structured and documented. **[5 Marks]**	• The code is poorly structured, with no naming convention used and no comments included. • The code is not well structured but	• The code structure can be somewhat improved or too few comments included.	• The code is mostly well structured, with some comments included.	• The code is excellently structured, easy to read, and with sufficient detail in the comments.	

PORTFOLIO OF EVIDENCE – WEB APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
	somewhat readable, and very few comments are included.				
	0 – 1 Marks	**2 – 3 Marks**	**4 Marks**	**5 Marks**	
Documentation: UML diagram accurately reflects the class structure. **[10 Marks]**	• No diagram is included, or the diagram doesn't reflect the application at all. • A partial diagram is included with most of the classes and methods missing.	• A partial diagram is included with at least half of the classes and methods included.	• A good diagram with only one or two mistakes.	• An excellent diagram that accurately reflects the design of the application.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Documentation: The user manual is well structured with useful screenshots. **[15 Marks]**	• Not included or almost no detail. • Some information is included.	• A partial user manual is included that covers the essential features.	• Mostly complete user manual included.	• Complete user manual included.	
	0 – 6 Marks	**7 – 9 Marks**	**10 – 11 Marks**	**12 – 15 Marks**	

PORTFOLIO OF EVIDENCE – WEB APPLICATION					
Marking Criteria	**Does not meet the required standard** ***(0% – 49%)***	**Meets the required standard** ***(50% – 64%)***	**Partially exceeds the required standard** ***(65% – 74%)***	**Greatly exceeds the required standard** ***(75% – 100%)***	**Feedback**
Documentation: The readme file provides enough information to run the app. **[10 Marks]**	• No readme file is included, or the readme file doesn't provide any helpful information about running the application. • The readme file contains information, but it is hard to understand or doesn't work.	• The readme file presents some information about running the app but is missing some important steps.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	