

ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

Course Title	Bachelors of Science in Software Development (Hons.)	Lecturer Name & Surname	Diane Desira Ryan Attard McIntyre
Unit Number & Title	ITSFT-506-1605 - Object Oriented Programming		
Assignment Number, Title / Type	Assignment 1: Implementing a Software Project		
Date Set		Deadline Date	15/01/2021
Student Name	Ismael Bendaoud	ID Number	0293600
		Class / Group	6.1A

Assessment Criteria	Maximum Mark
<i>KU6: Show ability to efficiently create and use the storage layer of a layered application.</i>	5
<i>KU7: Understand the use of LINQ queries in an Object-Oriented application.</i>	5
<i>AA3: Use LINQ to implement persistent data storage in an object-oriented application.</i>	7
<i>SE2: Assess and breakdown an application into different layers to identify where object-database mapped classes should be inserted.</i>	10
<i>AA5: Implement exception handling processes to respond to any exceptions occurring in an application.</i>	7
<i>SE3: Analyse and present an implementation of an Object-Oriented application.</i>	10
Total Mark	44

Notes to Students:
<ul style="list-style-type: none"> This assignment brief has been approved and released by the Internal Verifier through Classter. Assessment marks and feedback by the lecturer will be available online via Classter (http://mcast.classter.com) following release by the Internal Verifier Students submitting their assignment on Moodle/Unicheck will be requested to confirm online the following statements: <p>Student's declaration prior to handing-in of assignment</p> <p>✚ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy</p> <p>Student's declaration on assessment special arrangements</p> <p>✚ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.</p> <p>✚ I declare that I refused the special support offered by the Institute.</p>

Assignment 1 Implementing a Software Project

Assignment Guidelines

Read the following instructions carefully:

- The assignment coversheet should be the first sheet in your assignment. Moreover, the coversheet should be fully completed with all the necessary details.
- You are required to use the .NET Framework and the C# programming language for this assignment.
- All text\code *must be properly referenced*. In the absence of proper referencing, the assignment will be regarded as plagiarised.
- **Copying is strictly prohibited and will be penalized** in line with the College's disciplinary procedures.
- The deadline should be specified by your lecturer.
- You need to submit your application via a Moodle link. Ensure that you submit via your group link. You are also required to submit your work on Github. Ensure that you assign your lecturer as a collaborator to your assignment repository. Also, please make sure that the assignment repository has been set as **private**.
- You need to submit the .Net project, the script representing your database and a word document with all the code used in your application.
- The lecturer will hold a post-submission interview. Attendance to such interview is mandatory. Moreover, marks assigned to all criteria can be affected by the interview performance.

Scenario: Attendance System

You have been appointed as a software developer to implement an Attendance System for a school.

You need to ensure that the application uses the most efficient **Object-Oriented Techniques**, is structured in **three-tier architecture**, and has proper **ObjectRelational mapping (ORM)** to allow for efficient persistence.

The application should have the following persistent types. You are to use a Database First ORM approach (therefore, first you need to implement the database and then you need to map your database to an object-oriented application).

Teacher

<u>Field</u>	<u>Type</u>
TeacherID	Number
Username	Text
Password	Text
Name	Text
Surname	Text
Email	Text

Group

<u>Field</u>	<u>Type</u>
GroupID	Number
Name	Text
Course	Text

Student

<u>Field</u>	<u>Type</u>
StudentID	Number
Name	Text
Surname	Text
Email	Text
GroupID	Number

Lesson

<u>Field</u>	<u>Type</u>
LessonID	Number
GroupID	Number
DateTime	DateTime
TeacherID	Number

StudentAttendance

<u>Field</u>	<u>Type</u>
AttendanceID	Number

LessonID	Number
Presence	Bool
StudentID	Number

Your Object-Oriented application should be structured in a Three-Tier Architecture.

This attendance system should only be controlled by a teacher. Before you start testing the system, you need to create at least one teacher record in your database.

The following main menu should be displayed when the user (teacher) runs the application.

```
Main Menu
=====
1. Login
2. Exit
Enter Choice:
```

The user should enter a number which corresponds to a menu choice.

The **Exit** menu should show an exit message and terminate the application.

The **Login** option should redirect the user to the following:

```
Login
=====
Username:
```

The user needs to type in his/her username. Before asking for the password, the system should check if the username exists.

```
Login
=====
Username: testing
The entered ID does not exist in our system.
```

If the username does not exist, an error message should be displayed, and the system should be redirected to the main menu.

Otherwise, the user will be allowed to enter the corresponding password. An authentication check should be carried out to check if the password is correct. If not, an error message should be displayed. Otherwise, the teacher's menu should be displayed.

N.B. It would be ideal to fetch the ID for the teacher who successfully logged in. This will be required since for certain database functionalities to be carried out, the teacher's ID is needed. It is not user friendly to ask for the ID of a user who is already logged in. Thus, this should be stored once authentication is completed.

Hint: You can fetch the ID upon checking for an existing username or a matching password. For example, an ID which is not equal to 0 would indicate that the username exists, or the password has matched.

Teacher's Menu

Add Attendance:

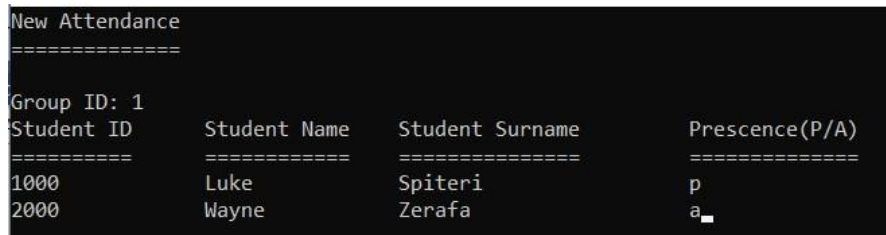
This menu option should first show all the groups that are saved in the database. The group ID and the group Name should be displayed. The system should then ask the teacher for the group ID. Validation should be carried out accordingly.

If the group ID is correct, a new lesson should be created.

N.B. To create a lesson, apart from the group ID, you need the logged in teacher's id (as suggested, this can be saved globally in the current layer/class) and date. Also note, the lesson ID could be set to auto-increment from the table itself (otherwise you would need to implement a validation to ensure that the ID is unique).

Following this, all students from this group should be fetched so that each student is displayed on the screen and the teacher will be allowed to enter either **p** or **a**. The character **p** would indicate that the student is present and a character **a** would indicate that the student is absent.

For validation, a loop should be used to wait for only **p** and **a** inputs from the teacher. If any other character is inserted, the system should keep on asking the teacher for the proper character.



```
New Attendance
=====
Group ID: 1
Student ID      Student Name      Student Surname      Prescence(P/A)
=====
1000            Luke              Spiteri              p
2000            Wayne             Zerafa               a_
```

Add a New Group:

For this menu option, the user needs to enter the group name and the course name.

N.B. the group ID could be set to auto-increment from the table itself (otherwise you would need to implement a validation to ensure that the ID is unique).

Add a New Student:

This menu option should first ask the teacher for the group ID. Validation should be carried out accordingly.

If the group ID exists, the teacher needs to enter all the new student's details.

Add a New Teacher:

For this menu option, the teacher needs to enter all the new teacher's details.

Check a student's attendance percentage:

This menu option should first ask the teacher for the student's ID. Validation should be carried out accordingly.

If the student ID is correct, all the student's attendances are retrieved. A check needs to occur to count only those attendances which marked this student as present. **N.B.** You need to use LINQ for this task.

Get all attendances submitted on a particular day:

For this menu option, the teacher needs to enter a date in the following format. A search needs to occur so that all attendances which are submitted on the given date, by the logged in teacher, are returned.

As previously suggested, the logged in teacher's ID should be saved and thus, can easily be used for this query.

```
Submitted Attendances
=====

Day: 14
Month: 10
Year: 2020

Number of attendances submitted on given day: 3
```

Hint: when a teacher submits an attendance, a lesson is created.

Edit Student:

For this menu option, the teacher needs to input the student's id so that that student can be edited. The teacher is then asked to input the new name, surname, and email for this student.

N.B. if not all details are to be edited, the teacher needs to re-enter the same value which is currently saved in the database.

If the student ID is incorrect, the system should return a value to indicate this and the presentation for this application, should keep on asking the teacher to re-enter all details (including the student's ID).

Hint: A do while loop can be used for this repetition.

```
Edit Student
=====

Student ID: 2000

New Details:

Name: Wayne
Surname: Mallia
Email: wm@mcaste.edu.mt
```

Error Handling

For proper error handling of your system, Try and Catch should be used to ensure that the entered IDs are in fact numbers. TryParse should also be used to ensure that all menu option numbers are also of a number type.

Marking Scheme

KU6 (5 marks)	Marks	SCORE
ORM mapping set up (1 Mark)	The Database has been properly mapped with the Object-Oriented solution.	
Saved changes (2 Marks)	Any new objects are being properly saved in the database as described in the given requirements.	
Retrieve data (1 Mark)	All necessary data is retrieved from the database.	
Edit changes (1 Mark)	The modified changes for objects are seen applied in the database.	
	/5	

KU7 (5 marks)	Marks	SCORE
Check Username (1 Mark)	All queries need to be implemented via LINQ and should follow the specific requirements given. Moreover, the mark will only be rewarded if the requirement works.	
Check authentication (2 Marks)		
Unique ID (Student + Teacher) (2 Marks)		
	/5	

AA3 (7 marks)	Marks	SCORE
Fetch all groups and check Group ID (1 Mark)	All queries need to be implemented via LINQ and should follow the specific requirements given. Moreover, the mark will only be rewarded if the requirement works.	
Student Percentage (2 Mark)		
Total attendances (2 Mark)		
Fetch Student to Edit + Correct Edit (2 Mark)		
	/7	

SE2 (10 marks)	Marks	SCORE
Presentation Layer (2 Marks)	All the requirements for the specific layer have been implemented. There should not be any violations in the given layer and the layer is following proper separation of concerns.	
Business Layer (2 Marks)		
Data Layer (2 Marks)		
Three-Tier Architecture (4 Marks)	The application is implemented in a proper three-tier structure and there should only be minor violations for this mark to be rewarded.	
	/10	

AA5 (7 marks)	Marks	SCORE
Login (1 Mark)	Authentication for login is functional and error is displayed as per requirements.	
Error in Registration (1 Mark)	Both validations (username and email) for the registration option are functional and error is displayed as per requirements.	
Try and Catch (2 Marks)	IDs are properly validated using a Try and Catch ().	
Menu Number or Err (1 Mark)	The menu number entered should be within the correct range for all menus in the application.	
TryParse() (2 Marks)	TryParse() works properly to test the menu option numbers.	
Question repetition using loops (1 Mark)	Implementation has been carried out as instructed in the given requirements (presence for adding a new attendance and correct student ID for the edit student option).	
	/7	

SE3 (10 marks)	Marks	SCORE
Student needs to be ready for the interview as per time allocated. All necessary files and applications should be made readily available before the interview starts. If this is not the case, all the marks for this criterion will be lost.		
Data Preparation (1 Mark)	All the necessary data for testing has been included in the database.	
Menus (2 Marks)	Development for menus has been implemented efficiently to allow for proper testing. Thus, the user can go back to main menu or previous menu at different points at runtime.	
Presentation of Prototype (2 Marks)	Student went through all the Assignment requirements and tested them out. This should include testing for proper error handling. Student should be prepared on how to go through the presentation. Marks will be lost if certain requirements were not presented by the student.	
Development Explanation (3 Marks)	The student should go through the implemented code and highlight development of important techniques.	
Interview Questions (2 Marks)	The student was able to correctly answer questions asked by the interviewer.	
	/10	