Dublin Business School

# Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Unit Title: | Advanced Programming |
| Unit Code: | ICT |
| Unit Lecturers: | Paul Laird |
| Level: | 8 |
| Assessment Title: | Practical Programming |
| Assessment Number: | 1 |
| Assessment Type: | Individual |
| Restrictions on Time/Length : | N/A |
| Individual/Group: | Individual |
| Assessment Weighting: | 50% |
| Issue Date: | 30/4/19 |
| Hand In Date: | 11/6/19 |
| Mode of Submission: | On-line **ONLY** |

Assessment

The following is the assignment as set out for the module of Advanced Programming. The assignment is an Individual assignment. It is suggested that code be written in C# with MS SQL Server localdb Backend.

The following table illustrates the percentage allocation for each individual part of the assignment.

*Part Breakdown of Marks*

Presentation Tier 15%

Business Tier 20%

Data Tier 15%

Database – SQL set-up and population 15%

Technical Merit and Innovation 15%

UML 10%

Code Structure, Layout and Adherence to Specifications 10%

**Assessment Specification**

You have been asked to develop a Student Management System for DBS. When the application is run a list of Students must be displayed to the user in a grid format. The software must allow users to enter details relating to a new Student in addition to updating existing students and removing a student from the system.

A sample of the data relating to a Student can be found in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| First Name | Surname | Student Number | County | Country | Course |
| Joe | Smith | 1712345 | Dublin | Ireland | Software Development |
| Mary | Jones | 1712346 | Galway | Ireland | Arts |
| John | Doe | 1712347 | Donegal | Ireland | Business |

The basic requirements are described in the following table.

|  |  |
| --- | --- |
| Requirement | Description |
| REQ 1 | The desktop application must be built using a **tiered** architecture. |
| REQ 2 | When the application is launched, a user must be presented with a screen to allow them to login.  A menu will always be present at the top of the application; however the contents of the menu are context specific. When the application is initially launched the menu should have the following structure.   * File   + Login   + Exit |
| REQ 3 | In order to ensure appropriate use of the application only internal staff maybe allowed use this software. As a result a user must login to the application. The username can be displayed in clear text, however the password should be masked using a \* for each character of the password.  If the user enters an incorrect username and/or password a message should appear informing them so. |
| REQ 4 | Once a user has successfully logged into the application the menu context will change. The menu should now display the following:   * File   + Logout   + Exit * Student   + New Student   + Edit Student   + Delete Student   + View Database History |
| REQ 5 | In addition to updating the menu upon a successfully login a grid should be displayed containing information relating to existing students. |
| REQ 6 | The user must be able to enrol a new Student. To do this they select the “New Student” menu item under “Student” from the menu bar.  The following data must be captured:   |  |  |  | | --- | --- | --- | | Name | Description | Data type | | First Name | The first name | String | | Surname | The surname | String | | Email | The email address | String | | Phone | The phone number | String | | Address Line 1 | The first line postal address | String | | Address Line 2 | The second line in the address | String | | City | The city or town postal address | String | | County | The county postal address | Drop down list of the counties of Ireland | | Level | The course can be either a Postgrad or a Undergrad | Radio buttons | | Course | Insert at least 5 courses | Drop down list | | Student Number | The new number for the Student | Int (8 digits) | |
| REQ 7 | The user of the application must be able to update/edit existing Students.  The following fields should be read only. That is the user should not be able to modify the student using this screen.   * Student Number * First Name * Surname * Course   All other fields should be pre-populated with the relevant data for the student being updated.  As per REQ6, the grid displaying all students should be updated to reflect any changes in information. |
| REQ 8 | The system should provide a means to serialise the details of a specific Students into an XML format. |
| REQ 9 | The system will need to be documented with detailed Class Diagrams, Use Case Diagrams and Sequence Diagrams. |

**NOTE:**

* You must submit all your code, project(s) and solution files along with any other supporting documentation.
* You must also submit all your SQL, including a create scripts for any tables and/or stored procedures you have used.
* You must develop your project using GitHub or a similar service for SCM, and must share the repository or make it public. It is expected that there will be commits every week throughout the time available.
* Marks may be deducted if files or code is not submitted/omitted.
* **An appropriate alternative application may be developed, focusing on a domain of interest, subject to it including appropriate data structures, database access and XML manipulation, structured in a three-tier architecture. Requirements must be submitted by 7th may if you intend to develop an alternative application.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria/**  **Mark** | **< 40** | **40 - 49** | **50 - 59** | **60 – 69** | **70 +** |
| **UML** | Difficult to read; not related to code | Messy, needs tidying; partially related to code | Some errors and mistakes; a lot related to code | Few errors and mistakes; fully related to code | Presented excellently with no errors |
| **Code& Technical Merit** | Insufficient or incomplete code structure | Some but insufficient and poorly structured code which doesn’t solve the problem | Sufficient solves problem but lack of attention to coding practices | Well-structured code that solves the problem | Excellent solution to problem |

**General Requirements for Students:**

1. Please read carefully the following problems. You are expected to submit both the UML diagrams and code as part of your submission. Save all your files in a folder and give it name: yourName. Compress the folder and upload it on Moodle.
2. It is your responsibility to ensure your files are uploaded correctly.
3. Students are required to retain a copy of their assignment.
4. When an assignment is submitted, it is the student’s responsibility to ensure that the file is in the correct format and opens correctly.
5. Students should refer to the assessment regulations in their Course Guide.
6. DBS penalises students who engage in academic impropriety (i.e. plagiarism, collusion and / or copying). Please refer to the referencing guidelines on Moodle for information on correct referencing.
7. All relevant provisions of the Assessment Regulations must be complied with. Penalties for late submission on assignments are as follows:
   1. 25% penalty for assignments submitted within 5 working days of the deadline.
   2. No marks for assignments submitted more than 5 working days after the deadline.
8. Extensions to assignment submission deadlines will be granted in exceptional circumstances only. The appropriate “Application for Extension” form must be used and supporting documentation (e.g. medical certificate) must be attached. Applications for extensions should be made directly to the Head of Year or Programme Leader in advance of the deadline date.