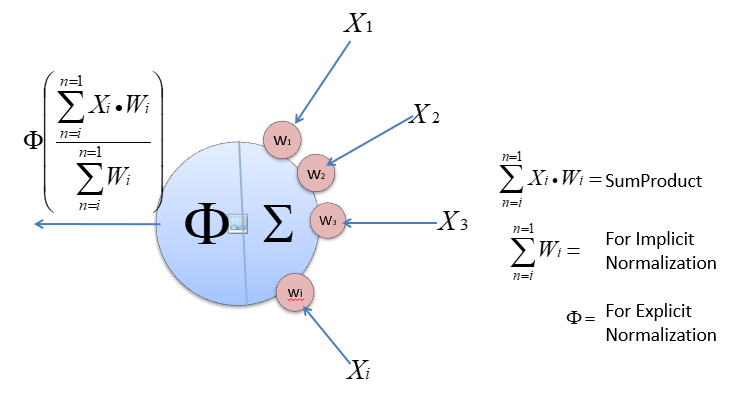
**Mini-Project**

This is an individual project. However, students are encouraged to have discussion in group. But the design and coding should be done by individual student.

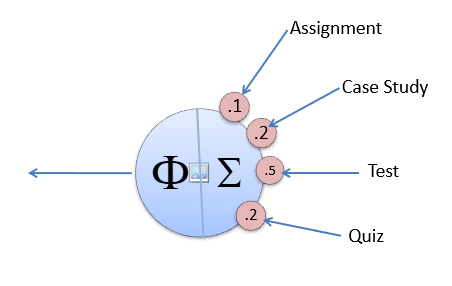
**Duration:** 2 Man/day

**Due Date:** Submit **before** 23/September/2023 (Saturday)

**Simple Measurement Unit:**



**Analogy:**



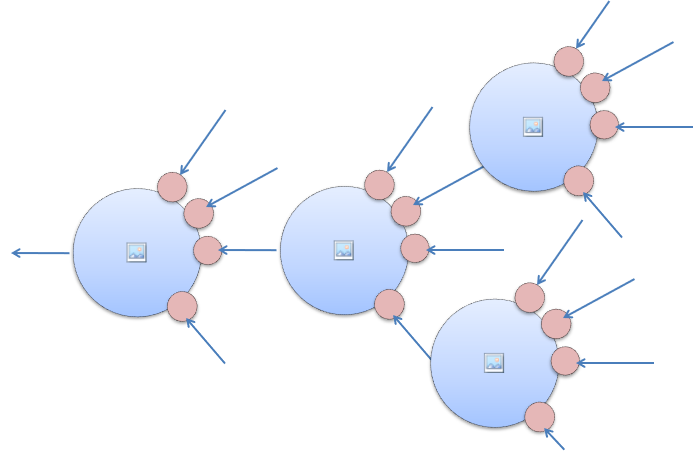
**Implicit Normalization:**

* The total weight need not be 100%. The formula will ensure the output will fall in between 0 to 100%. This implied if the output becomes the input, it will be in between 0 to 100%.
* It is done implicitly
* New sub measurement can be added or existing sub measurement can be removed without affecting other sub assessments.
* Example: The lecturer missed one of the assessments because of some reason. Or sometime ad hoc assessment is given.

**Explicit Normalization:**

* Under some abnormal circumstances, special normalization is needed.
* This need to be done explicitly by the authorized parties which justifications.
* Example: The lecturer expectation is too high. The faculty need to normalized the student result for the subject by bring up all the student marks.

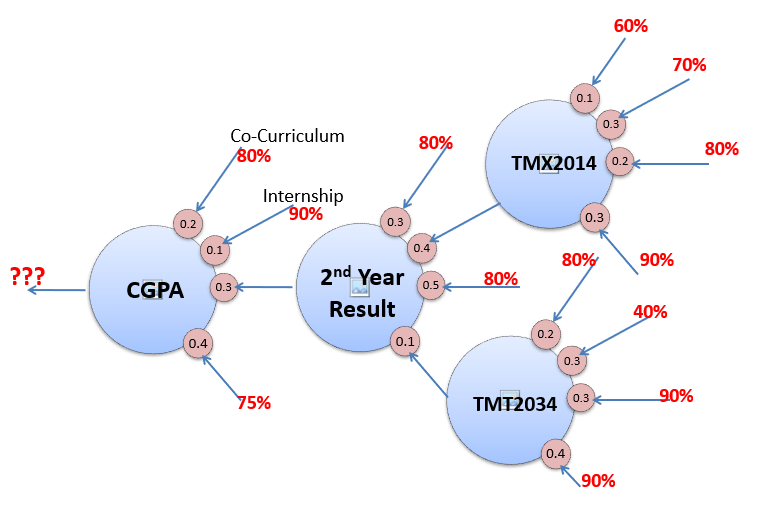
The Measurement System:



Root MU

* When linking measurement units, it forms a measurement system.
* The System has following properties
  1. The number of inputs for each measurement unit can be varies.
  2. The output each measurement units is linked to other measurement units as inputs, except the root measurement unit.
  3. To get the output of the root measurement unit will “fire” other measure units to generate outputs.
  4. Those inputs not link to any measure unit should link to any other value source

**CGPA System:**



**The mini project requirements:**

**Part-1:** Design and develop a set of foundation classes to implement the abovementioned measurement system. (This part should implement in Class Library)

**Part-2:** Apply these foundation classes and test it by solving the CGPA problem above. You can hardcode those terminal input values, or scores in this context (60%, 70%, etc). (This part should implement in Console project)

**Part-3:** Design and develop tables in a database under Microsoft SQL Server to keep students’ data and their scores for subjects. The design of database can be done using the SQL Server Management Studio by using an SQL Script file so that the trainer can recreate the database and records for assessment. The script should include all the database object and records creation. (This part dealing with SSMS)

**Part-4:** Demonstrate the unit test for the foundation classes developed with a Test project. Since this trainer is not for tester, just perform few unit tests will do. (This part should implement in Test Project)

**Part-5:** Create an GUI interface for user to manage the database. No need to make it too fancy. Just to demonstrate the DQL and DML with ADO.NET (This GUI should implement in WinForm)

**Part-6:** Write a simple web application just to show the result of each students’ final mark or other useful information to web browser page. (This part should implement in either ASP.NET)

**Part-7:** Create a Web Service to provide information about the measure settings in XML file. Test this web service with a console project by pulling the information from web service calls to form the measurement system dynamically. (This part should implement in SOAP-based WebService)

**Project Submission:**

Place all these projects in a single solution with name “MiniProject”. Write a simple deployment guide for the for the trainer to deploy your mini project to other computers. This is needed for the trainer to evaluate your mini-project. Clean and compress the entire project solution. include SQL script file and deployment guide (or other needed files) in a single Zip file with your <name>.zip. Send it to the trainer email ([ckleng1964@gmail.com](mailto:ckleng1964@gmail.com)) as attachment.

**Marking Scheme:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Measurement** | **Aspect** | **Weight** | **Description** |
| **Functionality of various project types** | Part-1 | 10% | Class Library |
| Part-2 | 10% | Console |
| Part-3 | 10% | DB |
| Part-4 | 10% | Unit Test |
| Part-5 | 10% | WinForm |
| Part-6 | 10% | WebForm |
| Part-7 | 10% | Web Service |
| **Language Features** | Parameters | 2% | Passing Machanisms, params, etc.. |
| Object Instantiation | 2% |  |
| Inheritance | 2% |  |
| Properties | 2% | get, set |
| Collections | 2% | Array, ArrayList, Generic List |
| Interface/Abstract Class | 2% | Apply any one of this |
| Polymorphism | 2% |  |
| Exception Handling | 2% | try, catch, multiple catchs, finally |
| Constructor | 2% | Overloading, Chaining, etc |
| Access Modifiers | 2% | Private, protected, public, internal |
| **Others** | ADO.NET | 5% | DQL, DML |
| Deployment Documentation | 5% | Information needed for the trainer to deploy the project for evaluation. |