# SOEN 487 – Winter 2020
# Assignment-III

2×9% (18%) of the Final Grade

Due Date: Apr 20, 2020 – 11:59 PM
Group Assignment: A maximum of three per team

## Using Web Services (2)

**Objectives**

The purpose of this assignment is to create a small application with an exposed secure public API.
Note: The topic of the implementation has already been approved.

**Overview**

In this assignment you are developing a small application on a topic of your choice that has been already approved. The list systems along with the development teams have been posted. You may choose any programming language / framework of your choice.
Using a framework / development platform other than the ones used in the class is encouraged **(R0)**.
The Project consists of the following components:

- The System Core
- The Web Service API
- The Web UI

Regardless of the development framework, the boundaries of the above three must be preserved **(R1)**.

***The System Core***

The system core implements the core of the system. It is essentially the business layer of your projects. All business objects, entities, etc. are defined in here **(R2)**.
It would be fine in case you do not use an object-oriented environment (i.e. functional environments, etc.)
It would be OK if you use non-standard business objects. For instance, using json data as business entities are completely fine.
In case you are implementing an API-like system (i.e. file conversion, image processing, etc.), this module MUST NOT be implemented as a web service. **(R3)**
It would be OK to use external web services. The System core may a client to external / public web services.
The system core must be secure. **(R4)**. It means, no calls are allowed, unless the caller is authenticated.
You may use a small user-database (a table or a configuration file) for authorized clients **(R5)**.
Any configuration items (i.e. connection to the data sources, external web services, etc.) must be put in a configuration file **(R6)**.
All errors / exceptions MUST be handled. A proper error codes / messages must be returned / thrown **(R7)**.

***The Web Service API***

The Web Service API exposes the System Core functionality via a standard web service interface, such as Resource API, Message API, or RPC API.

The Web Service API is a thin wrapper on the System Core and MUST NOT implement any business logic **(R8)**.

Possible data conversions may be required. In that case only utility, helper, data convertors, and mappers are allowed **(R9)**.

The Web Service API must receive security credentials from the client and pass them on to the system core. No additional authentication is required **(R10)**.

The Web Service API must properly handle errors, regardless of underlying layer. In case of REST implementation, appropriate error handling (via HTTP response codes, or via response data) must be implemented. In case of using SOAP implementation, proper SOAP fault must be generated **(R11)**.

***The Web UI***

The Web UI component is a small Web-UI thin client that is implemented on top of either the System Core or the Web Service API, allowing the end user to use the system.

NO business logic implementation is allowed in this layer **(R12)**.

**Deliverables**

**D1. The System Specification**

Provide a system specification document that describes the functionalities of the system, including the following components:

- Description of the system
- Functions of the system
- Use Case Diagram
- Class / Package Diagram*
- The API specification (list of services, methods, parameters, and error codes)

* In case a non-object oriented platform is used, components may be identified as classes.

**D2.** Implementing the three components, addressing the non-functional requirements R1 to R12.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). You and your teammate must be in the same section. Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted for both members. You must make sure that you upload the assignment to the correct directory of **Assignment 3** on Moodle. Assignments uploaded to the wrong directory will be discarded and no resubmission will be allowed.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention:

The zip file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID(s) number. For example, for the first assignment, student

12345678 would submit a zip file named *a1_12345678.zip*. If you work on a team and your IDs are 12345678 and 34567890, you would submit a zip file named *a1_12345678_34567890.zip*.

Submit your assignment electronically via Moodle based on the instruction given by your instructor as indicated above. **Please see course outline for submission rules and format, as well as for the required demo of the assignment.** A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. **You must keep a record of your submission confirmation.** This is your proof of submission, which you may need should a submission problem arises.

**Grading Scheme**

| D# | MX | MK | |
|---|---|---|---|
| D1.0 | /13 | | *spec text (4), use-case diagram (3), class / package diagrams (6)* |
| D1.1 | /6 | | *API spec* |
| D2.0 | /2 | | *R1* |
| D2.1 | /29 | | *System Core: overall functionality (20), R2-4 (3), R4(3), R5-R7 (3)* |
| D2.2 | /28 | | *Web Service: overall functionality (20), R8-R11 (8)* |
| D2.3 | /22 | | *Web UI: overall functionality (20), R12 (2)* |
| R0 | | | *bonus may apply* |
| Total: | /100 | | |

(D# - deliverable number, MX - max (out of), MK - your mark)