School of Information Technology and Electrical Engineering
INFS3204/7204 – Service Oriented Architecture

Assignment Three (10 Marks)

Due at 23:59 Sunday 08/10/2017

The goal of this assignment is to explore the WCF Framework and ASP.NET Web MVC Applications. All assignments will have to be developed with **Microsoft Visual Studio** using **C#** as the programming language. No other languages will be accepted. This assignment contributes to 10% of your overall grade. You must demonstrate this assignment to your lab tutors during your enrolled lab session in week 10. No Demo, No Mark! It is your own interest to ensure that your codes have been submitted to the BlackBoard before 23:59 Sunday 08/10/2017. You have only three attempts to submit your codes. Penalty will be applied for late submission. All submissions need to be in the form of zip/rar file format.

You are required to develop a Book Store Management System using ASP.NET MVC and WCF in this assignment and the assignment is divided into 2 tasks:

- **Task 1:**
  - **Building A Book Storage Web Service Using WCF** (2 Marks)
  - **Building A Book Store MVC Application and Invoking the Book Storage Web Service** (3 Marks)
- **Task 2:**
  - **Build A Book Purchase Web Service Using WCF** (3 Marks)
  - **Adding a new action method and a new view for Book Purchase into the Book Store Application** (2 Marks)

Please Note: The GUI of Prac Three is objective and no marks will be allocated to the layout design of this assignment. But you are responsible for all exception handling in both service and client sides in all required functions. During the demonstration, if your application crashes because of unhandled exception (either service or client side), you will lose marks for that function. For input validation, only the data types listed are accepted, no other types are allowed.

Preparation

- Before attempting to do this assignment, you should have the basic knowledge of how to use Microsoft Visual Studio 2015 or 2017 to create ASP.NET MVC Application and WCF service, as well as the basic knowledge of programming with C#. Please note that you need to choose .NET Framework 4.6 for your assignments.
- Notes: Please be advised that you need to meet the basic requirements for this assignment, which are mentioned in this document. However, you may try to design and implement your work more interestingly and comprehensively.

Task 1

Part I - Book Storage Web Service (2 Marks)

Data will be stored in a text file. The content of the text file is similar to the image below and the sample file is available in the Blackboard.

```
0470376848,Service Oriented Architecture (SOA) For Dummies,Judith Hurwitz,2009,$25.27,3
0470476248,Service-Oriented Architecture: Concepts Technology and Design,Thomas Erl,2005,$46.36,2
0485965215,Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services,Thomas Erl,2007,$43.16,1
0456235975,SOA in Practice: The Art of Distributed System Design,Nicolai M. Josuttis,2007,$24.59,1
0487965321,SOA Made Simple,Lonneke Dikmans,2012,$13.72,2
0412359687,SOA with REST: Principles Patterns & Constraints for Building Enterprise Solutions with REST,Thomas Erl,2012,$48.25,3
```

Create a WCF web service and define the following custom type (user-defined class) using data contract and data member (Please note that each class should have the specified fields with the specified data types):

1. **Book** (**0.4 Mark**)
   a. **string** ID
   b. **string** name
   c. **string** author
   d. **int** year
   e. **float** price
   f. **int** stock

This WCF web service should have the following 2 operations (Please note that each operation should only receive the specified parameters with the specified data types):

1. ReadBooks() (0.8 Marks):

   This operation reads all books information from file books.txt (Download from Blackboard) and returns a list of books. The return type should be **List<Book>** or **IEnumerable<Book>**.

2. **WriteBooks() (0.8 Marks):**

   This operation receives a list of books and writes them into the file books.txt. The parameter of this operation should be List<Book> or IEnumerable<Book>.

Part II- Book Store MVC Application (3 Marks)

Please Note: As the domain class **Book** has been defined in the **Book Storage Web Service,** you cannot define it again in the models of the MVC application**.** You can only use it as a proxy class in your MVC application. You are not allowed to directly read or write the file books.txt in the MVC application, and they must be done by invoking the operations in **Book Storage Web Service.**

Book Store MVC Application consists of the following elements (actions and views):

1. GetAllBooks() (0.75 Marks):

A page that lists all book records from Data Storage Web Service. (Note that Num is the index of a book in this book list and is not recorded in book.txt)

All Books in Store

Num	ID	Name	Author	Year	Price	Stock
1	0470376848	Service Oriented Architecture (SOA) For Dummies	Judith Hurwitz	2009	$25.27	3
2	0470476248	Service-Oriented Architecture: Concepts Technology and Design	Thomas Erl	2005	$46.36	2
3	0485965215	Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services	Thomas Erl	2007	$43.16	1
4	0456235975	SOA in Practice: The Art of Distributed System Design	Nicolai M. Josuttis	2007	$24.59	1
5	0487965321	SOA Made Simple	Lonneke Dikmans	2012	$13.72	2
6	0412359687	SOA with REST: Principles Patterns & Constraints for Building Enterprise Solutions with REST	Thomas Erl	2012	$48.25	3

2. AddBook() (0.75 Marks):

A page (similar to the image below) that allows users to add more book records to the Book Store Application. An AddBook() controller is needed for receiving the information of a book from the view and creating an instance of class Book, then it will add this book into existing book list and save the information to the text file via Data Storage Web Service.

Note that: The new book list after this operation needs to be displayed on the top of the page. Input validation is needed, and the input cannot be null. All the input values are subject to proper validation.

All Books in Store

Num	ID	Name	Author	Year	Price	Stock
1	0470376848	Service Oriented Architecture (SOA) For Dummies	Judith Hurwitz	2009	$25.27	3
2	0470476248	Service-Oriented Architecture: Concepts Technology and Design	Thomas Erl	2005	$46.36	2
3	0485965215	Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services	Thomas Erl	2007	$43.16	1
4	0456235975	SOA in Practice: The Art of Distributed System Design	Nicolai M. Josuttis	2007	$24.59	1
5	0487965321	SOA Made Simple	Lonneke Dikmans	2012	$13.72	2
6	0412359687	SOA with REST: Principles Patterns & Constraints for Building Enterprise Solutions with REST	Thomas Erl	2012	$48.25	3
7	0467297423	SOA	Kevin	2015	$20	2

Add ID 0467297423 Name SOA Author Kevin Year 2015 Price 20 Stock 2

3. DeleteBook() (0.75 Marks):

A page (similar to the image below) that allows users to remove book records from the system. An DeleteBook() controller will need to be created for the request from the corresponding view. For example, the field is year, and the value is 2007. Then, it deletes all the books that match the condition, and save the rest of the books back to the file via Data Storage Web Service. If the operation is successful, it returns true. Otherwise it returns false.

Note that: Input validation is needed, and the input cannot be null. All the input values are subject to proper validation.

All Books in Store

Num	ID	Name	Author	Year	Price	Stock
1	0470376848	Service Oriented Architecture (SOA) For Dummies	Judith Hurwitz	2009	$25.27	3
2	0470476248	Service-Oriented Architecture: Concepts Technology and Design	Thomas Erl	2005	$46.36	2
3	0485965215	Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services	Thomas Erl	2007	$43.16	1
4	0456235975	SOA in Practice: The Art of Distributed System Design	Nicolai M. Josuttis	2007	$24.59	1
5	0487965321	SOA Made Simple	Lonneke Dikmans	2012	$13.72	2
6	0412359687	SOA with REST: Principles Patterns & Constraints for Building Enterprise Solutions with REST	Thomas Erl	2012	$48.25	3
7	0467297423	SOA	Kevin	2015	$20	2

Delete Year 2007

Num
ID
Year

4. SearchBook() (0.75 Marks):

A search page (similar to the image below) that allows users to search book records in the system. For example, one user is using year as the search condition. Then, the page displays all the books that match the condition.
Note that: The supported fields should include Name, ID, Author and Year (Using Dropdownlist). Input validation is needed, and the input cannot be null. All the input values are subject to proper validation.

Task 2

Part I – Book Purchase Web Service (3 Marks)

Create a **WCF** web service and define the following custom types (user-defined classes) using message contract (**Please note that each class should have the specified fields with the specified data types**):

1. **BookPurchaseInfo (0.5 Mark)**
   a. **float** budget
   b. **Dictionary<int, int>** items

The budget means the total amount of money you will spend on purchasing books. For items, the key (int) is the Num (the index of a book in the book list file in task 1) of the book you want to buy, and the value (int) is the number of books with the Num you want to buy. The protection level for **items** should be highest (EncryptAndSign), since it is the user's sensitive information.

2. **BookPurchaseResponse (0.5 Mark)**
   a. **bool** result
   b. **string** response

The result indicates whether this purchase succeeds.

1) If the cost of books you choose is greater than your budget, this purchase fails, then the result is false, and the response should be "No enough money".
2) If the amount of books you want to buy is greater than the stock of the books, then it fails and the result is false, the response should be "No enough stocks".
3) If the purchasing succeeds, the response should be the number of money left after purchasing the books you choose.

This WCF web service should have the following operation (**Please note that each operation should only receive the specified parameters with the specified data types**):

1. **PurchaseBooks (2 Marks):**

   Input: **BookPurchaseInfo**

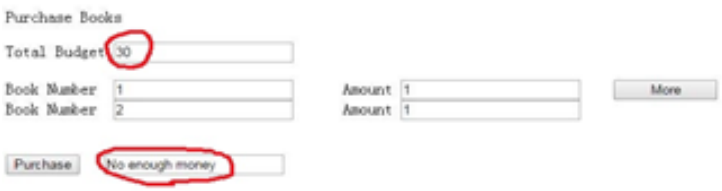
   Output: **BookPurchaseResponse**

This operation receives a **BookPurchaseInfo** and returns a **BookPurchaseResponse**.

Part II – Adding BookPurchase to the Book Store Application (2 Marks)

You are required to add a new action method and a new view for **BookPurchase** in the Book Store MVC Application. In the action method, you should call the operation **PurchaseBooks** implemented in Part I rather than implementing business logic again. Please find an example of the view below. Note that the "More" button is used to automatically create more fields to input the purchase items. All the input values are subject to proper validation:

- Total budget: positive float
- Book Number: positive integer, within the range of book list
- Amount: positive integer



--- End of Document ---