# CSE445/598 Assignments 1 (50 Points)

# Spring 2024

Assignment 1  Due: Saturday, September 2, 2023 by 11:59pm (Arizona Time),
plus a one-day grace period.

---

## Introduction

The aim of this assignment is to make sure that you understand the concepts covered in the lectures and in the text, including SOA, SOC, SOD, and their applications in software development. You will also follow a tutorial to obtain hands on experience of developing a simple service and a simple application that uses services. By the end of the assignment, you should have applied these concepts in developing simple services and simple applications that uses remote Web services.

## Section 1  Preparation Part (No submission required)

Reading: Textbook Chapter 1 and Appendix A. Chapter 1 gives you the basic concepts and Appendix A shows you the tutorials that create the operation applications.

No submission is required for this section of exercises. However, doing these exercises can help you better understand the concepts and thus help you in writing quizzes and exams.

1. Answer the multiple choice questions in Textbook section 1.7. Compare your answers with the answers given in the course web page in "Module 1 Course Overview and Introduction" > "Activities and Assessments" > "SOCSI_Exercises_Keys.pdf." Doing these exercises will help you prepare your weekly quizzes, the chapter tests, and the exams, as scheduled in the course calendar.

2. What are SOA, SOC, SOD, SOE, SOI, and SOSE? Briefly state their definitions based on your understanding.

3. What are the main differences between requirement analyses in the OOC paradigm and in the SOC paradigm?

4. What are the major benefits of separating an application builder from the service providers?

5. What are the main techniques in SOSE (service oriented system engineering)? For each technique, write one or two sentences to describe its purpose.

6. Compare and contrast the traditional software development process and the Service-oriented software development process. For each step of the development, write a paragraph to describe the purposes, responsibilities, functions of the step.

7. What is a service registry? What is a service repository? What are their differences?

8. An electronic travel agency needs to be developed. What is your responsibility if you are:

8.1  a service provider?

8.2 a service broker?
8.3 an application builder?

9. You plan to invent a unique online game.

9.1 Describe what you must do as an application builder, and what you can expect the service providers to do for you.

9.2 Describe your invention idea and list everything you must do as an application builder.

9.3 List everything that you can possibly find through service brokers.

10. List a few application areas where you believe SOC is a better fit than OOC. State your reasons and justifications.

11. What are the impacts of SOC paradigm to the IT market and to computer science graduates?

12. Download Visual Studio. You can download Visual Studio 2019 or 2022. After downloading, you need to add Windows Communication Foundation to be able to create a WCF service by running Visual Studio Installer. Mac VS may not support full VS features. Such as Windows Communication Foundation.

## Section 2 Tutorial Exercises: Using a Web Service in Application (No submission required)

These tutorials complete the lectures. They help you to complete the assignment questions in Section 3. If the services mentioned in the tutorial are not available, use another given in text appendix C (or check online at: https://venus.sod.asu.edu/WSRepository/) instead. Please also read text Appendix A for more details.

**Tutorial Exercise 1**: Creating a simple Web service

This tutorial shows how a service provider develops Web services for the application builders' use. The recommended platform is Windows Visual Studio **Community 2019** or **2022**. The Mac Visual Studio may not have the required template.

Step 1:

Start Visual Studio and choose "Create a new project". You can search "WCF" and then choose "WCF Service Application" template, as shown in Figure 1.

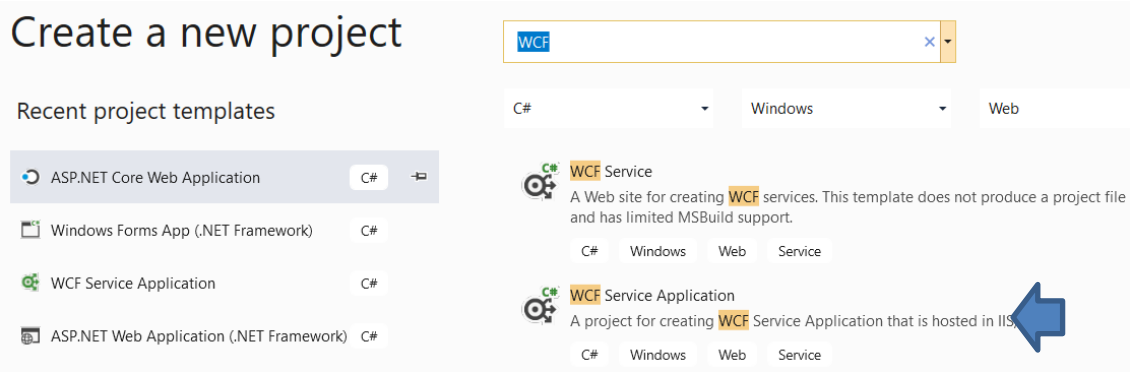



**Figure 1.** Creating a new project and choose WCF Service Application template

If you do not find this template, you need to start Visual Studio Installer. Run Visual Studio Installer and modify your Visual Studio adding components. When you add new components, you will not see Windows Communication Foundation in the Workloads tag. You need to click the "Individual components" tag and add Windows Communication Foundation, as shown in Figure 2.


**Figure 2.** Install Windows Communication Foundation through individual components

Step 2: Follow Textbook, Chapter 3, Section 3.2.1 or A1 Tutorial given in Assignment 1 folder, to develop a simple Web service using Visual Studio.

**Tutorial Exercise 2**: Using WCF Test Client

Follow the text Chapter 3, Sections 3.2.2 AND 3.2.3 or lecture slides to test your service using WCF test client.

**Tutorial Exercise 3**: Creating a Web Application to consume the service that you developed in Tutorial 1.

In Visual Studio, choose "Create a new project". Search "web" and choose ASP .Net Web Application. A project with a Default.aspx page will be created, among other files, as shown in Figure 3. Note, do not choose the template with "**Core**" in the name.

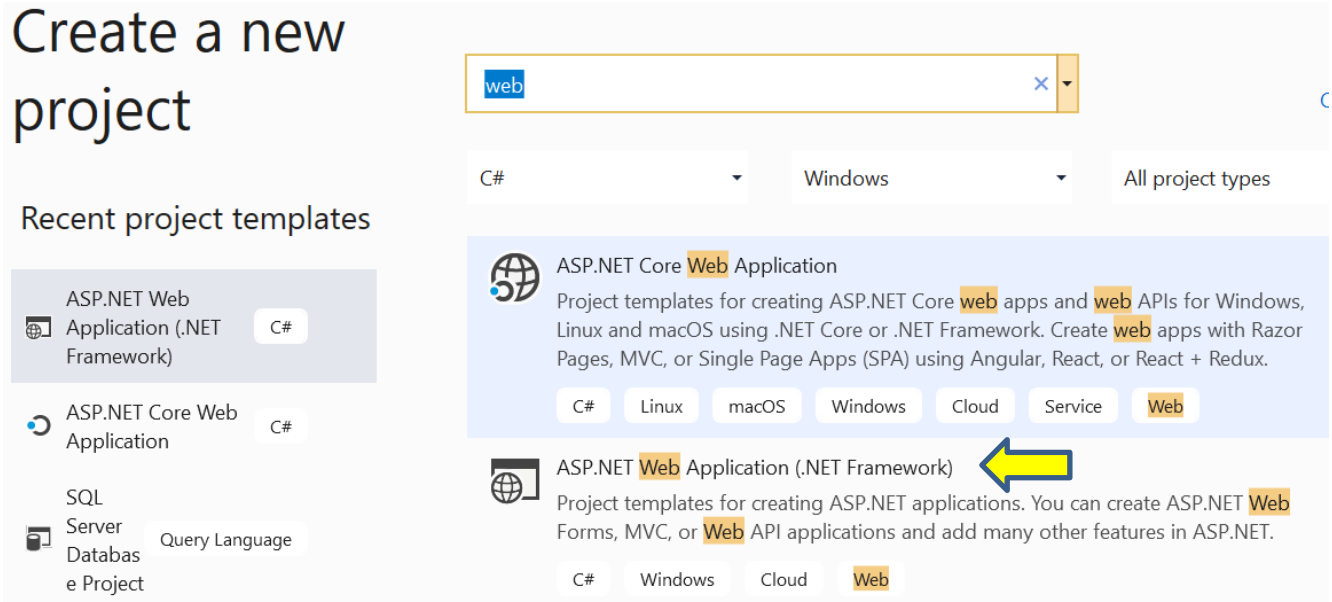



**Figure 3.** Create an ASP .Net Web Application

Next, you can follow the lectures and textbook, Section 3.6.3 to complete the Web application development.

This tutorial shows you how an application builder makes use of remote services to create an application that provides a Web GUI for accessing Web services.

**Tutorial Exercise 4**: Creating a Windows Forms App to consume the service that you developed in tutorial 1.

Although Web applications are more widely used, developing a desktop application is still needed in certain situations. This tutorial shows how to develop a Windows Forms-based desktop application.

In this tutorial, you will choose Windows Forms App (.Net Framework) as the template, as shown in Figure 4.

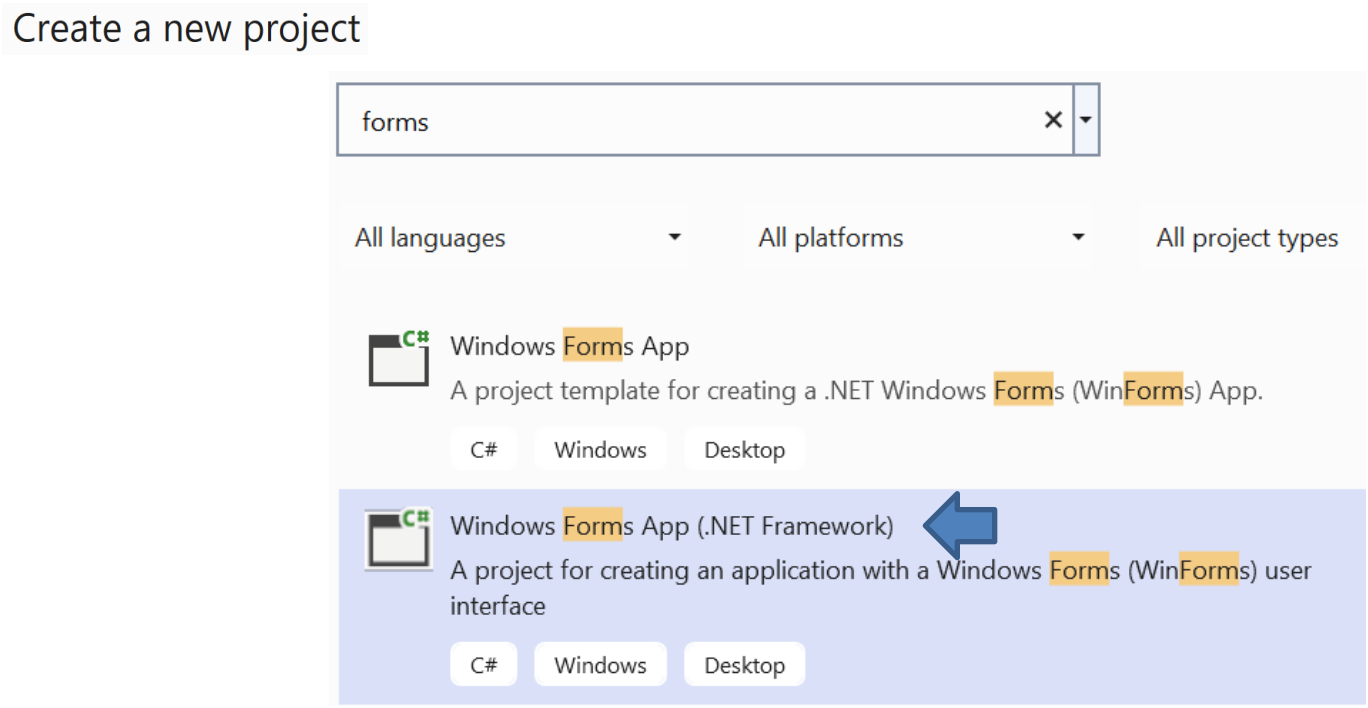



**Figure 4.** Create Windows Forms-based desktop application

For the remaining steps, you can follow the lectures and the textbook, Appendix A.1 and Chapter 3 Section 3.6.2, or follow Microsoft document to develop a Windows Forms App (.Net Framework):

https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022

You can further follow Text Section 3.6.3 to use the WCF services in your application. This tutorial shows you how an application builder makes use of remote services (.svc services) to create an application that provides a GUI for accessing Web services.

> Notice that, in order to test your application, you must have the service started first to make the object an active object! You can start the service by right-clicking the file **Service.svc** in the project and choose "View in Browser." Then, you will see the service URL in the browser address bar. Use this URL when you chose "Add Service Reference…".

**Tutorial Exercise 5**: Creating an image verifier using the image (CAPTCHA) service based on: http://venus.sod.asu.edu/WSRepository/Services/ImageVerifierSvc/Service.svc and http://venus.sod.asu.edu/WSRepository/Services/ImageVerifierSvc/TryIt.aspx

Follow the tutorial given in the textbook, Appendix sections A.3 to develop a Web application that verifies if a human user is entering a Web form. An image verifier is frequently used for preventing programmed attacks to a Web site that allows self-registration.

## Assignment 1: Web Services (Submission required, 50 points)

Assignment 1 is an individual assignment. Each student must perform and submit independent work. No cooperation with anyone else is allowed in completing individual assignments.

1. Follow the Tutorial Exercise 1 given in **Section 2 of this document** to develop a temperature convention Web service. The service contains two operations: [10 points]

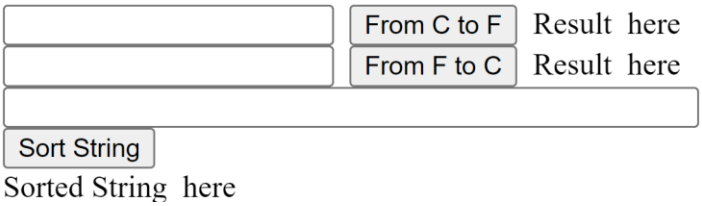
   int c2f(int c); // convert Celsius temperature to Fahrenheit temperature

   int f2c(int f); // convert Fahrenheit temperature to Celsius temperature

2. Develop a Web service to sort a string of real (float) numbers, separated by space. It returns a string of numbers, sorted and separated by commas. You can choose any sorting algorithm or use a sorting library function in this question. The service contains one operation: [10 points]

   string sort(string s); // sort a string of real (float) numbers, separated by space, returned numbers are separated by comma.

3. Follow the Tutorial Exercise 3 given in Section 2 of this document to develop a Web Application (called a TryIt page) to consume the temperature service and number sorting service that you developed in Questions 1 and 2. The service must be running on localhost in web browser when your application calls the service. Do not close the browser that hosts the Web services. The figure below shows a sample GUI design of the application. [10 points]




4. Based on Tutorial 4 given in Section 2 of this document and following the tutorial in text Appendix Section A.1, create a Web browser that can take any URL and display the content of the page in the text window that you created. You can also google search "Making a Web Browser in Visual Studio" to find the online document on this topic. [10 points]

5. In this question, you will add more features into the web browser that you created in the previous question. Choose two sub questions only from the following set of sub questions. If you implement more than two, we will grade the first two only. [Each sub question is 5 points]

| |
|---|
| 1) Add a simple calculator in your Web browser, with floating point +, -, *, and / operations. No service call is required in this question. [5 points] |

2) Add text encryption decryption function in your browser. Follow the example in text 3.6.3. However, instead of using the localhost service, you must use the service in the ASU Repository at: http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc [5 points]

3) Add the Get Stock Quote function in your browser. You can use the stock simulator service (http://venus.sod.asu.edu/WSRepository/Services/Stockquote/Service.svc), or discover your own real stock service (such as https://www.google.com/finance/), to build your application. For a given stock symbol, e.g., F, IBM, GOOG, TSLA, you must display all the values (one or more) returned in a readable format. [5 points]

4) Follow Tutorial 5 in this document to add the image verifier (CAPTCHA) service into your Web browser). [5 points]

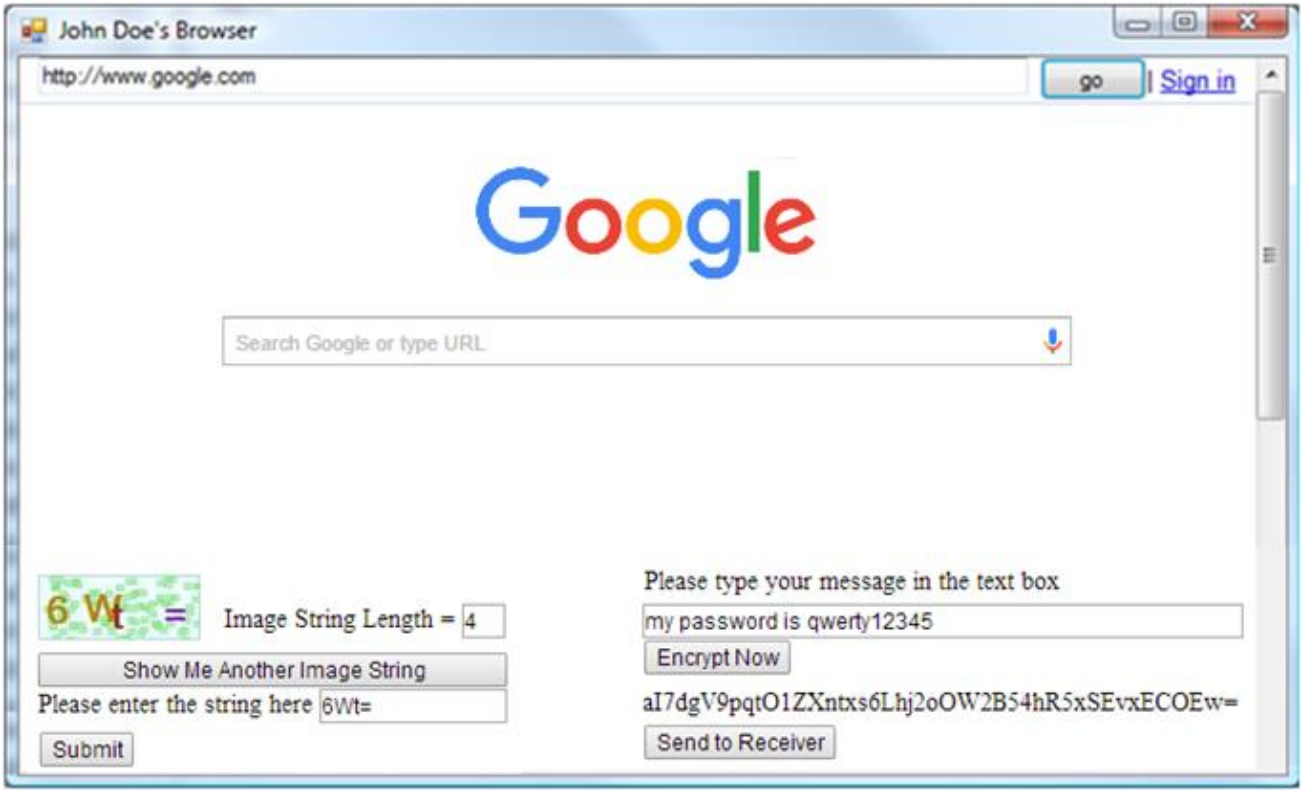
5) Find a temperature Web service (e.g., https://www.meteomatics.com/en/weather-api), and add the temperature into your browser. You can use any other temperature Web service or API that you can find. [5 points]

6) You can discover more services at http://venus.sod.asu.edu/WSRepository/repository.html or other public service repository and use one of the services in this assignment, except the Basic Three service. [5 points]

Note: From 2022, ASU requires all Web servers to use https (http with SSL security). http access will be denied. Thus, when you create an application to access any WSDL web service on ASU server, you need to open the application's Web.config file of your application and change all basicHttpBinding to basicHttpsBinding.

The figure below shows a sample layout of your browser. Notice that the sample components in the sample is different from what is required in this assignment. You must design your own layout to best display the required information. However, all parts of the information must be displayed in a **single page**.

If a particular service is not working, you can use another one with the same level of complexity, for example, the same number of input parameters.




## Assignment 1 Submission Requirement

You can put all assignment questions into a single Visual Studio solution. You may also put questions 1, 2, and 3 in one solution, and put questions 4 and 5 into another solution.

All submissions must be electronically submitted to the assignment folder on Canvas. Complete solutions with all the files must be submitted, so that the solutions can be unzipped and tested. Put all solution folder(s) into a single umbrella folder. Zip the umbrella folder for submission.

If you have saved a project/Website in a different folder, you can copy the folder containing the project/Website to the directory where the other projects are saved. Then go into Visual Studio and delete the project/website that was in a different place. Then right click the solution in Visual Studio and add existing project/website, browse to the new location and select the project/website to link the moved project/website into the solution.

Submission preparation notice: The assignment consists of multiple distributed projects and components. They may be stored in different locations on your computer when you create them. You must copy these projects into a single folder for Canvas submission. To make sure that you have all the files included in the zip file and they work together, you must test them before submission. You must also download your own submission from the Canvas. Unzip the file on a different machine, and test your assignment and see if you can run the solution in a different location, because the TA will test your application on a different machine. If you submitted an empty project folder, an incomplete project folder, or a wrong folder, you cannot resubmit after the submission linked is closed! We grade only what you submitted in the Canvas. We cannot grade the assignment on your computer or any other storage, even if the modification date indicated that the files were created before the submission due dates.

The Canvas submission may take a long time if the file is big. If your solution folder is too big in size, you may open the folder and identify and delete some unnecessary files, e.g., different language packages. After any deletion, you must test your program and make sure that you did not delete any necessary files.

## Late submission deduction policy:

- Grace period (Sunday): No penalty for late submissions that are received within 24 hours of the given due date.
- 1% grade deduction for every hour after the first 24 hours of the grace period (from Monday through Tuesday!)
- No submission will be allowed after Tuesday midnight. The submission link will be disabled at 11:59pm on Tuesday. You must make sure that you complete the submission before 11:59pm. If your file is big, it may take more than an hour to complete the submission!

## Grading and Rubrics

Each sub-question (programming tasks) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each sub-question, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

Please note that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each question / sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

| Compile | Code passed compilation | | | | Code failed compilation | | |
|---|---|---|---|---|---|---|---|
| Points | *pts* * 100% | *pts* * 90% | *pts* * 80% | *pts* * 70% - 60% | *pts* * 50% or 40% | *pts* * 30% or 10% | 0 or 1 point |
| For each sub-question | Meeting all requirements, well commented, and working correctly in all test cases | Working correctly in all test cases. Comments not provided to explain | Working with minor problem, such as not writing comments, code not working in | Working in most test cases, but with major problem, such as | Failed compilation or not working correctly, but showing serious | Failed to compile, showing some effort, but the code does not implemen | **0** points if no submission<br>**1** point if submitted files that do not answer any |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | what each part of code does. | certain uncommon boundary conditions. | the code fail a common test case | effort in addressing the problem. | t the required work. | questions required, for example, the files are empty, cannot be open, wrong files, etc. |