CSE 445 Project 5 (Assignments 8 and 9) (50+50 Points)

Fall 2022

Project 5 A8 Due: Saturday, November 19, 2022, 11:59pm, Plus a one-day grace period.

Project 5 A9 Due: Monday, November 28, 2022, 11:59pm, Plus a one-day grace period. No late submission for A9.

A8 Individual submission: Each member submits an individual assignment.

A9 Group submission: One group one submission. No duplicate submission. Duplicate copy can be submitted into the **Backup** Submission folder

Please read the submission and grade calculation detail at the end of this document.

Both assignments A8 and A9 must be submitted into the Canvas submission site AND into the WebStrar Server. We will read code from your Canvas submission and test your code from WebStrar

---

Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures, including the Web application architecture, components and structure, controls, and state management.

This is an individual and team project and the continuation of project 3. The project must be submitted into the Canvas AND into the Web server WebStrar.

As this project is critical and also it can impact the other team members, assignment 9 grade will not be dropped within the drop-the-lowest assignment grade policy. Please read the grading policy in Day 1 Lecture on “Weight and Grading Scale”. Note, assignment 8 grade can be dropped. However, you are still required to do assignment 8 in order to make sense for your assignment 9, even if you do not submit assignment 8. If you do not do project 8 work, your assignment 9 contribution will be reduced!

A team must work together to complete the project in a collaborative and coordinated manner. It means that the team members must discuss and know what other team members are doing and how they are doing, and how their components work with each other. On the other hand, a large part of the project has to be done independently and individually, which means that each member will take the lead in a part of the project, and each member will be individually assessed in the grading process.

You need to be in a team of 2 or 3 members, and you need to use WebStrar server. If your team has 2 or 3 members, you must continue with your team. If you are the only member left in your team, you need to form a new team with those who are also in one-member teams. You can also join a current team that has two members only. I have created a new forum called "Team building for Project 5". Please use that forum to form your Project 5 team if you are in a single-member team. After you joined a new team, you will continue to use your server assigned to you in Project 3. Note, we cannot change the server! Please read the FAQ Document in the Team Building folder in the course web page.

After you have created your new team, you must inform the TA and instructor of your new team and members.

Practice Exercises (No submission required)

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

1. Reading: Textbook Chapter 5 and Chapter 6 section 6.2.

2. Do the multiple-choice exercises in text for Chapter 5 and Chapter 6 for practice.

3. What kinds of Web-based computing models exist? Where is the computation (client side or server side) done in each of the models?

4. Explain how the files in ASP .Net Website application are organized in the application folder.

5. Explain what types of files exist in an ASP .Net Website application and what the functions of each type file are.

6. How do we create a user control, and how do we include the user control into a Web page?

7. What is the most frequently used function of the Global.asax file?

8. What kinds of state-saving mechanism exist, and what are the main features of each kind state-saving mechanism?

9. What is the most general state variable in ASP .Net? What type of data can be stored in this kind of state variable?

10. Compare and contrast the state management mechanisms: View State, Cookie, Session State, Application State, and Caching variables.

11. How are dynamic graphics generated and maintained in ASP .Net environment? Read text section 5.6.

12. What is the execution model of ASP .Net application in the tightly managed Web server?

13. Study for the questions 2 through 12 in text section 5.8, and study the questions 2 through 15 in text section 6.5. Make sure that you understand these questions and can briefly answer these questions. Study the material covered in these questions can help you prepare for the quiz and exam, and can help you understand the homework assignment.

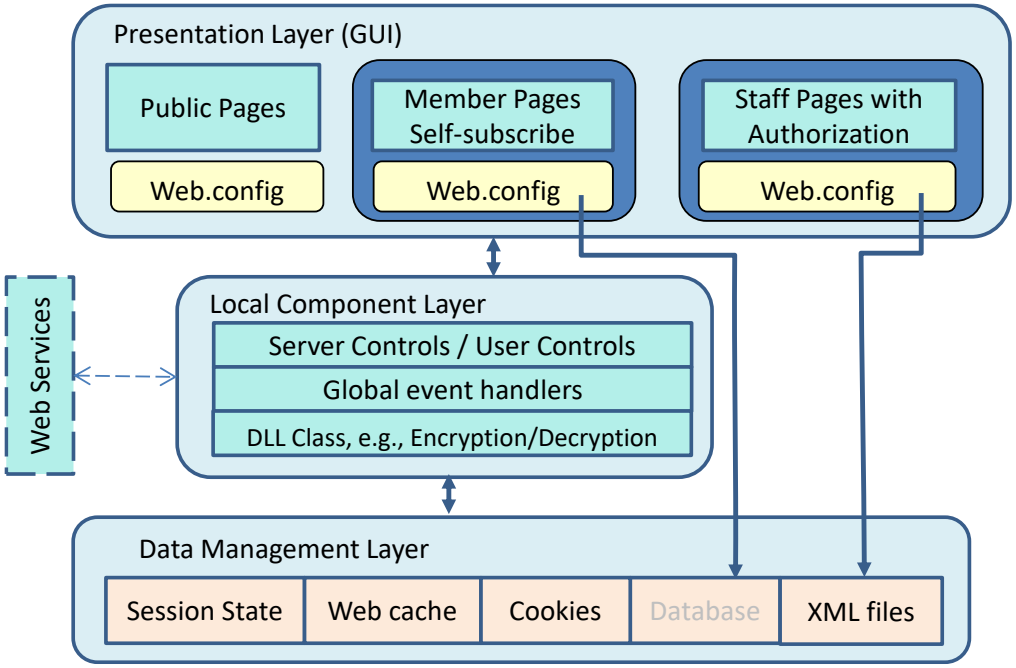
14. Having learned all the techniques that you need to build Web applications, now, you need developing your own Web application. Brainstorm among the team members and come up with your ideas of developing a sensible Web application. Ideally, this project solves a problem that you always wanted to solve, for example

    - Managing membership, finance, and activities of the club that you are an officer.

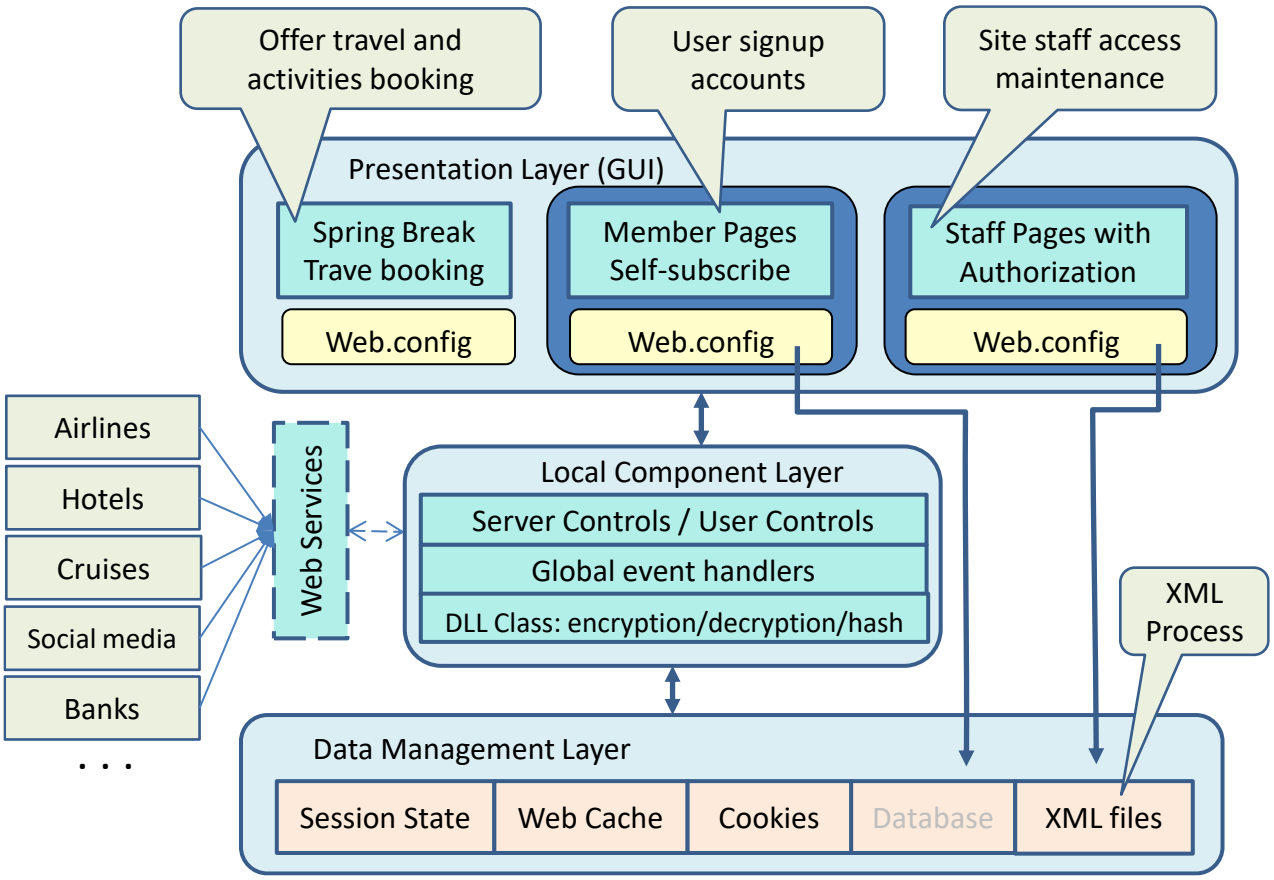
- Managing your hobby collections: When was each item purchased, price, current value.
- Loyalty center: allow businesses to check loyalty of a costumer based on buying history and habit;
- Developing a Web testing tool: analyzes a Web page at a given URL and calls the links in the page to discover broken links;
- An online store with recommendation list, like Amazon, eBay, etc.,
- Combine this project with your other course projects. In this case, you need to clearly state what part is submitted for this course, and what part is submitted for another course.

Project 5 (Assignments 8 and 9) Description: Submission Required

In this project, your team will develop a service-oriented Web application. The project must simulate a real or realistic application for the end users. The required architecture is shown in the following figure.



As an example of this general project requirement, the following diagram shows an instance. It is an example for you to better understand the requirement. You may or may not follow this example.



The system must be implemented as an **ASP .Net Web Application** (**ASP .Net Framework**) and must be deployed to the given Web server. Do not use ASP .Net Core Web application.

Your team may implement the application and use the services that you have outlined and developed in Project 3 and possibly in Project 4, or choose to implement a different application and use different services.

The application that you implement in this project must meet the following organizational, architectural, and functional requirements. The code must be well commented to explain how you implement the functions. The composed application must have at least the following layers of components.

1 Presentation/GUI layer and server controls, consisting of ASPX pages and server controls, which allow users to interact with the application using Forms Security. The application must have at least the following ASPX pages: [10 point for assignment 8 and 20 points for assignment 9]

a. Default page. This is a public page. You must introduce clearly what functionality the application offers, how end users can sign up for the services, how the users (TA/grader) can test this application, and what are the test cases/inputs. The page must contain a button to access the "Member" page and a button to access the "Staff" page. Login redirection must be implemented in the access controls: If the user is not logged in, the Login page will be activated. If the user has already logged in, the user should not see the Login page. In addition, all the components and services used in the application must be listed in a "Service Directory", similar to the one that you created in Project 3. The directory must include: provider name (member who is responsible for the component), component type (Web service, DLL function, user control, etc.), operation name, parameters and their types, return type, function description, and a link to a TryIt page (or

an item in the web presentation layer GUI) if the test interface is not explicitly implemented in the web application page. You can combine the TryIt pages into your application GUI and distribute them into the required pages (Default, Member, and Staff, and other GUI pages that you create) listed in this question.

b. Member page: In this page, you must introduce (explain) clearly what functions this page offers. You must allow users to register (self-subscribe) to obtain access to this page. For example, the account information page or the shopping cart page can be Member pages. An image verifier must be used when a new user signs up. You must create your own access control component and store the credentials in an XML file called Member.xml. Only the authenticated members can access this page. The password must be encrypted or hashed when adding into the XML files. You must use a local encryption/decryption or hash function that your team developed as a DLL library function. Calling the encryption/decryption/hashing Web service is not acceptable, as the password may be sent to the server in clear text.

c. Staff page: Access to this page is given by the administrator, who adds a staff member's username and password into an XML file called Staff.xml. You can either use a GUI for the administrator to add the credentials, or let the administrator to open the Staff.xml to add the credentials. For allowing the TA/grader to test this page, you must allow this staff credential to test this page: User name: "TA" and Password: Cse44598!

2 Local component layer (individual work). This layer consists of the following **types** of components: [20 points for assignment 8]

a. Global.asax file with at least one reasonable event handler.

b. DLL class library modules, to implement at least the hash or encryption/decryption functions.

c. Cookie for storing user profile and Session state for storing temporary states for sharing among the sessions.

d. User control. For example, put a captcha in a user control or a login window as a user control.

e. XML file manipulation, such as the searching and adding elements into the Member.xml file.

All **types** of components must be implemented in the application. Each member must choose at least two types of the local components from the list above to implement. Exception: If you end up with a one-member team, you must implement at least three types of components above. If all types have been covered, team members can choose the same **types** of components to implement, but not the same components. For example, two team members can implement a different user control in their part.

3 Remote service layer (individual work), consisting of sensible Web services developed by each team member. Each team member must develop at least one Web service that is used in the application. You can use the services that you developed in Projects 3 or 4, or develop new services. The services must be deployed into WebStrar.
Your team can use any number of public services discovered, including those in ASU repository. It is the developers' (your) responsibility to make sure that the services are available and reliable when the TA/grader grades the project. [10 points for assignment 8]

4 Integration of all layers and all components, including the GUI pages, local components, services, and data management consisting of both temporary states (session state / cookie) **and** permanent

states. You must use XML files as your permanent data storage to implement your application. Do not use text files or database in this project. [20 points for assignment 9]

5 Deploy all the components into the server for testing.
[10 points for assignment 8 and 9 points for assignment 10]

6 Submit into Canvas the code for grading (code reading) and the link to the server deployment for testing. Without submitting to Canvas, the points for all questions in Assignments 8 and 9 will be reduced by 50%!

Assignment 8 Submission (Individual Work and Individual Submission, 50 Points)

Each member must submit individual work (complete working solutions) into the Canvas submission site AND into Web server. We will read the code from your Canvas submission and test your code from WebStrar. You should submit your own parts of code only. However, if all parts from all members have been integrated into one application, and your parts will not work without other's parts, you could submit other's code in your submission. In this case, you must make clear what parts are done by you and you claim credits for your parts only.

You must submit your server deployment URLs of the ASP .Net web pages into the Canvas, so that we can find and test your deployment. The main URL of the application must be in the Default page.

The questions mentioned below correspond to the same questions in the project description section.

Question 1: You submit the Application and Components Summary Table (see an example of the table at the end of this document) that lists all your components, so that we can test your components. If your team has integrated all components, you can submit the integrated table. In this case, you must make clear which components are developed by which members through the provider name, so that we can find your contribution in the table. The page must have a TryIt function for each component that you developed. You can combine the TryIt pages into your application GUI and distribute them into the required pages (Default, Member, and Staff, and other GUI pages that you create) listed in this question. If your components are implicitly used and are not visible/testable from your team's integrated GUI, you must add test access points to make them testable by, for example, adding additional TryIt buttons and labels. For example, hashing the password is not testable -- tester cannot test if you have implemented the hashing function using the normal GUI. Thus, you should add a button and a label to show that you have actually implemented the hashing function. [10 points]

To obtain this 10 points, you do not have to include all the application functions described in this project. Instead, you just need to include a part of the functions. The minimum requirement is to implement an ASPX page with the directory table that can invoke the components in questions 2 and 3. You can consider this page as a TryIt page for questions 2 and 3. However, this page must be a part of the overall page that your team will integrate in the Part 2 Submission.

Question 2: Submit the local components that you have developed individually and link them to your TryIt page in question 1. [20 points]

Question 3: Submit the Web service that you have developed individually, as well as the other services that you have used. In the TryIt it page, you must call the services. [10 points]

Question 5: Deployment. All code must be deployed into WebStrar. If your code is running on local host only, you will not receive the deployment points. [10 points]

Question 6: Each member submit code into Canvas for grading (code reading). Note, even if all your Assignment 8 code and implementation have been included in Assignment 9 submission, you still need to submit your individual assignment 8 in order to receive the points for assignment 8.

Assignment 9 Submission (Group Work and One Submission per Team, 50 Points)

Submit the **Integrated** Application into the Canvas submission site and into WebStrar server. It must include the integrated directory table showing all the components and the providers (team members who developed the components). Member page and Staff page must be implemented in assignment 9. One team member must be designated to submit the integrated application into the Canvas. In the Canvas submission, you must submit your WebStrar deployment URL of the main (Default) page, so that we can test your deployment. The points are assigned in the following questions:

Question 1: Your team (one team member) must submit the Application and Components Summary Table that lists all members' components, so that we can test all components and the integrated application. The Service Directory must make clear which components are developed by which members through the provider name, so that we can find your contribution in the table. The page must have a TryIt function for each component that the team developed. If the components are implicitly used and are not visible/testable from the team's integrated GUI, you must add test access points to make them testable by, for example, adding additional TryIt buttons and labels. [20 points]

Question 4: Integration of all layers and all components, including the GUI pages, local components, services, and data management consisting of both temporary states (session state / cookie) and permanent states. You must use XML files as your permanent data storage to implement your application. Do not use text files or database in this project. [20 points]

Question 5: Deploy all the components into the server for testing. [10 points]

Question 6: One team member must submit entire project code into Canvas for grading (code reading). Notice that if the team members have submitted the integrated application as Assignment 8, one of the team members still needs to submit the integrated application again into the Canvas submission site for Assignment 9. We will not take the submission from in Assignment 8 for Assignment 9 grading.

No duplicated submissions for Assignment 9. If two members submitted, we will deduct 5 points for failure in coordination, and if both copies are graded, we will use the lower grade as the team grade. Notice that we have limited TA/grader resource and we have a hard deadline to complete grading of this last project. We will create a backup submission site in Canvas. You cannot see from you're your Canvas whether or not your team member has submitted. If you are not sure whether the designated team member has submitted the project in the Canvas, you can submit a copy into the backup site, just to make sure. No grade deduction for submitting redundant copies into the backup submission site. We will not grade the submissions in the backup site. However, if no submission is found for a team in the assignment submission, we will grade one submission from the backup site.

Example Application and Components Summary Table

Application and Components Summary Table			
The application is deployed at: http://webstrarX.fulton.asu.edu/Default.aspx (X is your site #)			
Percentage of contribution: Member 1 name: 34%, Member 2 name: 30%, Member 3 name: 30% The percentage will be used to scale the grade. Note, if a member does not provide the components in assignment 8, the percentage of contribution will be reduced.			
Provider name	Page and component type, e.g., aspx, DLL, SVC, etc.	Component description: What does the component do? What are input/parameter and output/return value?	Actual resources and methods used to implement the component and where this component is used.
All the following contents are examples. You must enter your own contents that you implemented.			
Member 1 name	aspx page and server control	The public Default page that call and link all other pages	GUI design and C# code behind GUI
Member 1 name	User control	Log in page and authentication verification	C# Code behind GUI. Linked to the Default page
Member 1 name	XML file	Store usernames and passwords	Linked to the login page and the hashing function
Member 2 name	DLL	Hashing function: Input: String Output: String	Use library class and local component to implement this library function. It is used in the saving data into XML file.
Member 2 name	SVC service	Output annual KW number for a given panel size at a given zip code location. SolarPower Inputs: zip code and size Output: integer	Retrieve information from national database at: http://graphical.weather.gov/xml/ It linked into the Default page.
Member 2 name	Cookies	Store user ID and password. The password requires 8 characters, which must include upper and lower case letter, digit, and a special character.	GUI design and C# code behind GUI using HTTP cookies library. It is linked to the login page.
	…	…	…
Member 3 name	SVC service	Solving an equation system Inputs: input matrix Output: vector	This component is in finding the optimal solution among different conditions and parameter values
…	…	…	…

Make sure that you have the deployment URL in the second row of the page!

Members Grade Calculation for Team Project

For Assignment 8, it is an individual assignment. Normal rules apply. For Assignment 9, it is a team assignment.

For a team assignment, the member grade *Gi* for member *i* is calculated and scaled through the formula:

$$G_i = S * \frac{N * P_i}{\sum_{i=1}^{N} P_i}$$

where

G_i: The grade of member *i* (G_i <= 50. If G_i > 50, set to 50)

S: the overall score for Assignment 9 before scaling (*S* is between 1 and 50)

P_i: Contribution percentage of member *i* (P_i is between 1% and 100%)

N: The number of active members in the team (*N* = 2 or 3). If a member dropped or does not make any contribution, the member does not count in calculation.

General Submission Requirement

All submissions must be electronically submitted to the assignment folder where you downloaded the assignment paper. All files must be zipped into a single file.

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 choose your own location to store the project when you create the project. Then, you can copy these projects into a single folder for the blackboard 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 blackboard. Unzip the file on a different machine, and test your assignment and see if you can run the solution in a different machine, because the TA will test your application on a different machine.

If you submitted an empty project folder, or an incomplete project folder, we cannot grade your resubmission after the due date! We grade only what you submitted before the submission due date. Please read FAQ document in the course Web page for more details.

Late submission deduction policy:

- Grace period: No penalty for late submissions that are received within 24 hours of the given due date.
- For A8 only: 1% grade deduction for every hour after the first 24 hours of the grace period (from Monday through Tuesday!
- For A9, no late submission! The submission link will be disabled on Tuesday at 11:59pm!
- No submission will be allowed after Tuesday midnight for any assignment. 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 notice 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 sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% -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 what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed to compile, showing some effort, but the code does not implement the required work.	0 points if no submission 1 point if submitted files that do not answer any questions required, for example, the files are empty, cannot be open, wrong files, etc.