# CSE 445/598 Assignment 5: The Final Project (100 Points)

# Fall 2014

Part 1 Due: Saturday, November 22, 2014, 11:59pm (Arizona Time),   
Part 2 Due: Friday, December 5, 2014, 11:59pm (Arizona Time). No Grace Period for Part 2.

Both parts must be submitted into the Blackboard’s submission site AND into the WebStrar. We will read code from your Blackboard 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 a team project and the continuation of assignment 3. The project must be submitted into the blackboard AND into WebStra server.

Although a team works together to complete the project in a collaborative and coordinated manner, a large part of the project has to be done independently and individually.

# 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.
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. Discuss the relationship between the dependency and callback in the insert() method of Cache class.
12. How are dynamic graphics generated and maintained in ASP .Net environment? Read text section 5.6.
13. What is the execution model of ASP .Net application in the tightly managed Web server?
14. 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.
15. Having learned all the techniques that you need to build Web applications, now, you need develop 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.,
    * A mobile application to manage your shopping list and best places to buy, taking into account time, driving distances, and cost.
    * 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 Description (Submission Required)

In this project, you will develop a service-oriented Web application. The project must simulate a realistic application for the end users. The required architecture is shown in the figure below.



The system should be implemented as an **ASP .Net** **Website** **application** and must be deployed to the given Web server. As an alternative, your team could choose to use **ASP .Net** **MVC** architecture, instead of the Website application architecture. In this case, you can organize your components differently according to MVC architecture. However, you must still implement all the components defined in this assignment document. You can read textbook Chapter 5, Section 5.8.2 to learn MVC architecture.

Your team may implement the application that you have outlined in Assignment 3, or choose to implement a different application. The application implemented 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, consisting of ASPX pages and server controls, which allow users to interact with the application. The application must have at least the following ASPX pages: [10 point for part 1 and 20 points for part 2]
   1. Default page. This is a public page, and you must introduce clearly what functionality the system offers, how end users can sign up for the services, how the users (TA) can test this application and what are the test cases/inputs. All the components and services used in the application must be listed in a “Service Directory”, similar to the one that you created in Assignment 3. The directory must include: provider name (member who is responsible for the component). Component type (Web service, DLL function, user control, caching, etc.), operation name, parameters and their types, return type, function description, and link to a TryIt page if the test interface is not explicitly implemented in this Default page. You can combine the TryIt pages into your application GUI and distributed them into the required pages listed in this question.
   2. Member page: In this page, you must introduce clearly what functions this page offers. Users can register (self-subscribe) to obtain the access to this page. An image verifier must be used when a new user signs up. You can use the built in Account management for this purpose, which will save user name and password into the build-in database. You may create your own access control component and store the credentials in an XML file.
   3. Admin page: This page allows the administrator to add user name, password, and a role into an XML file. There are two possible roles: Manager and Staff. The password must be encrypted or hashed when adding into the XML file. 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. Only the administrator can access this page. For the TA to test this page, you must allow this admin credential to test this page: User name: TA and Password: CSE445598ta!
   4. Manage and Staff pages: These two pages must have authentication and authorization access control. The managers can access both Manager page and Staff page, while the staff members can access the Staff page only. The user name, password, and role are stored in an XML file that is maintained by the administrator. You are not allowed to use the built-in account management for this part, in order to exercise XML manipulation learned in Chapter 4. Since the password is encrypted or hashed in the XML file. You must use a local encryption/decryption/hashing 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.
2. Local component layer (individual work). This layer consists of the following types of components: [20 points for part 1]
   1. Global.asax file with a reasonable event handlers.
   2. DLL class library modules, to implement at least the hash or encryption/decryption functions.
   3. Cookie for storing user profile and Session state for storing temporary states for sharing among the sessions
   4. XMLfile data caching with cache dependency and cache removal functions (Do not select this unless you want to read the Web caching lecture before it is given)
   5. User control, and the control must have the fragment caching function implemented.
   6. Code behind the server controls that implement the presentation layer.

Each member must choose at least two components from the list above. For the first four components (a, b, c, and d), no more than one team members can choose the same components. For the last two components (e and f), multiple members can choose the same component, but they must implement different functions using these components. For example, two members can choose user controls to implement their different functions.

1. 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 service that you developed in Assignment 3. 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 TAs grade the assignment. [10 points for part 1]
2. Integration of all layers and all components, including the GUI pages, local components, services, and data management consisting of both temporary states (session state and cache) **and** permanent states (XML file and database) to implement your application. [20 points for part 2]
3. Deploy all the components into the server for testing. [10 points for part 1 and 10 points for part 2]
4. Submit code into Blackboard for grading (code reading).

# Part 1 Submission (Individual Work, 50 Points)

Part 1 Due: Saturday, November 22, 2014, 11:59pm (Arizona Time),

Each member must submit individually the following work into the Blackboard submission site AND into the WebStrar. We will read code from your Blackboard submission and test your code from WebStrar. The questions mentioned here correspond the same questions in the project description section.

**Question 1**: You submit the Service Directory 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. Make clear the part of your work through the provider name, so that can find your contribution in the table. The page must have TryIt function for each component that you developed. If your components are implicitly used and is not visible/testable from your team’s integrated GUI, you must add the test access points to make them testable, for example, add additional TryIt buttons and labels. [10 points]

For this question, you do not need to include all the 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 that will 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 question1 [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: Submit code into Blackboard for grading (code reading).

# Part 2 Submission (Group Work, 50 Points)

Part 2 Due: Friday, December 5, 2014, 11:59pm (Arizona Time). No Grace Period for Part 2.

Submit the **Integrated** Application into the Blackboard submission site and into the WebStrar. One team member must be designated to submit into the Blackboard. No double submissions. 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 resource and we have a hard deadline to complete grading of this last assignment. We will create a backup submission site in Blackboard. If you are not sure whether the designated team member has submitted the project in the blackboard, you can submit a copy into the backup site. 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 Part 2 submission, we will grade one submission from the backup site.

Notice that, if your team has submitted the integrated application in Part 1, one of the team members still need to submit the integrated application again into the Blackboard Part 2 submission site. We will not take submission from Part 1 for Part 2 grading.

# Grading of computer programs

The TA will grade your program following these steps:

(1) The TA will read your program and give points based on the points allocated to each component, 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.

(2) Compile the code. If it does not compile, 40% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 12 if the program fails to compile.

(3) If the code passes the compilation, the TA will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 20% of the points given in (1) will be deducted.

(4) The TA will test the deployed application in the server assigned to your team.

Please notice that the TA will not debug your program to figure out how big or how small the error is. You may lose 40% or 20% of your points for a small error such missing a comma or a space!

# Final Project Submission Policy:

For Part 1 submission, the normal late submission policy applies.

For Part 2 submission, the following submission policy applies:

* No grace period;
* Five percent (5%) grade deduction for every hour after the submission due date! Notice that the penalty is 5% per hour, instead of 1% in the previous assignment, as we do not have buffer zone at the end of the semester for any delay. For example, if your team submits Part 2 at 10am on Saturday, you are 10 hours late, and each of the team members will lose 50% of your Part 2 points. If your team submits Part 2 at 8pm on Saturday, you are 20 hours late, and each of you will have 0 grade for Part 2.