**INB / INN 373**

**Web Application Development Assignment**

Worth: 40%

Due: 27th May (proposal due 2nd April)

**Introduction**

Web sites containing interactive and dynamic content today allow us to:

* Buy and sell shares.
* Use electronic home banking services.
* Buy books and other commodity items.
* Pay telephone and other bills.
* Pay our taxes.
* Enrol in university courses.
* Buy and sell second hand goods at auction.
* Play poker, chess, etc
* etc

Your task in this assignment is to create an interactive and dynamic web site using .NET technologies. You will need to meet various implementation requirements, but the actual content of the web site is entirely up to you. Be creative; don’t just go for the same old boring electronic shopping cart style web application. Choose an application area that you find exciting and interesting; strive to think of something slightly different from the norm. Feel free to use it as an opportunity to develop a prototype for a web application that may be of future use to you, your employer, a club or an organization you belong to. You might also consider it as an opportunity to create a showcase application that you can show to potential employers in the future. Don’t, however, let real world needs dictate your requirements to a point where the application becomes too large. The goal is to create a relatively small application that is extremely well designed, architected and implemented.

**Group Work**

You are to complete this assignment in groups of 1, 2 or 3, (ie, entirely by yourself, or with one or two other members of the class). Groups of size two are strongly encouraged. If you work with anyone else then you must make only a single submission on behalf of the entire group. A group may not implement the same application idea as another group.

**Plagiarism Warning**

Your group must develop 100% of the code submitted for the assignment. It may not be based either in whole, or in part on any existing web site implementation developed either by you or by others. Doing so will be regarded as plagiarism and may result in you obtaining a zero grade for the assignment.

**Basic Requirements**

1. The focus of this assignment is on the development of interactive and dynamic web pages. Don’t waste time and effort creating lots of “fancy” pages containing only static content - you will not get any extra credit for them.
2. Your application should be able to be viewed in any standard web browser, such as Internet Explorer, Netscape or Opera (although it may look slightly different in different browsers) and on any operating system. This means that you should produce only standard HTML and ECMA script for client side processing, i.e. no client-side ActiveX or .NET controls.
3. Your web application should persist all important information in a server-side SQL Server database. Your web application should both read and write this information.

**Approval**

Before commencing work on your project you must submit a proposal to the unit coordinator that briefly outlines your project:

1. What’s the application generally about and why you chose it.
2. What functionality will the user interface provide (ie what will the main web pages allow you to do)?
3. What information will be stored in the database?

You should submit your proposal in hard copy to the tutors in the practical classes. Feel free to talk to the unit coordinator about your ideas prior to submitting your proposal. The unit coordinator may wish to discuss and possibly refine your submitted proposal. Your proposal must be submitted prior to the April 2 deadline or marks will be deducted from you final assignment grade. Please try to get your proposals in well in advance of this date to avoid the final rush - the sooner the better; you can always submit a modified proposal if you later change your mind.

**Logical Architecture**

Your application must be developed with 3 logically separate tiers (see Figure 1). The components coloured blue are the parts you need to develop. Each tier should be implemented as a separate component (.NET assembly) in order to emphasize this logical separation.

**The Presentation Tier**

The presentation tier will consist of a number of ASP.NET web pages (.aspx)**.** You should implement at least two such web pages, but definitely no more than ten. The presentation tier should contain only presentation layout (HTML) and basic presentation logic. Any “deeper” application logic should be implemented in the business tier. Any presentation logic (C# or VB.NET source code) that is present in the presentation tier should be contained only in code behind files (i.e., do not make use of the ability to embed server-side source code directly in amongst the HTML in the aspx page itself).

**Data Tier**

SQL Server

Database

**Presentation Tier**

**Business Tier**

.NET Framework

ASP.NET

ADO.NET

Web Pages

incl.

Code Behind

IIS

Web Server

**IE or**

**Netscape**

Web

Browser

Internet

Data

Access

Objects

Business

Logic

Objects

Tables

Stored

Procedures

Figure 1: Logical 3-Tier Architecture

**The Business Tier**

The business tier will consists of one or more .NET classes that implement short lived business actions. Note that business objects are not used to maintain state from one web request to the next. Any state that needs to be persisted between successive web requests should be managed by the data tier. In the simplest case, these business classes contain only static methods; alternatively, new instances of these business objects can created as required by the presentation tier for each incoming web request. The business layer is responsible for enforcing business policies and procedures, so all access to the data tier should be done via the business tier, even if only to more easily cater for future business policy extensions. The business tier is also responsible for presenting the raw information from the data tier in a more business-relevant fashion.

**The Data Tier**

The data tier consists of one or more .NET classes which provide “wrapper” methods that use ADO.NET to call SQL queries or stored procedures. The data tier may also define typed data sets which are used in some cases to return information to the business tier. The data tier is also responsible for opening and closing database connections. Such connections should be left open for as short a period as possible to aid scalability.

The data tier also consists of the tables, stored procedures, views, indexes, triggers, etc that you create in the SQL Server database. You should start by designing a schema for your database, i.e. a list of your tables and the columns contained within each table. The schema should be designed in such a way that information can be retrieved easily and accurately but without storing unnecessary or redundant information. Your tutor will be able to provide help if you are not experienced in database schema design.

**Levels of Achievement**

Your assignment will be assessed from both an external and internal point of view.

1. The external assessment will consider the:
2. Extent of the functionality
3. Correctness of the functionality
4. Simplicity and Intuitiveness of the UI.
5. The internal assessment will consider the:
6. Adherence to coding best practice.
7. Adherence to architectural best practice.
8. Appropriate use of a wide variety of optional requirements (see below).

**Minimum Requirements**

The requirements described so far constitute the minimum requirements that you need to satisfy if you wish to pass the assignment. If you are aiming for more than simply a passing grade (> 65%) then you should aim also to meet one or more of the following optional requirements. If you are aiming for a grade of 7 then you should aim to meet many of these optional requirements (note: you are **not** required to meet **all** of these optional requirements). Remember however that these optional requirements constitute only part of the marking criteria, and that it essential that you meet the minimum requirements in an elegant fashion before contemplating any optional requirements. It is strongly recommended that you develop your application in an incremental fashion, where you first develop a working system that meets all of the minimum requirements before contemplating any of the optional requirements.

***A working prototype of one of your pages (successfully connected to the database) must be deployed to the fastapps04 server and demoed to your tutor during the week 9 tutorial session.***

**Optional Requirements**

* Use AJAX to provide a richer/more interactive user experience.
* All access to the database from the data access layer should be done via stored procedures. Stored procedures are more efficient than issuing queries dynamically. They can also help to protect the integrity of the database and provide data in a more business oriented form.
* One of the beauties of .NET is the ability to use a wide variety of programming languages. Illustrate this ability by using a variety of programming languages to implement your assignment. For example, you might use C# to code the business and data access components and VB.NET to code your ASP.NET web pages.
* .NET allows us to go beyond traditional N-tier systems by using web services to integrate separate applications. Illustrate this capability by exposing and/or consuming a SOAP style web service. Note that QUT’s firewall and internet volume charging mechanisms may make it difficult for you to access web services that are hosted outside of QUT.
* Add security to your application by implementing ASP.NET Forms authentication and authorization.
* Create and use a web forms user control, or for the more adventurous, a ASP.NET Server Control.
* Make use of advanced web controls such as DataGrid, Repeater, Calendar, etc.
* Use data binding to link web controls to data sources. Do not, however, link web controls directly to tables in the database; all data should still be accessed via the business and data access tiers.
* Use typed datasets.
* Use ASP.NET performance optimization techniques such as caching.
* Use COM+ Enterprise services such as transactions.

The preceding list is intended only to give you examples of some of the advanced features of .NET that can be used to make your application better or make your development easier. You are encouraged to seek out other **appropriate** .NET features that you can also make use of productively. Your assignment should be a showcase of such features, but make sure that the addition of such features does not turn your original simple and elegant solution into an ugly mess. Simplicity and elegance must take priority over additional features.

**Physical Architecture and Deployment**

All of the development and the majority of your testing can and should be performed on a standalone Windows PC. The PC will need a:

* Web browser such as Internet Explorer,
* IIS (comes with most versions of Windows OS),
* Visual Studio.NET 2012,
* The .NET Framework version 4.5
* SQL Server 2012

All of this software is available in the S block labs. It can also be burnt to a CD (refer to Microsoft [MSDN academic alliance](http://software.fit.qut.edu.au/msdnaa/guide.html)) and installed on home computers (for educational purposes only).

Ultimately, however, you will need to deploy your web application to a central Windows server machine from which it will be tested and graded by tutors. Your web pages, business objects, data access objects and SQL Server will still all reside on the same physical machine.

Each INB/INN373 student will have their own directory and database on the server which they alone are authorized to access. Groups should deploy their web application to the directory and database of only one of its members (recorded in your assignment proposal). The default entry page for the group’s site should clearly list the full names and student numbers of all students in the group, together with basic information about your web site and how to use it. This documentation should all fit on **one screen**. Your user interface should be designed to be simple and intuitive to use, so your grader and others shouldn’t have to read pages of instructions in order to be able to use your site effectively.

**Deployment Instructions**

You must deploy your application to the **fastapps04** Windows server. You will not be able to login interactively to this server machine but you will be able to upload your files to a directory that has been created for your use.

To access the server from off campus you will need to use a VPN connection (<https://secure.qut.edu.au/itservices/qut/connectoffcampus/sas/index.jsp>).

Use Windows Explorer to access your directory using a UNC path, e.g. \\fastapp04.qut.edu.au\*n1234567*

If asked to authenticate, make sure you enter your username as "QUTAD\n*1234567*".

You should copy your entire web site to your directory on the server, including all binary, source, configuration and project files. In addition to copying your files to the server, you will need to configure your web application directory(s) as an IIS application directory using a specially designed web admin page:

http://fastapps04.qut.edu.au/Configure

Only the root directory of your web application(s) should be configured as an IIS application directory. You can create subdirectories within a web application but they should not be configured as IIS web applications – doing so will cause your application not to run.

This same page will be used to configure any .NET Enterprise service components that you might create.

You will be able to test your deployed web application from a web browser running either in the student labs or from home or work via, eg:

**http://fastapps04.qut.edu.au/n1234567**

Remember, however that all development and the majority of your testing should be done on your local PC.

You will have a disk quota of 10MB on the server and your SQL Server data and log files are each limited to 2 MB. It is your responsibility to make sure that your entire web application (including source code) fits within this quota.

You will also be able to connect to the instance of SQL Server running on the server in order to create tables, stored procedures, etc. You can use SQL Server Management Studio to connect to the server using:

Server type: Database Engine

Server name: fastapps04.qut.edu.au

Authentication: Windows Authentication

Note, this may not work from outside of QUT due to firewall restrictions.

**Coding Best Practice**

* Large volumes of comments are **not** required in your source code. The code however should be self documenting by carefully choosing identifier names and writing in a clear, simple and elegant fashion. Your code should read like a book. Code with the expectation that your code will be critiqued by thousands of people. Don’t simply consider “does it work?”, but rather “is it as clear, simple and elegant as it could possibly be?”
* Keep your methods short and simple, no user defined method should be longer than a screen.
* Don’t replicate code or logic.
* Use private methods and attributes where ever possible to maintain abstraction.

**What to Submit**

1. Deploy on the server (fastapps04):
2. All source, binary, configuration and project files.
3. A printed report submitted via assignment desk containing:
4. A list of any (admin) user names or passwords that might be required by graders to test your web application.
5. A brief summary of the status of your system including any known issues or bugs. [1 page max]
6. A web site flow diagram (see Figure A.1 of the textbook). [1 page max]
7. A single UML class diagram showing all user defined classes from all three tiers (partitioned into tiers). The diagram should also show the relationships (inherits, contains, uses, etc) between those classes expressed using correct UML notation. [1 page max, use A3 if necessary]
8. A printout of the SQL Server database diagram showing all user defined tables and their columns. It should also show the primary keys and foreign key relationships between these tables. You can create such a diagram by right clicking on the “Diagrams” tab of your database in SQL Server Enterprise Manager and select “New Database Diagram ...”. [1 page max]
9. A bullet point list of the optional requirements that you have implemented, including your own options not included in the suggested list of requirements. [1 page max]
10. **Do not** include a printout of any of your source code (we don’t have enough trees!)
11. **Do not** include anything else in your report. The entire report **must** be no more than 10 pages.