# .Net Web Registration Application

**Submitted by: Vasantha Gudiwada**
University of Illinois at Springfield
2356 William Maxwell Lane Apt # 401,
Springfield, Illinois-62703
(217)206-1310 [Home], (217)836-2308[cell]
vgudi01s@uis.edu

**Advisor: Kamyar Dezhgosha**
University of Illinois at Springfield
One University Plaza, MS UHB3111
Springfield, IL 62703-5407
Daytime: (217) 206-7243
kdezh1@uis.edu

**ABSTRACT**

The term n-tier refers to the number of logical levels or layers the various components of an application occupy. N-tier architecture provides a model for developers to create a flexible and reusable application. By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than rewrite the entire application over, if they decide to change technologies or scale up.

This paper reviews the design, development and implementation of an application based on n-tier architecture. From the end users point of view, this application abstracts the implementation details and provides them with the required features and facilities. It is also intended for users and developers in the field of information technology. This paper also provides literature overview on various architectures.

**Keywords:** N-tier technology, ASP.NET, C#.NET, ADO.NET, Required Field Validators, Regular Expression Validators.

# 1 Introduction

Creating N-tier applications is not a new concept. Since VB4 days Developers are building such applications. N-tier application architecture provides a model for developers to create a flexible and reusable application. By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than rewrite the entire application over, if they decide to change technologies or scale up. .NET provides great support for building such applications.

The .NET framework allows for applications written in different programming languages to integrate deeply with each other, enabling current development skills to go further without re-training. Memory, threads, and processes are managed by the .NET framework to ensure that memory leaks do not occur. ASP .NET monitors running web applications and can automatically restart them at administrator-defined intervals, thereby making the applications very reliable. With the elimination of DLLs (Dynamic Link Library), .NET framework has made it easy to deploy, run and manage applications. The .NET framework includes an evidence-based security system designed for today's Internet environments. By collecting evidence about where an application came from, who created it, its digital signature, as well as what the application is trying to do and combining that evidence with a security policy, the .NET framework's runtime environment can make very fine-grained decisions about whether to run an application.

## 1.1 Background

Center for Teaching and Learning (CTL) at UIS offers a large number of technology training workshops each semester for faculty, staff and students. At present, CTL does the registration process through 3 desktop applications like Word, Excel, Access and Outlook.
The author had proposed and developed a web application using .NET framework. This application uses n-tier architecture and can be deployed on any network-enabled system.
Some of the key features of the application would be:

- The students would be able to register and manage their course anytime with ease.
- The students would be able to do a look-up to find the list of courses that they have taken in the current and previous semesters.
- The administrators would be able to update and monitor the courses anytime with ease.
- The registration process would be fully automated with the elimination of using Excel, Word and Outlook completely.

## 1.2 Architecture of n-tier Applications

N-tier design came about as a result of the failings of the client/server model. There are many goals that an n-tier application design should achieve. Here are some of them.

- If the underlying data access methods are changed, the client-side code should not have to change.
- All data access routines should be exposed as objects instead of function calls. For example, it is much easier to use ADO than the ODBC API calls.
- SQL should be eliminated from the client-side code. The client code should just be concerned with methods and properties.
- Table and column names should be eliminated from the client-side code. Typed datasets can present table and column names as properties, providing an IntelliSense list, as opposed to having to type in a string name. This means at compile time, checks can be made for data types and names of columns.
- The client code should not care where the data comes from. It should just care that it can retrieve and modify the data in some object and the object will take care of the details.
- The coding on the client side should be simplified. Instead of using many functions, the application should be able to use objects with properties and methods.
- It becomes easier to create and use the classes than the function calls.
- It becomes easier to add functionality to your applications, and change the functionality, without breaking the client-side code.

(Mitchell)

In n-tier architecture the entire application is divided into several pieces. These pieces can be logical or physical. Each piece performs a specific task such as displaying user interface or data access. There can be any number of tiers or layers of such pieces. Hence, the name n-tier. Many times the terms tier and layer are used interchangeably. However, most commonly, applications have 3 distinct tiers or layers. They are:

- Presentation Layer
- Business Logic Layer
- Data Access Layer

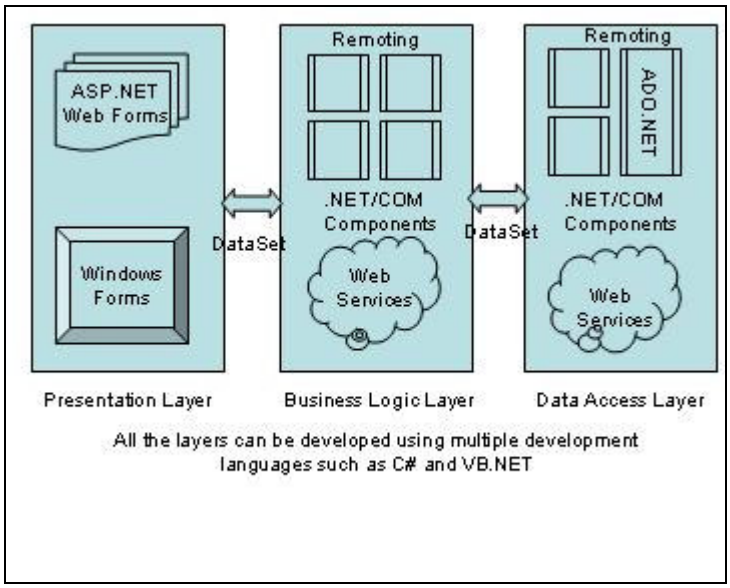



***Figure 1: A Typical N-Tier Model***

Presentation layer is a piece of software that deals with user interface of your application. Displaying data to the end users and allow them to interface with it is the core functionality of this layer. In most of the cases the data entered by the end users need some kind of validation or further processing. This is the responsibility of Business Logic Layer. Finally, the application data needs to be stored and retrieved in some data store (RDBMS, XML etc.). Data Access Layer handles this task.
The basic steps of the process are:

- User requests for some application data.
- The data access layer retrieves the data and is forwarded to the presentation layer via business logic layer. Sometimes data access layer gives this data directly to presentation layer.
- Presentation layer receives the data to be displayed via business logic layer.
- The user changes the data and initiates the appropriate action (such as insert, or update).
- The business logic layer validates the data submitted by the user.
- If the data is valid it is handed over to data access layer for updating into the database.

(Joshi)

### 1.2.1 Presentation Layer

This is the layer that provides an interface for the end user into your application. It works with the results/output of the Business logic layer to handle the transformation into something usable and readable by the end user. There are two main options for creating presentation layers in .NET. They are: Windows Forms or ASP.NET Web Forms. Using windows forms you can create traditional form-based desktop applications. Windows form applications can offer rich user interface elements and response times to the user.

However, they suffer from the drawback that they need to be installed on each and every machine. Though they can be used in "smart client" mode they are more or less the same as traditional VB forms. The most appealing option to develop presentation layer is ASP.NET web forms. Traditionally web pages suffered from the drawback of limited user interface elements. ASP.NET web server controls bridge the gap to a great extent. Controls such as DataGrid, DataList and Calendar provide rich UI in very few lines of code.
(Miller)
Presentation layer options can be coded in variety of languages such as C# or VB.NET. Here, again support for multiple development languages comes handy. You can use your choice of language to develop the UI.

### 1.2.2 Business Logic Layer/Business Tier

Business logic layer primarily consists of components that perform the task of business validation, business workflow and other similar things. This layer has nothing to do with HTML, ADO or SQL. .NET components form this layer. COM components can also be used in .NET via interop. However, this will degrade the performance.
ASP.NET Web Service can also serve as business logic layer. However, they should not be used as replacement to components in all the situations. They should be used only if the business validation happens at some "external" place than your network. The components that are developed need not always reside on the same machine. Using .NET remoting you can create and distribute them on multiple physical machines.
(Miller)

### 1.2.3 Data Access Layer

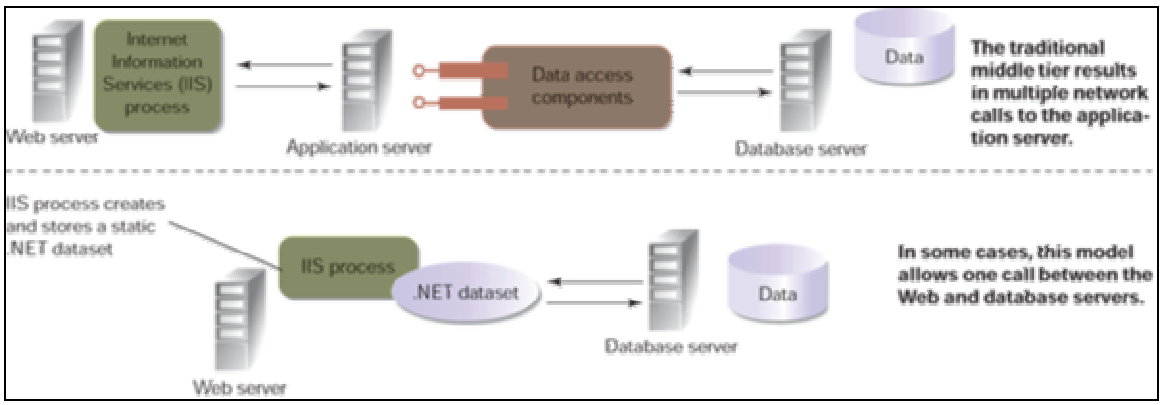



*Figure 2: Comparison of traditional middle-tier data access and .NET dataset*

Traditional middle-tier data access (Figure 2) might fall by the wayside when compared to a .NET dataset cached on the local server (bottom). This not only decreases latency, it improves application throughput by decreasing overall network utilization.

This layer contains some generic methods to interface with your data. This Data Layer contains no data business rules or data manipulation/transformation logic. It is merely a reusable interface to the database.

This layer deals with data manipulation actions such as inserts, updates, deletes and selects. The data can reside in RDBMS or XML files or even flat files. The data access layer should be designed in such a way that other layers need not have any knowledge about underlying data store.
(Miller)
ADO.NET is the data access technology under .NET. Though ADO.NET allows connected data access via DataReader classes, more focus is on disconnected data access. DataSet plays a key in this mode. In some rare cases you can also use ADO for data access but its use should have valid reasons. Again .NET components form this layer. As stated earlier, classic COM components may also be used. Web services can also form data access layer. This is especially true if the database does not have a data provider. In such cases, some custom code may be written to connect with the data and populate DataSet with it and then return DataSet to the caller. In addition to ADO.NET built-in RDBMS capabilities such as stored procedures and functions can also be used.
(Yang)

# 2 Different Types Of Architecture

## 2.1 Two-Tier Application Architecture

A typical two-tier application is a client application using ADO.NET communicating directly with a database server, like Microsoft SQL Server™ (see Figure 3). There are no intervening layers between the client application and the database other than ADO.NET.

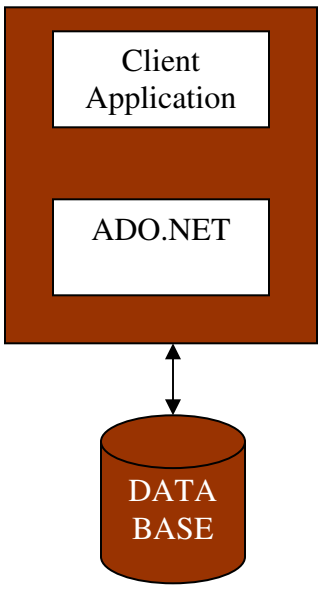



***Figure 3: A two-tier application***

### 2.1.1 When to Use a Two-Tier Architecture

A two-tier application is appropriate for a small application that does not have a lot of forms, or perhaps no forms at all. It may also be appropriate for prototypes where the final version of the application uses one of the other n-tier techniques described here. A two-tier application is not well suited, however, to an enterprise environment, because development and maintenance time and costs can become unmanageable. (4 Database Consulting)

### 2.1.2 Typical Implementation Options

There are several techniques that can be used to develop a two-tier application. All use ADO.NET with a client interface, such as a desktop or Web-based application, and a database, such as SQL Server. Ideas for how to approach the architecture of a two-tier application include: (4 Database Consulting)

- Use data binding techniques to connect an ADO.NET dataset directly to controls.
- Write code that accesses ADO.NET objects to load data manually into controls on your user interface.
- Use a combination of the above two techniques.
- Code business rules directly on the forms with any of the above techniques.

## 2.2 Three-Tier Architecture Application

Unfortunately the 2-tier model shows striking weaknesses that make the development and maintenance of such applications much more expensive. 3- and n-tier architectures endeavor to solve these problems by primarily moving the application logic from the client back to the server.

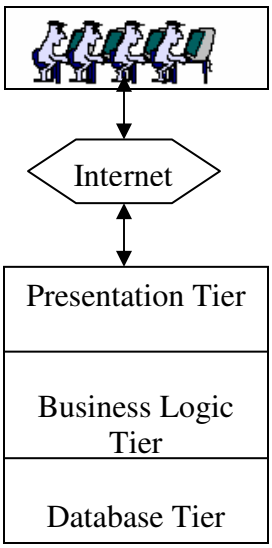



***Figure 4: 3-tier architecture***

In a typical 3 – tier application, the presentation layer is developed using ASP.NET. The business logic tier is developed using C# or VB.NET and the database tier is nothing but the data storage which could be any database server or a relational database. (4 Database Consulting)

### 2.2.1 When To Use This Technique

A three-tier application is appropriate for either a Web-based or Microsoft Windows® application. This technique comes in handy when you need the richness of a desktop application, but users connect to it from many different locations and access the data across an HTTP interface.

### 2.2.2 Typical Implementation Options

The following development techniques are most widely used to create a three-tiered application:

- Each layer can be deployed in geographically separated computers in a network.
- The characteristic of the tier communication is that the tiers will communicate only to their adjacent neighbors. For an example, The Presentation Tier will interact directly with the Business Tier and not directly with Data Access or Data Tiers.
- Datasets returned from the business logic layer can be bound directly to the controls on the forms.
- Datasets created can be used to manually load data into the various controls on the forms.

(4 Database Consulting)

## 2.3 Three-Tier Application Using an XML Web Service

Another design option is to use an XML Web service to separate the database's access to another component that returns the data to the front-end application. Figure 4 illustrates this design option.

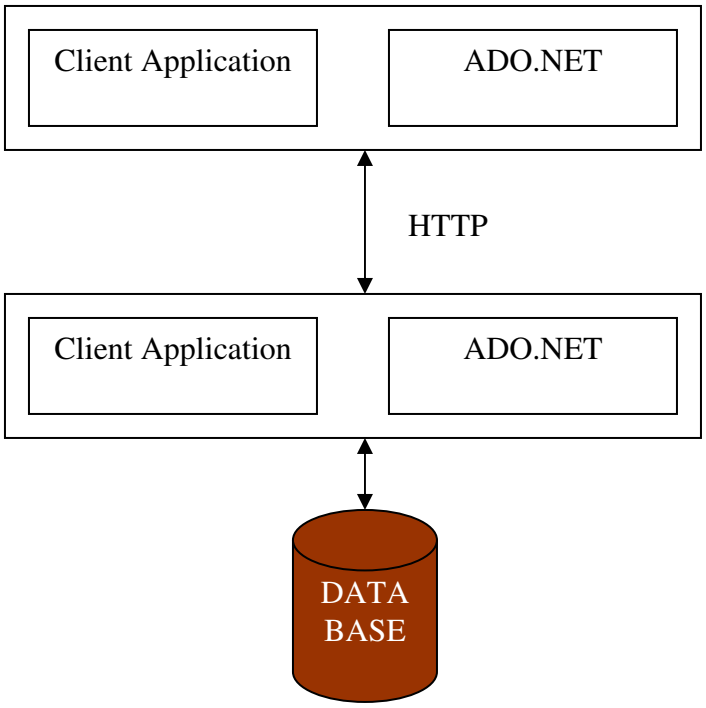



*Figure 5: A three-tier application using XML Web Service*

### 2.3.1 When To Use This Technique

A three-tier application using an XML Web service is appropriate for either a Web-based or Microsoft Windows® application. This technique comes in handy when you need the richness of a desktop application, but users connect to it from many different locations and access the data across an HTTP interface. (4 Database Consulting)

### 2.3.2 Typical Implementation Options

The following development techniques are most widely used to create a three-tiered/XML Web service application:

- All SQL resides within the XML Web service. Datasets are built on the server and returned as an XML stream to the client where they can be rebuilt into datasets.
- Datasets returned from the XML Web service can be bound directly to the controls on the forms.
- Datasets returned from the XML Web service can be used to manually load data into the various controls on the forms.
- All business rules are coded directly on the forms.

(4 Database Consulting)

## 2.4 Three-Tier Application using .NET Remoting

This type of application architecture is almost exactly the same as the three-tier application using an XML Web service. The only difference is that you use .NET Remoting instead of an XML Web service to wrap up the data access layer. Figure 5 illustrates this design option.

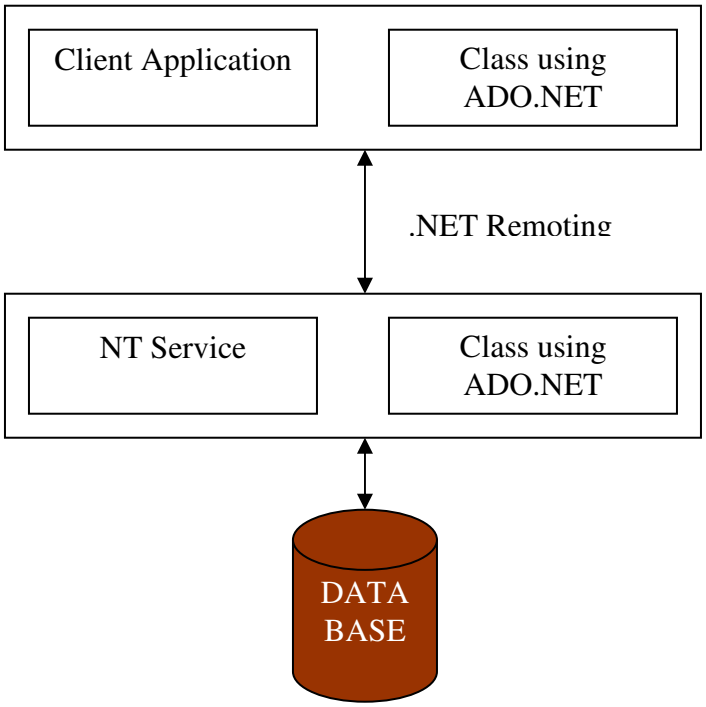



***Figure 6: A three-tier application using .NET Remoting***

### 2.4.1 When To Use the Three-Tier .NET Remoting Technique

A three-tier application using .NET Remoting is appropriate for an application that must be distributed between computers on a LAN. This may be for business reasons or because the cost of the work involved justifies the cost of the network call.(4 Database Consulting)

### 2.4.2 Typical Implementation Options

You will most likely use the following development techniques to create this type of application.

- All SQL resides within the component called through the Remoting service. Datasets are built on the server and returned as an XML stream to the client where they can be re-built into datasets.

- Datasets returned from the Remoting component are bound directly to the controls on your forms.
- Datasets returned from the Remoting component are used to load data manually into the different controls on your forms.
- All business rules are coded directly on the forms.

(4 Database Consulting)

# 3 Design and Implementation

This section describes various elements involved in completing a project and delivering the final product (web application) to the end users.

## 3.1 Requirement Analysis

This web application is developed using ASP.NET technology, C# as the database programming language and Microsoft Access as the database to store course and user information. A list of hardware and system requirements for the application is included in the paper.
The web application would be maintained by the administrators and used by the faculty, staff and students attending UIS. Some of the proposed functionalities for the administrators supported in this web application are:

- Ability to enter and edit the information about the new users (including new administrators).
- Ability to create, delete and edit the list of training classes offered in each semester.
- Ability to search the database based on UIN (University ID Number), course number.
- Ability to register students for the classes.
- Ability to send reminders to the students, one week prior to the date the course is offered.

Some of the proposed functionalities for the users (Faculty, Staff and Students) supported in this web application are:

- Ability to enter and edit the their personal information.
- Ability to view the list of courses offered each semester and also to register for them.
- Ability to cancel registration for a particular course.
- Ability to search the database to produce a list of courses taken in a specific semester of a year or all the courses taken so far.

## 3.2 Analysis/Design

The architecture of the application is in 3-tier, where tables are stored in a MS Access 2000 database. Business components were written in C#, while the presentation layer is in ASP.NET. The 3 tiers are:

- Presentation Layer
- Business Logic Layer
- Data Access Layer

The detail of each component and their design is described in the following sections.

### 3.2.1 Design of Presentation Layer

The presentation layer was created using ASP.NET. This layer included ASPX pages and user controls such as DataGrids, TextBoxes, Labels, and Buttons.

### 3.2.2 Design of Business Logic Layer

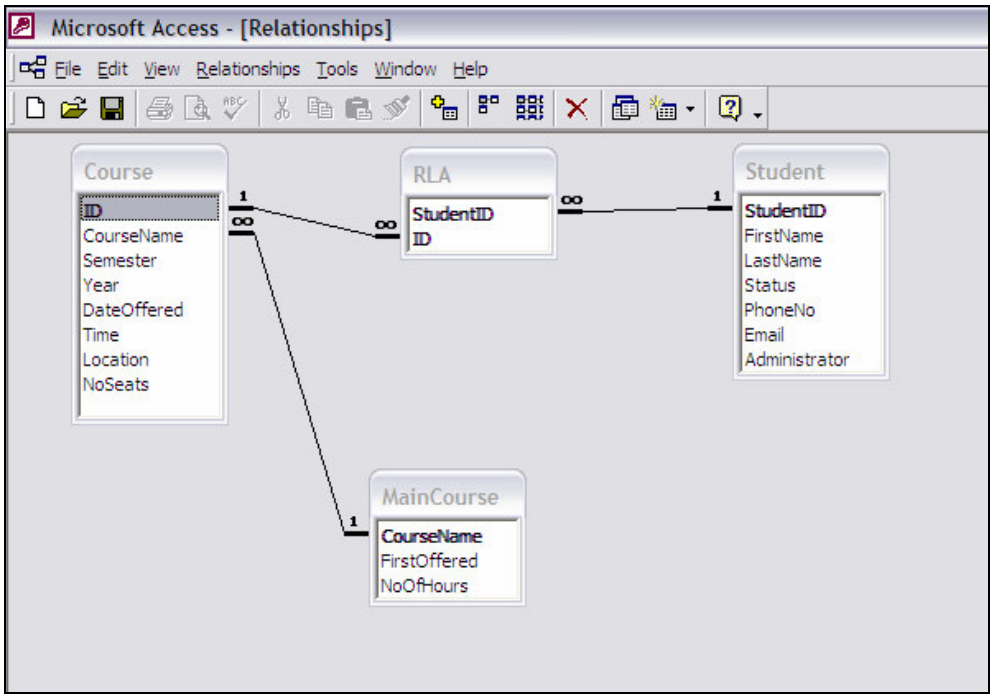
This layer is responsible for preparing or processing the data retrieved and sends it to the presentation layer and data access layer. Since forms are an integral part of the developed application, it is essential that a user's query into a form fit the specified guidelines. ASP.NET includes several validation controls that can be used to simplify the entire validation process. Two controls that are used for this application are RequiredFieldValidator, and the RegularExpressionValidator.

**Required Field Validators**: As the name implies, the RequiredFieldValidator is used on all the required form fields.

**Regular Expression Validators:** This validator ensures that the form field entry corresponds to a regular expression. Regular expressions are used in pattern matching and substitution. The regular expression interpreter takes your grammar, compares it with the string you are doing pattern matching on, then returns a true or false. True meaning it found a match, false meaning it did not. If the result were false, then the "Error Message" would be displayed. For example, the date entered should be of the form "mm/dd/yyyy" or the SSN should be of the form "000-00-0000".

### 3.2.3 Design of Data Access Layer

The application data needs to be stored and retrieved in some data source. Data Access Layer handles this task. For this application, data access layer is developed using MS Access 2000. The database is created using 4 tables (Student, Course, MainCourse, RLA).




### 3.3 Implementation

Since the web application was developed in an iterative process, features were coded as soon as the design for that feature had been completed. If design changes were required due to requirement changes or flaws that were discovered, work on this phase was suspended and the analysis and design phases were reinitiated to make the necessary design changes. Once those changes were made the implementation phase continued. Whenever a design change was required, a new design version was created.

## 4 Evaluation of the Application

### 4.1 Unit Testing

Unit tests are the building blocks of testing. A unit test ensures that individual software components produce expected results, thus fulfilling their contract. Regression testing combines unit tests to verify that software components work together correctly after changes are made. Unit tests also help to understand how the individual pieces that form an application work and perform, before the final assembly process begins.
All the web pages are tested to make sure that they save the data to the correct table of the database and into the appropriate fields of the table. They are also tested to make sure that they retrieve the correct information and the data is updated correctly. The application is also tested for its speed.

### 4.2 Functionality Testing

The goal of functionality testing is simple, to find out how people use the application. Tests were conducted using one-on-one and in small groups. The degree to which the application meets the required goals was determined primarily by the evaluations of testers.

## 5 System Requirements

This web application was written in C#. Microsoft's ASP.NET technology was used to implement it. Internet Information Server (IIS v5.1) was used as the web server to support server side scripts and Microsoft Access 2000 was used as the database for storing user and course information. SMTP server was used for emailing facilities provided on the web registration application. The estimates for the recommended minimum system requirements are:

**System**: PC Pentium IV or other comparable processor, 200 MHz or greater
**RAM**: 128 MB or greater
**Operating System**: Windows XP
**Free Hard Disk**: 2MB for deployment

## 6 Conclusion

The testing of the Application passed very well. All the interfaces were viewed properly. Two different groups tested the Client application. The testers were satisfied with the functionality and the speed with which the application executed. There were no delays in broadcasted messages. The testers who tested the Client application were satisfied with the features available.

The future work involves enabling alumni to access the application. The alumni would also be able to register for classes. The additional functionality would be to allow for credit card or debit card transaction processing as alumni might have to pay $10/hr depending on the final decision made by the University of Illinois at Springfield.

## References


4 Database Consulting. "Technologies: Robust Architecture." http://www.4databaseconsulting.com/architect.html (August 2004).

Duthie, Andrew G. "Microsoft ASP.NET Programming with Microsoft Visual C#.NET Step by Step." 2003.

Joshi, Bipin. "N-Tier Applications and .NET." *DotNetBips*. http://www.dotnetbips.com/displayarticle.aspx?id=213 (August 2004).

Miller, Richard Lt. "N-Tier Application Architecture." *TechiWarehouse*. March 31st 2003. http://www.techiwarehouse.com/Articles/2003-03-31.html (July 2004).

Mitchell, Scott. “Displaying custom classes in a datagrid.” *4GuysFromRolla*. http://aspnet.4guysfromrolla.com/articles/102302-1.aspx (September 2004)
Ullman, Chris, Kauffman, John., Hart, Chris., and Sussman, David. “Beginning ASP.NET 1.1 with VB.NET 2003”, *Wiley Publishing Inc.,* 2004.
Yang, James. “Boosting your .NET Application Performance – Data Application Blocks.” *Developer Fusion Ltd*. Jan 20th 2003, http://www.developerfusion.com/show/3058/6/ (August 2004).

## Acknowledgements

First I would like to thank my parents, brother and sister for all of their support through the years. I would never have gotten where I am today without their blessings. I would also like to thank my advisor, Dr. Kamyar Dezhgosha, for all his guidance and help with this project. I also thank the other members of my committee Jingyu Zhang and Dungey Keenan, for all of their help with this project.
Finally, I would like to thank all of my friends, at the University of Illinois at Springfield, especially Mary Elizabeth Smith, Kandice Biggs, and Emily Welch for their time and effort with giving valuable suggestions and testing the final application.