Journal of Information Systems Education, Vol. 18(1)

Using Reengineering as an Integrating Capstone Experience

Victor Matos
Rebecca Grasser
Cleveland State University
Cleveland, Ohio, 44114 USA
matos@cis.csuohio.edu rgrasser@acm.org

ABSTRACT

This paper presents an example of integrating IT skills using an interesting real life problem. We describe how the reverse- and forward-engineering of the USA National Do Not Call registry was used in our capstone course to illustrate the fusion of different (but interdependent) issues and techniques learned in the IT program. The purpose of the registry is to maintain a list of residential and personal phone numbers whose owners want to keep out of the reach of the telemarketing industry. We believe this experience is rich in educational possibilities; it is very appropriate for a technically oriented Information Technology program and can be conducted in a typical one-semester capstone or senior design project course. The project begins with an exhaustive investigation of the existing artifact leading the student to the discovery of the original model, and its processes, business rules and data structures. The various UML diagrams representing the specifications collected in the discovery phase are used to forward engineer a functionally equivalent database solution using the Microsoft .NET platform. The project touches on ethical issues concerning the legality of reverse-engineering and hints at possibilities on producing similar designs such as a "Do-Not-Email" Registry.

Keywords: Capstone project, senior design project, reengineering, National Do Not Call Registry, UML modeling, web programming, Microsoft .NET applications, database systems.

1. INTRODUCTION

This paper describes an educational experience suitable for advanced IT students whose academic programs include courses on systems analysis and design, web-based programming, and database development. We believe this example is ideal for a senior design project or capstone course requiring the participants to work in integrating concepts and skills around a familiar *real life* problem which is realistic but moderate in length and complexity.

The subject of this study is the *National Do Not Call Registry* (DNC) ("Rules and Regulations", 2003). The *DNC* gives you a choice about whether to receive telemarketing calls at home. The goal of our project is to produce a *functionally equivalent* version of the internet-based component of the DNC system.

Students engage in a process of *reverse engineering* the existing registry in order to discover and learn its underlying structure and design. This approach is consistent with the case-study and problem based learning methods which help students to develop individual and group working skills, including effective communication and critical thinking (Herreid, 1994). Students working on this problem will assume a sequence of increasingly demanding roles. Some of those roles and their corresponding assignments are:

(a) *Researcher*. Investigate and collect published references about the DNC,
(b) *Author/Reporter*. Write an executive summary report describing the purpose of the registry,
(c) *User*. Register own home/personal phone,
(d) *Analyst*. Based on the knowledge acquired from references and their experience of registering their own phones, students are asked to *guess* what processes, business rules, and data structures are used to support the registry,
(e) *Designer*. Provide graphical depictions of the existing DNC model suitable for implementation of the *new* model,
(f) *Developer*. Apply a current technology (we use Microsoft .NET solutions) to construct an equivalent version of the registry,
(g) *Administrator*. Provide maintenance, security, recovery, and catastrophic operational plans.

Those roles are typical of a successful systems development team, in which business analysis, software development, database support and operational maintenance need to work harmoniously to provide an effective solution to the user's problem. Asking the student to *be* different actors is useful in illustrating how a system is assembled to meet a particular goal, showing alternative courses of action

that might be taken to get there, as well as those situations that could cause the system to fail.

The paper is organized as follows. The next section – *playing researchers*- contains background material on the DNC registry, and a short review of what the re-engineering process is about. Then we present a summary of the typical material produced by a *systems analyst/designer* after dissecting the real DNC model. Section four is dedicated to a detailed discussion of the techniques applied by a Microsoft .NET *developer* to implement an internet solution based on the specifications previously made by the designers. Finally we provide conclusions and additional directions that the case could take.

2. BACKGROUND

2.1 What is Reverse and Forward Engineering?

Reverse engineering is a process of investigation whose goal is to develop a set of specifications for a complex entity by a meticulous examination of *real samples* of that system. It generally involves an orderly extraction of the fragments of the item under examination and synthesizing abstractions that represent the structural composition and/or the most essential and characteristic forms of behavior of the original item. As indicated by (Chikofsky 2005), when those concepts are applied to software systems, we realize that most of the effort is made in "gaining a basic understanding of a system and its structure with the final objective of either reproducing a similar version of the software system, or to gain a sufficient design-level understanding to aid maintenance, strengthen enhancement, or support replacement". Therefore, *reverse engineering* is based on analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction.

Forward engineering is the constructive process of moving from high-level abstractions and logical designs made from the *old* system to the physical implementation of an equivalent – and perhaps improved – *new* system. In forward engineering the assessment and adjustment of a subject system is crucial in the re-constructing of that entity in a new form. Reengineering generally includes some form of reverse engineering (to achieve a more abstract description) followed by some form of forward engineering or restructuring. This process may include additions, changes, and/or elimination of features not met by the original system.

In our lab experience we have two definitive goals: first we must understand how the DNC system works, and, secondly we must implement a similar *product* using a particular computer technology (Microsoft .NET solution).

2.2 Methodology for Researching the Do-Not-Call (DNC) Registry

The first step in the evolution of this study requires the students to gather and synthesize relevant information about the registry. Sources include journals, newspaper and magazine articles, official reports prepared by congressmen on each side of the issue, documentation posted on web pages by the Federal Communications Commission (FCC), etc. Participants - working in groups or individually - must prepare a short executive summary of their findings.

The second step of the process requires the participants to act as a typical consumer. Students are asked to register their personal/home phone number using the internet version of the registry. The purpose of the task is to gain full understanding of the actions taking place in a typical interaction between a consumer and the existing DNC system. Some of the suggested strategies include flawless and wrongful situations such as (a) registration of a new and valid phone number, (b) registration of a phone already signed for, (c) partial registration of a phone number followed by intentional omission of confirmation steps, etc. All relevant information should be noted and schematically represented using UML diagrams (to be discussed later).

The following section contains a brief synopsis of the typical statements reported by the students about the origins and operational characterization of the actual internet version of the registry as implemented by the FCC.

2.2.1 What exactly is the Do-Not-Call (DNC) Registry? According to year 2002 estimates made by the Direct Telemarketing Association, an average of 60 million telemarketing calls were made daily. Those unsolicited calls were aimed at selling all kind of products and services ranging from cheaper mortgage loans to unforgettable summer vacations in the Caribbean. Under pressure from many consumer-advocacy groups the U.S.A. Congress intervened and drafted legislation to regulate the massive volume of calls made by those companies. In February 2003, Congress passed H.R. 395, known as the *Do-Not-Call Implementation Act* and the Appropriations Committee granted $18 million for the implementation of the program. This legislation promised the consumers protection from unwanted telemarketing calls. The law includes penalties of up to $11,000 in fines for those telemarketers who ignore the registry and continue to call listed numbers.

The National Do Not Call Registry is managed by the Federal Trade Commission (FTC), the nation's consumer protection agency. It is enforced by the FTC, the Federal Communications Commission (FCC), and state law enforcement officials. By August 2003 about 50 million phone numbers were already included in the DNC registry.

2.2.2 The Registry and its Business Rules: Our pedagogical strategy at this point is based on class discussions where the students have an opportunity to present and summarize the functions of the registry and other related issues. The business rules describing the different transactions are synthesized as *consumer-registry* interactions. Other aspects such as *merchant-registry* and *registry-merchant* interactions were recognized but not elaborated.

It should be noted that students actively participated in these discussions and it was clear that the majority of them were in favor of the concept of the registry. We believe that this is important as students tend to create a better solution for an idea that they support.

2.2.3 DNC Business Rules: Identifying and summarizing the DNC business rules is arguably the most critical task of

the reverse engineering phase. The what, whom, when, where, and why of the transactions performed by the DNC registry is collectively represented by these rules. They do not provide a view of the physical organization or internal data structures but rather offer an insight into the behavioral depiction of operations performed by the registry.

During the process of finding and documenting the DNC set of business rules, most students found the registry's *Questions & Answers* web-page to be very useful. The following excerpts represent a portion of the internet-based registry's operational description:

1. Home and personal phones numbers can be registered, verified and removed from the DNC list by calling 1-888-382-1222 or accessing the web site at www.donotcall.gov.
2. Telemarketers covered by the National Do Not Call Registry have up to 31 days from the date you register to stop calling you.
3. You may register up to three telephone numbers at one time on the National Do Not Call Registry website.
4. You will receive a separate confirmation email for each number you wish to register online. You must open each email and click on the link in each one to complete the registration process. This must be done no later than 72 hours after receiving the email.
5. If you have more than three personal telephone numbers, you will have to go through the registration process more than once to register all of your numbers. There is a limit on the number of phone numbers you can register in this manner.
6. Your phone number will remain on the registry for five years from the date you register (unless you choose to take it off the registry or your phone number is disconnected). You can click on the *Verify a Registration* button any time to check your expiration date.
7. If your number is disconnected and then reconnected, you may need to re-register. Also you must re-register after changing calling plans or other services, or changing the billing name on the account.
8. Calls from or on behalf of political organizations, charities, and telephone surveyors would still be permitted, as would calls from companies with which you have an existing business relationship, or those to whom you have provided express agreement in writing to receive their calls.
9. If your number has been on the National Do Not Call Registry for at least 31 days (starting January 1, 2005) and you receive a call from a telemarketer that you believe is covered by the National Do Not Call Registry, you can file a complaint. To file a complaint, you must know either the name or telephone number of the company that called, and the date the company called you. While the FTC does not resolve individual consumer problems, your complaint will help investigate the company and could lead to law enforcement action.

The DNC Registry also includes a phone registration process which is ignored in this article.

3. MAPPING THE FINDINGS OF THE RE-ENGINEERING PROCESS

As a result of the class discussions and the registration of their personal phone numbers in the list, the participants gain exposure to the inner working of the registry. At this point we ask the students to summarize their observations, and report those annotations in the form of different UML diagrams; more specifically *use-case, sequence,* and *transition* diagrams (Rumbaugh, et al 2005). The students initiate this phase from a conceptual level, however, a useful byproduct of this task could be an early sketching of potential data structures needed to support the operations observed in the real system.

In essence, students are acting in new roles as Analysts and Database Designers. The first – and perhaps most important - representation will be a *use-case diagram.* The goal of this task is to capture and represent a concise high-level behavioral illustration of the system.

Figure 1 depicts a first cut *use-case* diagram summarizing the activities of the DNC registry. It identifies two actors, the *Consumer* and the *Merchant*, and introduces seven main activities. The *Consumer* is any person trying to sign his/her personal or residential phone in the registry and the *Merchant* represents the telemarketers. Each of the seven cases consists of an identifier and an optional list of the business rules it relates to (business/operational rules are introduced in section 2.2.3). For instance the case "Delete Phone" is described by the business rule number 1 of section 2.2.3, i.e. a phone could be removed from the DNC after calling from that particular phone the toll free number 1-888-382-1222.

At the consumer-registry level the main tasks are the Registration, Confirmation, and Verification of a phone number. However figure 1 includes other operations such as removal of a number from the list, acquisition of information at different levels of detail, and providing the merchants with updated copies of the list.

In order to provide more details about each of the cases, students are asked to develop additional UML descriptions. Figure 2 shows an *activity diagram* detailing the Registration-Confirmation tasks introduced in the *use-case* diagram as the cases "Register Phone" and "Confirm Phone".

At this level of the reverse-engineering process the students may begin to produce their own versions of *how* the system works. To support their speculations it is acceptable to believe there is a physical database on which those operations take place. Although the exact structure of that database is unknown to the students, it is appropriate for them to *guess* the potential nature and composition it might have. The following statements paraphrase the behavior of the registry in terms of operations executed on that (to be developed) database

- After a consumer's registration web page is submitted and all inconsistencies and omissions are corrected, a summary web page is sent back to the consumer. The consumer has two options, either to ignore the message or to acknowledge the registration request by clicking the *accept link* of the summary page.

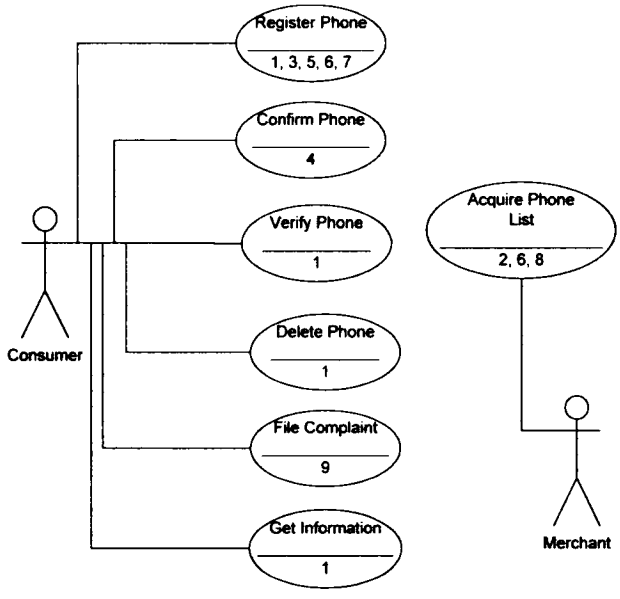


Figure 1. Do-Not-Call Registry Use-Case Diagram.

- After a consumer's registration web page is submitted and all inconsistencies and omissions are corrected, a summary web page is sent back to the consumer. The consumer has two options, either to ignore the message or to acknowledge the registration request by clicking the *accept link* of the summary page.
- If the user chooses to accept, the web page is sent to the registry. The phone(s) held in the page are temporarily entered in the database.
- Each database record could include the fields <*Key, eMail, PhoneNumber, RegistrationDate, ConfirmationDate, ExpirationDate*>. The temporary entries are marked as unconfirmed pending for final consumer's action. Each new record entered in the database has a unique database key and includes the dates of registration, confirmation and expiration.
- For each temporary record entered in the database the registry must prepare a *confirmation* email. The email will include the date of the request, the phone number, and a hyperlink that the consumer must click on to close the confirmation process. The hyperlink includes the database key associated to the current phone number.
- The consumer is given 72 hour to respond to the email. Messages sent but not responded to represent temporary records that must be deleted at the end of those 72 hours.
- If the consumer clicks on the confirmation hyperlink, the appropriate temporary record is updated. The key embedded in the hyperlink is used to reach and update the database record. The update is done by adding an expiration date of five years after today's date.

Figure 3 is a *sequence* diagram showing an alternative explanation of the Registration and Confirmation processes. We have found that some students prefer to work with the material as an activity diagram, while others prefer the equivalent sequence diagram.

The vertical lines in the sequence diagram are called lifelines and represent the object's life during the interaction. This representation is particularly useful to the object-oriented developer as a way of depicting the overall flow of control that takes place in a typical interaction. Inter-object *messages* are represented by solid arrows between the involved lifelines; messages are invocations to the methods supplied by each object. Information returned by the invoked methods is denoted by broken lines. The order in which those messages are sent is shown from top to bottom of the diagram. Messages have at least a unique name and a list of optional parameters. Special messages – called *self delegation* – represent recursive communication in which the object calls itself and are denoted by arrows back to the same lifeline.

3.1 A Preliminary Database Design

Based on the knowledge already acquired by observation of the real registry, the students are asked to prepare a relational database structure capable of supporting the different operations already described for the DNC registry. A tentative implementation of a physical database design is drafted in Figure 4. The structure of this *one table solution* appears in Figure 4(a) as a Microsoft Access file called the *Master* table. Figure 4(b) is a fragment of the actual repository.

Each record in the Master table represents a transaction posted on the DNC registry. The state of a transaction could be twofold; it is either a *pending request*, or a *confirmed registration*. Each record of the Master table consists of seven fields. The first is a unique identifier (**ID** field) which in the case of MS-Access is implemented as a serial *Autonumber*. The next field (**PatronEmail**) is the consumer's email address. The next fields are the phone's area code (**AreaCode**) and number (**PhoneNumber**). The last three fields are dates used to record the moment in which the consumer first signs his/her phone number in list (**RegistrationDate**), the date in which the consumer responds to the automatic email generated by the registry to confirm his/her demand (**ConfirmationDate**) and the date in which the confirmed request will expire (**ExpirationDate**).

At this stage in the forward engineering of a new solution, the emphasis is on effectiveness as oppose to efficiency. Rather than requesting the students an *ideal* and *complete* representation of data in the form of a fully normalized multi-table database, what is important is that the student's design somehow could be used to realistically support all of the operations identified early on. Optimizing data structures could be done at a later time, when more understanding of the needs of the system is reached. For instance, the preliminary design in Figure 4(a) does not include a **DeleteDate** field; however it is relatively simple to add such data at a later time. Also, the combination AreaCode and PhoneNumber could be made unique (not admitting duplicates), but we choose to ignore it for now

For the time being the above suggested database representation can handle the main operations in the following way:

- *Registration*: Create a database record. Include unique ID value, area code and phone number. Make RegistrationDate be today's date. Set ExpirationDate

equal to (RegistrationDate + 3 days). Set ConfirmationDate to *null*.

- *Confirmation*: Change ConfirmationDate to today's date. Set ExpirationDate equal to (ConfirmationDate + 5 years).
- *Verification*: Use a phone index approach for fast retrieval of a phone number. If the number is found in the database, respond positively to the request, otherwise the number is not registered (or it has already expired).
- *Deletion*: If the email supplied in the request matches the value stored in the database proceed to removal by setting ExpirationDate to today's date. Move the record to a historical file.
- *Distribution* (to merchants): Extract from database (at least) the AreaCode/Phone and ExpirationDate of confirmed records whose ExpirationDate has not yet been reached.

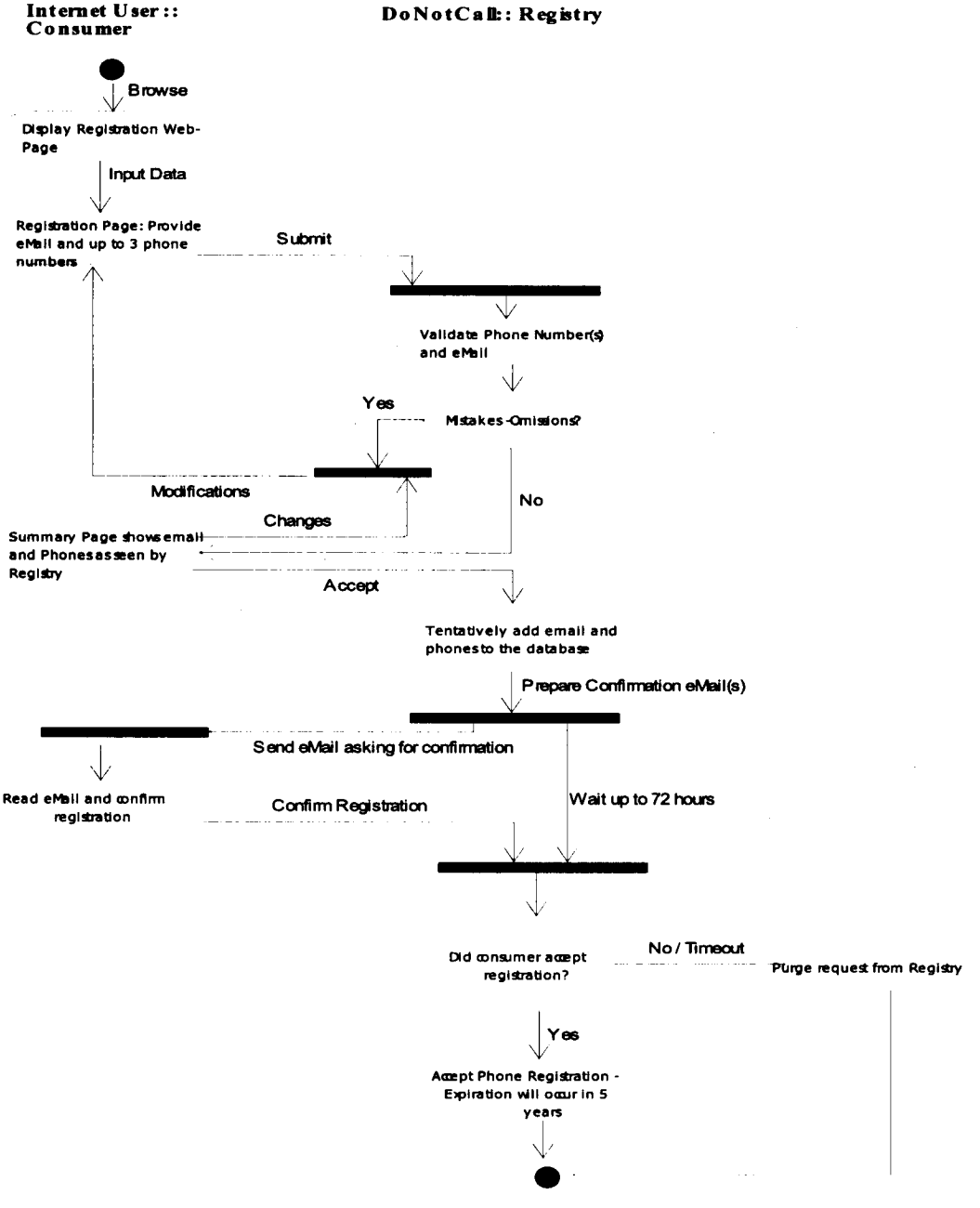


Figure 2. Activity Diagram: Registration-Confirmation Process

This preliminary solution could be used as a starting point in the development process. We recognize there are important issues that still need to be discussed. These include, but are not limited to, database accessibility, reliability, concurrency control, recovery, security, encryption, and periodic backing-up of the physical database.

4. FORWARD ENGINEERING OF A WEB-BASED SOLUTION (MyDNC)

In the following phase of the project the students proceed to make a software artifact that mimics the behavior of the real DNC registry. The workbench for the forward engineering process is based on current Microsoft .NET technologies including: Active Server Pages (ASP.NET), Active Database Objects (ADO.NET), and Visual Basic .NET.

A mandatory constraint for this phase is the requirement to reproduce the same user-friendly nature of the original DNC registry. Efforts must be made to duplicate the *look and feel* of the real DNC web site. For brevity we will call the new system **MyDNC**. Figures 5 - 7 show our rendition of the main home page, and the sequence of pages used for phone registration.

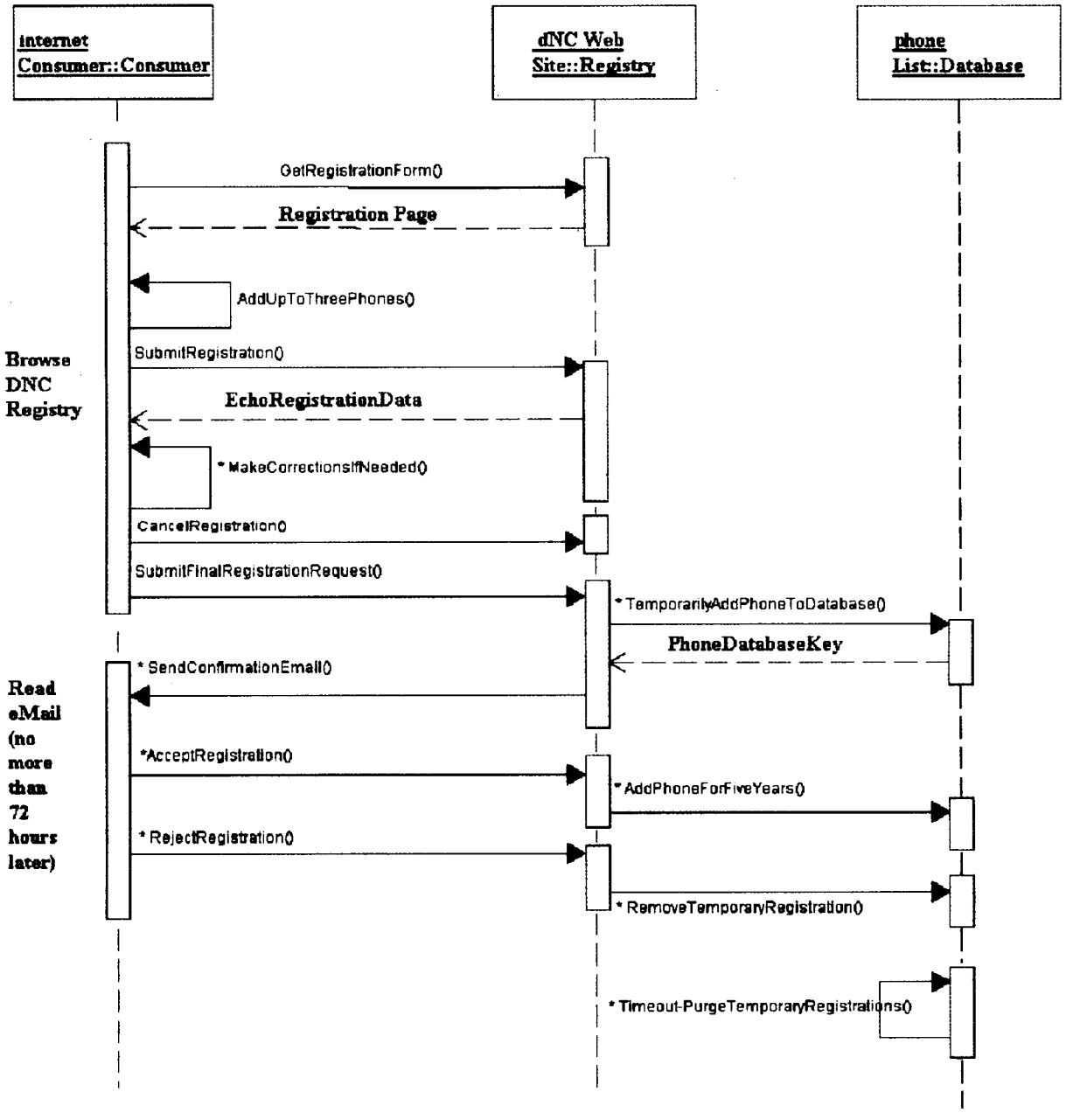
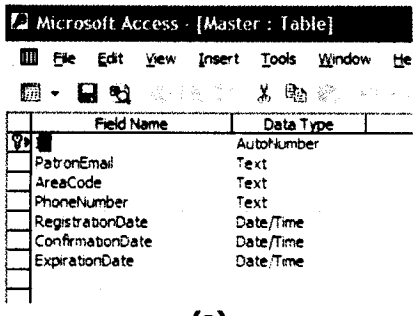
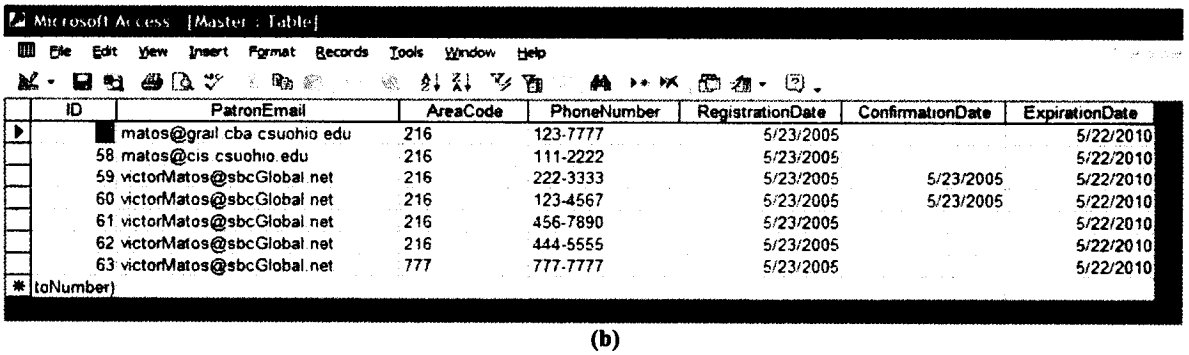


Figure 3. Sequence Diagram: Registration – Confirmation Process.



(a)



(b)

Figure 4. A simple relational database design for the Do-Not-Call Registry.

We limit the following discussion to the Registration-Confirmation process illustrated in the diagrams provided in Figures 1, 2, and 3. Other processes such as Verification, Elimination, Filing of Complaints, and Requesting Information have been left as possible future extensions of the project.

4.1 Implementing the Software Solution

Each student should create his/her own local version of MyDNC as a stand-alone *ASP.NET Web Application.* The development software in the students' machines includes (a) Windows XP Professional with the Internet Information Services (IIS) installed, (b) Visual Studio 7 (defaulting to Visual Basic .NET Development), and (c) the Internet Explorer.

4.1.1 Microsoft .NET Tools: ASP.NET (Active Server Pages) is a Microsoft language-independent technology that enables a server computer to dynamically produce pages or services that can be rendered inside a web browser (the clients) (Balena 2002). ASP.NET consists of a group of predefined objects that facilitate the programming of web applications; some of those objects are *Request, Response, Session, Application,* and *Server*. The Request object allows the user to grab data from incoming pages, the Response helps with the writing of dynamic pages, Session provides a mechanism for global inter-page memory, and Server grants access to server-side resources such as databases, etc.

When the students initiate the development of *myDNC* software, Microsoft Visual Studio creates a new directory in the web default folder of the server (in most cases the name is *C:\InetPub\wwwRoot\MyDNC*). This web directory will contain a collection of *.aspx* web Forms. Each web Form holds any number of graphical user interface (GUI) controls as well as code to attend their events. The GUI controls could be either traditional HTML design elements – such as such as text-boxes, pictures, labels, buttons, etc. - or the more sophisticated .NET GUI counterparts. The computer code serving the events of an *.aspx* page is made of client-side and server-side sets of routines.

Client-side subroutines are locally interpreted by the virtual machine in the client's browser. Those scripts are interspersed with the page's HTML tags. Local scripts are written using traditional client-side web-languages including VBScript, JavaScript, etc. For .NET web applications, server-side code is written with any Visual Studio .NET language (we choose VB.NET). This type of routines offer richer possibilities for the developer, however, they are more resource demanding and require a round-trip navigation.

Unlike Windows applications, ASP.NET solutions are not meant to keep a permanent link or state between the client and the server. This *stateless* feature is inherent to HTTP and requires the developer to create code that can simulate the concept of page-to-page memory storage. ASP.NET programmers have several alternatives to overcome the stateless nature of HTTP. Some of the choices include *Application state, Session state, Cookies,* and *persistent databases.* We use the ASP.NET *Session* object in the implementation of our solution. In particular we rely on its support for *collections* as a global array-like memory place holder whose visibility encompasses the entire application.

4.1.2 The Final Product: The Microsoft Internet Information Services (IIS) tool allows the students to host - in their computers - operational stand-alone web sites. In

particular, *MyDNC* web-pages (or more precisely Web Forms) could be directly accessed using a browser such as Internet Explorer. For instance, the URL required by Internet Explorer to reach the main page of the student's application (*MyDNCHome.aspx*) should be:

http://LocalHost/myDNC/MyDNCHome.aspx

The home page in Figure 5 acts as a main switchboard for the registry services. The buttons in the home page are responsible for the calling of other pages stored in the local MyDNC directory. Options such as Registration, Verification, Filing a Complaint, etc, are available by clicking the appropriate buttons. Like any other web application, the attention to events is provided by the application software. For this project we have chosen to write server-side code as much as possible, however we also include an example of local scripting.

In Figure 5, if the consumer pushes the button labeled *Register Now*, the associated event handling routine invokes the ASP.NET *Server* object to execute on its behalf the method that transfers control to another page. Assume the page being called is named *MyDNCRegister.aspx*. The supporting VB.NET "code-behind" subroutine is as follows:

```
Private Sub btnRegisterNow_Click(...) Handles
btnRegisterNow.Click
  Server.Transfer("myDNCRegister.aspx")
End Sub
```

The service routine for the other buttons in Figure 5 is similar to the above fragment. The only difference is the actual name of the Web Form being called.

4.1.2.1 Step One: Registration – Data Collection: A rendition of the registration page appears in Figure 6. The interior frame identifies the page as "Step One" of the process and provides brief instructions for the consumer as well as text boxes to collect input data. Up to three phone numbers (including area code) could be entered on the page. The consumer – for verification purposes - must enter his/her email address twice. Local (client-side) validation could be done. As an option, simple facts could be locally checked, using the ASP.NET *Validator* controls (*Required Field, Range,* and *Regular Expression Validators*). For instance; area code must be present and it should be a number less than 999, phone numbers are made of exactly seven digits each (range 0000000-9999999), email addresses have a format defined by the pattern xxx@yyy.zzz, etc.

4.1.2.2 Step Two: Temporary Registration: Once the consumer has finished the entering and editing of data, he/she could click on the *Submit* button. The page is sent to the web server where another page (call it *MyDNCCheckErrors.aspx*) may perform further validation of the supplied data. In addition to syntactical verification, some logical problems could be addressed, such as: is the area code a legitimate value, is the phone number already registered, and so on. Figure 7 shows a verification page that reflects the data entered in the previous step (Figure 6).

An appropriate moment for the preparation of the Web Form shown in Figure 7 occurs during the *Page_Load* event of *myDNCCheckErrors.aspx*. The skeleton of this server-side code fragment is given below.

The function *IsPostBack( )* returns the *True* value only when the page is displayed for the first time. Please notice the use of the *ASP.NET Session* object as a way of preserveing data. For instance *Session(6)* holds the consumer's email string. This value is extracted from the txtEmail textbox defined in the web Form shown in Figure 6. The construct *Request.Form("txtEmail")* relies on the ASP.NET Request

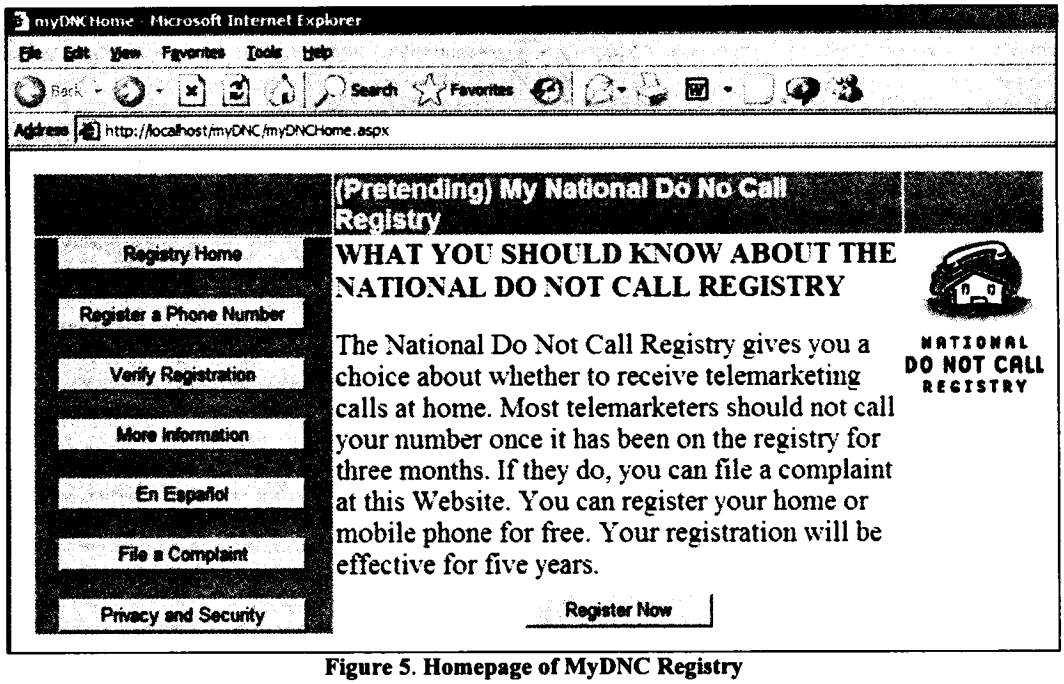


Figure 5. Homepage of MyDNC Registry

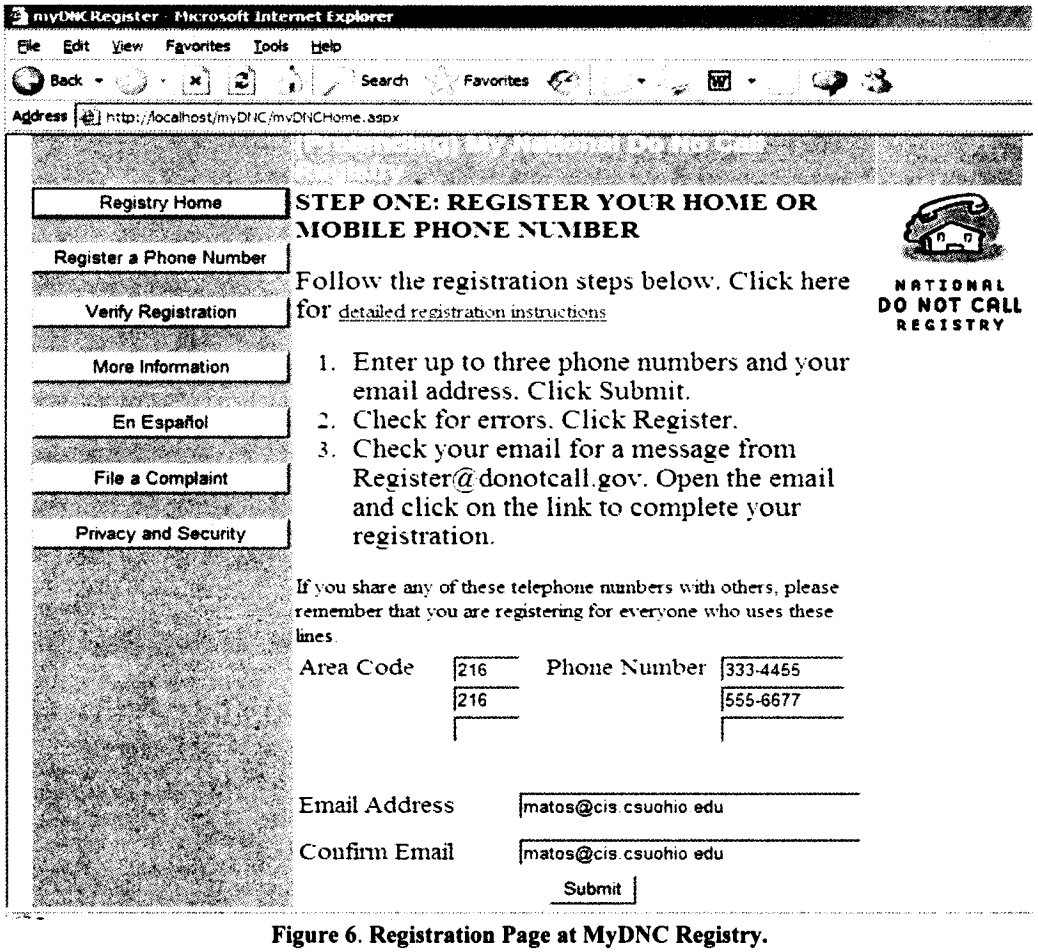


Figure 6. Registration Page at MyDNC Registry.

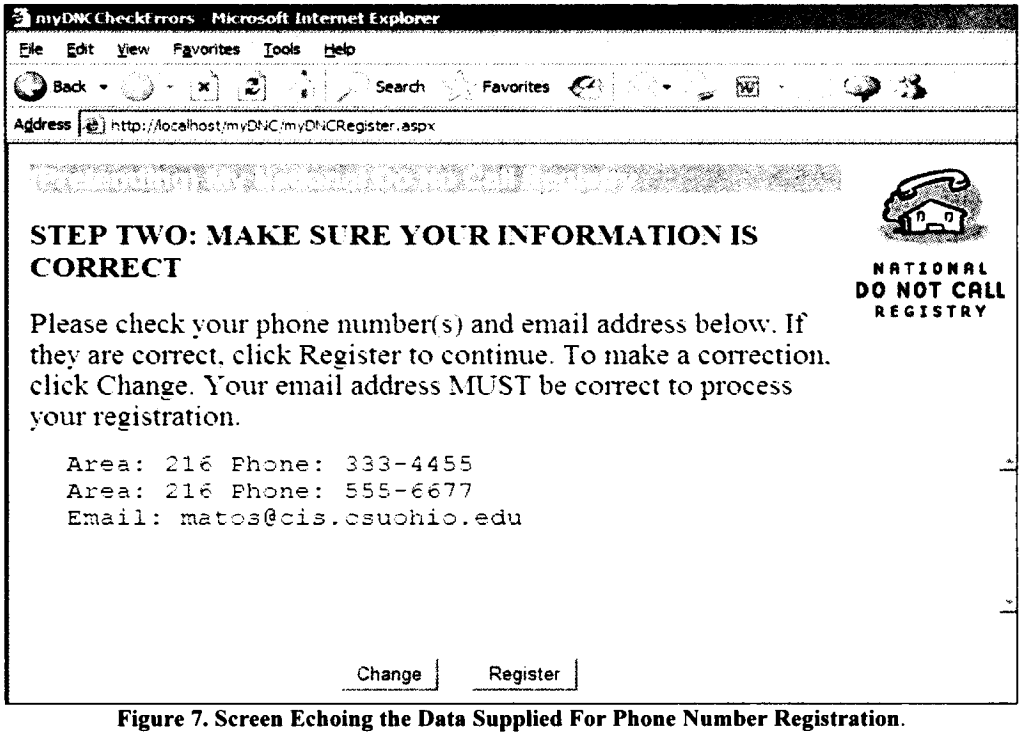


Figure 7. Screen Echoing the Data Supplied For Phone Number Registration.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
          System.EventArgs) Handles MyBase.Load
    'run this segment only the first time the page is loaded
    If Not IsPostBack Then
      'create ASP.SESSION variables initialize them
      Session(0) = Request.Form("txtArea1")
      Session(1) = Request.Form("txtPhone1")
      Session(2) = Request.Form("txtArea2")
      Session(3) = Request.Form("txtPhone2")
      Session(4) = Request.Form("txtArea3")
      Session(5) = Request.Form("txtPhone3")
      Session(6) = Request.Form("txtEmail")
      '-----------------------------------------------------
      '--- TODO: check data here (validate area code,
      '--- phone must be numeric, spaces/hyphens/dots, etc.
      '-----------------------------------------------------

      'Echo all the data on the textbox so consumer could see
      'what we have at received at the server side.
      Dim i As Integer, myLine As String
      'up to three lines holding AreaCode & PhoneNumber
      For i = 0 To 2
        If Session(2 * i) <> "" Then
          myLine &= " Area: " & Session(2 * i) & _
               " Phone: " & Session(2 * i + 1) & vbCrLf
        End If
      Next
      myLine &= " Email: " & Session(6)
      Me.txtMsg.Text = myLine & vbCrLf
End Sub
```

object for the parsing of the <Name:Value> pairs sent by the calling Web Form. Any other subroutine – in the same or another page – could recall this email value by examining the contents of *Session(6)*. ASP.NET creates the sense of "global memory" by making the Session variables available to any web-page for the current user for the current session. This server-side subroutine finishes with a loop in which each combination of area code and phone number is placed on a line and assigned to a textbox for display purposes.

The verification page (*myDNCCheckErrors.aspx*) offers the consumer the option of making some changes (clicking the *Change* button) or completing the second step of registration by pushing the *Register* button. As the students will discover, the first action is very simple to take care of, but the second is more complex.

If the consumer decides to make some changes, the registration screen seen in figure 6 is sent back to the user and the cycle is repeated. This rather simple navigation process is similar to clicking the *Previous-Page* button provided by most web browsers. However, we implemented this idea by defining the *Change* button as a primitive (or *Intrinsic*) *HTML* control with a *client-side* piece of VBScript code attached to its *click* event. The HTML definition of the button is *<INPUT type="button" value="Change" name="btnChange">*. The corresponding service routine uses the *Back(1)* method of the *Window.History* object to take the navigation one page backwards. The routine follows:

```
<script Language="VBScript">
Sub btnChange_OnClick()
  window.history.back(1)
End Sub
</script>
```

If the consumer chooses to register her phone number(s), a set of intensive server-side operations take place. The main goal of this step is to assemble and insert temporary record(s) in the physical database. A temporary record represents the solicited - although uncommitted – request made by the consumer. The VB code is given below; lines have been numbered to facilitate the discussion.

Line 4 defines the local arrays Area(3) and Phone(3). Those arrays will be used to facilitate the use of the <AreaCode:PhoneNumber> pairs stored in Session variables. In lines 10-12 three ADO.NET objects: myCnn, myCmd, and myDR are defined. ADO.NET is a new middleware standard used by the Microsoft .NET platform for reaching a variety of data sources including relational databases. myCnn is an instance of the Connection class; it creates a pipe between the application and the database on which messages and data will travel. The user messages will be phrased in the form of SQL statements. Those SQL retrieval or maintenance messages will be assembled and managed inside the Command object defined in line 11. The data returned by the server will consist of relational tables. Clients have several ways of collecting those answer tables, in our case myDR is an ADO.NET DataReader object which will provide a simple read-only client-side data repository.

```
1   Private Sub btnRegister_Click(ByVal sender As System.Object, ByVal e As
                System.EventArgs) Handles btnRegister.Click
2       '--- Prepare TEMPORARY database records
3       '--- get data from Session object
4       Dim area(3), phone(3) As String
5       Area(1) = Session(0) : phone(1) = Session(1)
6       Area(2) = Session(2) : phone(2) = Session(3)
7       Area(3) = Session(4) : phone(3) = Session(5)
8       email = Session(6)
9       'prepare the .NET Connection, Command, and DataReader objects
10      Dim myCnn As New OleDb.OleDbConnection
11      Dim myCmd As New OleDb.OleDbCommand
12      Dim myDR As OleDb.OleDbDataReader
13      '--- define a link to the local MS-Access Database
14      Dim cnnStr As String = "Provider=Microsoft.Jet.OleDB.4.0;" & _
15                  "Data Source=C:\DoNotCall\DoNotCall.mdb"
16      myCnn.ConnectionString = cnnStr
17      myCnn.Open()
18
19      Dim i, myRecAffected, theGeneratedID As Integer,
20      Dim mySQL, myID As String,
21      Dim myDate1, myDate2, myDate3 As Date
22      Dim ts As New TimeSpan(5 * 365, 0, 0, 0, 0)
23      myDate1 = Format(Now, "short date") 'today's date
24      myDate3 = myDate1.Add(ts)       'five years from now
25
26      '--- prepare the SQL INSERT commands to enter the new user request
27      For i = 1 To 3
28        If phone(i) <> "" Then
29          'insert the new phone in the Master file
30          mySQL = "Insert into Master (PatronEmail, AreaCode,
31                       PhoneNumber, RegistrationDate,
32                      ConfirmationDate, ExpirationDate)
33              values  ('" & _
34               email   & "', '" & _
35               area(i)  & "', '" & _
36               phone(i)  & "', #" & _
37               myDate1  & "#, null, #" & _
38               myDate3  & "# )"
39          myCmd.CommandText = mySQL
40          myCmd.CommandType = CommandType.Text
41          myCmd.Connection = myCnn
42          myRecAffected = myCmd.ExecuteNonQuery()
43
44          'find the ID (AutoNumber) of the recently inserted record
45          mySQL = " select ID from Master " & _
46              " where areaCode = '" & area(i) & "'" & _
47              "  and PhoneNumber='" & phone(i) & "'"
48          myCmd.CommandText = mySQL
49          theGeneratedID = myCmd.ExecuteScalar
50
51          'create (one at the time) session variables: ID1, ID2, ID3
52          'and assign to each the generated ID
53          myID = "ID" & i
54          Session(myID) = theGeneratedID
55        End If
56      Next
57      myCnn.Close()
58      Server.Transfer("myDNCConfirmation.aspx")
59    End Sub
```

In order to use the ADO.NET components, the application code must include the line "Imports System.Data.OleDb". The System.Data.OleDb namespace is the .NET Framework Data Provider for OLE DB. This namespace describes the collection of classes used by the application to access a disconnected OLE DB data source in the .NET managed space.

The connection string of line 14-15 identifies the database software provider and the physical database location. In our code Microsoft.Jet.OleDB.4.0 refers to the MS-Access driver. For security reasons the database must be held in a special type of directory that allows remote web users the possibility of reading and updating the database. For additional information on this topic, see appendix A.

Lines 21-25 define and prepare date variables *myDate1* and *myDate3* holding respectively the current date and one that is five years away from today's date.

The for-loop in line 27 is used to assemble and insert each of the three possible phones given by the consumer. *mySQL* is the string holding the corresponding SQL-insert command. Each of these commands will request the database in Figure 4 to accept a new temporary record. In each case the pieces of the insert operator are *glued* together: email, area code, phone, registration date, and expiration date are put into a comma delimited string. This string is the text that the command object *myCmd* will send as a message to the database server. For example, the following SQL insert command represents the request for a temporary record on behalf of the first phone number in Figure 6.

```
Insert into Master (PatronEmail,
 AreaCode, PhoneNumber, RegistrationDate,
ConfirmationDate, ExpirationDate)
 values ('VictorMatos@sbcGlobal.net',
'216','123-4567',#5/23/2005#, null,
#5/22/2010#)
```

Lines 39-42 show how *MyCmd* is told what SQL command, code type, and connection object should be used. In line 42 the command is executed as a non-query. Under this format the server does not return a table, however a control value indicating the total number of database records affected by the operation is sent back to the application. Also notice the presence of a *null* value in the argument list. This null represents the *ConfirmationDate* which is - if any - a date in the next three days range given to the consumer to respond.

Lines 44-49 illustrate a special case of database retrieval. The string *mySQL* is phrased as a SQL-select command, requesting from the server the internal identification number assigned to the newly inserted record. This one single value is easily retrieved using the *ExecuteScalar* method of the command object. The variable *TheGeneratedID* will be used to tell the key of the temporary record in the database. We will explain later how this ID number is used in the email to be sent to the user asking for confirmation.

The last two lines of the registration routine close the connection object and request the server to display a Web Form called *myDNCConfirmation.aspx*.

4.1.2.3 Step Three: Confirmation and Permanent Registration: The purpose of *myDNCConfirmation.aspx* is to acknowledge the consumer's request for registration and to initiate the preparation of the third and final step required for his/her acceptance. The students will need to insert several temporary records in the Master table. These lines hold the internal database key, area code and phone number. For each of the temporary records that were created and inserted into the Master table, the system will prepare individual pieces of email and deliver each of those messages to the consumer. The goal of these messages is to get the final approval from the consumer.

For each phone number *temporarily* registered in the database, the VB.NET (SendEmailNow) code below will assemble and deliver to the consumer a piece of email. A sample of such mail appears in Figure 8. The mail text includes a hyperlink that ends with an encrypted identifier. In our example, the cyphertext material is ***?xYzAbC?66***. We will use this string to illustrate the mechanism to update the pending database requests. The value 66 represents the internal database key associated to the phone number (216 333-4455) that is being confirmed.

The equivalent of a signature is obtained by asking the consumer to click on the hyperlink that is included in the e-mail's text. In order to finalize the registration, the email must be responded to within three days of its creation. If this is done in a timely fashion, the *ConfirmationDate* in the temporary database record is set to *today's date* and the record will be held in the system for five years. Within a few days, telemarketers will be informed that this number is "off limits".

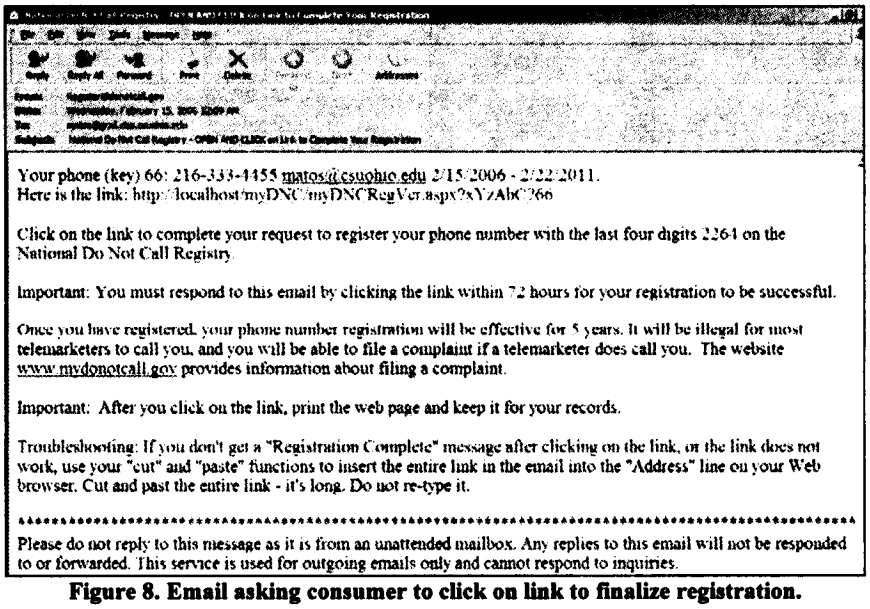
The following VB.NET illustrates how the email pieces are made and transmitted. Again, code lines are numbered to facilitate the discussion.

In lines 1-18 the subroutine *SendEmailNow* defines an email object (line 2), fills the different parts of the email using the data from each temporary record (lines 5-9), identifies the mail server (line 11) and releases the message (line 12). The subroutine *PrepareBody* (line 7, 21-41) produces the encrypted hyperlink and concatenates it with the rest of the letter.

Later on, if the consumer clicks on the link that appears in the email, the web form *myDNCRegVer.aspx* will be invoked and the encrypted string will be supplied as an argument to the called page. This page could use its *Page_Load* event to decipher the input argument, use the internal database key (value 66 in our running example), update the corresponding temporary record by setting the confirmation date as *today's date*, and finally create the simple page shown in Figure 9. This last web page informs the consumer about the permanent registration of his/her phone number in *myDNC* registry.

The following VB.NET code is part of *myDNCRegVer.aspx* page. It performs the final steps of registration by attempting to change the *ConfirmationDate* of a temporary record to the current date. The subroutine in lines 1-31 updates the temporary record whose ID key is known to us. It uses the ADO.NET *Connection, Command,* and *DataReader* already discussed during the pre-registration of the phone number. The SQL statement in line 18 tries to set the *ConfirmationDate* to the day of confirmation. The command is executed as a *non-query* operation (line 22). The code is rather simplistic and has room for several situations not discussed here. For instance, we did not check for the elapsed period between pre-registration and confirmation dates to exceed the limit of three days. These and other improvements are left to the students. The final lines 42-43 dynamically make a simple web page announcing the *permanent* registration of the phone number which will remain valid for the next five years.

```
1   Private Sub SendEmailNow(ByVal theEmailTo, ByVal theID,
                ByVal theArea, ByVal thePhone)
2    Dim myEmail As New System.Web.Mail.MailMessage
3    Try
4     'assemble the email asking the user to click URL-link for confirmation
5     myEmail.To = theEmailTo
6     myEmail.From = "myVerify@donotcall.govx"
7     myEmail.Body = PrepareBody(theEmailTo, theID, theArea, thePhone)
8     myEmail.Subject = "National Do Not Call Registry - " & _
              "Your Registration Is Confirmed"
9     myEmail.BodyFormat = Web.Mail.MailFormat.Text
10    'using my school mail server
11    System.Web.Mail.SmtpMail.SmtpServer = "grail.cba.csuohio.edu"
12    System.Web.Mail.SmtpMail.Send(myEmail)
13    Catch ex As Exception
14      txtSpy.Text &= vbCrLf & "Problems-" & ex.Message 'only for DEBUGGING
15      Exit Sub
16    End Try
17    txtSpy.Text &= vbCrLf & "Done mailing" 'only for DEBUGGING!
18  End Sub
19
20
21  Public Function PrepareBody(ByVal theEmailTo As String, ByVal theID As Integer,
                ByVal theArea As String, ByVal thePhone As String)
22   Dim theBody As New System.Text.StringBuilder
23   Dim myDate1, myDate2, myDate3 As Date, ts As New TimeSpan(5 * 365, 0, 0, 0, 0)
24
25   myDate1 = Format(Now, "short date") 'request: todays date
26   myDate3 = myDate1.Add(ts) 'expiration: five years later
27
28   'TODO: use the ID-key to encrypt a more complex key.
29   'last piece of the hyperlink is the database key of the current record
30   Dim theLink As String
31   theLink = "http://localhost/myDNC/myDNCRegVer.aspx?xYzAbC?" & theID
32   theBody.Append("Your phone... " & theID & " " & theArea & " " & thePhone & _
          "theEmailTo & " " & myDate1 & " " & myDate3 & vbCrLf)
33   theBody.Append(". Here is the link: " & theLink & vbCrLf)
34   theBody.Append("Click on the link to complete your request to " & _
          "register your phone " & _
          "number with the last four digits " & Right(thePhone, 4) & _
          "on the National Do Not Call Registry. ")
35   theBody.Append(".... ") 'See Figure 9
40   Return theBody.ToString
41  End Function
```



Your phone (key) 66: 216-333-4455 matos@csuohio.edu 2/15/2006 - 2/22/2011.
Here is the link: http://localhost/myDNC/myDNCRegVer.aspx?xYzAbC?66

Click on the link to complete your request to register your phone number with the last four digits 2264 on the National Do Not Call Registry.

Important: You must respond to this email by clicking the link within 72 hours for your registration to be successful.

Once you have registered, your phone number registration will be effective for 5 years. It will be illegal for most telemarketers to call you, and you will be able to file a complaint if a telemarketer does call you. The website www.mydonotcall.gov provides information about filing a complaint.

Important: After you click on the link, print the web page and keep it for your records.

Troubleshooting: If you don't get a "Registration Complete" message after clicking on the link, or the link does not work, use your "cut" and "paste" functions to insert the entire link in the email into the "Address" line on your Web browser. Cut and past the entire link - it's long. Do not re-type it.

****

Please do not reply to this message as it is from an unattended mailbox. Any replies to this email will not be responded to or forwarded. This service is used for outgoing emails only and cannot respond to inquiries.

Figure 8. Email asking consumer to click on link to finalize registration.

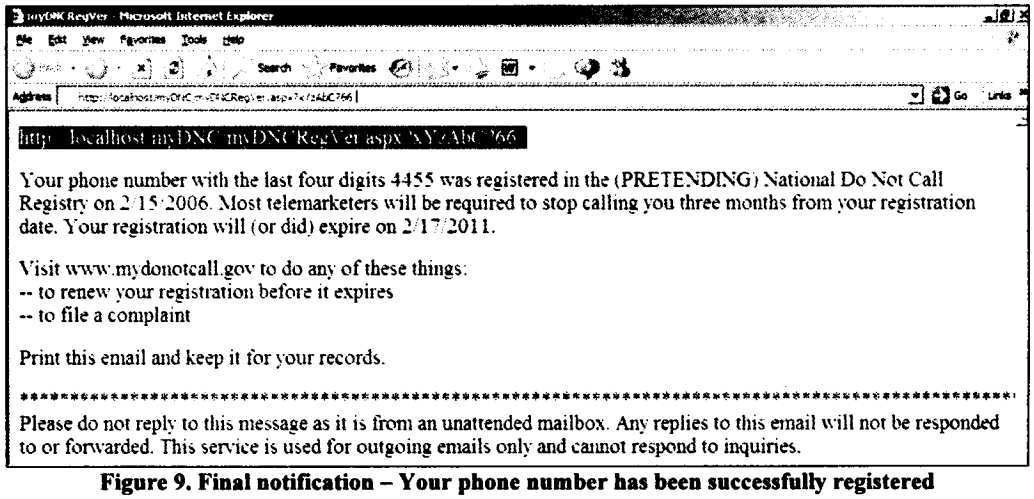


Figure 9. Final notification – Your phone number has been successfully registered

```
1   Private Sub SetDBConfirmationDate(ByVal theIDValue)
2     'CONFIRMATION of a phone number is represented as the entering -in the database record-
3     'of an expiration date (5 years after request date)
4     Try
5       Dim myCnn As New OleDb.OleDbConnection
6       Dim myCmd As New OleDb.OleDbCommand
7       Dim myDR As OleDb.OleDbDataReader
8
9       'setup the connection to the MS-Access database
10      Dim cnnStr As String = "Provider=Microsoft.Jet.OleDB.4.0; " & _
                "Data Source=C:\DoNotCall\DoNotCall.mdb"
11      myCnn.ConnectionString = cnnStr
12      myCnn.Open()
14
15      Dim myRecAffected As Integer, mySQL, thePhone As String, myDate1 As Date
        myDate1 = Format(Now, "short date") 'todays date
16
17      'change the confirmation date in the existing record whose DB key is: theIDValue
18      mySQL = "UPDATE Master SET ConfirmationDate=#" & myDate1 & "# where ID=" & theIDValue
19      myCmd.CommandText = mySQL
20      myCmd.CommandType = CommandType.Text
21      myCmd.Connection = myCnn
22      myRecAffected = myCmd.ExecuteNonQuery
23
24      If myRecAffected <> 1 Then
25        txtSpy.Text = "Failure - " & mySQL      'for DEBUGGING - improve!
26      End If
27      myCnn.Close()
28    Catch ex As Exception
29      txtSpy.Text = ex.Message
30    End Try
31  End Sub

32  Private Function PrepareMsg(ByVal thePhone, ByVal myDate1, ByVal mydate3)
33    'Prepare the body of the email notifying the user about the successful
34    'registration of his/her phone number
35    Dim theBody As New System.Text.StringBuilder
36    theBody.Append("Your phone number with the last four digits ")
37    theBody.Append(thePhone)
38    theBody.Append(" was registered in the (PRETENDING) 'MY-National Do Not Call Registry' on " & _
            myDate1 & "." & vbCrLf)
39    theBody.Append("Print this email and keep it for your records." & vbCrLf)
40    theBody.Append("**********************************************************" & vbCrLf)
41    theBody.Append("Please do not reply to this message as it is from an unattended mailbox.")
42    Return theBody.ToString
43  End Function
```

5. CONCLUSION

In this paper we discussed a complete and technically challenging IT experience. The problem is based on the reengineering of the National Do-Not-Call Registry which is controlled by the U.S.A. Federal Trade Commission. The goal is to produce a functionally equivalent software system, with the obvious limitations that a classroom project imposes. Students participating in this one-semester project showed a great deal of interest in completing the assignment. We argue that the sense of familiarity with the subject, as it looks like a *real life* issue, and the relatively simple technical requirements for implementation make the project an effective candidate for success.

All students in the course were given a syllabus similar to the one in Appendix B. The syllabus provides a step-by-step breakdown of the tasks at hand, and allows the instructor to realistically plan the experience. Every student participated in all aspects of the task, and each student was required to individually submit each of the deliverables listed. The project is planned for a 15-week semester. It includes many individual checkpoints, class discussion, small student-initiated research projects, conceptual and physical modeling, and report and code preparation. In the semesters where this project was made available to our students, we did not allow them to work in groups. This decision was made as the number of students was small enough that individual effort was appropriate. However, team collaboration can be an excellent option for larger class sizes and/or where group work is a course outcome.

According to Martin (1998), "The capstone projects course presents the final opportunity for students to reflect upon the principles and operationalize the skills they have learned in all of their courses through the design, development and implementation of a large scale project". To allow students to reflect on what they have learned, we recommend students write an *Impact* paper, where they discuss how the registry will affect (a) the telemarketing industry, (b) non-profit fundraising organizations, (c) the average telephone owner, and /or (d) IT professionals. (See, for example (Martin, 1998 – Appendix D)) An excellent opportunity to discuss these items is given in the syllabus during the week 3 and week 13 class discussion.

In this project, students were asked to carefully dissect and annotate the behavioral characterization of the internet based portion of the registry. This observation of the real registry was performed in conjunction with the actual registration of the student's personal/home phone(s). Special attention was given to the identification of the business rules governing the registry.

We noted several interesting observations during the semester. The students did not seem to immediately grasp that their simple physical model could be a proper solution to the problem. They spent considerable time looking for a more complex solution rather than considering that their simple approach was satisfactory. In their research, the students found that the implementation costs of the actual system ran in the order of $20 million. They could not reconcile the simplicity of their solution with the enormity of this figure.

Several problems important to the IT professional were identified during the evolution of this project. We did not elaborate on all of them but made a point in identifying issues such as (a) legality of the reverse engineering process (Hansen 2004), (b) ethical aspects of the registry and the free-speech rights of the telemarketers, (c) securing the data against intruders and impersonators, (d) creating maintenance plans including daily housekeeping, backup, recovery, catastrophic failures, etc.

The instructors need to be aware that there may be some problems with the processing of student-generated email. Many institutions prevent (or heavily filter) the sending of mail from non-local accounts or certain areas of campus to avoid potential security risks. It may be necessary for the instructor to involve local IT resources to allow this project to go forward.

One issue we discovered was that with the emphasis on the reengineering of the system, we were unable to spend a satisfactory amount of time on the administrative portion of the experience. We recognize that there are several important administrative and security issues which we were unable to address due to lack of time and resources. These topics include (a) planning for periodic backups, (b) creating profiles for different kinds of users working on the registry (such as DEVELOPER, AUDITOR, OPERATOR, MANAGER, etc.), (c) adding security provisions (mirroring sites, redundant disks in the same location), (d) identifying threats (disruption, destruction, disaster, unauthorized access), (e) assessing the risk of each threat and preparing a response for each case, and (f) creating *controls* to mitigate or stop threats (consider redundancy, fault tolerant servers, disaster recovery plans, anti-virus software, security policy, passwords and encryption, firewalls, etc).

In addition, the instructor may consider alternative methods of implementation emphasizing, for instance, public relations issues behind filing complaints against merchants, and the processing of those complaints, verification of all data, deleting records from the system, and attending consumer requests for information. Other major variations closely related to this project can include similar registries such as a *National Do-Not-Email, Do-Not-Mail, Do-Not-Fax,* and *Do-Not-SMS/Text-Me.*

Even with our failure to cover some activities – normally included in actual software development projects – we believe that this experience is not only appropriate and feasible but fun for the students who enthusiastically embraced the problem and worked towards interesting solutions.

6. REFERENCES

Balena, Francesco (2002), Programming Microsoft Visual Basic.NET (Core Reference). Microsoft Press, ISBN 0-7356-1375-3.

Chikofsky, Eliot (2005), "Reverse Engineering: A Valuable Double-Edged Sword." Foreword for Eldad Eilam's Reversing: Secrets of Reverse Engineering. Wiley.

Hansen, Evan (2004), "US Court: Reverse engineering is 'presumptively legal' ".CNET News.com, 11:40 BST, March 01, 2004.

Herreid, Clyde Freeman (1994), "Case Studies in Science A Novel Method of Science Education". Journal of College Science Teaching (pp. 221-229), February.

Martin, C. Dianne and Elaine Yale Weltz (1998). "From Awareness to Action: Integrating Ethics and Social Responsibility across the Computer Science Curriculum. Third Report from the Project ImpactCS Steering Committee". Available at http://www.seas.gwu.edu/~impactcs/paper3/pg1.html. Accessed August 2005.

Microsoft Corp. Technical Articles 175168 and 316675. Microsoft Knowledge Database.

"Rules and Regulations Implementing the Telephone Consumer Protection Act of 1991". U.S.A. Federal Communications Commission. CG Docket No. 02-278, Report and Order, 18 FCC Rcd 14014 (2003).

Rumbaugh, James, Ivar Jacobson, and Grady Booch (2005), Unified Modeling Language Reference Manual, The, 2nd Edition. Addison Wesley Professional. ISBN: 0321245628.

AUTHOR BIOGRAPHIES

Victor Matos is an Associate Professor of Computer and Information Science at Cleveland State University in Cleveland, Ohio.

Rebecca Grasser is an Associate Professor of Information Systems at Lakeland Community College in Kirtland, Ohio.

Appendix A: Changing a Directory to Allow Web-Requested Read/Write Operations

The Internet GUEST account (IUSR_MACHINE), which is by default part of the "Everyone" group, does not initially have *write* permissions on remote database files (.mdb). This is the most common reason for the exception "*Operation must use an updatable query*". To fix this problem, use the Security tab in Explorer to adjust the properties for this file so that the Internet Guest account has the correct permissions (Microsoft Knowledge Database)

1. Global Change from Basic to Advanced Sharing Properties

1. Execute the Windows-Explorer.
2. Select Tools | Folder Options | View
3. Uncheck the entry "Use simple file sharing (recommended)".
4. Apply | OK.

2. Select Directory and Allow user

1. Select directory that contains the shared file.
2. Right-Click on the folder.
3. Choose Properties | Security
4. Click on Add... to enter anonymous web user.
5. Enter the line **<MachineName>\aspnet.** Where *MachineName* is the machine on which ASP.NET is installed on.
6. Click *Check Names* ... to verify the validity of the (anonymous) user name.
7. Select the *aspnet* user you just created and give him/her *Modify* and *Write* access to the directory.

3. Changing the Share/Change options of a Directory using Basic Sharing Properties

Assume the file we want to share/modify on the web is called *myDoNotCall.mdb*. Let the path to the database be **c:/myDoNotCall/myDoNotCall.mdb**. The folder c:/myDoNotCall must be set as "shared" and "modifiable". To do that, follow the instructions

1. Create the folder c:/myDoNotCall. Copy the sample database into the folder.
2. Right-click on the folder's name. Select *Properties*. A pop-up window appears showing several tags. Click on *Sharing*. Check the two options (a) *Share this folder on the network* and, (b) *Allow network users to change my files*.

Note.

Another (less desirable) option to work around the "Updatable query" exception is to store the database file into a subdirectory that has already been granted the Global and Shared privileges. Files under Windows-XP could be remotely accessed via web by just moving them to the folder

C:\My Documents\All Users\Shared Documents

This directory has already all the options set (including *Read, Modify, Write*) and allows *everybody* to access the files

Appendix B: Sample Syllabus and Short Instructor Notes

Date(s)	Notes
End of week 1	**Submit** team name, team logo, and names/email address of all team members
End of week 2	**Register** your phone number with the actual Do Not Call Registry. **Submit** resulting printouts
End of week 3	Class Discussion (1) Free speech of the telemarketing industry vs. the consumer's right to privacy. Take a position – which side did you chose and why? (2) The legality of reverse engineering of existing systems and artifacts. Copyright laws. Take a position – which side did you chose and why?
End of week 4	**Submit** use-case diagrams
End of week 5	**Submit** required systems analysis including your sequence and transition diagrams.
End of week 6	Class Discussion **Agree** on a physical implementation of the conceptual model.
End of week 7	**Submit** required database documentation including **demonstration** of completed MS Access implementation
End of week 8	**Submit** screen prints of completed graphical user interface (GUI) for the home page (first entry page) (No functionality, GUI only)
End of week 9	**Submit** screen prints of completed graphical user interface (GUI) for the complete registration process (No functionality, GUI only)
End of week 10	*Begin* documentation process
End of week 11	**Submit/demonstrate** working application - up to, but not including, the email process (Insure your database is being correctly updated with all applicable information)
End of week 12	**Submit** screen prints of completed graphical user interface (GUI) for the email process (No functionality, GUI only)
End of week 13	Class Discussion: Security and System Management. Identification, assessment, and response to threats. Ethical issues surrounding data assurance, security, and privacy.
End of week 14	**Submit** the following documentation: Note: All manuals must be typed; have a table of contents and/or index; and finally a professional layout, easy to follow and understand **User's Manual:** How do users properly use your product? What happens when an error is received? How is it corrected? What technical issues might be encountered? Insure it can "stand alone" without you there to explain. Include a "How To" for every activity in your site, common and not-so-common errors explained and fixes suggested, and so forth. **Application Administrator's Manual:** How do administrators properly use your product? What happens when an error is received? How is it corrected? What technical issues might be encountered? Insure it can "stand alone" without you there to explain. Include a "How To" for every activity in your site, common and not-so-common errors explained and fixes suggested, and so forth. **Software Administrator's Manual** This manual has three parts (1) a test suite, (2) a security suite, and (3) installation instructions. **Test Suite:** You will need to provide a test suite and associated documentation that adequately and sufficiently tests every line of code in your application. Your test suite documentation should be such that another programmer can adequately test your project without you being in the room. *User Level Testing* For every user page, do you have screen shots of the page and all possible errors/error messages and corrections? *System Level Testing* System Charts included in test suite For every decision point/loop construct - have you described and then prevented "illegal" data points? Discussion of external issues, such as networking, browsers, and operating systems **Security Suite**: What are your security mechanisms? How have you tested them? **Installation Instructions**: Starting on a clean computer (only the operating system installed and web server software), install your application so that it is ready to be served. Work on your presentations and clean up any last minute coding.
End of week 15	**Submit/demonstrate** completed working application

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096