2000

Web-based workflow in secure collaborative telemedicine

Vijayanand Ranganath Bharadwaj
West Virginia University

Web Based Workflow in Secure Collaborative Telemedicine

Vijayanand Bharadwaj

Thesis submitted to the College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for
the degree of

Master of Science
in
Computer Science

Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2000

Keywords: Workflow, Internet, World Wide Web, Internet Security,
Automation, Smartcards, Mobile User

Abstract

Web Based Workflow in Secure Collaborative Telemedicine

Vijayanand Bharadwaj

Workflow involves the automation of a business process in whole or part with movement of information and tasks between parties according to a set of rules. The ubiquitous nature of the Internet and WWW prompts us to investigate how these entities can be fully exploited to improve the efficiency of any workflow process in a reliable, secure fashion.

This research effort aims to investigate the utility of the internet and specifically the WWW in providing a substrate for complex workflow and to demonstrate that workflow with a high degree of automation and ability to interface with disparate data sources can be conducted in a reliable, secure fashion with synchronous and asynchronous communication between mobile participants.

Paper based administrative workflow in healthcare has been automated to validate the claims made in this thesis. Web pages substitute paper forms and information needed is obtained from disparate data sources. Information is transferred across the Internet is using strict security protocols and communication is supported with the help of messaging. Users are not tied to any system and are mobile. Automation techniques decrease manual effort, minimize errors and improve the overall efficiency of the workflow process.

Acknowledgement

I am grateful to Dr Ramana Reddy, Director of the Concurrent Engineering Research Center, (CERC) and committee Chairman for his constant guidance, support and for allowing me to work on my Master's thesis at CERC, West Virginia University. I am very thankful to my committee members Dr. Sumitra Reddy and Dr.Bojan Cukic for their encouragement and enthusiastic support.

I owe a deep sense of gratitude to Mr.Ravi Raman, Asst. Director For Technology & Program Development (CERC) who has guided, motivated and encouraged me through all the stages of this project.

I would like to specially mention William Hunt, Igor Lapshin and Alan Butcher who are part of the Secure Collaborative Telemedicine Project development team at CERC for their cooperation, invaluable technical support and timely advice without which this effort would not have been possible.

I am indebted to the rest of the Secure Collaborative Telemedicine Project development group of research associates and research assistants whose friendly and helpful attitude made this research experience very enjoyable.

Many thanks to the administrative staff at CERC for their efficiency in helping me deal with the administrative formalities involved during my tenure at CERC.

Finally I recall with gratitude the help rendered by many others who gave their time and advice without which this project could not have been accomplished

Vijayanand Bharadwaj

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for this exercise

As the new millennium dawns, one cannot help but feeling the all-pervasive impact that the Internet, the world wide web and the web browser in particular have made to every walk of life of life, right from personal health, nutrition and fitness to electronic commerce, education, entertainment, sports and recreation. It is difficult to miss the rapid changes taking place in nearly all of the fields mentioned above due to the undeniable popularity of the Internet and it is apparent that more and more developers and solution providers are taking a “Web-Centric” approach to building their systems apart from the already prevalent Client –Server Models which use the Internet and Intranets.

However some of the major concerns for solution providers regarding a Web Based system are

a) How **reliable** is the system in terms of guaranteeing that information will be transferred to the correct recipient(s) immediately or at a specified time.

b) How **secure** is the system so that there is no eavesdropping or tampering of information in transit.

c) How fast is the system in terms of **providing interactivity** (synchronous communication) to the users.

d) Does the system provide means for **asynchronous communication** where the end parties may not be present in front of their terminals.

e) Can the system be used as a means for **decision support**.

f) Can the system be integrated into a **workflow environment**.

g) How **cost efficient** is the system in terms of deployment at the end user’s site .

h) Is the system **extensible** and will this affect the end–user in terms of upgrades.

i) Will it improve the **overall efficiency** of the process.

This thesis aims to demonstrate the use of the now ubiquitous web browser in a highly sensitive environment which demands that all of the above criteria be satisfied for success. In addition another important feature is the **physical mobility** of the client (user) i.e. it is desirable that an end user not be constrained with the physical location of work but be able to roam and access the system from any machine where a web browser is available.

1.2 Objectives

The primary objectives of this research effort are as follows

1. To use a web browser which is easily available (downloadable) to an end user as a multi - faceted client in a Web Based System.
2. To enable a workflow scenario which involves the following:
    a. flow of information
    b. decision support both automated and by human intervention
    c. synchronous communication between the user(s) and the system
    d. asynchronous communication between the user and other users

    The web browser with rich options for interactivity provides all of the above.
3. To show the level of reliability of the web browser as far as transfer of information is concerned.
4. To demonstrate that all the above can be done in a secure fashion.
5. To be able to extend the application without any overhead of upgrades or maintenance as far as the client is concerned.
6. To demonstrate the mobility of the end user.
7. Automation : To increase the degree of automation so that the user is freed of tasks such

as typing in large amounts of data and also provide for error checking of information entered by the user.

Most earlier solutions which incorporated the World Wide Web and the concept of Workflow provided intranet based solutions and did not address the security aspect. Also security was tied to the operating system and thus it made it necessary that the end user be part of some domain or the transaction was not secure. This effort aims at demonstrating that it is definitely possible to maintain a high level of complexity in a workflow scenario, and still be absolutely secure.

The author's endeavor is to show that such a solution is cost effective and improves the overall efficiency of the process since it does not involve the hassles of maintenance and upgrades on the end user's side. Thus it can support a large number of users.

1.3 Methodology

A non-conventional approach was used to complete the application. The **Rapid Prototyping** approach was preferred over conventional **System-Lifecycle** approach since all requirements could not be determined *apriori*. Another reason was that most of the implementation had to be tested as the process continued before one could finalize the requirements and if the test results were not satisfactory then alternate means would have to be found. Chapter 3 describes the technology used to build the system. The technology mainly deals the Internet and World Wide Web along with Smart Cards. The web servers, web browser, components used and other details are fully described in Chapter 3.

1.4 Scope of the Project and Usage Scenario

An example from the Healthcare Industry has been chosen due to the fact that the daily transactions involved in a Healthcare scenario involve highly confidential Medical records, namely patient records and other administrative information. Healthcare Transactions typically have to satisfy all the objectives mentioned above and as most of these transactions are currently being carried out using paper-based means to store and retrieve records and conventional means of communication (such as phone, fax and to some extent e-mail) are being used. The aim was to assess how the overall process could be improved by automating some parts and to see if it satisfied all the above mentioned objectives.

This research effort is a part of the existing project namely the “Secure Collaborative Telemedicine “which is being carried out by the development team at the Concurrent Engineering Research Center[7] at West Virginia University[49].

The above project also involves the extensive use of the Smart Card Technology , Vital Signs Server and Web Based Medical Records [35]. Some of the above applications are integrated with the author’s effort especially the Web Based Records and the Smart Card Technology.

In many spheres such as healthcare, legal, defense, education, an electronic workflow solution would be very useful compared to the existing manual procedures. All these fields have tasks which involve administrative duties and non administrative ones. For example, in the healthcare scenario the patient could be examined and his medical records could be accessed across the Internet and be updated. This project tries to demonstrate electronic workflow involving an administrative task in the healthcare scenario.

A scenario which is quite common in the Healthcare Industry is that of the physician placing a “Request for Exemption from Criteria” with an agency for approval of a medication being prescribed. Conventionally the whole process involves the physician having to fill out a paper based form which is available at the agency. This form contains information about the patient (who is already registered with the agency), physician’s information, information about the physician’s clinic, pharmacy information and the details about the medication which needs approval. This form is then faxed to the agency where it is processed by the agency personnel .

From this point onwards all the communication takes place either via phone or fax.

If the agency personnel require some additional information that the physician's office has not provided or some fields on the form or incorrect or incomplete the agency has no choice but to fax the from back to the physician's office. All this involves the physical movement of paper and this means that there could be delays if mail is used or if it is faxed, the paper may not noticed by the personnel and lie on the fax machine. Also if telephone is the primary means of communication then to obtain results the party to be contacted must be available and must have a phone at all times which may not be possible. Paper forms may also be lost, stolen, or destroyed. This can put highly sensitive and confidential information in jeopardy and could result in security of the information being compromised. Paper based forms also imply that at various points in the process the personnel may have to enter some of the information into a computer, if part of the system is automated. Hence there is manpower involved and a significant waste of effort, time and money to accomplish tasks which are routine and could have been automated. In addition this may lead to errors. Also requiring the physician to enter data manually is a waste of valuable time for the physician and as always there is error-prone. Hence there is an overall decrease in efficiency of the whole process.

The solution to this problem is to improve the whole process by automating it in such a way that it does not affect conventional means of work and that it can be extend to incorporate conventional as well as automated Workflow.

Some ways in which this process can be accelerated and made efficient are as follows:

1. Automated filling of electronic forms to save time, effort and minimize errors.
2. Immediate validation of submitted forms on-line to avoid the delay in having to send for corrections after the forms have been examined.
3. Electronic storage and retrieval of transaction information to avoid loss of information .
4. Asynchronous notification facility via email in addition to conventional means of communication to enable the client to access the information anytime he or she is on-line.

5. Allow the user to be mobile by allowing access to information anytime and from any computer.
6. All transactions between the client and server to be carried out using the strictest security standards available for such purposes.

The above can be made possible by the use of the Internet, World Wide Web and the security protocols which come with it along with a host of other technologies such as database management systems, smartcards, active components that work well with the WWW infrastructure.

By automating the whole process we expect to achieve the following:

1. Save time and effort of parties concerned, be they physicians or other personnel due to automation of the form filling process.
2. Minimize errors due to data entry
3. Ensure reliable transfer of data.
4. Improve the quality of interaction between the parties concerned mainly the speed.
5. Asynchronous notification via email to contact the right people. People can periodically check their email remotely and can access notification information.

1.5 The Role of the Web Browser

From an end user's point of view the Web Browser has become an indispensable tool. It can be a window to a vast galaxy of information and which will provide opportunity for almost every form of information and interactivity, be it plain-text, image or voice. System developers and solution providers are taking advantage of this aspect of the web browser and aligning their applications so that they utilize the full potential of the Internet and the various technologies associated with it.

Be it the Internet or an Intranet which enable transactions involving the flow of information and support decisions needing to be taken based upon information available, it is highly desirable that such processes take place quickly, reliably, securely and in a cost effective manner. The web browser provides a means for just that. It provides a solution provider a cost effective solution to the problem of deploying a client application at all end user terminals. An end user can just download a web browser and avail of all the features. Also today's web browser come with sophisticated features which enable high level of interactivity and most importantly, security.

Information technology solutions concerned with fields such as Healthcare, Law, Defense and Business and Banking are especially concerned about the security and reliability of the information flowing through their system and the web browser supports security protocols which can provide the necessary levels of security.

The highly interactive nature of the web browser in addition to the above features has demonstrated that Web-Based solutions are very stable and cost-effective and go a long way in improving the overall efficiency of any process.

1.6 Preview of Chapters

The subsequent chapters are organized as follows. Chapter 2 provides an in depth review of the various concepts involved with this body of work and a brief review of previous work done in this arena. Chapter 3 enumerates the various technologies involved along with a brief insight of the same. Chapter 4 discusses the design of this system and the implementation scheme of this system is treated in Chapter 5. Results of the work done, Conclusions and future work possible in this field are the topics discussed in Chapter 6.

Chapter 2

Background

This chapter discusses concepts involved in this research effort and the previous work relevant to it. A brief overview of the concepts demonstrated in this effort will help to bring out the commonalties and the differences between this effort and its precursors.

Some of the key concepts involved are those of workflow, collaboration using the World Wide Web, Web Browsers as Clients, Server Side and Client Side Scripting, Client Side Authentication, Active Components, Internet Security and some facets of Smart Card Technology. Work done in the arena of Workflow using the World Wide Web and how this effort differs from the previous research is also discussed here. This chapter focuses on the concepts whereas the next chapter "**Technologies Used**" provides a brief overview of the actual technologies which demonstrate these concepts.

2.1 Concepts

Some of the fundamental concepts upon which this thesis is based are discussed below.

2.1.1 Workflow

According to the Workflow Management Coalition[1][52], workflow is defined as follows

"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

[1] http://www.aiim.org/wfmc/mainframe.htm

Workflow is the computerized facilitation or automation of a business process, in whole or part. Various day to day activities carried out in an environment involving people, business process, tasks and information flow which support decision making processes constitute a “workflow”. During the course of a workflow there may be changes in the process, information and status of tasks. This workflow may be automated or may be manual. However in the context of this effort, workflow does involve a large degree of automation. A very simple example could be that whenever an individual receives an e-mail regarding a certain subject of interest she sets up her software such that it automatically forwards that piece of mail to all the members of a specific group who share the same interest. A very complex example could be one which involves extensive flow of information across remote systems and remote initiation, monitoring and control of business processes or “work activities”. Communication between various parties involved could be synchronous or asynchronous and use conventional (mail, phone, fax) and modern means of communication (email, pagers, video conferencing).

There has been a growing acceptance of workflow technology in numerous application domains such as telecommunications, manufacturing, production, finance and banking, laboratory sciences, healthcare, shipping and office automation.

The usage scenario mentioned in Chapter 1 involves workflow which is highly automated and involves the business process of filling and sending an Exemption claim and also the process of examining the claim and either approving, denying or taking some other action based upon the content of the request. In addition the task of notification means there is an element of communication.

2.1.2 The Internet and the World Wide Web

The Internet [44] is a huge computer network which can be termed as a “network of networks” [20] that connects millions of computers globally and provides world-wide communications to businesses, homes, schools, and governments.

The Internet has grown explosively in the 1990s. There are now more than twelve million server computers on the Internet, each providing some type of information or service. The number of users of the Internet is more difficult to measure. Since each service on the Internet is used by many people, many millions of users are currently online. The number of users and services on the Internet continues to grow as the variety of the services increases.

In describing the Internet and its uses various authors and developers choose to do so with respect to differing levels of abstraction ranging from the physical components (the wires, cables, routers, bridges and other associated hardware) to the highly complex applications which run on the servers and clients. We confine ourselves to the software programs which provide services and are called Servers and the programs which make use of these services which are the software Clients.

Perhaps the most popular Internet service or application, the **World Wide Web**, has accelerated the growth of the Internet by giving it an easy to use, point and click, graphical interface. Users are attracted to the World Wide Web because it is interactive, because it is easy to use, and because it combines graphics, text, sound, and animation into a rich communications medium.

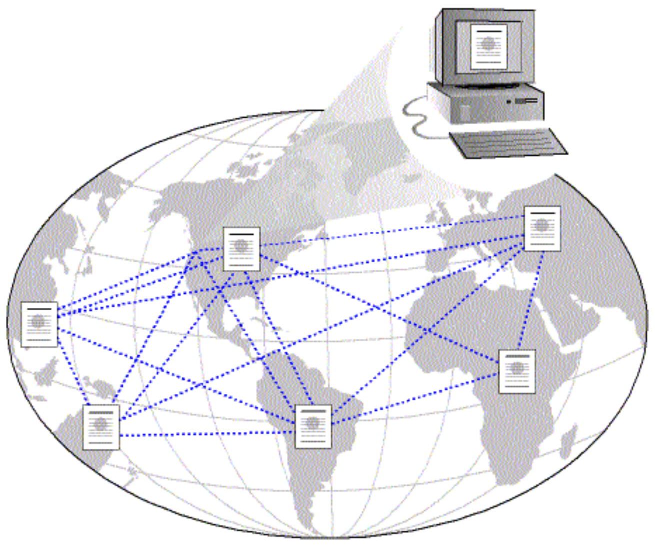


Figure 2.1 The Internet

The **World Wide Web** is the most popular and fastest growing Internet service. It uses hypertext links called ***hyperlinks*** to locate and retrieve *pages* from World Wide Web servers. World Wide Web pages combine sound, graphics, animation, text, and software programs into dynamic documents. Users view the World Wide Web from ***Web browsers***, client programs that can display the HTML (Hyper Text Markup Language) in which pages are written.

2.1.3 Servers and Clients

Every computer program that communicates on the Internet is either a server or a client.

A *server* offers a service to other machines on the network. For example, a *file server* contains files of interest to users around the network — the service it offers is the storage and retrieval of the files. There are thousands of file servers on the Internet, some offering files of a particular type, such as artwork (a file can contain an image), weather maps, or tourism information.

A *client* requests a service from a server. Most services are designed so that specialized client software must be running to interact with the server. For example, to get a file from a file server, a client program on your machine must format and send the request to a program running on the server. The server program locates the file on the server machine and sends a copy of it back to the client program which saves it or opens it for immediate use. The client and the server use a common method for communicating and for passing the file from one machine to another, called a *protocol*.

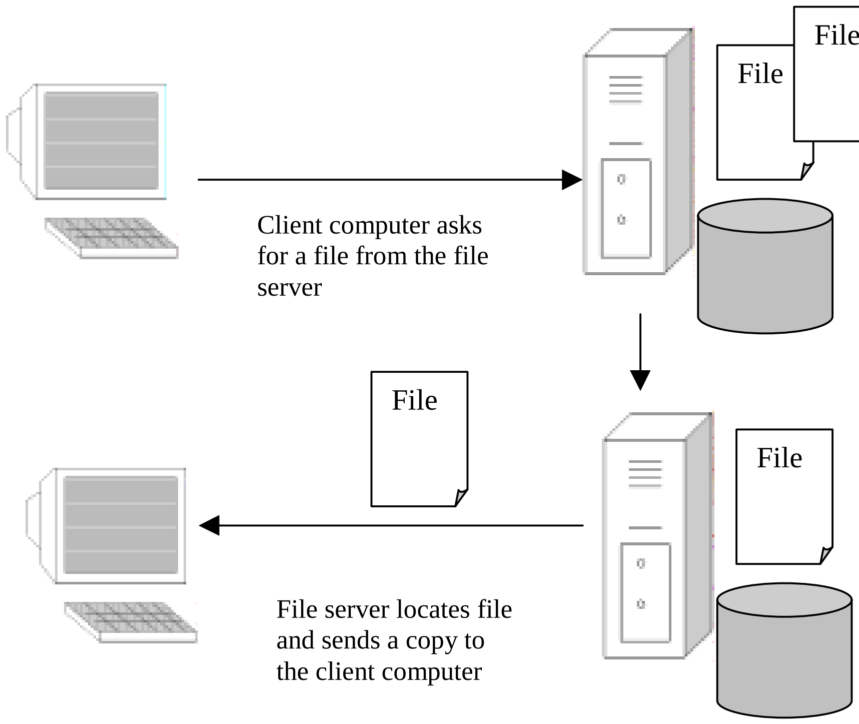

File
File
Client computer asks for a file from the file server
File
File
File server locates file and sends a copy to the client computer

Figure 2.2 Servers and Clients

2.1.4 Web Servers and Web Browsers

The World Wide Web being an internet application is essentially built upon a Client-Server model. The basic building blocks of the World wide web which can be thought to be the "**Web Servers** " and "**Web Browsers**".

A World Wide Web **client** is called a **Web browser** or simply a ***browser***, and a World Wide Web **server** is called a ***Web server*** or sometimes just a ***server***. Web browsers and Web servers use a set of communications rules called *Hypertext Transfer Protocol* (HTTP).

Web Servers are software programs which host web pages and can run applications on the native operating system. These web servers can connect to databases, other networks and communicate with other hosts. They respond to the Client requests for a particular page and send it back to the client computer.

A **Web browser** is a program that navigates through the World Wide Web and displays pages. The Web browser requests a page from a server based on its Internet address. It retrieves the document from the server and displays the contents.

As HTML evolves, Web browsers are evolving with it, adding more advanced functionality. For example, Microsoft Internet Explorer displays ActiveX Controls and runs VBScript programs and Java applets.

2.1.5 Workflow using the World Wide Web

The Internet and more specifically the World Wide Web has been chosen as the substrate for this example because of its ability to render itself very easily to a complex workflow paradigm. The existing protocols of the World Wide Web can be used to accommodate a workflow model.

In the last few years, pervasive network connectivity, explosive growth of Internet and web technologies has changed our computational landscape to distributed, heterogeneous and network centric computing model from centralized, desktop oriented, and homogenous computing. This has raised challenging requirements for workflow technologies in terms of being required to support heterogeneous, distributed computing infrastructures, interoperability, scalability and availability. Basic units of any organization are the work processes performed in it. A process refers to a business process and its corresponding information process. Workflow management involves everything from defining and modeling processes up to synchronizing the activities of performers (information systems or humans). Subsequent sections will enumerate recent efforts by others to customize the existing protocols and to come up with new ones which offer true interoperability among different workflow systems.

The following few sections explain some of the core concepts based upon which the World Wide Web applications of today are based on. Initially web pages were simple, text only, static pages and were meant to be read only. However they have evolved with the addition of scripting, dynamic HTML, ActiveX components and Applets into dynamic pages. They can also be made to interact with databases and legacy systems. This has lead to the formation of the Active Web Site concept. The Active Web Site has become a highly interactive medium which can deliver rich content and can form a substrate upon which large distributed applications can be developed and run.

2.1.6 Web Pages, Static and Dynamic

Hypertext Markup Language(HTML)

HTML [53][48][24] is the language of the World Wide Web. An HTML file is what a web browser uses to generate a web page. One can call the World Wide Web a vast collection of HTML files residing on hard drives of computers across the world along with a transport protocol(Hypertext Transport Protocol or HTTP) which enables the file transfer between computers. HTML files are just text files in a human –readable format. Web browsers perform the task of interpreting these html files.

In its infancy the World Wide Web was made up of plain HTML files. These files resulted in web pages which were **static**. Though they did display color and graphics along with text, they did little in terms of increasing the level of interactivity with the user and did not harness the power of the client's machine fully. In fact every time the programmer had to show the user something different based upon the user's actions, a totally new page had to be displayed which meant that a new request had to be made to the web server and then the server would respond with the appropriate page. Even currently many web sites host such pages.

This resulted in web technologies evolving towards **"Dynamic Web Pages"** or more "**Active Web Pages** ". This involved displaying different web pages based upon different users needs from the same site, changing the appearance of the page "on the fly" after the complete page had been downloaded while the user interacted with it, reacting to user's actions and more all without making a request to the server for a fresh page. Hence the downloaded page contained all the necessary logic to act to the user's actions by running code on the user's local machine.

The key concepts which influenced this paradigm were the Document Object Model and Scripting.

Dynamic HTML (DHTML)

DHTML [18][8] introduces the concept of a **Document Object Model** wherein a browser can display a web page and then provide a standard mechanism to access the various parts of the displayed document. In fact the basic idea is that every part of the web page is an Object with properties and events which can be manipulated by the user. Even the Web browser itself is considered as an object. Thus a developer can programmatically control the behavior of every element on a web page. Thus interactivity between the user and the web page increases by a large amount due to the fact that event handlers to users actions can run on the users client machine after the complete page has been downloaded.

This removes the need to request new page every time the display has to change or some other action has to be taken in response to the user 's actions.

Scripting

Scripting [18] completes the picture as far Dynamic Web pages are concerned. Scripts are blocks of program code that are "embedded" in web pages and are interpreted at run time i.e. the text file containing the HTML also contains additional code . The browser takes actions based upon this code. Scripts are written in a Scripting Language and act as the framework which tie the Document Object Model to the User's actions and provide a rich level of interactivity.

Some other technologies which make the web more interactive are Active X controls and Java applets. Scripting can occur both on the Client side as well as the Server side as described below.

2.1.7 Server Side Scripting

A server side script is one which can execute on the server (web server) and not on the Client's browser. This script can control the flow of what is to be displayed to the user based upon the user's requests and actions so that only the page is customized for the user. These scripts can also connect to databases on the server side and run applications on the native operating system or those running remotely.

Server side scripts can also make use of the Underlying messaging services and use the communication protocols to send messages to different hosts.

2.1.8 Client Side Scripting

Client side scripting involves the same concept as server side scripting excepting that the scripts execute on the Client's browser. The scripts make use of the Client's resources. The Client's security settings determine the degree of access that the scripts may have on the local machine.

2.1.9 Components

Component based software [42] is a technique used to overcome the inherent complexity of software by improving maintainability, enabling re-use of code and increasing production of robust software. This involves selecting a number of items of software, each of which performs a specific item of functionality and gluing them together to form a single integrated solution.

A software component can be defined as a combination of data and code which acts off that data, and together can be considered as a single unit. The data and code associated to the component define everything that the component represents (state) and what it can do (behavior). These characteristics are specified by the *Component's Definition.*

An object is a created instance of a component. A component can have many instantiations thus many objects.

Component Objects are created according to the COM or Component Object Model standard.

The primary architectural feature of the Component Object Model, is that it allows COM to completely preserve encapsulation of data and processing, a fundamental requirement of a component software standard. From the Internet and World Wide Web perspective, some of the advantages of using components are as follows:

a) Components can be implemented in a number of different programming languages and used from clients that are written using completely different programming languages. This is because COM, unlike an object-oriented programming language, represents a binary object standard, not a source code standard.
b) Components can be updated and reused if they are written according to the COM standard.
c) Tasks which are difficult to complete using standard scripting techniques can be accomplished using Components. This thesis demonstrates one such important example where a component can easily interface with the local Smart Card Reader on the client machine which a Script cannot do.
d) They can be easily downloaded as they are ”thin” and do not take too much of the Client machine’s resources though they can access the resources.

Components are further classified into Server Side Components and Client Side Components. Server side components run on the server and use the local server resources .

They can be designed to access various other applications and can perform a wide variety of complex tasks thereby extending the Web server’s potential from just being host to web pages to a complex work environment. In the following chapter we shall see a description of the ActiveX Data Objects and ASPEmail which are example of a server side components.

Client side components get downloaded onto the Client’s machine and run there . The tasks they can perform may be limited due to the security considerations imposed upon them. The

HealthCardUpdate Control and the HealthCardAccess Control are two such client side components that have been used in this project.

2.1.10 Active Web Sites and Active Web Pages

Currently, the concept of an Active Web Site and Active Web Pages have made significant strides in Internet Application Development. Active Web sites are built using a combination of a number of technologies and languages. These can be used to generate not only extremely attractive, highly interactive web pages but also actual complete Internet Applications that can access complete databases and legacy systems. Active Web sites involve incorporate Server Side Processing of the Web Pages, Advanced Messaging and Communication between various heterogeneous systems

In the context of workflow they provide the Workflow Engines needed by a workflow application.

Some of the technologies which enable a web site to be truly active are CGI or Common Gateway Interface, Active Server Pages, ActiveX Components (both Server side and Client Side), Active Messaging Platform, Active Data Objects, Dynamic HTML and Scripting, Extensible markup Language (XML), to name a few. Subsequent chapters discuss some of the technologies and the manner in which they have been incorporated into the design and development of this system.

2.1.11 Active Platform

A core concept on which an Active Web Site is based is the **Active Platform**. The Active Platform software architecture is a development platform that provides developers with an easy way to build applications that take advantage of the best of the Internet and the best of the PC. For the first time, developers can now target a consistent, unified platform from client to server to build robust Web-based application with a broad set of easy-to-use tools. For the end user, the

Active Platform allows a seamless browsing experience, regardless of the operating system being used. The Active Platform uses Server side tools to build applications which run on the server and can perform tasks such as interfacing with databases. Additionally server side tools can also create modules or components which can perform tasks on the client machine as long as the client also follows the standards of the Active platform. The clients in turn can access the applications running on the server via the Internet and World Wide Web, download the components created and work with them to perform tasks without worrying about the operating system on the server side and other such details.

The Active Platform is based on three core technologies: Active Desktop, Active Server, and ActiveX which will be explored in greater detail in the following chapter.

2.1.12 Server Side Processing and Client Side Processing

Server side processing involves the server processing scripts and instructions sent to it by the client browser, running and calling remote applications and sending the results back to the browser.

Client side processing involves the client browser in processing and decision making based upon scripts or instructions which would be usually sent to the server. This means running of scripts, components and accessing the local machine's resources.

Both these modes of processing have advantages and disadvantages and this research effort exploits both to a high degree. Chapters 4 and 5 describe the use of both modes along with their pros and cons.

2.1.13 Client Side Validation

This refers to the process in which data submitted by the user is not directly sent to the server but is actually checked and validated in a sort of pre processing effort.

This is an example of client side processing and has numerous advantages in terms of connections with low bandwidth and data integrity

2.1.14 Internet Security

In view of the fact that increasingly sensitive information in large amounts is being sent across the internet, it is of paramount importance to ensure that the information reaches its destination in a secure fashion. The sender is assured that the information is being sent to the correct party and the receiver is assured that the information is being received from the desired party. Also the receiver is sure that the information received is not bogus. Finally no one should be able to eavesdrop on the information and the information is not tampered with while in transit. Chapter 3 discusses some important aspects of Internet Security and how they have been incorporated in this thesis.

2.1.15 Smart Card Technology

A smart card [19][33] could be considered as follows:

"A smart card is a credit card sized plastic card embedded with an integrated circuit(IC) chip and thereby also known as the Integrated Circuit card(ICC). This integrated circuit chip stores information and controls who uses this information and how."

The smartcard is a tamper resistant and portable device and can carry a variety of useful information. A wide range of information can be carried on the smartcard beginning with simple identification information of its owner to complex medical records and insurance information (depending upon the capacity of the card and the information format).

Apart from this feature smartcard's can carry security information such as private keys and digital certificates, which play a major role in internet security.

Portability and convenience are the major advantages of Smart Card technology. This combined with the fact that smartcard information can be accessed by Components adhering to the COM standard and that this extracted information can then be transferred across the internet in a secure fashion, gives scope for exploitation of this useful technology in a wide range of application fields. This thesis explores one such scenario pertaining to the healthcare industry.

Another important feature of smartcard technology is the high-level and variety of security they offer to protect the card, data and the system supporting the card from fraudulent use. A smartcard may carry everything from the Data Encryption Standard (DES), which provides secure authentication of system components and data , encipherment of data, and the certification of data , to biometric information such as finger prints, for positive identification of the cardholder. Secure access control mechanisms provide the ability to store very sensitive information. Also the fact that the smartcard itself provides a secure environment for computations, such as generation of digital signatures is a plus since this environment cannot be compromised by viruses and trojan attacks on a PC.

And finally with the evolution of standards for interfacing the PC with the smartcard (PC/SC) the picture is complete and this technology is rapidly gaining popularity.

The detailed analysis of this technology is beyond the scope of this report.

However [19] and [33] provide detailed descriptions about the technologies' various features, architecture, standards and security mechanisms.

It must be noted that the Smartcard is one example of a secure information repository which can be integrated with a web based workflow and that the GINA is a method of accessing this in a secure fashion. Other systems can be still used like databases along with password protection.

2.1.16 Workflow In Healthcare

With the increasing pressure of having to deliver a multitude of services at lower costs but maintain or improve the quality of care many health care providers have turned to embrace technology and take a leaf out of the books of other industries which have used the concept of *workflow*[6][16]. The specific needs of the healthcare industry make it an ideal match with workflow concepts.

Healthcare involves complex procedures that include both clinical and administrative tasks.

As a result, workflow increases efficiency and effectiveness through the maximal use of relevant, timely information. This pressing requirement makes healthcare unique among industries in its ability to take advantage of the information benefits provided by the implementation of workflow concepts and hence been chosen as a scenario for this research demonstration.

Healthcare related protocols can be divided into Clinical Protocols and Administrative protocols.

For example, a critical pathway attempts to sequence care tasks, coordinate medical and non-medical care resources, and set a defined timeline to ensure that milestones are met. Administrative protocols, which involve processes such as obtaining prior authorizations and providing clinic referrals, are healthcare workflow processes as well.

Other Administrative tasks include checking membership eligibility, completing necessary insurance or encounter forms, providing Advisory Committee on Immunization Practices (ACIP) immunization guidelines, tracking height and weight to norms, and distributing patient education sheets (e.g., immunization side effects).

The major benefits of using a Workflow Process in Healthcare reflect the objectives of this research effort and are as follows:

Automation of the workflow process in healthcare makes physician orders standardized in format, reducing the likelihood of error, duplication or omission. Further, as completed tasks are documented, an easily accessible record of what care patients receive is created. This clear, accurate record of specific patient information improves communication among providers. Moreover, this information is available at the ***point of service***, when and where the clinician needs it most. Finally, the logic branching in healthcare workflow allows flexible implementation of previously agreed upon courses of action which produce the best outcomes. This decision support permits the clinician to concentrate on the more complex clinical choices presented by patients, rather than the routine clinical tasks, such as immunizations or cancer screenings, which can best be managed through automation.

2.2 Previous Work Done

This section mentions briefly, some of the previous work done in the realm of workflow on the Internet and some of the latest developments in research by the major special interest groups involved in workflow research. We look at the efforts in primarily two areas, one workflow itself and the other , workflow and the Healthcare scenario.

2.2.1 Workflow

The Workflow Management Coalition(WFMC)

The Workflow Management Coalition [52], founded in August 1993, is a non-profit, international organization of workflow vendors, users, analysts and university/research groups. The Coalition's mission is to promote and develop the use of workflow through the establishment of standards for software terminology, interoperability and connectivity between workflow products.

Consisting of over 130 members, spread throughout the world, the Coalition has quickly become established as the primary standards body for this rapidly expanding software market.

Some of the Achievements of the WFMC have been publishing of the Reference Model and Glossary, defining a common architecture and terminology for the industry. A major milestone was achieved with the publication of the first versions of the Workflow API (WAPI) specification, covering the Workflow Client Application Interface, and the Workflow Interoperability specification. The Audit Data specification was added in 1997, being followed by the Process Definition Import/Export specification. A further version of WAPI covers

Application Invocation API's, completing the Coalition's initial deliverables across the five interface functions. Further work includes the completion of a common object model with object bindings for IDL and OLE, interoperability extensions for security, and additional interoperability models. The Coalition has validated the use of its specifications through international demonstrations and prototype implementations. In direct response to growing user demand, live demonstrations of a workflow interoperability scenario have shown how business can successfully exchange and process work across multiple workflow products using the Coalition's specifications.

The Software Workflow Access Protocol (SWAP)

The main objective of this working group[46][4] is to define requirements and develop Internet based Workflow Access Protocol to instantiate, control and monitor the workflow process instances across heterogeneous workflow engines. The group has come with two important documents 1) a requirements document for the software access workflow protocol and 2) a specification for the same , both of which are Internet Drafts. The key idea is to allow workflow systems to interoperate across heterogeneous settings and to that extent the use of the Extensible Markup language (XML) [11] and other such features have been advocated.

A paper published by two of the group members (Greg Bolcer, Gail Kaiser)[46] introduces the concept of SWAP and its objectives.

The Endeavors Project

This project [10][17][41][9] is conducted by the Irvine Research Unit in Software at the University of California at Irvine. **Endeavors** is an open, distributed, extensible process execution environment. It is designed to improve coordination and managerial control of development teams by allowing flexible definition, modeling, and execution of typical workflow applications. Endeavors is a complete workflow management system which can utilize the WWW and has an architecture based upon objects. Endeavors provides an open, extensible process support infrastructure. The system uses a layered object model to provide for the object-oriented definition and specification of process artifacts, activities, and resources.

Interaction between system components is event driven. Behavior of process objects is specified through the use of handlers: code invoked by the object in response to events received. Stored locally or loaded from a remote source, handlers are bound to objects at runtime and may be changed dynamically in the course of process execution. Activity networks associate activities by control-flow, data-flow, and resource-flow relationships. A rich user interface provides for their dynamic specification and control of their execution. Networks are executed by interpreters that traverse a network and send appropriate events to objects to invoke the objects' behaviors.

Oz and OzWeb

The **Oz** and **OzWeb** [34][22] projects at Programming Systems Laboratory (Columbia University) involved building a workflow management system which was rule-based and used thew World Wide Web as a substrate. This system is a multi-user collaborative workflow system and uses the concepts of rule based "worklets". It involves the modeling of the workflow activities and tasks as objects. Actions taken by the workflow engine are called "rules" which act upon these objects.

WebWork : Meteor's Web-Based Workflow Management System

WebWork[37][38] is part of the Large Scale Distributed Information Systems Lab (LSDIS) [21] research effort at the University of Georgia (Athens) and is meant to be a workflow management system which solely uses web technologies. It consists of a complete Workflow Application

Development Environment[37][38] where a developer can build a workflow application with the tools provided and upon completing the build process one gets web based application with the complete CGI scripts and HTML pages. The CGI scripts control the flow of actions and the HTML pages provide the user interface.

2.2.2. Workflow and Healthcare :- The Healthcare Commercial Agencies

Some commercial organizations have formulated solutions specifically tailored for the healthcare industry and are as follows.

Healtheon Platform

The Healtheon Platform [12] is a multi-tier, CORBA-based distributed application framework that allows reliable, simultaneous access by large numbers of users. It provides an open-architecture, object-oriented , middleware environment that allows the easy development of applications and integration of disparate applications and data, based on industry-standard interfaces. Within this framework, Healtheon has developed a rich set of tools and services which provide integrated security, workflow management, a rules engine, fault tolerance, scalability, manageability and data exchange. Also internet security using the latest protocols has been incorporated.

Healthflow –The Araxsys Solution

Created by Araxsys industries, Healthflow [23] is a workflow management system designed for the healthcare industry which uses web based technology to design, deploy and execute workflow applications. It allows the user to automate processes, manage tasks, monitor patient records and report and analyze.

2.3 Comparison of this effort and other work done in this field

Some of the similarities and differences between this research effort and some of work mentioned in this chapter are as follows:

1. Objective of this research project

This research project focuses on the use of web based technologies to build and manage a workflow application. Unlike the others it does not involve the building of a workflow management system or a system which can design, deploy and control a workflow application. However due to the use of certain server side technologies (such as Active Server Pages) in the building of this application it can be demonstrated that actually constructing the web pages for a workflow applications is relatively easy and does not require extensive theoretical models or the need to implement the workflow logic with Object Oriented languages. With tools such as Visual Interdev [47] one can easily integrate ASP, HTML, Scripts, Database Access easily to build fairly complex and robust workflow applications from the designs available. The use of scripting and ASP provides an easy way of implementing the workflow application which is lightweight (ASP pages are text pages only) and does not involves running make files or a build process.

2. Internet Security

Security is an all important aspect and most of the earlier work done in this area do not implement this feature or it has been implemented using simple techniques .This effort investigates some of the features of current web based protocols which enable one to use workflow on the web.

3. Mobility of the user.

This is made possible by the use of Smart Card technology and ActiveX Components. This feature is unique because the user does not have to use any one particular machine to access the application. No machine specific features need be used.

4. No server side extensions

Some of the work mentioned above (Endeavors) needed to use server side extensions which allowed the HTTP servers implement the PUT operation, but the server chosen for this project does not need such customizations.

4. Asynchronous communication

Asynchronous communication between parties involved has been implemented with the help of messaging systems like the Exchange Server. On-line interaction provides Synchronous communication. What this application lacks is a provision whereby a process can be remotely started and then be controlled or monitored. (Active X Components can be implemented to do some of this work remotely)

Chapter 3

Enabling Technologies

This chapter provides a brief overview of some of the core technologies used in this project and how they are related to each other. This chapter also illustrates how the concepts introduced in chapter 2 have been implemented in today's industry standards, products and solutions. These technologies have been used to demonstrate the manner in which the objectives of this project have been met. This includes Active Server Pages, Secure Sockets Layer, ActiveX components, DHTML and CSS, Smartcard and the GINA along with others.

3.1 Windows NT server

The foundation for most of the inter-related technologies used in this project is the Operating System itself which is Microsoft's Windows NT and the architecture known as the Active Platform. Windows NT is a single robust and high performance network operating system which can act as a file server, print server, application server, communication server, database server and a Workstation or Client. Windows NT security is developed according to the C2 security standard required by the U.S. Department of Defense's security evaluation criteria. In this project this server forms the hub or basis of all applications being run. The client too is a Windows NT workstation.

3.2 Microsoft's™ Active Platform

The Active Platform is the architecture that addresses seamless integration along with the dynamic delivery of applications and rich content. It encompasses both the Active Client and the Active Server. Its complete support for ActiveX and Internet standards enables developers to integrate HTML, Scripting, Components and transactions to build powerful, scalable and highly available applications that run across the enterprise.

3.2.1 Active Server

The foundation of the Active Server is the Windows NT server along with the Internet Information Server (IIS), Active Server Pages (ASP) and the BackOffice, security network and data components which form suite of servers and products.

ActiveX is the “software glue” that integrates these different items. This provides a set of tools to create distributed applications where different parts of a solution can reside on different machines. The services are structured into individual Components which make it easy to organize, maintain and enhance the system as a whole.

The use of the DCOM or Distributed Component Object Model standards allows the components to be deployed over multiple clients and servers.

3.2.2 Active Client

The Active Client is Microsoft’s operating system-independent client support for HTML and ActiveX technologies and is a set of components that are included with the Internet Explorer.

3.3 ActiveX

ActiveX (or 3rd generation Object Linking and Embedding OLE technology) is a framework that allows software components to co-operate even if they have been written by different vendors, at different times using different tools and different languages, and if the objects are located in the same process, same machine or distributed over multiple machines. Put simply, ActiveX provides the **software plumbing** between software objects and insulates the component developer from such complexities.

ActiveX actually encompasses a number of technologies and is not just one thing. Each ActiveX technology defines the interfaces between the objects to implement the particular technology. For the Internet, examples include:

- **ActiveX Documents** – enables the browser to support non-HTML documents
- **ActiveX Scripting** – enables script logic, included with a downloaded Web page or a server-side ASP page, to be executed
- **ActiveX Controls** – provides a method of packaging client components for reuse across platforms and development environments which can then be dynamically downloaded as needed and used within a web page
- **ActiveX Server Components** – enables the Web Server (IIS and ASP) to interface to server software components .Some examples are the ActiveX Data Objects (ADO) and the Ad Rotator Component.

To most users, ActiveX is transparent and they just see the effects of these technologies . Underneath ActiveX is the generic **Component Object Model** (COM) which defines the binary interface between objects. The original specification of COM always allowed for co-operating objects to be located on different machines, but this was not implemented until Windows NT 4.0 and was then called **Distributed COM** (DCOM). Because of this, in reality DCOM and COM are now the same.

3.4 Component Object Model, ActiveX and Software Components

ActiveX is the software which enables us to use components. Underneath ActiveX is the Component Object Model standard which defines the binary interface between objects. DCOM or the Distributed COM allows for co-operating objects to be on different machines . Though the original COM specification allowed for the objects to be on separate machines, it was implemented only with Windows NT 4.0 onwards and was then called DCOM.

3.5 BackOffice

Providing additional value to Windows NT is a number of dynamic products and components which work together to provide a comprehensive Active Server Solution. Most of these server components are grouped under the BackOffice portfolio of products. Some of the products which have been used in this project are Internet Information Server (IIS) 4.0, Active Server Pages (ASP), Exchange Server 5.5, SQL Server 6.5. Some others are Microsoft Transaction Server, Message Queue Server(MSMQ), Index Server and NetShow Server to name a few.
A short description of the products used in this project follows.

3.6 Internet Information Server

Microsoft's IIS 4.0 provides a very high level of integration with the Windows NT system and BackOffice Product suite and is the Web server for this project. It provides a platform for the publishing of information and the delivery of applications over both the Internet and intranet. Apart from providing the functionality of a simple web server which is being able to deliver static HTML files , IIS enables developers to create rich web based applications which can fulfill users demand for services, dynamically generate web pages with the content based on the identity of the user, interface with backend services and databases
and do all this in a secure fashion according to the internet standards such as HTTPS protocol. IIS supports the industry wide standard Common Gateway Interface or CGI but the alternative approach it takes is of ISAPI or Internet Server Application Programming Interface standard. Microsoft have implemented ISAPI extensions as a Dynamic Link Library(DLL) which means it is loaded only once , on first demand and then stays resident in the same process as the IIS Web Server. Some of its features are.

a. Standards supported

IIS supports the industry standard HTTP 1.1 protocol but is backward compatible with its predecessor HTTP 1.0. File transfer and new are supported with FTP and NNTP respectively and one can send e-mail from a web application due to the SMTP support provided.

b. Application Development

IIS can be used to develop sophisticated web applications as it hosts Active Server Pages which together with HTML, and Script Logic(JavaScript , VBScript) enable the generation of dynamic page content. IIS can also call on services from other servers and interface with databases.

c. Crash protection from misbehaving components is provided by running Web applications as separate processes(Process isolation).

d. Transactional Active Server Pages which enables the logic within scripts and components to be bounded such that all database actions are committed together or the database is reverted back to its original state.

e. Security IIS4.0 allows web files, scripts and executables to be made available for secured access to an anonymous user or for restricted access by particular users or groups of users. IIS4.0 is tightly integrated with the NT security subsystem and provides a single infrastructure for managing users accounts and access permissions to system resources.

It also includes Microsoft Certificate Server to provide for the issue and management of X.509 digital certificates. Th industry standard SSL is supported to protect data sent over the network.

All these features such as performance characteristics, user access rights and permissions, security and so on can be turned on or off with the help of the Administrative tool which is the Microsoft management Console which can be used to Administer the IIS.

3.7 Active Server Pages

Microsoft® Active Server Pages (ASP) [31][32][5][54] is a ***server-side scripting*** environment that one can use to create and run dynamic, interactive Web server applications. ASP is an ISAPI extension which builds on top of the ISAPI infrastructure to provide a server-side application framework. With ASP, one can combine HTML pages, script commands, and ActiveX components to create interactive Web pages or powerful Web-based applications. ASP applications are easy to develop and modify.

With ASP, one can easily use ActiveX components to perform complex tasks, such as connecting to a database to store and retrieve information. In ASP pages one can use any scripting language for which there is a scripting engine installed that follows the ActiveX Scripting standard. ASP comes with scripting engines for Microsoft® Visual Basic® Scripting Edition (VBScript) and Microsoft® JScript™ so that one can immediately begin writing scripts. ActiveX Scripting engines for PERL, REXX, and Python are available through third-party developers.

By adding script commands to HTML pages, one can create an HTML interface for an application. By creating ActiveX components, one can encapsulate an application's business logic into reusable modules that can be called from a script, from another component, or from another program.

3.7.1 The Active Server Pages Model

An ASP script begins to run when a browser requests a **.asp** file from a Web server . The Web server then calls ASP, which reads through the requested file from top to bottom, executes any script commands, and sends a Web page to the browser.

The ASP Model for execution is as shown below.

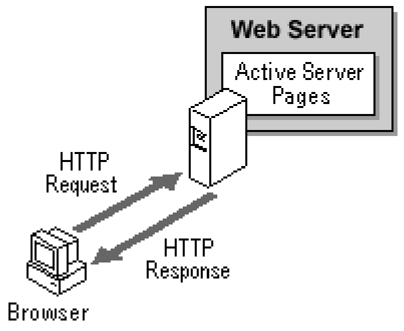


Figure 3.1 Active Server Pages Model

Because the ASP scripts run on the server rather than on the client, the Web server does all the work involved in generating the Web pages that that is sent to browsers. One need not worry whether a browser can process ASP scripts: the Web server does all the script processing, transmitting standard HTML to the browser. Server-side scripts cannot be readily copied because only the result of the script is returned to the browser. End users cannot view the script commands that created the page they are viewing.

3.7.2 Creating Active Server Pages

An Active Server Pages (ASP) file is a text file with the extension **.asp** that contains any combination of the following:

- Text
- HTML tags
- ASP script commands

It’s easy to create an **.asp** file. For any HTML file to which one wants to add scripts, the file must be renamed by replacing the existing .htm or .html file name extension with .asp. To make the .asp file available to Web users, the new file should be saved in a directory on the Web site (making sure that the directory has Script or Execute permission enabled). When one views the file with a browser, one sees that ASP processes and returns an HTML page. One can now add script commands to the .asp file. ASP uses the delimiters <% and %> to enclose script commands.

For Example the following is a valid piece from an ASP page.

```
<HTML>
<BODY>
This page was last refreshed on <%= Now %>.
</BODY>
</HTML>
```

The VBScript function **Now** returns the current date and time. When the Web server processes this page, it replaces `<%= Now %>` with the current date and time and returns the page to the browser: The output seen by the user when viewed with a browser is a shown below

```
This page was last refreshed on 8/1/97 2:20:00 PM.
```

ASP pages are a collection of text, HTML, ActiveX components and built in ASP objects.

Files with Dynamic HTML and Cascading Style Sheets (which involve the use of Client Side Scripting) can be combined with ASP pages to provide rich alternatives which can involve both server and client side processing.

3.7.3 The Active Server Pages Object Model

Active Server Pages provides built-in objects that make it easier for a developer to gather information sent with a browser request, to respond to the browser, and to store information about a particular user, such as user-selected preferences.

A brief description of the object model is a below

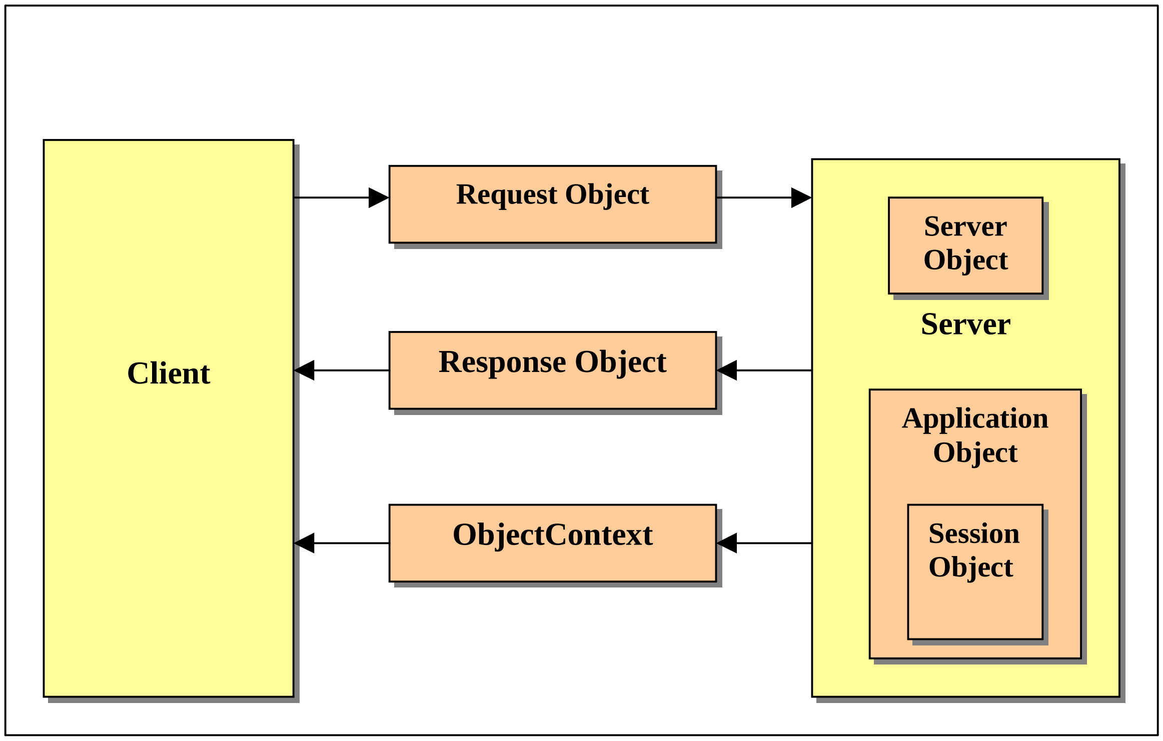


Figure 3.2 ASP Object Model

Application Object

The **Application** object is used to share information among all users of a given application.

Request Object

The **Request** object is used to gain access to any information that is passed with an HTTP request. This includes parameters passed from an HTML form using either the `POST` method or the `GET` method, cookies, and client certificates. The **Request** object also gives one access to binary data sent to the server, such as file uploads.

Response Object

The **Response** object is used to control the information sent to a user. This includes sending information directly to the browser, redirecting the browser to another URL, or setting cookie values.

Server Object

The **Server** object provides access to methods and properties on the server. The most frequently used method is the one that creates an instance of an ActiveX component (**Server.CreateObject**). Other methods apply URL or HTML encoding to strings, map virtual paths to physical paths, and set the timeout period for a script.

Session Object

The **Session** object is used to store information needed for a particular user session. Variables stored in the **Session** object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire time the user is accessing pages in a application. One can also use **Session** methods to explicitly end a session and set the timeout period for an idle session.

ObjectContext Object

The **ObjectContext** object is used to either commit or abort a transaction initiated by an ASP script.

3.7.4 ASP and Components:

ASP has the ability to interface with external COM compliant software components, including those supplied with ASP, those provided by Windows NT and BackOffice products and third party components. For example ASP is provided with ADO or ActiveX Data Objects that provide a high performance interface to databases that are ODBC or OLE DB compliant

We shall look at two such examples used in our project with which ASP works closely.

3.8 ActiveX Data Objects

ASP is provided with the Data Access Component which consists of the ActiveX Data Objects (ADO). ADO is a connection mechanism that provides access to data of all types
especially that stored in s relational database. ADO can interface with relational databases through the Open Database Connectivity (ODBC). Any data source for which an ODBC driver is available can be used.

This includes, SQL server, Oracle, Access and even Excel and text files. ADO is built upon another layer called OLE DB which provides a uniform data interface through the methods and properties it maintains internally.

ODBC is one kind of data that ADO and ASP can access, but not the only kind. ADO exposes an object model which can be manipulated to access a relational database. Some of the Objects are the **Connection** object which is used to set up a database connection between ASP and the database and the **Recordset** object which gives access to data returned from executing a SQL query, a stored procedure, or by opening a table to access the data.

3.9 ASPEMail (Persits Software)

The other server side component used in this project is a freeware component known as ASPEmail (version 4.2) [2] developed by Persits Software. AspEmail 4.2 is an active server component for sending email messages using an external SMTP server in an ASP or VB environment. AspEmail 4.2 supports multiple recipients, multiple CC, multiple Bcc, multiple file attachments, HTML format, embedded images, and non-US ASCII character sets.

3.10 Exchange Server

Microsoft Exchange [28] is an enterprise-level mail server product that is designed to run on Windows NT. It is intended primarily store and route e-mail messages. It can also be used for collaboration features, such as group scheduling, contact sharing, and the hosting of discussion folders. Exchange server has been used as a mail server in this project by using its Internet mail Service (IMS)feature.

3.11 SQL Server

Microsoft SQL Server [30] is the relational database system used for this project. The reason being the relative ease with which it interfaces with the host of other products being used in this project and the fact that it provides high performance for a client server computing environment.

3.12 Active X Scripting –Server Side and Client Side

Scripting is an easy way to make our Web pages come alive and allows the browser to interact with the user and software components (i.e. trap events, invoke methods and access properties). It involves incorporating high level script commands into the HTML document that automatically get invoked by the browser when loaded or by the user clicking on something.

ActiveX provides a flexible architecture for adding any scripting language to an application. ActiveX Scripting allows **script hosts** to invoke scripting services within **script engines**. Further the hosts and engines can be from different software vendors and can implement different languages, since the plumbing between the two is handled by COM/DCOM. The ActiveX Scripting specifications define the interfaces that a script host and script engine must support. The script language, syntax and execution rules are defined by the vendor of the script engine. All script logic is interpreted on the fly by the script engine, there is no concept of compiling the scripts. This is shown as follows:

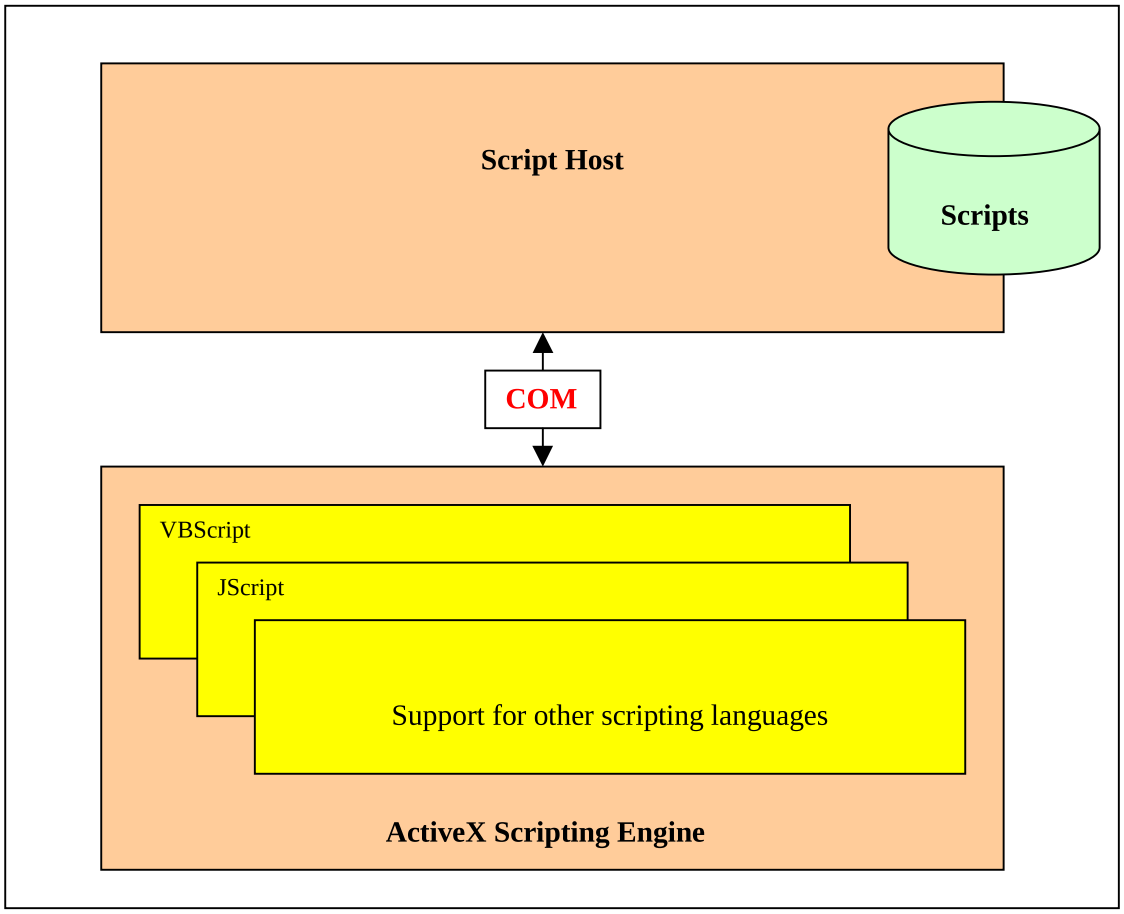


Figure 3.3 ActiveX Scripting

Internet Explorer 5, Active Server Pages and Visual InterDev are examples of script hosts. you **VBScript** and **JScript** are examples of script languages. VBScript is a subset of Visual Basic for Applications (as used by the Microsoft Office Products) which in turn is a subset of the popular Visual Basic programming language. JScript is Microsoft's implementation of the Netscape's JavaScript.

The real power of scripting comes with the ability to interact with other objects. This enables accessing an object's properties, invoking methods and detecting events, and of course all this happens using COM/DCOM under the covers. Accessible objects are either:

- Intrinsic (built-in) objects exposed within the script host – often referred to as an **object model**.

- Executable software components: which are packages of reusable code that usually serve a specific function and don't have to be resident on the browser and can therefore be downloaded when needed.

  An ***ActiveX component*** is a file containing code that performs a task or set of tasks. Components perform common tasks so that you do not have to create your own code to perform these tasks. For example, a stock ticker component might display the latest stock quotes on a Web page. Components provide objects that you use in your scripts to perform tasks.

Server Side scripting has been examined already in Section 3.1 which discussed how ASP uses server side scripting which is a form of ActiveX scripting

Client-side scripting can be inserted into an HTML page using the **`<SCRIPT>`** tags pair, and so can be incorporated within minimal impact. To identify the script language, the **`LANGUAGE`** attribute is used, as shown in the following template.

```
<HTML>
<HEAD>
   <TITLE>Document title
   </TITLE>
   <SCRIPT LANGUAGE="VBSCRIPT">
      Client-side VBS scripting logic
   </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

The client-side scripting can interface to all element objects in a web page (e.g. tags, images, text, etc.), browser objects (e.g. windows, frames, history, etc.) and 'talk to' any object that has been included in the page. From this, we can see that scripting is what really enables the *Dynamic* in Dynamic HTML. We can now change any part of the Web page as a user interacts with the page.

3.13 HTML, DHTML, CSS, VBScript and Java Script, JSCript

Chapter 2 discussed the evolution of the world wide web and traced the progression of the WWW from delivering static HTML content to its present dynamic and interactive form.

In this project HTML, DHTML, CSS and Scripts form the backbone of the application in terms of the presentation layer and the a few details regarding the technologies are as follows.

This project has been coded with HTML 4.0 which is the current standardized version by the World Wide Web Consortium(W3C).

Cascading style sheets have been used to enhance the look and feel and provide a user friendly view to the web pages. It also helped make efficient use of the “real estate” provided on a web page in being able to position elements effectively for maximum usability.

The latest stable version is CSS1 by the W3C which has been revised on January 11th 1999.

A mixture of both Scripting Languages VBScript and JScript (which is Microsoft’s version of JavaScript) have been used on this project. The primary reason was to be able to reuse some of the code written previously in other files. Also one could demonstrate that a single page could contain both scripts with functions written in one being called by another language’s modules. All the above technologies along with the Browser Object model comprise to form Dynamic HTML which has been used extensively in this project as will be demonstrated in the Implementations Chapter to achieve the objectives of this thesis.

3.14 Internet Security and the Secure Sockets Layer (SSL)

This section describes some of the concepts of Internet security and the Secure Sockets Layer [14][15] which have been used in this project.

Some of the fundamental concerns as far as Internet Security is concerned are as follows.

Users sending information over the internet would like to make sure that this information is ***not*** subject to

Eavesdropping : Information remains intact, but its privacy is compromised. For example, someone could learn about a credit card number being sent, record a sensitive conversation, or intercept classified information.

Tampering : Information in transit is changed or replaced and then sent on to the recipient. For example, someone could alter an order for goods or change a person's resume.

Impersonation. Information passes to a person who poses as the intended recipient. Impersonation can take two forms namely **Spoofing.** A person can pretend to be someone else. The other is **Misrepresentation.** A person or organization can misrepresent itself.

3.14.1 Public-Key Cryptography

To address the threats listed above and take precautions against them a set of well established standards called Public-Key Cryptography have evolved

Public-key cryptography facilitates the following tasks:

Encryption and decryption allow two communicating parties to disguise information they send to each other. Th e sender encrypts, or scrambles, information before sending it. The receiver decrypts, or unscrambles, the information after receiving it. While in transit, the encrypted information is unintelligible to an intruder.

Tamper detection allows the recipient of information to verify that it has not been modified in transit. Any attempt to modify data or substitute a false message for a legitimate one will be detected.

Authentication allows the recipient of information to determine its origin--that is, to confirm the sender's identity.

Nonrepudiation prevents the sender of information from claiming at a later date that the information was never sent.

3.14.2 Encryption and Decryption

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. **Decryption** is the process of transforming encrypted information so that it is intelligible again. A *cryptographic algorithm***,** also called a **cipher**, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one
for encryption and the other for decryption.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a **key** that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is computationally expensive, and in some cases impossible for all practical purposes.

3.14.3 Symmetric Key Encryption

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in the figure below

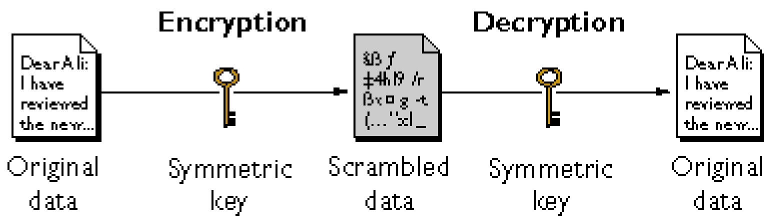


Figure 3.4 Symmetric Key Encryption

3.14.4 Public Key Encryption

Public-key encryption (also called **asymmetric encryption**) involves a pair of keys--a public key and a private key--associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with your private key. The Figure below shows a simplified view of the way public-key encryption works.

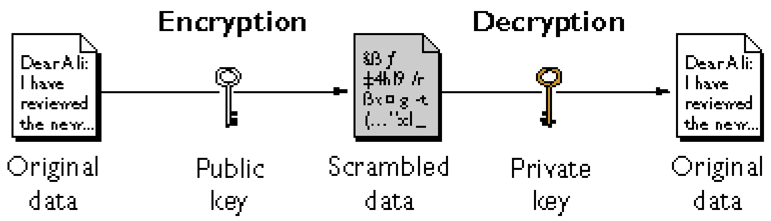


Figure 3.5 Public-Key Encryption

In general, to send encrypted data to someone, you encrypt the data with that person's public key, and the person receiving the encrypted data decrypts it with the corresponding private key. The reverse of the scheme shown in Figure above also works: data encrypted with your private key can be decrypted only with your public key.

3.14.5 Digital Signatures

Tamper detection and related authentication techniques rely on a mathematical function called a **one-way hash** (also called a **message digest**). A one-way hash is a number of fixed length with the following characteristics: (1)The value of the hash is unique for the hashed data. Any change in the data, even deleting or altering a single character, results in a different value. (2)The content of the hashed data cannot, for all practical purposes, be deduced from the hash--which is why it is called "one-way."

Using your private key for encryption and your public key for decryption is a crucial part of digitally signing any data. Instead of encrypting the data itself, the signing software creates a one-way hash of the data, then uses ones private key to encrypt the hash. The encrypted hash, along with other information, such as the hashing algorithm, is known as a digital signature.

The Figure below shows a simplified view of the way a digital signature can be used to validate the integrity of signed data.

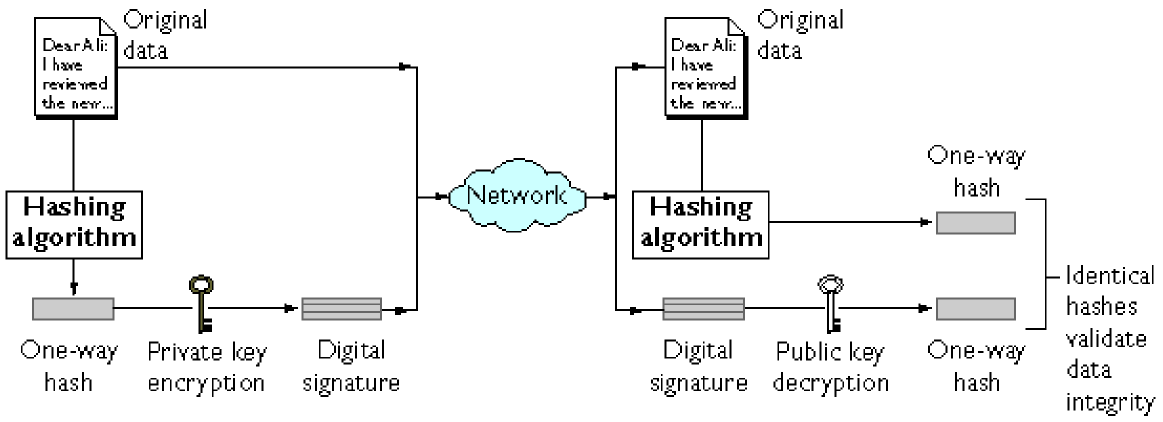


Figure 3.6 Digital Signatures

Figure above shows two items transferred to the recipient of some signed data: the original data and the digital signature. To validate the integrity of the data, the receiving software first uses the signer's public key to decrypt the hash. It then uses the same hashing algorithm

that generated the original hash to generate a new one-way hash of the same data. (Information about the hashing algorithm used is sent with the digital signature, although this isn't shown in the figure.) Finally, the receiving software compares the new hash against the original hash. If the two hashes match, the data has not changed since it was signed. If they don't match, the data may have been tampered with since it was signed, or the signature may have been created with a private key that doesn't correspond to the public key presented by the signer.

The significance of a digital signature is comparable to the significance of a handwritten signature. Once you have signed some data, it is difficult to deny doing so later--assuming that the private key has not been compromised or out of the owner's control. This quality of digital signatures provides a high degree of **nonrepudiation**--that is, digital signatures make it difficult for the signer to deny having signed the data. In some situations, a digital signature may be as legally binding as a handwritten signature.

3.14.6 Certificates

A certificate is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key. Certificates work much the same way as any of these familiar forms of identification such as passports and drivers licenses. **Certificate authorities (CAs)** are entities that validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate-issuing server software (such as Microsoft Certificate Server). The certificate issued by the CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee or a server). Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possessed by the entity identified by the certificate. They also contain the digital signature of the issuing CA along with other information such as expiration date and so on which help validate the certificate and its owner. The CA's digital signature allows the certificate to function as a "letter of introduction" for users who know and trust the CA but don't know the entity identified by the certificate.

3.14.7 Authentication

Authentication is the process of confirming an identity. In the context of network interactions, authentication involves the confident identification of one party by another party. Authentication over networks can take many forms. Certificates are one way of supporting authentication.

Client authentication refers to the confident identification of a client by a server (that is, identification of the person assumed to be using the client software). **Server authentication**

refers to the confident identification of a server by a client (that is, identification of the organization assumed to be responsible for the server at a particular network address).

3.14.8 Certificate-Based Authentication

Client authentication based on certificates is part of the SSL protocol. The client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. The server uses techniques of public-key cryptography to validate the signature and confirm the validity of the certificate. The next section explains the SSL process.

3.14.9 SSL protocol

The SSL protocol [39] runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

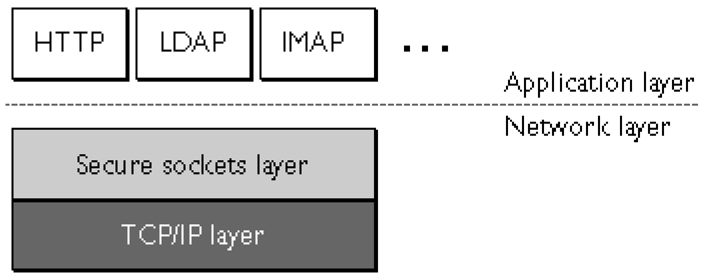


Figure 3.7 Secure Sockets Layer Protocol

The SSL protocol includes two sub-protocols: the **SSL record protocol** and the **SSL handshake protocol**. The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions:

1. Authenticate the server to the client.
2. Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
3. Optionally authenticate the client to the server.
4. Use public-key encryption techniques to generate shared secrets.
5. Establish an encrypted SSL connection.

An encrypted SSL connection requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. SSL connection is protected with a mechanism for detecting tampering--that is, for automatically determining whether the data has been altered in transit.

The SSL Handshake

The SSL protocol uses a combination of public-key and symmetric key encryption. An SSL session always begins with an exchange of messages called the SSL handshake. The handshake allows the server to authenticate itself to the client using public-key techniques, then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows.Optionally, the handshake also allows the client to authenticate itself to the server.

The process is briefly described below:

1.The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.

2.The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL.

The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client’s certificate.

3.The client uses some of the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successful authenticated, the client goes on to Step 4.

4.Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the premaster secret for the session, encrypts it with the server’s public key (obtained from the server’s certificate, sent in Step 2), and sends the encrypted premaster secret to the server.

5.If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client’s own certificate to the server along with the encrypted premaster secret.

6.If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the master secret.

7.Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity--that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.

8.The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.

9.The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.

10.The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

It’s important to note that both client and server authentication involve encrypting some piece of data with one key of a public-private key pair and decrypting it with the other key:

In the case of server authentication, the client encrypts the premaster secret with the server’s public key. Only the corresponding private key can correctly decrypt the secret, so the client has some assurance that the identity associated with the public key is in fact the server with which the client is connected. Otherwise, the server cannot decrypt the premaster secret and cannot generate the symmetric keys required for the session, and the session will be terminated.

In the case of client authentication, the client encrypts some random data with the client’s private key--that is, it creates a digital signature. The public key in the client’s certificate can correctly validate the digital signature only if the corresponding private key was used. Otherwise, the server cannot validate the digital signature and the session is terminated.

The following figures show the steps involved in Server and Client Authentication.

Server Authentication

As explained in Step 2 of The SSL Handshake, the server sends the client a certificate to authenticate itself. The client uses the certificate in Step 3 to authenticate the identity the certificate claims to represent.

To authenticate the binding between a public key and the server identified by the certificate that contains the public key, an SSL-enabled client must receive a ***"yes"*** answer to the four questions shown in Figure below.

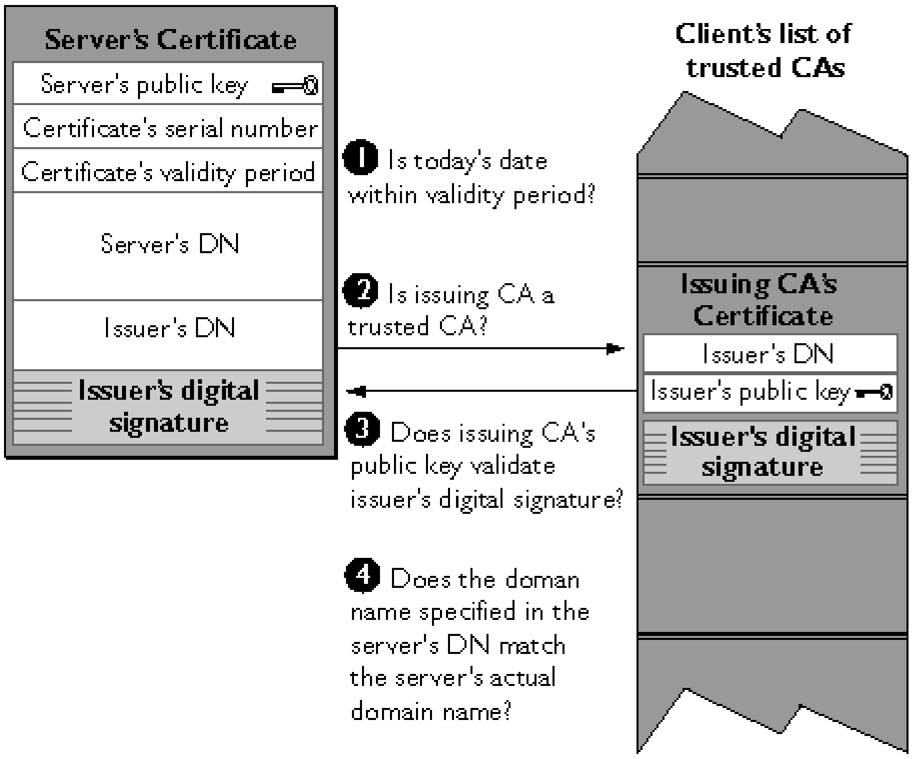
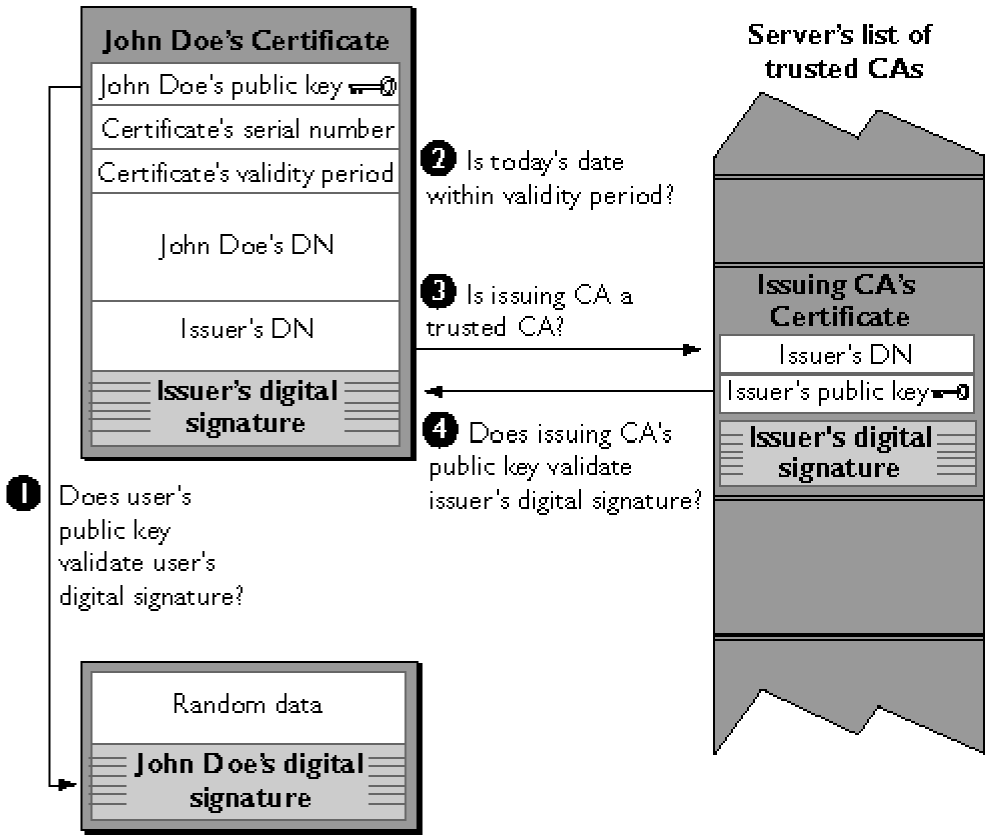

Server's Certificate
Server's public key
Certificate's serial number
Certificate's validity period
Server's DN
Issuer's DN
Issuer's digital signature
Client's list of trusted CAs
1 Is today's date within validity period?
2 Is issuing CA a trusted CA?
3 Does issuing CA's public key validate issuer's digital signature?
4 Does the doman name specified in the server's DN match the server's actual domain name?
Issuing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature

Figure 3.8 Server Authentication

Client Authentication

SSL-enabled servers can be configured to require client authentication, or cryptographic validation by the server of the client's identity. When a server configured this way requests client authentication (see Step 6 of The SSL Handshake), the client sends the server both a certificate and a separate piece of digitally signed data to authenticate itself. The server uses the digitally signed data to validate the public key in the certificate and to authenticate the identity the certificate claims to represent. To authenticate the binding between the public key and the person or other entity identified by the certificate that contains the public key, an SSL-enabled server must receive a *"yes"* answer to the first four questions shown in Figure below.



John Doe's Certificate
John Doe's public key
Certificate's serial number
Certificate's validity period
John Doe's DN
Issuer's DN
Issuer's digital signature
Server's list of trusted CAs
2 Is today's date within validity period?
3 Is issuing CA a trusted CA?
Issuing CA's Certificate
Issuer's DN
Issuer's public key
Issuer's digital signature
4 Does issuing CA's public key validate issuer's digital signature?
1 Does user's public key validate user's digital signature?
Random data
John Doe's digital signature

Figure 3.9 Client Authentication

3.15 SmartCard Technology

Schlumberger's Multiflex card and Reflex 72 card readers were chosen for development of this system. Multiflex cards provide 8K of EEPROM for storing application data on the card. The original reader for this project was SCR60. However Reflex 72 was preferred as it was PC/SC compliant, which meant that any application which was written using Microsoft's Smartcard API could access this Smartcard reader . In addition the Smartcard modules could access any reader which was PC/ SC compliant. This was done to aid interoperability over a whole class of reader drivers (PC/ SC reader drivers) .

Chapter 4

Architecture and System Design

The design of this application has been done keeping in mind the objectives listed in Chapter 1. Namely that the internet and more specifically the world wide web be used as a substrate to provide a secure workflow environment.

4.1 Design Requirements

As we want to deal with the internet it implies that the application will be designed as a Client-Server application where a web server will host web pages and these web pages can be accessed by client browser applications. There needs to be some sort of processing engine which processes the web pages and also monitors the interaction between the client and the server. There also needs to be means for asynchronous notification using say email and this implies the need of some such notification service. To maintain the state of every transaction and to provide the correct context when needed there has to be some sort of facility which enables persistence of information. We also desire that the user be mobile i.e. in no way should the user be tied to a machine or a domain and must be able to access the workflow application in a secure fashion using whichever machine available.

Over and above all this one must not forget the need to incorporate a secure means where the user is authenticated and is allowed access to only information meant for him or her.

In other words the following are the **Design Requirements** in a broad fashion.

1. The world wide web as a substrate for workflow
2. Facility on the server side to process user actions and to take decisions and automate the workflow , in other words the need for a “Workflow Engine”
3. Avoidance of error on the user’s part by enhancing the degree of automation for the user.

4. Ability to communicate both synchronously and asynchronously between the various possible parties involved.
5. All transactions should be done in a secure fashion.
6. Mobility of the user and ability to securely access the application.

The above design requirements provide the rational for designing the application and implementing it in the way it has been done. This chapter provides the overall system design and points to the technologies used to implement it. However it must be noted that the implementation and means used are not tied with the design and in fact other implementation could be used to demonstrate the same proof of concept.

Let us look at the various components of the system architecture in lieu of the requirements.

4.1.1 World Wide Web as a Workflow Application Substrate

A web server will provide the service of hosting the various web pages which constitute the application or the workflow environment. Web browsers which will act as Clients to this application will contact a server and request pages as per a Workflow sequence determined by the server side logic. All this occurs through the application level protocol which is Hyper Text Transfer Protocol or HTTP[43]. HTTP protocol deals completely with the whole process of the request and response of information over the world wide web and is quite complex underneath . The server side logic is what determines which page is shown to which user at what time during the whole process and far as the workflow application is considered, the user can take actions based upon what is displayed to the user on the web pages. The server also processes the users replies, takes actions and displays the results or asks for further input in terms of actions from the client. Hence there is an interaction between the Web Server and the Client Browser(s) forming the basis of the application.

In this application the Web Server used is Microsoft's Internet Information Server 4.0 which hosts the web pages for this application. The client can be any browser for example Microsoft's Internet Explorer or Netscape Navigator.

4.1.2 Workflow Engine

The hub or crux of this application is the workflow engine or the main controller which decides how the whole business process unfolds. In our case this means that the web server needs a way to not only just supply the web pages but also must have some way of processing the user's actions and be able to take decisions as to which applications to call and finally display the results of the background processing back to the user. In other words there needs to be some sort of server side processing which works hand in hand with the web server. This server side processing engine which processes scripts forms the workflow engine. It must be noted that the fundamental advantage of designing a workflow application with a server-side scripting based approach is the fact that it provides very easy extensibility to the workflow logic. The script written in the web page requested could be as simple as a "case statement" which calls different applications on the server side depending upon the users actions or choice. Hence the scripting engine processes the server side scripts along with information obtained from the user. In fact the processing logic is in the scripts themselves.

Active Server Pages provide the Server side scripting environment and the workflow engine.

This workflow engine can be easily extended by modifying the scripts to include new cases, scenarios and functionality. The Active Server Pages Environment provides a set of objects which can be manipulated with scripting languages such as VBScript and JavaScript, and thus allows for easy customization.

4.1.3 Asynchronous and Synchronous means of Communication

Synchronous communication between the server and client user is automatically provided by the whole act of requesting a web page, the web page being sent as a response by the web server and the user performing certain actions and sending information back to the web server. This information is then processed by the server and a response is then sent back to the user.

Thus any transaction which requires the user to respond immediately while the user is still on-line can be termed as a synchronous transaction and this is easily facilitated by the internet.

On the other hand if the server side wants to notify the user of something and the user is not connected to the server then this communication will have to be Asynchronous. Now any form of communication such as regular postal service, paging, courier service, fax etc can be used. However Email is the obvious choice for its convenience and reliability.

Email can be sent by the server side script as the logic demands by invoking certain server side Active Components which can make use of the underlying mail service.

This implies that first of all there needs to be an active component which can provide an interface between the scripts in a web page and the native mailing service which can be the Simple Mail Transfer Protocol(SMTP)[45] . This also implies that there is a need for a SMTP service which can be invoked by the active component. This service may be on the same operating system or on other systems but must be accessible by the web server which hosts the scripts that require a mailing service. It is here that a mail server is used which can provide the service. In this work Microsoft’s Exchange Server 5.5 has been used to provide its SMTP service through its Internet Mail Service feature.

As regards the active component a third –party freeware solution was used which could be easily included into the scripting environment and which provided a very easy programmatic interface to access the SMTP service.

4.1.4 Security

All transactions in this application must take place in a secure fashion which imply that the following should be true.

a) Sender of information must be sure that the information sent is to a trusted source.

b) Information received must be authenticated i.e. it must come form a trusted source.

c) Information should not fall into the wrong hands during transfer. (No eavesdropping)

d) Information in transit must not be tampered with.

All this can be accomplished by using the Secure version of the HTTP protocol or HTTPS which creates a Secure Sockets Layer session between the sender and the receiver and ensures the above. Microsoft's Internet Information Server has an option to turn on SSL which ensures that a SSL session is created between the server and the Client browser. The browser's such as Internet Explorer and Netscape Navigator are SSL compatible.

4.1.5 Persistence and Storage of Transaction Context

To enable the system to store the information being transferred, particularly that which is relevant to the transaction and which helps to maintain the context of transactions we need to use a repository of some sort. A database has been used to store the relevant information sent by the client to the server. This information may be kept as part of the records on the server side. This information provides the context of the message within the notifications such as email messages . This helps the client receiving the message to understand the nature of the message and the context of the message.

Microsoft SQL server 6.5 is used to provide the Database support. The web application can access the database through the Active Data Objects Interface. Hence all the usual functionality that one may associate with a database management system can be utilized through the web application.

4.1.6 Mobility of the user through the use of Point of care machine and SmartCards.

For a user to be mobile he or she should not be tied down to any one particular machine or domain. The user must be able to “roam” to any machine with the suitable client and be able to access the workflow server.

To enable this, a machine is needed which acts as just a gateway to the internet and WWW and at the same time ensures that the user can be properly authenticated to the server.

As SSL and Client Authentication Certificates have been used in this project there was a need to somehow ensure that this authentication information (certificate) be part of the user and not the machine to enable the user’s mobility.

Smartcard Technology provided a very viable solution to this requirement. A smartcard can store a certificate and hence it can store the certificate issued to the user by the server side rather storing the certificate on the machine.

However there still is a need to be able to transfer this certificate from the card onto the machine so that the client browser could use this in the SSL handshake process.

Also an additional requirement was to enable a machine to able to accept a card holder’s name and Personal Identification Number (PIN) which would allow a card reading application previously installed on the machine to unlock the card and get to the stored information. This would ensure that even if the card was stolen the user could not access the information on the card without knowing the PIN of the owner of the card.

Both these requirements were met with the help of the GINA (graphical identification and authentication).

This module was developed at CERC in order to provide for secure logon mechanism on to a computer using a smartcard. When a user boots up a computer running Windows NT, winlogon.exe is executed. This DLL takes care of the logon process. In turn winlogon.exe calls GINA to perform various duties. WinNT comes with the default MSGina.dll . It can be replaced by any other DLL which wants to customize the logon process. The GINA allowes the user to securely logon by entering the PIN after inserting the smartcard into the card reader. It authenticates the user and performes the task of transferring the user's certificate from the card to the machine.

The complex details of this process are beyond the scope of this work and formed the application developed at CERC [33].

An additional advantage of this module is that it allows the machine to belong to some domain and it does not replace the normal functioning of the machine in terms of user logon. Different users with domain accounts can login as usual since the logon screen appears to be the same as usual provided by Windows NT. A regular logon proceeds as if nothing has changed. This implies that a non card holder user can proceed with the secure logon (providing he has a domain account) and use system. Mutsuddi [33] lists the various advantages of this mode of access and discusses the detailed workings of the same.

4.1.7 Enhancing the degree of Automation.

As the fundamental requirement of this project is to provide a smooth workflow to the whole process and avoid the user having to manually interfere with the process unless absolutely essential, it was important to incorporate this in the design. One of the key ways is to avoid the user having to type in information which could anyhow be obtained from some repository.

This meant we could avoid the waste in time, effort and the errors that came with looking at some file and copying the information (manually) into the web pages.

Thus the workflow application has to be capable of extracting the information from its original source. It is here that the notion of Smart Card Technology helps our task.

The smartcard can contain information which needs to be supplied to the workflow application.

This information can be read by an application which can interface with the smart card and then fill the same into the web page. However in keeping with the aims of this project it is desirable to somehow automate this process. This can be achieved by the Active Component technology which provides an interface to the smart card reader. It also provides us with a way to fill the extracted information into the web page fields.

It is important to note that the web server can now interact directly with a client side application or interface to an information repository through the Active Component Interface.

The component in our effort is a ActiveX control which provides an interface to the smart card reader. This control is downloaded from the site which hosting the web pages and it reads information from the smart card and populates the fields on the web page.

Thus it is to be noted that there can be multiple smartcards serving different purposes.

One can be used to hold information only for one user and the other can be to hold information in addition to security certificates and keys. In fact this project makes use of two cards as will be described in the usage scenario.

4.1.8 Client Side Requirements

On the Client side all that is needed is a web browser and an email client. In fact even a web browser can act as an Email client if the recipient has an account on an email server which has a web interface. The web browser must be SSL compatible and by using an email client which can process email with HTML in its body, one can work more smoothly by say just clicking on the links provided in a email message which directly launches a browser.

Microsoft’s Internet Explorer and Outlook Express have been used as the Web browser and email client respectively. Of course the smartcard readers and associated software must be installed also.

Chapter 5 provides the specifications of the software used in this project.

Before we look at how actually the application has been designed and how the various modules fit in, let us examine the usage scenario for this application.

4.2 Usage Scenario

The following specific usage scenario was used to demonstrate the project’s design objectives. It is a workflow scenario which involves the administrative functions in Healthcare provider unit. It can be divided primarily into client and server side to illustrate its components and interactions.

4.2.1 Client Side

The setting is a physician’s office (Healthcare provider) where a patient is examined and after diagnosis and prescribing medication the physician wishes to submit a “Request For Exemption for Criteria” to an agency with which the patient is registered and which also knows the Physician’s organization. Both the physician and the patient are issued smartcards.

These cards can contain a variety of information. But we are interested in a subset of information on these cards.

a) On the physician card which is also called the Health Professional Card (HPC) there will be present the identification information for the card holder (PIN) and the information (certificate) which will allow him or her to access the web server.

b) On the Patient card (PC) there will be patient’s identification information (PIN)and the other demographics and medical details.

There are four main cases possible for the physician to access the Agency's web server based upon the availability of the smartcard for both the physician and the patient. The choice of the authentication mechanism used creates some sub choices as will be discussed in table 4.1

Case 1. There is a point of care machine present with the GINA and a physician has a Health Professional Card(HPC) which holds his identification information and the certificate. The patient has a smartcard with her information on it.

Case 2. The physician has a HPC ; the patient does not have a PC.

Case 3. The patient has a PC ;no HPC for the physician.

Case 4. Neither card is present.

The ***implications*** that the three cases have on the overall scenario are as follows.

a. Security and access to the web server.

This is a must and has to be supported in all the cases. If the HPC is present and the physician has a certificate on the card then this certificate can be used to gain access to the web server. If there is no HPC then one has to use a certificate on a machine which means the user will have to have a windows NT account and will not be mobile. (there could also be a case that the physician has a HPC but no certificate on it [*])

b. Degree of Automation

If the HPC is present then the user can first access the system with it by inserting it into the card reader ad typing his PIN and the physician information can be accessed from it. Later the workflow logic can immediately ask for the patient card which can also be inserted into the card reader(after removing the HPC) . Here the system no longer asks for the patient's PIN as it uses the initial information from the HPC to access the patients card. The details of this process are

- The access to the server will have to be done using a machine which has a certificate it. In other words the physician must have an account on the machine.

part of the workings of the HealthCardUpdate Control created at CERC. This control has been directly used in this application.

If there is no HPC then the patient's PIN will have to be typed in and this means that the patient's PIN will be in the clear for the physician and other personnel (which may not be desirable to the patient form a security point of view).

Also if there is no PC then any information needed form the patient's card will have to be manually filled in which greatly reduces the degree of automation .The implementation for this project demonstrates all these scenarios.

Before we look at all the cases in further detail let us take note of the various tasks that the user on the physician's side must perform irrespective of the cases involved.

1. The user accesses the agency's site by launching his browser and typing the URL (or clicking on a shortcut). Hence a user has been able to initiate the "Workflow Application" using the World Wide Web substrate. Now this client can be remote or he can be on an intranet. But the fact is that the WWW has been used to launch a process and the workflow engine which is on the agency's server has been contacted and is initiated.

2. An SSL session is established between the two hosts based upon the SSL handshake protocol discussed in Chapter 3. The user is asked to present his or her client authentication certificate which is verified. If the user is given access to the site and if the user is visiting the site for the first time then the Active Component which performs the task of extracting the information from the physician's and the patient's smart card is downloaded onto the physician's machine. Along with the ActiveX Component a page containing a form for the Request for Exemption for Criteria is shown to the user. This Active Component asks the user to insert the smart card into the reader. The user (physician or patient depending upon which card is present) has to type in the Personal Identification Number (PIN) and the Active Component verifies that the PIN number typed in matches the one on the card. If not the component will not allow the user to view the information on the card. Thus security is enforced here with the help of Smart Card

technology so that a person who does have someone else's smart card cannot obtain access to that information unless he or she has the correct PIN. If verified correctly the active component then reads information from the card and fills in the form for "Request for Exemption from Criteria". Thus the security aspect of the Requirements is enforced from the word go in this design.

3. Once the Physician completes the form he or she can submit the form. It must be noted that this also tries to minimize the user from having to type in as much information as possible . Hence reflecting the advanced degree of automation which tries to make the Workflow process as smooth as possible. Also before the form is submitted the scripts on the page check to see if the information is valid and try to detect mistakes which can be easily rectified such as invalid data entered by the user and the mistakes in formats such as dates and so on.

   Thus client-side validation (because all the error checking is done before any information is submitted is sent to the server) makes the process faster and is especially useful in situations where bandwidth is at a premium by avoiding the user having to submit the form to the server and then receive a reply just because of few minor errors.

4. The information submitted by user is stored in the Agency database and the user is given an identification number for his transaction using which the user can check the status of his transaction. This is the confirmation of the user's actions .

   This completes one part of the workflow which is the synchronous part since after this the user as well as the patient will no longer be online and connected to the agency's web site.

Since we are now cognizant of the tasks which the user has to perform on the physician's side we can now examine how these tasks can be accomplished based upon the cases mentioned above. The table 4.1 lists the various cases in greater detail and how they affect the initial working of the application

Table 4.1 Cases involved in the Workflow Application due to the availability of HPC and PC

HPC	PC	Location of Certificate	Method of access to application	Transaction Security enforced	Degree of automation
✓	✓	HPC	Point of care machine	Yes. Using certificate on the HPC .	Insert HPC followed by PC. Only type HPC Pin Other information is automatically extracted form the PC.
		Local machine	Windows NT account	Yes. Using certificate on the machine	
✓	✕	HPC	Point of care machine	Yes. Using certificate on the HPC	Manually done no extraction of information from patient card.
		Local machine	Windows NT account	Yes. Using certificate on the machine	
✕	✓	Local machine	Windows NT account	Yes. Using certificate on the machine	Insert PC and type PIN. Patient information extracted from PC.
✕	✕	Local machine	Windows NT account	Yes. Using certificate on the machine	Manually done no extraction of information from patient card.

For example the first case in the above table can be interpreted as follows:

As both the cards are present and if there is a point of care machine then the physician gains access to the agency web server and the certificate from his card is used for client authentication. The opening page of the agency presents the user with choices for the method of filling in the form based upon the card(s) the users have. The choices could be 1) HPC and PC ,2).Only PC 3).Manual and so on. The user can choose the option which says one needs both the HPC and the PC. The corresponding ActiveX Component which has been downloaded now prompts the physician to follow a sequence such as 1) enter HPC then 2) type PIN and logon 3)enter PC and so on which brings the user to the form . This form gets filled in by the information extracted by the ActiveX component form the cards. The rest proceeds as explained in steps 1 to 4 above.

Other cases can be interpreted just as this one. The only difference being in the initial methods of access.

The reason that the other cases have also been demonstrated is that we want to show that if one is not able to achieve total automation due to the availability of client side technology the same applications (with minor modifications) can be accessed securely and the tasks can still be completed.

4.2.2 Server Side

1. The second part begins with the agency's personnel connecting to the agency's web site.

Here again a secure session is established with SSL and the client is authenticated with the help of certificates. The user can access a task list with tasks to be processed and access the request made by the physician's office which will show up as one of the tasks. This task list is filled by information obtained from the database. The request can be processed and a reply is sent automatically to the physician's office via email from the web application itself. This is the example of asynchronous communication since the receiver may not be connected to the agency's server. This notification message contains the context of the transaction in the form of the identification number of the transaction. All the agency personnel's processing results for a particular transaction are stored in the database. The agency personnel can also have the same set up as the physician is concerned namely certificates on smartcards and so on.

2. The physician can access his email and upon clicking on the link(or typing the URL in web browser if the email agent does not support HTML) provided for the particular transaction , he or she is taken to the agency's site where once again an SSL session is established and the user is authenticated with the help of his or her Client Authentication Certificate. The physician can examine the response from the agency and take further action such as providing more information if needed and re-submit the form.

3. The form is once again submitted and the new information (if any) is stored by the system in the repository. This information can be accessed by the agency personnel as part of their tasks. Thus the workflow loop can continue.

4.3 Overall Design and System Architecture

The overall system architecture incorporating all the above mentioned components looks as shown in the figure below. The usage scenario discussed so far can be demonstrated in the design as discussed below.

As shown in the figure below the scenario can be split into the client and server sides.

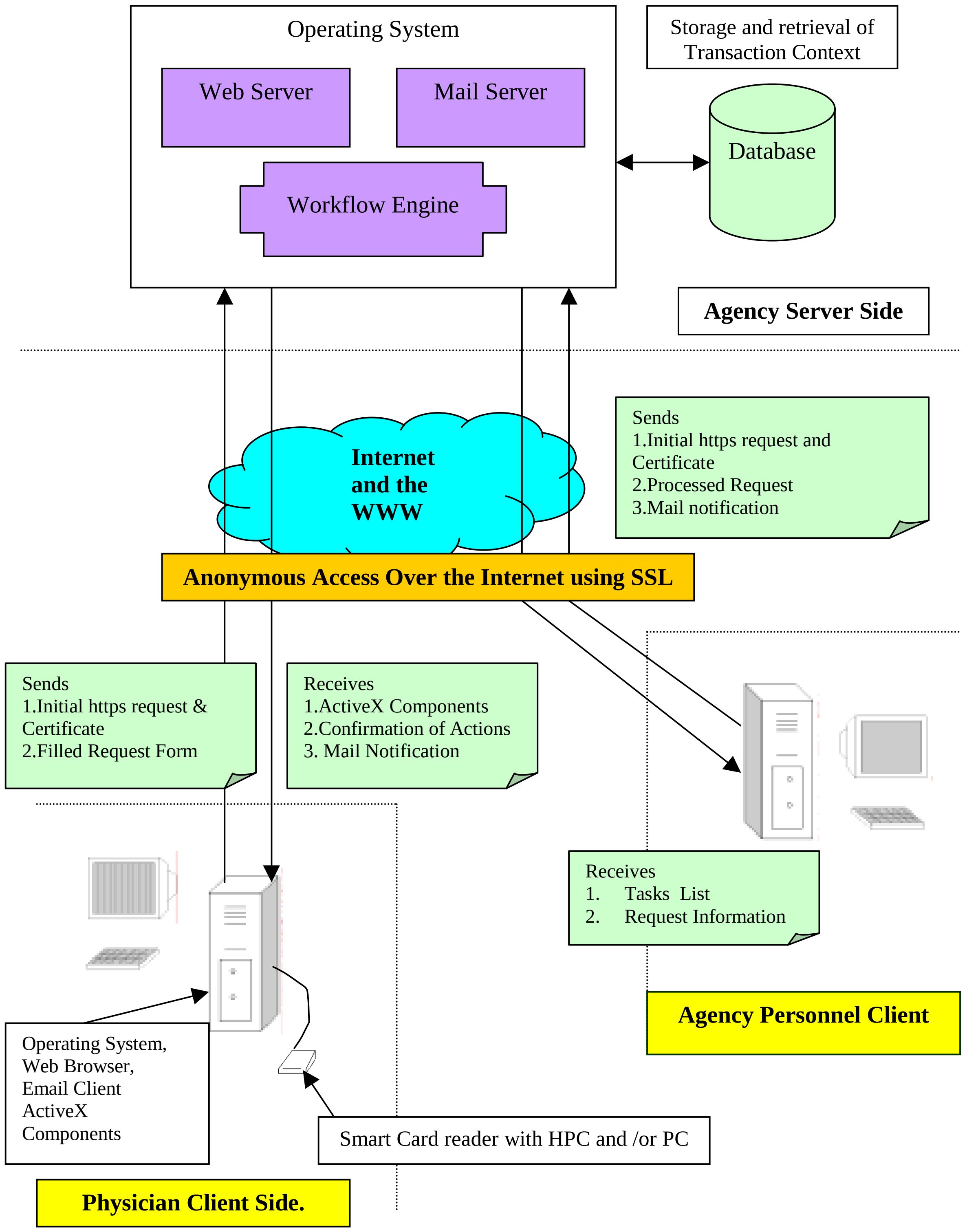


Figure 4.1 Complete Design of the Workflow Application

The major components on the Client side are

1. Client machine :- This can be an ordinary workstation or point of care machine. In either case it must have the correct versions of the operating system, web browser and email client. Also a smartcard reader will be connected to it and the necessary drivers to access the reader must be installed. If the machine is a point of care machine then the GINA module must be installed which also provides the functionality to transfer the certificate from the HPC to the machine's registry. GINA also allows the normal working to proceed as it was.

   This machine may be on a Windows NT domain on the physician's side. Thus it will allow the physician to have an email account on the organization's server. This enables the asynchronous communication.

2. On the Server Side:-

One can see that the server side has the following components

1. The operating system.
2. Web Server .
3. Workflow engine which is in the form of the logic within the web pages hosted by this web server
4. Mail server which performs the duties of a regular mail service for the organization
5. Database system to store the context of transactions.

Also there must be a client machine (workstation) which can be used by the Agency personnel to access the information submitted by the physician , process the request and submit the results of the processing. This machine should have a web browser with which the personnel can access the transactions which are displayed in the personnel's task list for the day..

3.Communication Channels and Security:-

As shown in the figure two forms of communication will be supported

1.Synchronous :- which occurs between a client and server when the former is on-line.

The figure shows the information being sent and its direction

For instance the after the client requests the starting page in the workflow application form the web server ,

1. Information pertaining to the SSL handshake is sent across by both parties.
2. The client sends the Client Authentication Certificate .
3. The server sends the ActiveX Component back based upon the clients choice of the available cases (HPC and PC card availability)
4. The client submits the form which has been filled.
5. Server sends a confirmation of actions.

The above six are some of the main pieces of information being sent. There may be others depending on the actual logic implemented. All this is done in secure fashion using SSL. The Uniform Resource Locator in the case of such transactions begins with a **"https"** as opposed to the standard **"http"**.

for example the URL could be "https://RationalDrugAgency.com"

2. Asynchronous :-

This is in the form of the E-mail message sent to the client form the web server which is done automatically when agency personnel submits her processing to the web server.

Even if this message is intercepted and the person tries to use this URL to get to the site, he will be denied access as he will not have the required client authentication certificates.

The next chapter discusses the implementation details of this project. This includes specifications of the hardware and software used, the files used and the logic in the workflow application, the transactions and schema of the storage system and other details.

Chapter 5

Implementation

This chapter discusses the implementation of this thesis. It includes the following aspects:

1. Rational for the implementation
2. Description of the development and testing facility
3. Description of the infrastructure in terms of hardware on both the client and server sides.
4. Description of all software components used in the project and how they interface with each other.
5. The logic used behind the workflow application and the way it accomplishes the tasks.
6. Description of the files created and the “page flow” or an illustration of how the various software modules call each other and at what point.

5.1 Rational for the implementation

This thesis has evolved along with the Secure Collaborative Telemedicine project which is being carried out the Concurrent Engineering Research Center at West Virginia University. It is one of the major components of the suite of applications created at CERC which include the Web Based Patient Medical records and the SmartCard Applications. CERC has a testbed facility which acts as the development and testing facility for this project. Most of the tools required to develop this project were already used by the CERC development team and the easy availability of these systems combined with the added advantage of leveraging the experience of the CERC team in this venture prompted this implementation scheme.

5.1.1 Description of the Facility

The facility consists of Dell computers which run Windows NT version 4.0 (Service Pack's 4 and 5). The facility consisted of machines which have both the Windows NT Server and Workstation Operating System installed.

5.1.2 Other Systems

As this thesis provides a proof of concept, it is worth noting that other systems (platforms, servers, clients) can also be used .

5.2 Hardware Infrastructure

5.2.1 Server Machine

This section provides a brief overview of the infrastructure in the testbed which was used for this project. The testbed consists of machines which run both the Windows NT operating system Server and Workstation version. The machine which was chosen to run the Windows NT Server is named **`“raccoon”.`** This machine was designated as the Domain Controller for the domain known as **`“demo”.`** This machine has two network cards, one which connects the other client machines and the other which connects this machine to the “**internet or rest of the world”.**

5.2.2 Security policy at the CERC testbed

Due to the security policy of CERC it was decided not to open this domain to the rest of the world, no machine on the net or outside this domain would have access to this domain. However only the server raccoon would be allowed to access the internet for information and research purposes.

5.2.3 Client Machine

The client machine on this domain is a Windows NT workstation which known as **`“taylor”.`** This machine would act as the physician’s office client. It may be noted that it may seem that this is an intranet of sorts and not the internet, (this is due to the security policy of CERC) however the application still runs as if there is remote access to the agency’s web site, i.e. anonymous access is still possible as will be discussed shortly. In fact when the client accesses the web server it does so “anonymously” which is as if it is across the internet.

5.2.4 Agency Personnel’s Client machine

Thus the setting at the testbed facility is that **`raccoon`** is the agency server and **`taylor`** is the client at the physician’s office. In reality the two are on separate domains and have no connection at all as far domain and accounts are concerned.

To simulate the agency’s client the server **`raccoon`** itself is used. So a person can log onto the **`demo`** domain and access his task list. Once again the agency personnel may be at a remote destination. But the access is anonymous and does not make use of the Windows NT account and password so this actually simulates the real scenario.

5.2.5 Smartcard reader connected to Physician’s machine.

The card reader used in this project is the Reflex 72 SmartCard Reader by Schlumberger. This is attached to the machine used by the physician. The corresponding drivers for this reader (from Schlumberger) and the smartcard base components from Microsoft must be installed on the same machine before the card can be used.

The following figure below shows the complete setup and the table summarizes the hardware infrastructure.

(Please Note that the names of the domain and hosts and IP addresses are implementation specific and have no bearing on the actual application i.e. one can use different machines and configure the domain and host names as required. The application is “portable” in that sense. All that is needed is transfer of the files and any system with a similar set up can function as the substrate)

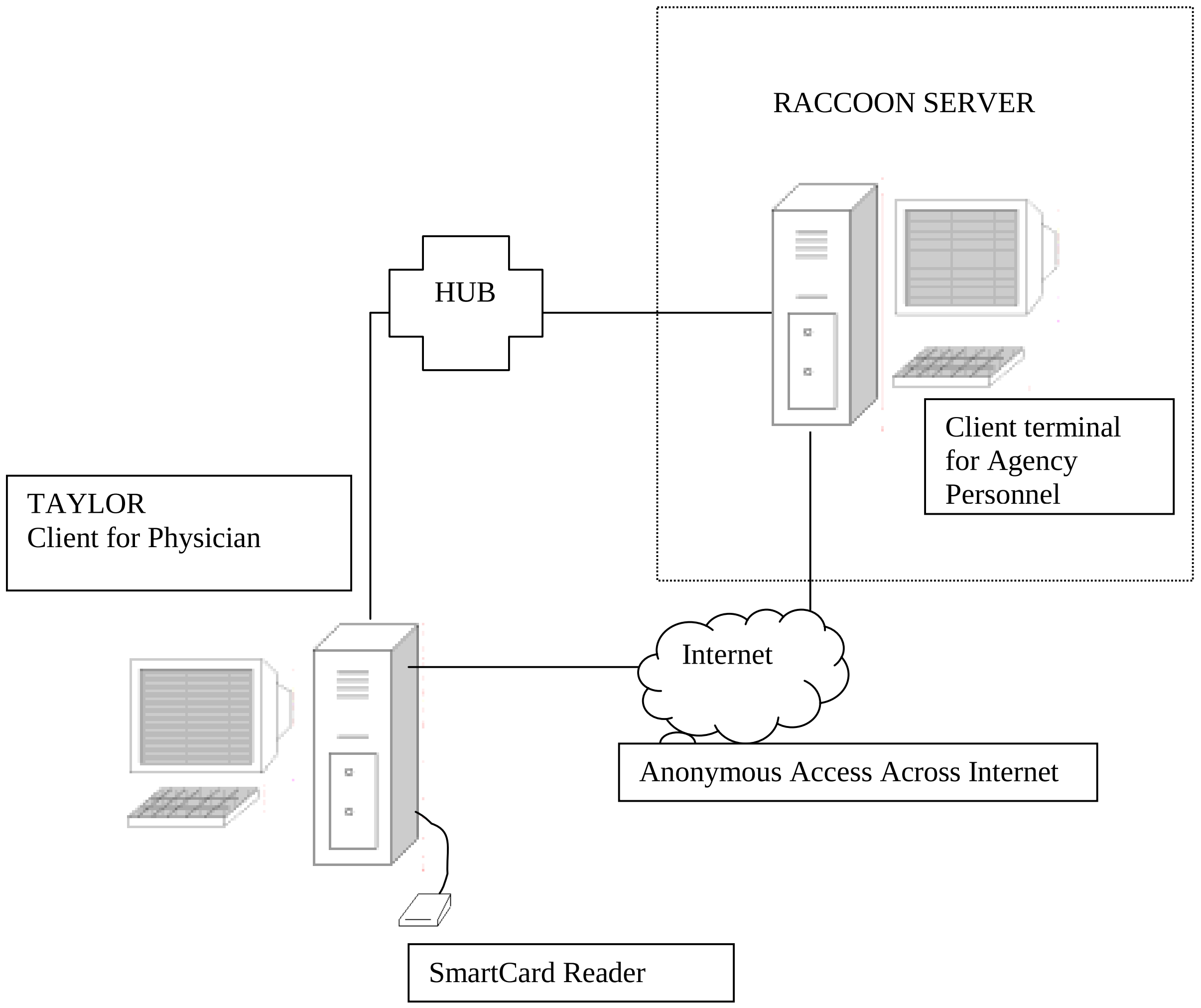

RACCOON SERVER
HUB
Client terminal for Agency Personnel
TAYLOR
Client for Physician
Internet
Anonymous Access Across Internet
SmartCard Reader

Figure 5.1 CERC Testbed Setup, The DEMO Domain

The figure shows the server raccoon which is also the domain controller of the demo domain and the client `taylor` which is connected on the same domain. It is also seen that `taylor` will act as the client at the physician's office and internet access will be simulated with the anonymous access . The client at the agency will be simulated using the server itself but the access mechanism is the same.(across the internet)

Table 5.1 CERC testbed facility description

Type of Network	Windows NT Network	
Domain name	DEMO	
Number of machines	2	
Description of Machines		
Name and IP address	**Function**	**Specifications**
raccoon 192.168.1.2	Domain controller and server	Dell XPS Pentium Pro200n 128 MB Main Memory
taylor 192.168.1.3	Client	Dell XPS Pentium Pro200n 128 MB Main Memory
Other hardware and utilities needed		
Reflex 72 SmartCard reader and Multiflex SmartCard from Schlumberger.(Card reader on client machine)		

5.3 Software Infrastructure

This section describes the software components used in this projects along with details about their versions and how they interface with each other.

It looks at the details of some of their features which were important to this project and the necessary options and settings made to ensure the same. Keeping these settings in mind subsequent sections talk about the interactions possible when the application is used and how these settings play a part in meeting the objectives of this project.

The software components are distributed between the server and client machines as follows::

Table 5.2 Description of Software used in this project

Server/Client Side	**Software name**	**Function**	**Version and other details**
Server Side Machine	Windows NT Server	Operating System and Domain Controller. Hosts other servers and programs in this project.	Version 4.0. Service Pack 5 build 1381
	Microsoft Internet Information Server	Web server and ASP host. Acts as the web server for the Agency site and forms the core of the workflow application processing with ASP	Version 4.0 .Comes with the Windows NT Option pack 4.0
	Microsoft Exchange Server	Mail Server . Its SMTP mail service is used through its Internet mail Service feature.	Version 5.5. Service pack 3
	Microsoft SQL Server	Used as the RDBMS for the project. Stores the transaction infromation.	Version 6.5.

	Active Server pages 2.0 Active Data Objects	Comes with Windows NT option pack 4.0 and IIS. Forms the workflow engines by processing the ASP pages. ADO is used by ASP to connect to the SQL database.	ASP Version 2.0.
	Microsoft Internet Explorer	Used as a browser when the Agency Personnel is using the machine as a client.	Version 5.0

Server/Client Side	Software name	Function	Version and other details
Client Side Machine	Windows NT Workstation	Operating System and workstation client software.	Version 4.0. Build 1381 Service pack 5
	Microsoft Internet Explorer	Used as a browser when the physician is using the machine as a client.	Version 5.0
	Microsoft Outlook Express	Mail Client which supports HTML content	Version 5.0
	Reflex72 SmartCard Driver Microsoft SmartCard Base Components.	---	---

5.4 Software Settings and selected options

We look at the settings and options specific to each of the software pieces used in this project. The next section describes how all these pieces fit together and work in a sample interaction.

5.4.1 Operating System

Windows NT Server Operating System 4.0 [50] (Service Pack 5 build 1381).

This is the basic operating system for the whole application on the server side. As mentioned before the Server version is used on the server side and the workstation version is used on the client side.

5.4.2 Web server

The Microsoft Internet information Server 4.0 [13] acts as the Web Server and hosts the ASP pages which are part of the workflow system. This server can be administered with the help of the MMC or Microsoft Management Console which is a program which lets the user with administrative privileges access the server and set options which reflect the working of the server. The main steps to create to host a web application on this server are as follows.

1. The files constituting the web application (web site files) must be stored on the Inetpub\wwwroot directory which is automatically created in the C: drive (assuming the IIS has been installed on the C drive). For example C:\inetpub\wwwroot\workflowapp
2. A virtual directory must be created and it must be associated to this directory.
3. This now forms the initial directory of the web application. Files associated with this application must be stored within this directory.
4. Security options [40]that must be set for this directory (using the MMC) are as follows.
   a) **Enable Anonymous access:**
   b) **Require Client Certificate:**

The rational for the above choices is as follows:

Normally, all users attempting to establish a WWW (HTTP) connection with the web server log on as anonymous users. When a user establishes an anonymous connection, the web server must log on the user with an anonymous or guest account (that is, a valid Windows NT user account to which you apply restrictions limiting the files and directories that the anonymous user can access).

For preventing anonymous users from connecting to your restricted content, one can configure the web server to *authenticate* users. Authentication involves prompting users for unique user name and password information, which must correspond to a valid Windows NT user account, governed by Windows NT File System (NTFS) file and directory permissions that define the account's level of access.

The IIS web server will authenticate users *only* under the following circumstances:

- Anonymous access is disabled.
- Anonymous access fails because the anonymous user account does not have permission to access a specific Windows NT File System (NTFS) file or resource.

If either of the previous conditions are true, the web server will refuse to establish an anonymous connection and attempt to identify users with the authentication method that has been enabled. IIS supports Basic, Windows NT Challenge/Response, and SSL client certificate authentication. By enabling different combinations of these authentication methods, in addition to setting up the anonymous user account, one can establish varying levels of control in determining which users connect to the Web content.

Basic Authentication

When Basic authentication is enabled, the user's web browser renders a dialog box where users can enter their previously assigned Window NT account user names and passwords. The Web browser then attempts to establish a connection using this information. If the server rejects the information, the web browser repeatedly displays the dialog box (the number of times depends on the web browser’s configuration) until the user enters a valid user name and password, or closes the dialog box. After the web server verifies that the user name and password correspond to valid Windows NT account, the user can establish a connection. Although widely used, this method is not recommended unless one is confident that the connection between the user and your web server is secure since web browsers using Basic authentication transmit user name and password information in an unencrypted form.

Windows NT Challenge/Response Authentication

IIS supports Windows NT Challenge/Response authentication, which authenticates users without requiring the transmission of actual passwords across a network. Currently, Microsoft Internet Explorer, version 2.0 or later, is the only Web browser that supports this authentication method.

When Windows NT Challenge/Response authentication is enabled, the user’s Internet Explorer browser proves its knowledge of the password through a cryptographic exchange with the web server. The actual password never travels over the network and the user is not prompted for account information.

However, if the authentication exchange initially fails to identify the user, Internet Explorer will prompt the user for a Windows NT account user name and password, which it will process using the same Windows NT Challenge/Response method. Internet Explorer will continue to prompt the user until the user enters a valid user name and password, or closes the prompt dialog box.

Client Certificates for Authenticating Anonymous Users

This is the approach followed in this implementation.

With client certificates one can regulate which users are allowed to establish an anonymous connection with the web server. When a user attempts to establish an anonymous connection, the web server can check whether the user submitted a client certificate which is valid and issued by this server or by a source which this server trusts.

Enabling IIS to use SSL (Server Certificates)

To activate the web server's Secure Sockets Layer (SSL) security features [1][51], one must obtain and install a valid *server certificate*. Server certificates are digital identifications containing information about the web server and the organization sponsoring the server's web content. Functioning in the same way as conventional forms of identification, a passport or driver's license, a server certificate enables users to authenticate your server, check the validity of web content, and establish a secure connection.

Certificates are sometimes issued and endorsed by a mutually trusted, third-party organization, called a *certificate authority*. The certificate authority's primary responsibility is confirming the identity of those seeking a certificate, thus ensuring the validity of the identification information contained in the certificate. Alternatively, one can issue one's own server certificates.

The whole process can be done with the Key manager Utility which comes with the IIS . This utility allows one to install server certificates and one can use the Key Manager to create, import, and export Secure Sockets Layer (SSL) encryption key pairs, which enable the server to negotiate a secure link with a user's browser. The details of this process can be found at the above references. The server certificate for the raccoon server was obtained from the Microsoft Certificate Server (which is part of Windows NT Option pack 4.0) which was installed at CERC on the separate server known as FAYETTE which belonged to different domain. The demo domain machines were configured in such a way that this certificate authority was trusted in all its transactions.

Active Server Pages 2.0

ASP pages are hosted and processed by the IIS and the asp pages along with other include files and ordinary.html reside in the virtual directory for the application.

5.4.3 Mail Server

Microsoft's Exchange Server 5.5 [25][26][27] acts as the mail server for the agency site. This Server works on the Windows NT platform and is based upon Microsoft's Active Platform which deals with messaging and collaboration. Exchange server can also be integrated with the applications such as Microsoft Outlook and works well with Email and web browser's such as Internet Explorer which can be its clients.

In this project due to the CERC security policy and fact that the physician's machine is also a client on the same domain as the server is, an email account for the physician has been provided on the exchange server (which takes the name of its host machine – **`raccoon`**). This email id is the same one used on the physician's certificate. This is because the workflow application through ASP's ClientCertificate collection picks up the email property and uses this id to send a notification back to the physician. Thus the physician does not have to type his email id and is assured of being notified at the email id provided on his certificate.

In reality the mail server which holds the physician's mail is different from the agency's mail server and could be on the physician's domain or could be on some remote location or could be even a commercial email service which the physician could access.

Exchange Server creates a mailbox for every user which has a Windows NT account. In other words all users with accounts on a particular Windows NT server get their complete "Inbox", "Outbox", "Sent Mail", folders and so on if there is an Exchange Server Associated with it.

Email Client's such as Outlook Express and Outlook can access the user's mail and Exchange uses the user's Windows NT account username and password information for logging on to the server. The whole mail setup is managed by the Internet Mail Service feature which is a service that exchange runs in the background. It provides various properties and access control mechanisms where by users can be allowed selective access and security policy can be enforced.

5.4.4 Database System

SQL Server 6.5 [29] runs on Windows NT and has been installed on `raccoon` and acts as the relational database management system for this project. Using the SQL Enterprise Manager and ISQL_w utilities one can manage the server and create , work with and delete database. Assign user accounts on the server which can access specific databases and make modifications. These include creating tables, views , stored procedures. Populating the tables with data, deleting the objects and so on.

All this has to be done by one having an administrative account on the Windows NT server.

A user account was created on this database to access a database specifically created for this application. This database was named “`workflow`”.

A single table was created which could hold the information related to the transactions involved in this project. This table was not normalized as this was a proof of concept but normalization and additions to the database could be made as part of future work possible.

The table is called **“`transactionT`”** and its schema is as shown below.

Table 5.3 Database Schema

Field Name	Data Type
taskStatus	varchar(15)
personnel	varchar(50)
requestDate	datetime
certSerialNumber	varchar(50)
phyEmail	varchar(50)
pid	varchar(20)

lastName	varchar(20)
firstName	varchar(20)
middleName	varchar(20)
month	varchar(4)
day	varchar(4)
year	varchar(8)
phyDEA	varchar(20)
phyPhone	varchar(20)
phyFax	varchar(20)
phyLastName	varchar(20)
phyFirstName	varchar(20)
phyMiddleName	varchar(20)
phyAddress	varchar(50)
phyCity	varchar(20)
phyState	varchar(20)
phyZip	varchar(20)
pharNABP	varchar(20)
pharPhone	varchar(20)
pharFax	varchar(20)
pharName	varchar(20)
pharAddress	varchar(50)
pharCity	varchar(20)
pharState	varchar(20)
pharZip	varchar(20)
fromMonth	varchar(4)
fromDay	varchar(4)
fromYear	varchar(8)
toMonth	varchar(4)
toDay	varchar(4)

toYear	varchar(8)
medication	text
dose	text
directions	text
diagnosis	text
transactionStatus	varchar(20)
rational	text

Some of the fields in the table which deserve mention are::

1. **`pid`** :- This is the identification number for a patient.
2. **`requestDate`**:- The date and time of the request made.

Together the above fields form the **unique transaction id** for a transaction.

3. **`certSerialNumber`**:- This is number unique to each certificate. This helps the system to compare the certificate serial number of the client certificate accessing the system for a particular transaction for the second time. If the workflow application wants to ensure that only one physician may be associated with a particular patient's request (transaction) then it can compare the two numbers and only if they match will it allow the client to proceed and view the processing results. This is the current scenario. However a physician may want a colleague to view this transaction and this will not be possible since though both have a valid certificate issued by the same organization the second physician is not authorized to view this particular transaction. Depending upon the security policy of the Agency this comparison may be made or ignored. All that needs to be changed is the ASP script which does the same.
4. **`transactionStatus`**:- This indicates if the request has been either "Approved", "Denied" or is being sent back to the client "Requesting further clarification".
5. **`personnel`** :- Name or identification number of the personnel processing this request.

The other fields correspond to the sample request form used in this project and consist of patient, physician, pharmacy and medication information.

5.4.5 Active X Components

Two ActiveX components have been used in this application. One is known as the **HealthCardUpdate Control** . This is meant to be a control which allows both reading and writing to the smartcard using a web interface. However in this project only its card reading capabilities have been used. This component is downloaded (the first time the client visits the agency web site) and subsequently if it is already present on the machine it will not be downloaded unless a newer version is created. In any case the user will be prompted before the component is downloaded and can check its security certificate and the fact that it is signed. This component is used in the case when both the HPC and the PC are present with the client.

If only the PC is present then the **HealthCardAccess** component is downloaded. This performs the same function of extracting information from the card but it does it, only with the patient card.

The ActiveX components for the application were created by the CERC development team which worked on the Smart Cards.

As these components were written according to ActiveX Scripting specifications and their functionality was exposed in terms of properties and methods, the components could be manipulated with VBScript and JavaScript. The **`object.property`** and **`object.method`** were used to access information in the card where the **`object`** was the component itself. The object is inserted into a .html or .asp page with the help of the <OBJECT></OBJECT> tags

Whenever a page with the <OBJECT> tags loads the component is downloaded and its functionality can be accessed

For example to read the card the syntax using VBScript would be :

```
varResult = SmartCard.readCardData(pin)
```

Here **`SmartCard`** is the object.

`readcardData()` is a method and the `pin` and `varResult` are input and output parameters respectively.

5.4.6 Server side components

A server side ActiveX component has also been used to send email messages from a web page. This is the ASPEmail component developed by the Persits Software [2]. The syntax is very similar to the one used above. On the server side ASP calls the method belonging to its Server Object to create an instance of a component.

For example the syntax (using VBScript) would be

```
<%
            Set Mail = Server.CreateObject("Persits.MailSender")

             Mail.Host = "smtp.smtp-server.com" ’Specify a valid SMTP server
             Mail.From = "sales@veryhotcakes.com" ’Specify sender’s address
             Mail.FromName = "VeryHotCakes Sales" ’Specify sender’s name

             Mail.AddAddress "andy@andrewscompany.net", "Andrew Johnson, Jr."
             Mail.AddAddress "paul@paulscompany.com" ’Name is optional
             Mail.AddReplyTo "info@veryhotcakes.com"
             Mail.AddAttachment "c:\images\cakes.gif"

             Mail.Subject = "Thanks for ordering our hot cakes!"
             Mail.Body = "Dear Sir:" & Chr(13) & Chr(10) & _
                "Thank you for your business."

             On Error Resume Next
             Mail.Send
             If Err <> 0 Then
               Response.Write "Error encountered: " & Err.Description
             End If
             %>
```

The `Host` property is very essential and is given the name of the SMATP server which in our case is “`raccoon.demo.net`”.

The additional advantage is that one can also send HTML based email i.e. mail which has an HTML page in its body so that one can access its links from the web page or point to the reference. This object supports a property known as `isHTML` (boolean) which is False by default. If set to True, AspEmail will set the CONTENT-TYPE of the message body to TEXT/HTML and an HTML based message can be sent.

Section 5.6 which describes the file structure of the application points to the places in the applications in which these objects have been instantiated and used.

5.4.7 Client Browser and Mail Client

Microsoft's Internet Explorer 4.0 and 5.0 are the two clients used here as far as the browser is concerned. Version 4.0 has been used on `taylor` and 5.0 on `raccoon`. This browser is able to support SSL transactions and certificates.

The mail Client for the physician's office is Outlook Express 5.0. This is a sophisticated email client which allows the user to read and send mail in HTML format along with the standard text format. One is able to launch a web browser from within this application.

This facility has been made use of in the scenario. Upon clicking on a link in the body of the notification message in the email client, the physician's web browser launched is he is taken to the agency's web site.

5.5 Application Set-up

The following figure shows how the complete system appears with all the software components in place.

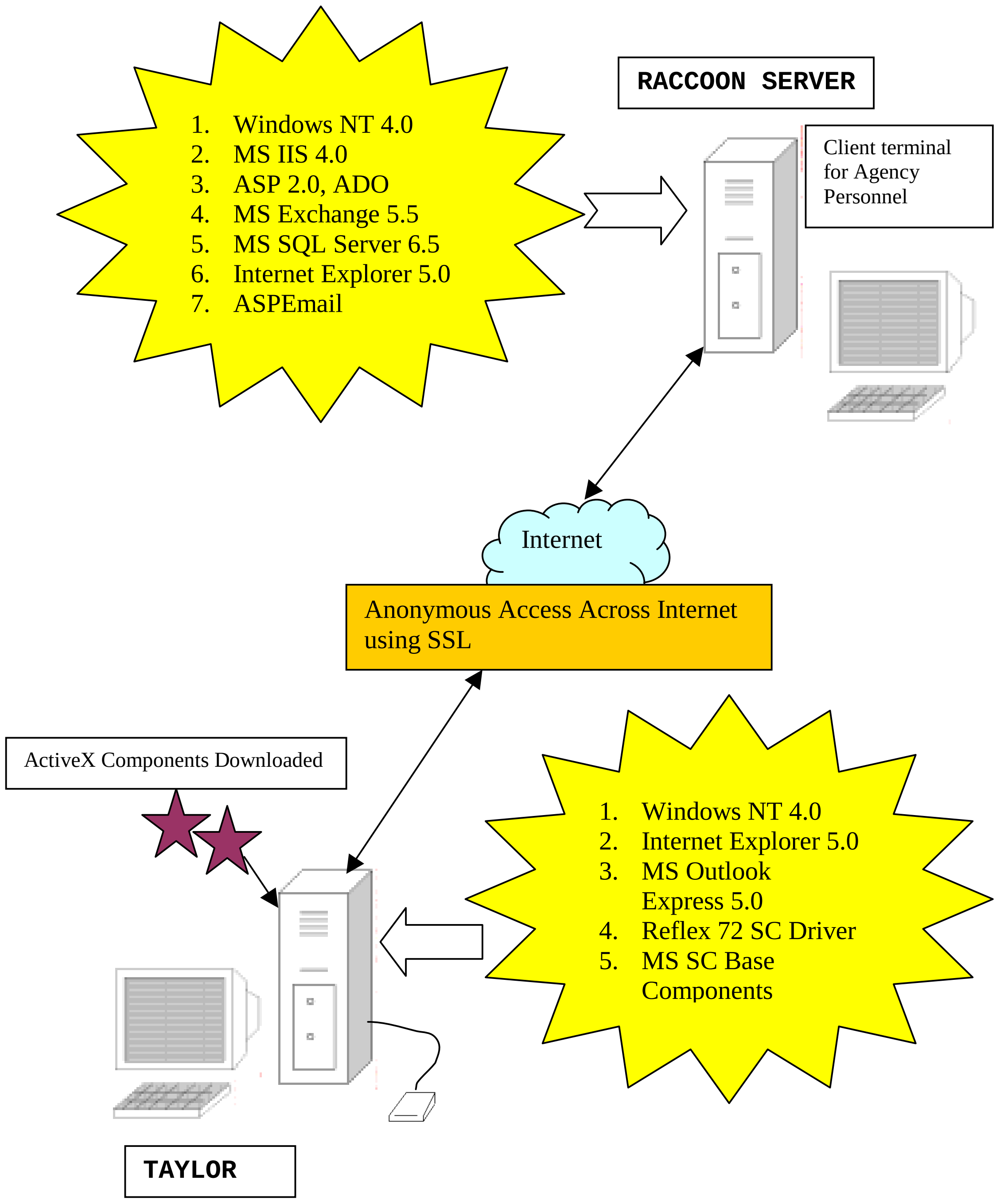


Figure 5.2 Software Installation on the System

5.6 Workflow Engine and the Workflow application

The “workflow engine” for the application is the logic embedded in the Active Server Pages which make up this project. The Active Server Pages Scripting Host which is on the IIS server processes the pages and provides for the flow of events. It sends responses to user actions and decides what is sent back to the user.

The next section describes the files used in this applications and how the “page-flow” i.e. the way in which the various pages are called to provide a smooth sequence of operations. There are a total of **34** files in this application. They are a mixture of ASP ,HTML and ActiveX Component files. The HTML files were created for those parts which did not require any server side processing. Each file represents a downloadable page from the web server and a link in that page to another file implies another page.

Table 5.4 discuss the file architecture of this project in terms of the name of the file, its type and its function in the overall project. The order chosen to describe the files more or less corresponds to the sequence with which one would encounter them if one ran the application.

Figure 5.3 illustrates the file architecture along with the links between the files.

Name of the Application :: **`workflowapp`**

Virtual Directory :: **`workflowapp`**

Link to launch the application:: https://raccoon.demo.net/workflowapp

Table 5.4 Summary of files used in this Application

File Name	Main Purpose	Links to other files	Special Features
1. default.html	The start page of the application when the client (Physician)accesses it by typing the above URL.	1.validate .asp	Automatically redirects to the link.
2. validate.asp	Checks to see if the client certificate presented is valid and if so directs to the appropriate page depending upon the client. If certificate is invalid then displays message to that effect .	1.mainFrameset.asp 2.second.asp 3.→ agencyFrameset.asp	1. if the client is a physician then the link is mainFrameset.asp 2. if the client is an agency personnel client then link is agencyFrameset.asp 3. if a client(physician) is coming to this site to view information about an already initiated transaction then link is second.asp
3.mainFrameset.asp	A page containing frames with two pages.	1.→ some_town_hospital _banner.html 2.choices.asp	-------------------------
4.some_town_hospi tal_banner.html	Displays a banner.	none	-------------------------
5.choices.asp	Displays choices for the client to choose to use the application.	1.hpcFrameset.asp 2.→ main_hospital_fram e.html 3.→ manualFormFrames et.asp	1.If the client has both the HPC and PC then he can choose to go to hpcFrameset.asp 2. For only PC the client can go to main_hospital_fram-e.html 3. For neither card the choice leads to manualFormFrames-et.asp
6.hpcFrameset.asp	Displays two frames which contain the banner and login page for this case.	1.→ some_town_hospital _banner.html 2. hpcInitialize.asp	The activeX Component for this scenario (HealthCardUpdate

			Control) is downloaded in this page.[1]
7.hpcInitialize.asp	If the object obtained from the parent frameset page is correctly instantiated and no errors found, it accepts the login information from the user and validates it. If validated it directs him to the form page.	1. onLoadCheckError-Msg.asp 2.hpcForm .asp 3. mainFrameset.asp	1. tests to see if the page was loaded without a parent document (not as part of frames) or if there are some errors in ActiveX object initialization and if so redirects to the error message page onLoadCheckErrorMsg-.asp 2.Redirects to hpcForm.asp if login is correct.
8.onLoadCheckErrorMsg.asp	Displays the error message and directs to the frameset page for this case.	1. hpcFrameset.asp	-------------------------
9.hpcForm.asp	If the object is valid and has no initialization errors it displays a form and uses the object to read from the card and fill the form's fields. It validates the user's entries before submitting them.	1.SCPhynotinitialize-d.asp 2. onLoadCheckError-Msg.asp 3.processForm.asp 4.	1. If the object has not been initialized from the previous page it redirects the user to SCPhynotionitialized.asp 2. Same function as in file hpcInitialize.asp

[1] The ActiveX Components are downloaded in the frameset pages in both the scenarios. This is because the frameset page acts as a container for other pages. Hence the other pages can get a handle to the object present in this frameset page. Even though different pages may be loaded and unloaded the object will persist since the frameset page always persists. Thus one can maintain the context of the session once this frameset is loaded . This is important since if the object had been downloaded in an ordinary web page it would disappear as soon as the page was replaced. Some initialization and clean up functions may also be performed on the object in this frameset page.

		noPatientInfoMsg.as-p	3. When user fills the form and submits , processForm.asp is called. 4. If no patient information exists when the object does a read on the card it displays the page noPatientInfoMsg.a-sp
10. noPatientInfoMsg.a-sp	Displays the error message and directs to the choices page for this case.	1. choices.asp	--------------------------
11. SCPhynotinitialized-.asp	Displays the error message and directs to the hpcInitialize page .	1.hpcInitialize.asp	--------------------------
12.processForm.asp	Stores the forms information in the database and informs the user of the link which can show her the current status of this request.	1. viewRequest.asp	---------------------------
13.viewRequest.asp	Retrieves the transaction information based on the id and displays it .	-------------------------	-----------------------------
14. main_hospital_frame.html	Displays two frames which contain the banner and login page for this case.	1. some_town_hospital_banner.html 2. login_smartcard.html	The activeX Component for this scenario (HealthCardAccess Control) is downloaded in this page.
15.login_smartcard.html	Checks to ensure that the object is valid ,initializes the object and then accepts the user's login information. If valid the user is shown the form page.	1. form.asp 2. mainFrameset.asp	------------------------
16.form.asp	If the object is valid and correctly initialized it displays the form to be filled and fills in the fields from the card. It validates the user's entries before submitting them.	1. Scnotinitialized.html 2. processForm.asp	1.If the object has not been initialized from the previous page it redirects the user to Scnotinitialized.html 2. When user fills the form and submits ,

			processForm.asp is called.
17.Scnotinitialized. html	Displays the error message and link to the login_smartcard.html page	1.login_smartcard.ht -ml	--------------------------
18.manualFormFra-meset.asp	Displays two frames , the banner and the form for the manual entry of information	1. manualForm.asp 2.some_town_hospit al_banner.html	--------------------------
19. manualForm.asp	Displays the form which has to be filled in by the user manually. It validates the user's input and submits the form.	1. processForm.asp 2. mainFrameset.asp	-------------------------
20.agencyDefault.a -sp	This is the starting page for the agency personnel who want to log onto the agency website.	1.validate.asp	Automatically redirects to the link.
21. agencyFrameset.asp	Displays two frames, one with the banner and the other with the task list for the user(agency personnel).	1. some_town_hospital _banner.html 2. taskList.asp	-----------------------
22.taskList.asp	Retrieves the list for tasks for the agency personnel and displays them.	1. task.asp	1. The task id. numbers on the page are links to the task.asp page with the appropriate parameters.
23. task.asp	Receives the task id from the tasklist.asp page and retrieves and displays the details of the transaction for that particular id. Calls the sendMail.asp page which notifies the client after the user has processed this transaction.	1. sendMail.asp	The agency personnel's additions to the transaction are validated before they are submitted.
24.sendMail.asp	Composes a mail based upon the transaction's processing and sends it to the client after obtaining his/her email id from the database.	1. taskList.asp	In the mail message body there is a link to the validate.asp page with the transaction id as a parameter so that the client can click on it to reach the agency web

			site again.
25.second.asp	The physician enters the agency web site through this page (after validate.asp calls it) when he clicks on the link in the email notification. This file displays all the information submitted by the physician so far , the agency's response and if needed the user will have to fill in the required fields depending on the status of the request (if the agency requires further details).	1.secondProcess.asp	The user's entries are validated before the form is submitted.
26.secondProcess.asp	This page updates the database with the additional information submitted by the client to the agency in response to the agency's request for further details.	1.viewRequest.asp	It displays the link which can show the user the current status of this request.
27. HealthCardUpdate.cab	This is the cab file which contains the HealthCardUpdate Control.	-----------------------	This gets downloaded in the page which refers to this file in the <OBJECT> tag.
28.HealthCardAccess.cab	This is the cab file which contains the HealthCardAccess Control.	-----------------------	This gets downloaded in the page which refers to this file in the <OBJECT> tag.
29.aspemail.zip	This control must be registered on the server using the regsvr32 command. This file provides the ASPEmail Component for the mail and notification feature	-----------------------	---------------------------
30.adovbs.inc	This is an include file containing the VBSCript constants used in the database access code.	------------------------	Included in all files which make use of constants in their database access code (for VBSCript)

31.adojavas.inc	This is an include file containing the JavaScript constants used in the database access code.	-------------------------	Included in all files which make use of constants in their database access code (for JavaScript)
32.hpcConstants.js	This include files consists of constants which are used for error conditions in the HPC card access scenario. (The HealthCardUpdate Control)	-------------------------	Included in the files which access the methods on the HealthCardUpdate object.
33.SC_testprocedures.js	This .js file consists of some procedures used to check the status and validity of the card object (ActiveX components) in the web pages.	-------------------------	Included in all web pages which checked the objects' validity.
34. transcationT.txt	Contains the schema for the database used in this application.	-------------------------	Must be run on the SQL server to create a table.

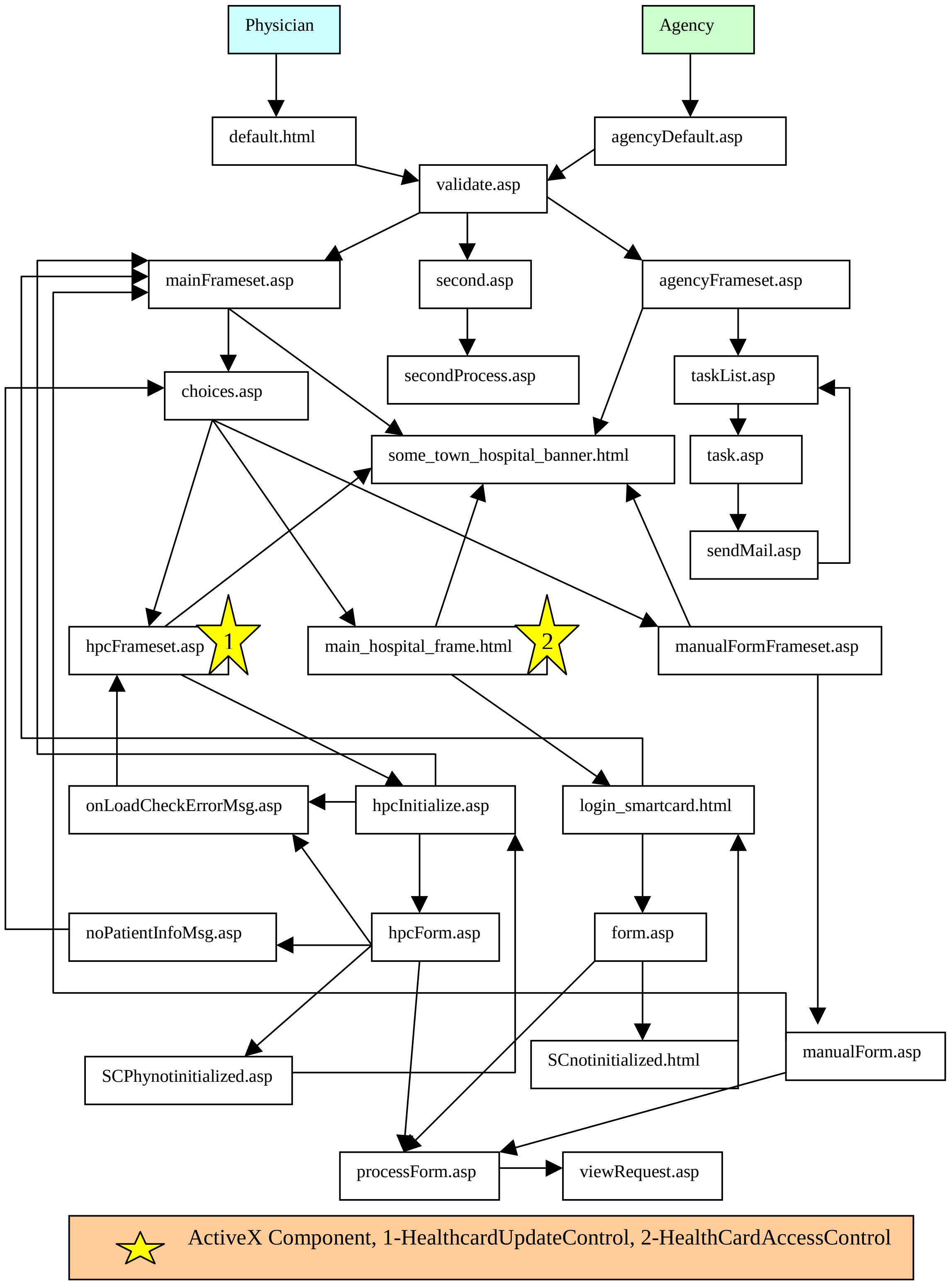


Figure 5.3 File System

5.7 Sample Scenario

A brief sample scenario with respect to the above file structure could be as follows.

1. Let us assume that the physician does have an HPC and the patient has a PC. However the physician's certificate is on the local machine and the physician has a valid Windows NT account on that domain.
2. After logging onto the machine the physician launches his browser and logs onto the site https://raccoon.demo.net/workflowapp. The SSL connection is established ; the browser prompts the user to select the client certificate to send to the server (assuming he has more than one certificates) and the default.html page redirects him to the validate.asp page.
3. If the client certificate is invalid the user is denied access and the scenario terminates.
4. If the certificate is valid then the user is sent to mainFramset.asp which has the choices.asp page which displays the three scenarios of use (HPC and PC, only PC , neither card)
5. The user chooses the HPC+PC combination and is taken to the page hpcFrameset.asp which prompts the user (the first time) that he can download the HealthCardUpdate Control .

   The user agrees to this and the component is downloaded and registered on the client's machine automatically. This object resides on this frameset page and its child pages can access this object by referencing it with the parents name or just using the `parent.objectName` syntax. This page contains the two frames, one a banner and the other contains the page which initializes the object and prompts the user to insert the HPC card into the card reader and enter the PIN number into the form provided. The page validates the user information and if the PIN is valid it prompts the user to enter the patient card. Upon entering the patient card it redirects the user to the hpcForm .asp page. If any errors occur such as the user typing an invalid PIN or the object is not properly initialized, the error message are displayed and the user is allowed to re-login or given a choice to exit as the case may be.

6. The hpcForm.asp page displays a form which is needed to be filled in by the user (A form for the Request for exemption from criteria). The ActiveX component extracts the patient information from the PC inserted in the card reader and populates the form. The user has to fill in some details too. Upon submitting the form , client side validation takes place which prompts the user of any invalid data entered and suggests the user to rectify the mistakes.
7. The form is submitted and the processForm.asp page is called which stores all details of this transaction in the database and also notifies the user of the transaction id (in the form of a link) which the user can click to see the data submitted and its current status.
8. This ends the initial interaction as far as the physician is concerned. When the agency personnel launches this application he will use the URL https://raccoon.demo.net/workflowapp/agencyDefault.asp, an SSL session is established and the agencyDefault.asp page is launched. The browser lists the certificate that the user has on the machine he is using and the user has to pick the appropriate one which the client sends along with the request. This page redirects the client to the validate.asp page which will validate the client certificate sent by the browser. If the certificate is invalid the client is not allowed to access the site.
9. If the client is allowed to enter the site he is taken to the agencyFrameset.asp page which contains the banner and the taskList.asp page. The taskList.asp page contains the list of requests submitted so far along with their id's, and current status. Namely have they been approved, denied or is the agency requesting further clarification. The user can click on the transaction id's and is shown the task.asp page which contains all the details of that particular the request . The user can view them and process them as needed. When the user submits his processing results , the system calls the sendMail.asp page which stores this on the database and sends a mail notification to the client who had submitted this request informing the client that his request has been processed and that he must follow the link provided in the message body to get to see the results of the processing. It is in this page that the server side ActiveX

component, ASPEmail is used. This ends the part where the agency personnel accesses the workflow application.

10. When the physician checks his email he will find the notification from the agency. The physician can click on a link provided which launches his browser and the browser places a request for the validate.asp page over a secure SSL connection and the client certificate is also sent by the browser. Validate.asp validates the certificate sent by the user and the user is shown the page second.asp which displays that particular transaction which the user has previously initiated. The user can view the agency's decision of approval or denial or fill in more details if the agency has requested further clarification.
11. On submitting this form the user's changes are processed by the page secondProcess.asp which updates the database with these changes and informs the user of the same.
12. When the agency personnel next check their task list as before this transaction will be present in the list and upon clicking the link they will be shown the updated request made by the physician. Thus the workflow loop for this scenario is complete. When a task is approved or denied there is no further action taken on it by the agency side and there is no link to those tasks in the task list page.

The figure below shows the sample interaction as described above.

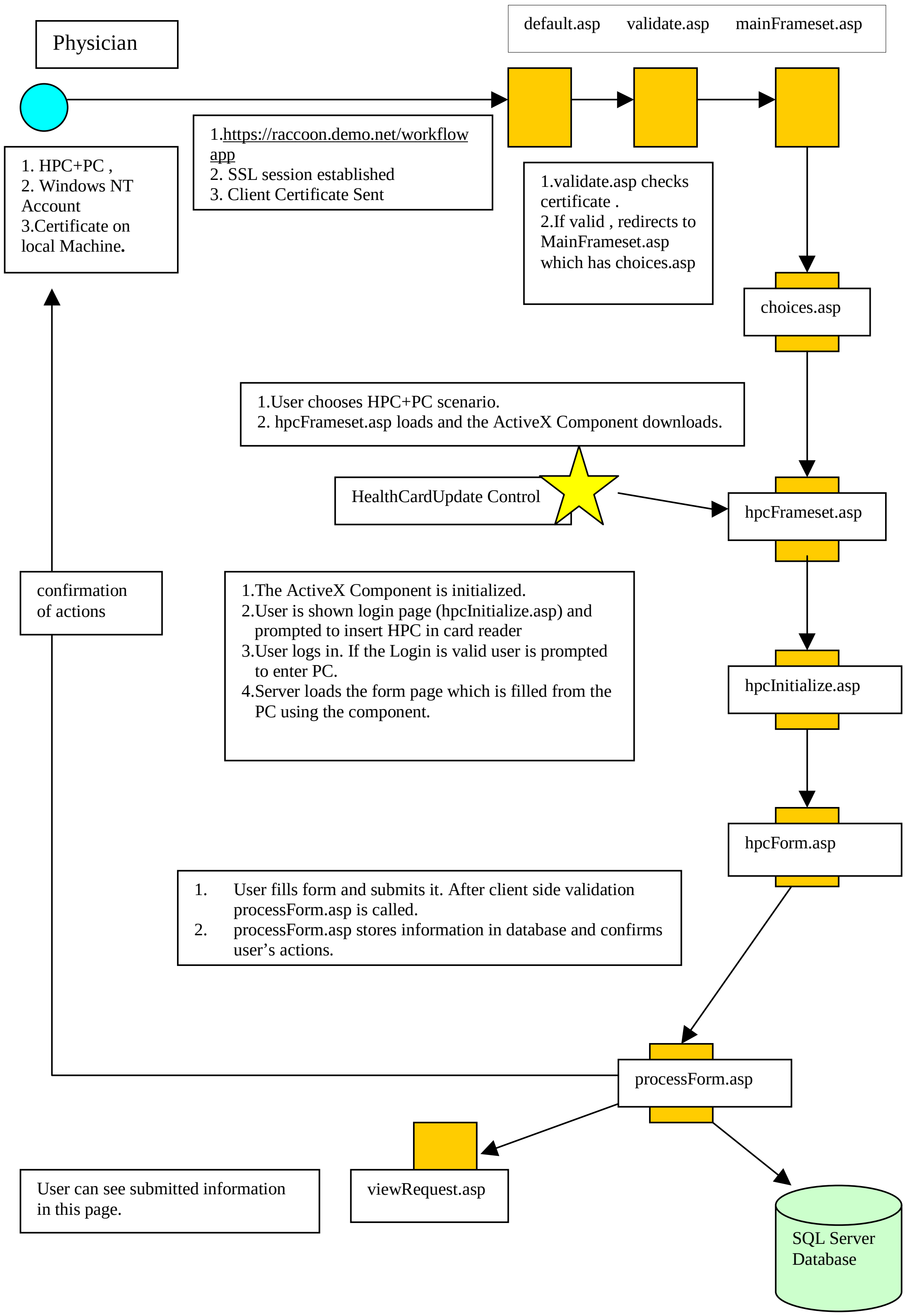

Physician
default.asp
validate.asp
mainFrameset.asp
1.https://raccoon.demo.net/workflowapp
2. SSL session established
3. Client Certificate Sent
1. HPC+PC ,
2. Windows NT Account
3.Certificate on local Machine.
1.validate.asp checks certificate .
2.If valid , redirects to MainFrameset.asp which has choices.asp
choices.asp
1.User chooses HPC+PC scenario.
2. hpcFrameset.asp loads and the ActiveX Component downloads.
HealthCardUpdate Control
hpcFrameset.asp
confirmation of actions
1.The ActiveX Component is initialized.
2.User is shown login page (hpcInitialize.asp) and prompted to insert HPC in card reader
3.User logs in. If the Login is valid user is prompted to enter PC.
4.Server loads the form page which is filled from the PC using the component.
hpcInitialize.asp
hpcForm.asp
1. User fills form and submits it. After client side validation processForm.asp is called.
2. processForm.asp stores information in database and confirms user's actions.
processForm.asp
viewRequest.asp
User can see submitted information in this page.
SQL Server Database

Figure 5.4 Initial Request by Physician(Steps 1 to 7)

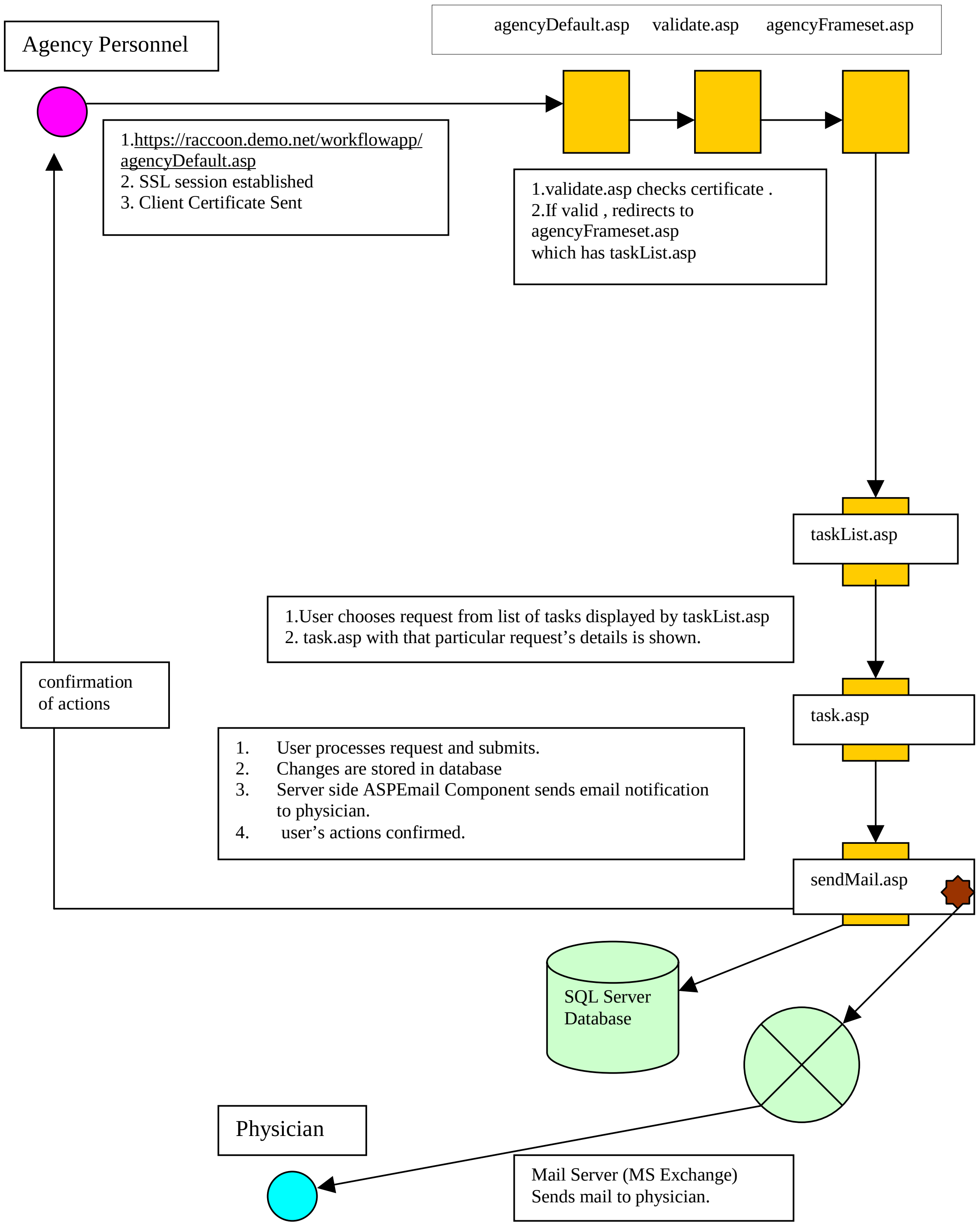

Agency Personnel
agencyDefault.asp
validate.asp
agencyFrameset.asp
1.https://raccoon.demo.net/workflowapp/agencyDefault.asp
2. SSL session established
3. Client Certificate Sent
1.validate.asp checks certificate .
2.If valid , redirects to agencyFrameset.asp which has taskList.asp
taskList.asp
1.User chooses request from list of tasks displayed by taskList.asp
2. task.asp with that particular request's details is shown.
confirmation of actions
task.asp
1. User processes request and submits.
2. Changes are stored in database
3. Server side ASPEmail Component sends email notification to physician.
4. user's actions confirmed.
sendMail.asp
SQL Server Database
Physician
Mail Server (MS Exchange)
Sends mail to physician.

Figure 5.5 Agency Personnel's Processing of the Request (Steps 8 to 9)

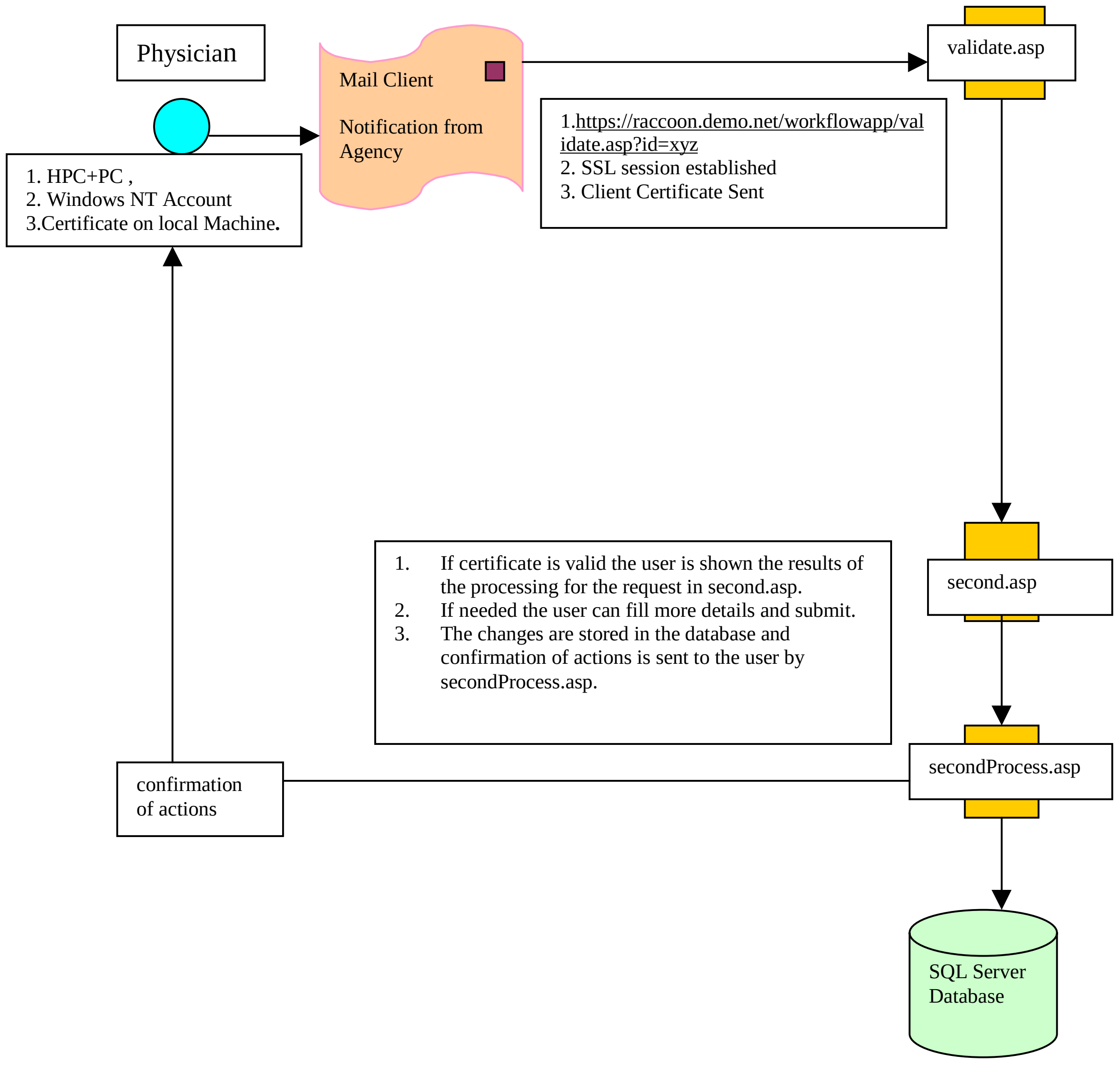

Physician
1. HPC+PC ,
2. Windows NT Account
3.Certificate on local Machine.
Mail Client
Notification from Agency
validate.asp
1.https://raccoon.demo.net/workflowapp/validate.asp?id=xyz
2. SSL session established
3. Client Certificate Sent
second.asp
1. If certificate is valid the user is shown the results of the processing for the request in second.asp.
2. If needed the user can fill more details and submit.
3. The changes are stored in the database and confirmation of actions is sent to the user by secondProcess.asp.
secondProcess.asp
confirmation of actions
SQL Server Database

Figure 5.6 Completion of Workflow Loop by Physician(Steps 10 to 12)

5.8 Software development tools used in this project.

Microsoft's Visual Studio: The integrated development environment provided by this application was used, especially the editor which allowed sophisticated editing features such as drag and drop of text, support for different file formats including **.sql, .asp** and **.html** and color coding of different types of code , html, script , CSS within a single file which made the editing easier.

Front Page 98:- This Front page editor was used to create tables with the WYSIWYG editing capabilities. This was easier than coding by hand since Front Page 98 automatically converts the design view into corresponding HTML syntax.

Chapter 6

Results ,Conclusions and Recommendations

This chapter summarizes the entire effort of this thesis in terms of the testing scenario, problems faced during the development of this project, conclusions and impressions derived from this research experience and the recommendations for future work.

6.1 Testing Scenario and Results obtained

The entire system was deployed and tested on the CERC testbed facility and the results of this exercise were as was expected from a system configured in that manner. There were no instances of the server side "crashing" during various operations across the different interfaces such as ASP to ADO and SQL server or ASP and Exchange Server. Even on the client side the ActiveX Components behaved as expected and did not display any error prone behavior.

Some of the specific tests carried out to check the security enforced in this application were as described below and the behavior exhibited are as follows.

Table 6.1 Important Tests carried out and their Results

Testing factor	**Test Carried Out**	**Behavior**
1.Security	1. Access web server without a client authentication certificate	1. Access Denied
	2. Access web server with an invalid (expired) client authentication certificate.	2.Access Denied
	3. Access information about a transaction belonging to another physician with a valid certificate	3.Access Denied since each transaction has a unique Client – CertificateSerialNumber Associated with it.

2.Client Side Authentication	1. Submitted one or more Empty fields in the form when the fields were required,	1.Client side scripts informs user of the same and indicates the field(s) which need to be filled.
	2.Submitted invalid data such as alphabets in the date field.	2.Client side script informs the user of the same and indicates the field(s) which needs to be corrected
3. Smart Card ActiveX Components	1. Invalid PIN or No PIN entered	1.Informs the user of the same and allows the user to re enter the PIN
	1. Using a wrong type of card or invalid card for a particular case. (HPC &PC or PC only)	2. Informs the user of the same and prompts the user to enter the correct type of card in the card reader.
4. Browser tests	1. Check if displayed links reach the correct pages.	1.All links open the correct pages they are meant to.
	2.Back, Refresh, Forward buttons clicked.	2.All show the desired pages as they should

As regarding the volume of traffic to the web site and performance of the web server tests can be carried out but it must be noted that the web server provides access selectively based upon the security policy and certificates of the organization and hence this should not be a problem in the real world. Even then the operating system and web server and RDBMS used have undergone rigorous tests regarding traffic and performance and should be able to take on a large number of users.

The above system was also migrated to Laptops running Windows NT and no change in performance (degradation or errors) was noted.

6.2 Problems faced during development of this application and their solutions

During the design and implementation of the a few problems were faced in development of this project which are as follows:

a) One of the ideas was that in addition to the security rules enforced it was desirable to actually encrypt data being transmitted. This meant that the client's Email signing certificate should be present with the server with which the server could encrypt some information and send it back to the client so that only that client could decrypt the information. Normally users wishing this use a tool such as Outlook/Outlook Express and send a digitally signed email to the recipient who can use a similar tool which will automatically add the sender's certificate to its list and use that certificate to perform the encryption when needed. This means a manual sign and send by the client is needed. But this was not desirable since it went against the objective which was to automate the client side efforts. Hence this was not implemented in the current version of the application.

Solution: A solution to this is to create a ActiveX Component which automatically appends an email signing certificate to the form submitting procedure that the client has to perform. This certificate is then extracted and stored by the server (if it does not already have a valid one from the same client). This can then be used to encrypt data which needs to be sent back. The email certificate on the client side could be on his smartcard or on the local machine.

b) This problem pertained to the development environment rather than being a true design and implementation problem. To obtain certificates from the FAYETTE server at CERC one has to connect to its Certificate Server /Enrollment Page using the Internet. But the testbed facility was secluded and this meant that to get a certificate for a particular account on the demo domain one had to remove the machine from the domain and connect it to the Internet, change its TCP/IP settings and reboot to be able to contact the FAYETTE server. Then one had to reverse the process and bring the machine back on the **`demo`** domain to be able to test the certificate. This was a tedious process since it had to be done a few times in the initial development phase.

Installing the Certificate Server on the Demo Server itself should have been done to alleviate this problem but was not done due to lack of time available.

6.3 Conclusions and Impressions drawn from this research experience.

1. All the objectives discussed in Chapter 1 were satisfactorily met in this research effort.

a) The internet and more specifically the world wide web was used a substrate for collaboration and workflow in a highly sensitive field of healthcare

b) Conventional servers and browsers were harnessed to provide a complex workflow scenario which included flow of information, enable decision making (both manual and automated), asynchronous and synchronous communication between parties involved

c) To demonstrate that the web browser is a suitable, reliable client for a complex application and the possibilities it provides in terms of high interactivity and delivery of rich content.

d) A high degree of automation was demonstrated through the use of component technology delivered over the internet through the use of smartcards.

e) The use of smartcard technology also made possible the concept of a mobile user.

f) Last but not the least all this was accomplished in a secure fashion without compromising the performance of the system.

2. There are quite a few of the shelf workflow products available which provide the users ability to create sophisticated workflow applications relatively quickly. For example Microsoft's Outlook provides quite a few advanced workflow options along with scripted agents and web

access. However some of the advantages to using solution which involves building from the ground up with web pages and scripting technology as opposed to an off the shelf product are

a) Web technologies due to their ubiquitous nature are very popular and HTML, VBScript, JavaScript, ASP are relatively easy to learn and use. Thus a shorter learning curve provides incentive to use the above means.

b) One can customize a workflow application written ground up easily and create various complex scenario's whereas one has to use the limited options provided in an off the shelf product. Also off the shelf products may not cater to different scenario's for example the needs for a healthcare workflow solution may be different from one in the education field.

c) Web pages constructed based upon sound software engineering principles can help in easy maintenance and modifications of the workflow application which can lead to better extensibility.

d) Also the options for presentation that a web interface offers where one can easily incorporate a variety of text, color, image and multimedia far exceeds that of any one application.

e) The internet supports interoperability by integrating with legacy systems. This is done with host of technologies such as XML. Now only the content need be transferred securely and presentation can be left to the client side. Use of Extensible Style Sheets (XSL) on the client side provides very rich presentation options for the client. Some of these alternatives are not yet supported in off the shelf products.

6.4 Recommendations for future work possible:

1. The application could be made interoperable with heterogeneous systems and also interface with different data sources which are part of legacy systems. This can be done by using technologies such as XML which aid interoperability as they emphasize on content and not on presentation.
2. The workflow application could be enhanced by providing some means to start a process or

task remotely, monitor its progress, accept notifications from it and stop it after it has completed the task. The SWAP protocol can be incorporated to introduce this aspect since the protocol aims to accomplish such workflow which is accountable and interoperable.

3. The current scenario could be enhanced to provide a more complex chain of events.
4. Users could be given options to customize their pages used such as the task list page and the visual interface elements can be more user friendly such as drop-down selection boxes in place of text fields to avoid typing.
5. SmartCard technology is making rapid strides in interfacing with the PC and Microsoft and other leading companies are coming with better solutions and products in this area.

   This technology may be fully exploited in some ways to handle the current tasks in an improved manner. Also a part of the application namely the GINA module and the use of the module to transfer the certificate from the card onto the machine [33] work only with Microsoft's products and the Windows NT platform. Making this platform and browser independent side is the next step.
6. In place of sending only the notification to the client confidential information could also be sent which must be encrypted. This encryption can take place on the server side with the use of ActiveX components after the Email signing certificate is obtained from the client side as described in the section 6.2 part a.
7. Microsoft Exchange server provides some workflow and collaboration relate features through the use of Collaboration Data Objects and Routing Objects. It would be worthwhile to investigate their potential in extending this application. In addition the versatility of the Outlook Client could be used.
8. The next major step could also be in developing an application on the server side could create a workflow management system by providing the skeleton code in web pages when the user selects and clicks options provided in the application. This skeleton could then be modified to suit one's purpose. The advantage of this is that it avoid the user from having to code the

trivial and often repeated tasks in every single page. Thus increasing speed and productivity of the application development.

9. It would be appropriate to introduce formal methods to model the entire workflow management system to be built, during the design phase. This could be done with the help of GUI tools or state machine diagrams. The advantage of establishing a design framework for such applications is that the design can be analyzed and the inherent weaknesses can be found out and eliminated. This leads to better design and correct implementation of the system. Also theoretical models for such workflow applications can be turned into templates which enable easy development of applications where a majority of the routine functions can be automatically implemented and customizations can be made. This assures that a basic workflow application will be created in a standardized error free format which can be enhanced.

In conclusion we can see that this project can open a number of exciting avenues for further research and development in this field.

Bibliography

1. *Adding Windows NT and Internet Information Server Security Features*, Chapter 9 from Microsoft Internet Information Server 4.0 Training Kit, published by Microsoft Press http://www.microsoft.com/technet/iis/train9.asp

2. ASPEmail Component ,Persits Software Inc http://www.persits.com/

3. ASPToday Web site http://www.asptoday.com

4. Bolcer, Gregory Alan. Kaiser, Gail. "*SWAP: Leveraging the Web To Manage Workflow*"IEEE Internet Computing , January February 1999.

5. Brian Francis, Kauffman, Llibre et al., *Beginning Active Server Pages*, Wrox Press Ltd. (1998)

6. Chaiken, Barry P. MD, MPH, Corporate Medical Director " *Workflow in Healthcare A White Paper"* Araxsys Solutions .

7. Concurrent Engineering Research Center , http://www.cerc.wvu.edu

8. Dynamic HTML: The Next Generation of User-Interface Design (Backgrounders/WebDevelopment),http://msdn.microsoft.com/library/backgrnd/html/msdn_dynhtml.htm

9. Endeavors Project http://www.ics.uci.edu/pub/endeavors/endeavors.html

10. Endeavors Published Papers http://www.ics.uci.edu/pub/endeavors/docs/papers/

11 Extensible markup language(XML), http://www.w3.org/XML/

12. Healtheon http://www.healtheon.com/tech/index.html

13. Internet Information Server 4.0, http://www.microsoft.com/ntserver/web/exec/overview/WebFeat.asp

14. Internet Security ,Public Key Cryptography, Secure Sockets Layer http://developer.netscape.com/docs/manuals/index.html?content=security.html

15. Internet Security Concepts , http://developer.netscape.com/docs/manuals/security/pkin/index.htm

16. *Intranets: Powerful New Tools for Healthcare Organizations*, PeopleSoft and Arthur Anderson ,http://www.peoplesoft.com/cgi-bin/whitepaper.cgi?paper=0992-0298

17. Irvine Research Unit in Software http://www.ics.uci.edu/IRUS/

18. Isaacs, Scott. *Inside Dynamic HTML*, Microsoft Press(1997)

19.Kannan, Srivatsan . *Smartcard Based Authentication.* Master's thesis , Concurrent Engineering Research Center, West Virginia University, September 1997

20. Kurose James F. , Ross Keith W., *Computer Networking : A top-down approach featuring the Internet*., Addison Wesley (1999)

21. Large Scale Distributed Information Systems.University of Georgia (Athens)http://lsdis.cs.uga.edu/

22. Lee, Wenke. Kaiser Gail E., Clayton Paul D., Sherman Eric H. ", *"OzCare: A Workflow Automation System for Care Plans"* "1996 American Medical Informatics Association Annual Fall Symposium", October1996,pages 577-5811

23. Macrae Kenneth, MD Phd. *"The Araxsys Solution"*

24. Microsoft Developers Network (MSDN) library online , http://msdn.microsoft.com/library/

25. *Microsoft Exchange Administrators Companion,* Microsoft press

26. Microsoft Exchange Documentation installed with the Server (MS Exchange 5.5)

27. Microsoft Exchange http://www.microsoft.com/technet/exchange/default.asp

28. Microsoft Exchange Server Page http://www.microsoft.com/exchange/

29. Microsoft SQL Server http://www.microsoft.com/sql

30. Microsoft SQL Server http://www.microsoft.com/sql/

31. Microsoft Windows NT Option Pack Documentation http://msdn.microsoft.com/library/psdk/iisref/iiwaabt.htm

32. Microsoft's Active Server Pages site http://msdn.microsoft.com/workshop/server/asp/ASPover.asp

33. Mutsuddi, Monoreet . *Smart Card enabled security services to support Secure Telemedicine Applications.* Master's thesis , Concurrent Engineering Research Center, West Virginia University, 2000.

34. Programming Systems Laboratory Oz and OZWeb http://www.psl.cs.columbia.edu/,http://www.psl.cs.columbia.edu/edcs/edcs.html,ftp://ftp.psl.cs.columbia.edu/pub/psl/INDEX.html

35. Projects at CERC http://www.cerc.wvu.edu/projects1.htm

36. Schlumberger http://www.slb.com/smartcards/

37. Sheth Amit, Miller John et al , *"WebWork: METEOR's Web-Based Worflow management System"*(Large Scale Distributed Information Systems Lab. http://lsdis.cs.uga.edu/lib/lib.html) Journal of Intelligent Information Systems, Kluwer Academic Publishers , Boston. 1997

38. Sheth Amit, Miller John, et al, *"The Future of Web Based Workflow"*(Large Scale Distributed Information Systems Lab. http://lsdis.cs.uga.edu/lib/lib.html)

39. SSL protocol http://developer.netscape.com/docs/manuals/security/sslin/index.htm

40. Stabbert Scott, *Authentication and Security for Internet Developers*, ASP/Visual InterDev Training Lead ,Microsoft Internet Developer Support ,Microsoft Corporation (October 10, 1997) http://msdn.microsoft.com/workshop/c-frame.htm#/workshop/server/default.asp

41. Supporting Distributed Workflow Using HTTP, P. Kammer et al., International Conference on Software Process , (ICSP5), June, 14-17, 1998, Lisle, IL USA

42. The Component Object Model: A Technical Overview http://msdn.microsoft.com/library/techart/msdn_comppr.htm

43. The HTTP Protocol http://www.ietf.org/html.charters/http-charter.html

44. The Internet, FrontPage 98 Documentation ,Microsoft ,MSDN Online Library http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/officedev/fpage/005fp_2.htm

45. The SMTP protocol ftp://ftp.isi.edu/in-notes/rfc821.txt

46. The Software Workflow Access Protocol home page http://www.ics.uci.edu/~ietfswap/

47. Visual Interdev Page , Microsoft Corporation. http://msdn.microsoft.com/vinterdev/

48. Walther, Stephen. *Active Server Pages Unleashed* , Sams.net Publishing (1998)

49. West Virginia University ,attp://www.wvu.edu.

50. Windows NT, http://www.microsoft.com/ntserver

51. Windows NT Option Pack documentation Microsoft IIS\Server Administration\Security

52. Workflow Management Coalition, http://www.aiim.org/wfmc/mainframe.htm

53. World Wide Web Constortium (W3C), http://www.w3.org/MarkUp/

54. ZDnet Web site http://www.zdnet.com

RESUME

Vijayanand Bharadwaj

vanandb@hotmail.com

Education	**Masters in Computer Science (M. S)** **May 2000** Department of Computer Science and Electrical Engineering (http://www.csee.wvu.edu) West Virginia University (http://www.wvu.edu) Current GPA :: 3.91 **Bachelors in Electronics Engineering (B.E),** **May 1997** Bombay University, India.

Professional Experience	**Graduate Research Assistant** **(August 1997 -- Present)** Concurrent Engineering Research Center (http://www.cerc.wvu.edu) (West Virginia University) **Student Intern** **(June 1996 --May 1997)** Crompton Greaves Ltd Bombay , India

Technical Skills	**Technologies ::** Active Server Pages , ActiveX ,Dynamic HTML Active Data Objects, Collaboration Data Objects. Secure Sockets Layer and Internet Security. **Programming Languages Known ::** C/C++, Ada , Java **Scripting Languages and Web related Technology::** VBscript, JScript, JavaScript, HTML, Cascading Style Sheets **Development Environments familiar with ::** Microsoft Visual C++, Microsoft FrontPage 2000 and Visual InterDev 6.0 **Database Related ::** SQL , Oracle 8I and Microsoft SQL server 6.5 **Servers Technology ::** Basic System Administration of servers such as Microsoft Internet Information Server 4.0, Modest knowledge of Microsoft Exchange Server 5.5,Microsoft SQL Server 6.5, Personal Web Server **Platforms Worked on ::** Windows NT 4.0, SunOS Unix **Tools and Applications::** Microsoft Outlook , Outlook Express.

Projects	Member of the Secure Collaborative Telemedicine Development Team at CERC (sponsors National Library of Medicine). Responsible for computer programming and application development with special emphasis on Internet and the World Wide Web. 1. Involved in developing a Secure, Collaborative Workflow Environment using the Internet and the World Wide Web as a Substrate. This is also the focus of my Master's Thesis .The objective of this project is to demonstrate that the efficiency of a complex Workflow Process can be improved by using the Internet and WWW as a substrate. All transactions are carried out in a very secure fashion using SSL and certificates. The user is mobile and is not tied to a domain to access the application. This is facilitated through the use of Smartcard technology and ActiveX components. Automation of the process results in decrease in effort on the user's part and in turn helps in minimizing errors in the process. Communication between parties involved in this application is both synchronous and asynchronous. Windows NT server using IIS 4.0 provided the infrastructure. Exchange and SQL server provided the communication and database support. ASP, ADO, SSL, ActiveX, DHTML, CSS VBScript and JavaScript were used extensively to develop this project. 2. Participated in the design and development of a system using the World Wide Web for management of Patient Medical Records. This system uses the Internet to securely view, search and sort, scanned patient medical records. These records can be paper based notes or images such as X-rays and other medical records. This application has been integrated with a remote server and a CORBA based service is run on the server to which the web application could connect and obtain information. All transactions made in this application are completely secure and use SSL and Client Side Authentication with the help of Certificates. Database support is provided by Microsoft SQL server in this project. 3. Created a web based statistical tool which can be launched from the above application. This tool keeps track of the user's session and monitors the number of patient charts and records examined by a physician. 4. Developed a module in Visual C++ that connects to a database and performs basic databases functions such as add, update and delete. This module has been integrated into a larger project that uses Smart Card Technology as its hub.

Relevant Courses Taken	**Graduate Level Computer Science (CS)Courses** Internet Technologies (CS391V) Theory of Operating Systems (CS 356) System LifeCyle and Configuration Management (CS 391G) Advanced Analysis of Algorithms (CS 326) The Formal Specification of Programming Languages(CS 336) **Senior /Graduate Level Computer Science Courses** Data and Computer Communications (CS 268) Introduction to Object Oriented Programming in C++ (CS 210)

For a detailed description of all courses taken and a brief overview of the various projects completed at CERC and as part of the course work, please e-mail me at vanandb@hotmail.com