A PROJECT REPORT ON

Patient Monitoring System

**Submitted to Acharya Nagarjuna University for the partial fulfillment of the
Requirement for the**

Award of Degree for

Master of Computer Applications

Done by

Mr. R. Gopal Rao

**Hindu College PG Courses
Guntur**

CERTIFICATE

This is to certify that **Mr. R.GOPAL RAO**, bearing Roll No**. Y7MC13012** have developed Software project titled **PATIENT MONITORING SYSTEM** for **CHIGULLA SOFTWARE** as a partial Fulfillment for the award of the Degree of **M.C.A.**

HEAD OF DEPARTMENT

PRINCIPAL
HINDU COLLEGE P.GCOURSES

EXTERNAL

ACKNOWLEDGMENT

My express thanks and gratitude and thanks to Almighty God, my parents and other family members and friends without whose uncontained support, I could not have made this career in M.C.A.

I wish to place on my record my deep sense of gratitude to my project guide, **Mr. Ramesh Sr.Software Engineer, Chigulla Software, Hyderabad** for his constant motivation and valuable help through the project work. Express my gratitude to **Mr.Soma sekhararao**, Director of Hindu College P.G Courses for his valuable suggestions and advices through out the M.C.A course. I also extend my thanks to other Faculties for their Cooperation during my Course.

Finally I would like to thank my friends for their cooperation to complete this project.

R.Gopal Rao

ABSTRACT

In a given day, number of patients visits a hospital or a clinic. Many hospitals in India still manage the patient data manually. Hospitals will be able to save money and time if they have a good software program for managing patient's data. The idea is to develop web based patient management software that can be used to keep track of the patients registering in a hospital or clinic. Doctors and the rooms available in a hospital can be managed using this system. Also, this system should support accessing the previous visit histories of any patient, search for patients by name etc.

A few points to be noted about the system we are developing here

- A patient can be categorized as " In patient" or "Out Patient". If patient type is "In Patient", a bed will be assigned to the patient.
- A doctor will be assigned to each patient before the patient meets the doctor. Only one doctor can be assigned to a patient at a given time.
- A patient can visit the hospital any number of times

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of MS-SQL Server and all the user interfaces have been designed using the ASP.Net technologies. The database connectivity is planned using the "SQL Connection" methodology. The standards of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology, in mind. The specification has been normalized up to 3NF to eliminate all the anomalies that may arise due to the database transaction that are executed by the general users and the organizational administration. The user interfaces are browser specific to give distributed

accessibility for the overall system. The internal database has been selected as MS-SQL server 200.The basic constructs of table spaces, clusters and indexes have been exploited to provide higher consistency and reliability for the data storage. The MS-SQL server 200 was a choice as it provides the constructs of high-level reliability and security. The total front end was dominated using the ASP.Net technologies. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations. The database connectivity was planned using the latest "SQL Connection" technology provided by Microsoft Corporation. The authentication and authorization was crosschecked at all the relevant stages. The user level accessibility has been restricted into two zones namely.

CONTENTS

INTRODUCTION

1.1. INTRODUCTION TO PROJECT

Hospital administrators are often inundated with information about a large number of patients and their visits to the hospital that need to be organized and kept up-to-date. The patient management system is a web based application that is designed and developed for hospital administrators and doctors to organize information on patient visits. The system intends to facilitate several steps in the process from the patient registration and to the patient evaluation. During this process, there will be many tasks that have to be handled by this system including maintaining complete information. The main objective of the system is to provide the administration staff and doctors with an easily maintainable information system for patient registration, visit scheduling and patient tracking with latest information.

1.2. ORGANIZATION PROFILE

SOFTWARE SOLUTIONS

Chigulla Software is an IT solution provider for a dynamic environment where business and technology strategies converge. Their approach focuses on new ways of business combining IT innovation and adoption while also leveraging an organization's current IT assets. Their work with large global corporations and new products or services and to implement prudent business and technology strategies in today's environment.

CHIGULLA SOFTWARE RANGE OF EXPERTISE INCLUDES:

- Software Development Services
- Engineering Services
- Systems Integration
- Customer Relationship Management

- Product Development
- Electronic Commerce
- Consulting
- IT Outsourcing

We apply technology with innovation and responsibility to achieve two broad objectives:

- Effectively address the business issues our customers face today.
- Generate new opportunities that will help them stay ahead in the future.

THIS APPROACH RESTS ON:

- A strategy where we architect, integrate and manage technology services and solutions - we call it AIM for success.
- A robust offshore development methodology and reduced demand on customer resources.
- A focus on the use of reusable frameworks to provide cost and times benefits.

They combine the best people, processes and technology to achieve excellent results - consistency. We offer customers the advantages of:

SPEED:

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and evens before schedule.

EXPERTISE:

Our teams combine cutting edge technology skills with rich domain expertise. What's equally important - they share a strong customer orientation that means they actually start by listening to the customer. They're focused on

coming up with solutions that serve customer requirements today and anticipate future needs.

A FULL SERVICE PORTFOLIO:

They offer customers the advantage of being able to Architect, integrate and manage technology services. This means that they can rely on one, fully accountable source instead of trying to integrate disparate multi vendor solutions.

SERVICES:

Chigulla Software is providing it's services to companies which are in the field of production, quality control etc With their rich expertise and experience and information technology they are in best position to provide software solutions to distinct business requirements.

1.3. **PURPOSE OF THE PROJECT**

In a given day, number of patients visits a hospital or a clinic. Many hospitals in India still manage the patient data manually. Hospitals will be able to save money and time if they have a good software program for managing patient's data. The idea is to develop web based patient management software that can be used to keep track of the patients registering in a hospital or clinic. Doctors and the rooms available in a hospital can be managed using this system. Also, this system should support accessing the previous visit histories of any patient, search for patients by name etc.

A few points to be noted about the system we are developing here

- A patient can be categorized as " In patient" or "Out Patient". If patient type is "In Patient", a bed will be assigned to the patient.
- A doctor will be assigned to each patient before the patient meets the doctor. Only one doctor can be assigned to a patient at a given time.
- A patient can visit the hospital any number of times

1.4. **PROBLEM IN EXISTING SYSTEM**

- Cannot Upload and Download the latest updates.
- No use of Web Services and Remoting.
- Risk of mismanagement and of data when the project is under development.
- Less Security.
- No proper coordination between different Applications and Users.
- Fewer Users - Friendly.

1.5. **SOLUTION OF THESE PROBLEMS**

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

1. User friendliness is provided in the application with various controls.
2. The system makes the overall project management much easier and flexible.
3. Readily upload the latest updates, allows user to download the alerts by clicking the URL.
4. There is no risk of data mismanagement at any level while the project development is under process.
5. It provides high level of security with different level of authentication.

SYSTEM ANALYSIS

2.1. INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

2.2. ANALYSIS MODEL

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.
The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

- A preliminary design is created for the new system.

- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
    1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning an designing the second prototype.
    4. Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:

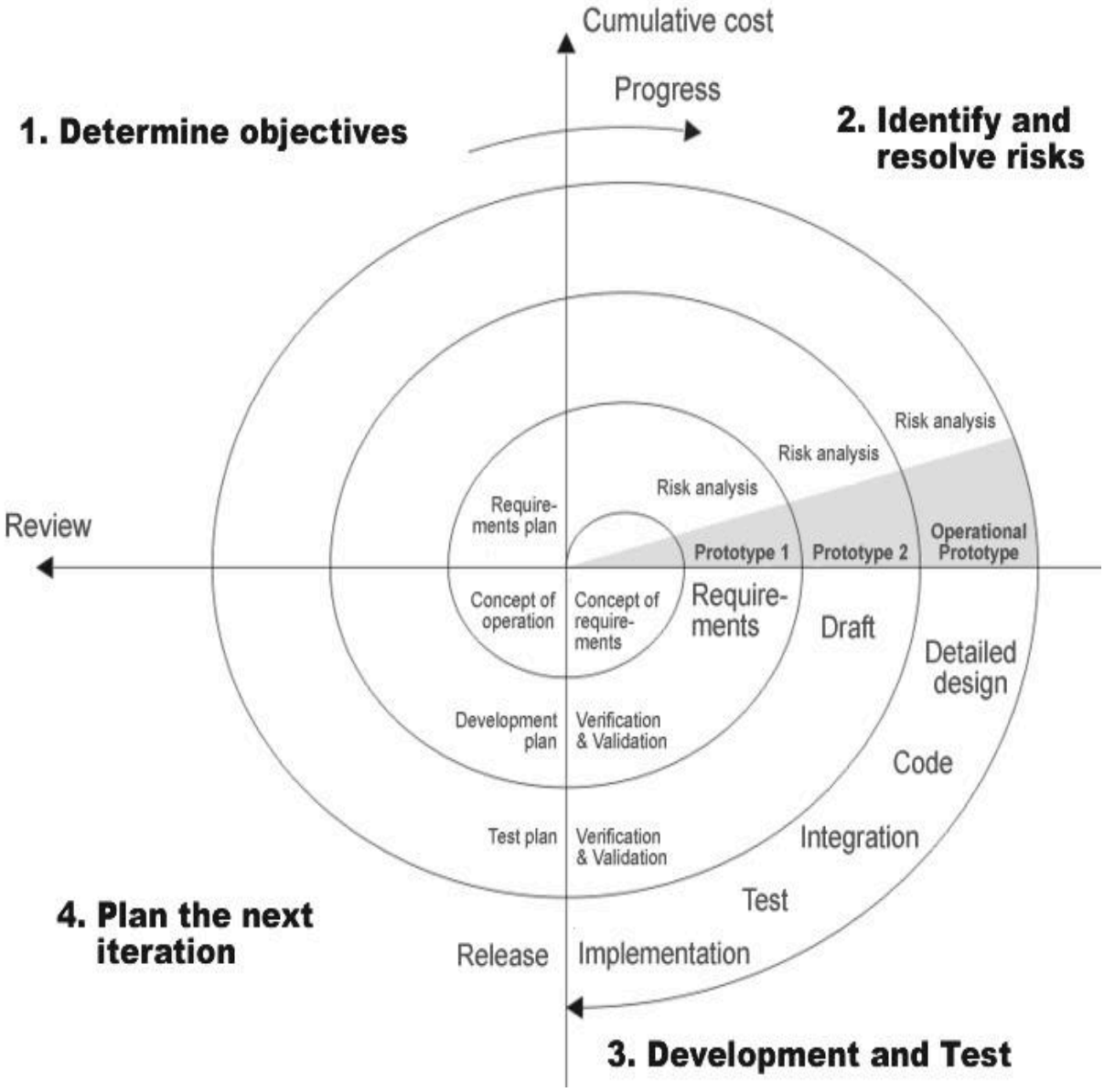


Fig 1.0-Spiral Model

2.3. STUDY OF THE SYSTEM

GUI'S

In the flexibility of the uses the interface has been developed a graphics concept in mind, associated through a browses interface. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The interfaces help the administrations with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The operational or generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.

NUMBER OF MODULES

The system after careful analysis has been identified to be presented with the following modules:

The modules involved are:

1. Administration
2. Employee
3. Doctor
4. Reports
5. Authentication

Administrator:-

In this module Administrator will have complete control of the system. She/he can Add/Edit/Delete patients, Add/Edit/Delete Doctors, Add/Edit/Delete Beds, Search for patients, Assign patients to doctors. He can search all the info about the Admitted Patient, Discharged Patient, Doctors, Medicine, Test, and Room.

Employee:-

This is Module is for employees who are working in that particular hospital. Admin will assign them user name and password by this they can enter in to their related page. An employee can enter the information about the Admitted Patient, he can add all type of charges like Room, Medicine, Test etc to particular Patient, and he can also maintain the information of the Patient who has discharged.

Doctor:-

Doctors who are related to that hospital can enter to their related page by their login name and password. They can see their information, change their account, they can see patients whom they checked, and Doctor can access a patient's record and update his observations about the patient in that particular visit

Reports:-

This module contains all the information about the reports generated by the admin of the patient admitted, discharged, medicine charges, room charges, test charges for a particular patient. Etc…

Authentication:-

This module contains all the information about the authenticated user. User without his username and password can't enter into the login if he is only the authenticated user then he can enter to his login.

PROJECT INSTRUCTIONS:

- Based on the given requirements, conceptualize the Solution Architecture. Choose the domain of your interest otherwise develop the application for ultimatedotnet.com. Depict the various architectural components, show interactions and connectedness and show internal and external elements. Design the web services, web methods and database infrastructure needed both and client and server.

- Provide an environment for upgradation of application for newer versions that are available in the same domain as web service target.

2.4. SYSTEM REQUIREMENTS SPECIFICATIONS

HARDWARE REQUIREMENTS:

- PIV 2.8 GHz Processor and Above
- RAM 512MB and Above
- HDD 20 GB Hard Disk Space and Above

SOFTWARE REQUIREMENTS:

- WINDOWS OS (XP / 2000 / 200 Server / 2003 Server)
- Visual Studio .Net 2005 Enterprise Edition
- Internet Information Server 5.0 (IIS)
- Visual Studio .Net Framework (Minimal for Deployment)
- SQL Server 2000 Enterprise Edition

2.5. PROPOSED SYSTEM

To debug the existing system, remove procedures those cause data redundancy, make navigational sequence proper. To provide information about audits on different level and also to reflect the current work status depending on organization/auditor or date. To build strong password mechanism.

NEED FOR COMPUTERIZATION

We all know the importance of computerization. The world is moving ahead at lightening speed and every one is running short of time. One always wants to get the information and perform a task he/she/they desire(s) within a short period of time and too with amount of efficiency and accuracy. The application areas for the computerization have been selected on the basis of following factors:

- Minimizing the manual records kept at different locations.
- There will be more data integrity.
- Facilitating desired information display, very quickly, by retrieving information from users.
- Facilitating various statistical information which helps in decision-making?
- To reduce manual efforts in activities that involved repetitive work.
- Updating and deletion of such a huge amount of data will become easier.

2.6. INPUT AND OUTPUT

The main inputs, outputs and major functions of the system are as follows

INPUTS:

- Head operator enters his or her user id and password.
- Operators enter his or her user id and password.
- Technicians enter his or her user id and password.
- Sub technicians enter his or her user id and password.
- User requests the reports.
- User requests the search.
- Head operator can edits the personal details and so on.

OUTPUTS:

- Head operator receives personal details.
- Operator receives the personal details.

- Technicians receive personal and technical details.
- Users receive requested reports.
- Displays search result.

2.7. PROCESS MODELS USED WITH JUSTIFICATION

ACCESS CONTROL FOR DATA WHICH REQUIRE USER AUTHENTICATION

The following commands specify access control identifiers and they are typically used to authorize and authenticate the user (command codes are shown in parentheses)

USER NAME (USER)

The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made (some servers may require this).

PASSWORD (PASS)

This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress type out.

Feasibility Report

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

3.1. Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an

easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.2. Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.3. Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

SOFTWARE REQUIREMENT SPECIFICATION

The software, Site Explorer is designed for management of web sites from a remote location.

INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope: This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

DEVELOPERS RESPONSIBILITIES OVERVIEW:

The developer is responsible for:

- Developing the system, which meets the SRS and solving all the requirements of the system?
- Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- Conducting any user training that might be needed for using the system.
- Maintaining the system for a period of one year after installation.

4.1. FUNCTIONAL REQUIREMENTS:

OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization.
- Internal Outputs whose destination is with in organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely with in the computer department.
- Interface outputs, which involve the user in communicating directly with

OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

- Will decimal points need to be inserted
- Should leading zeros be suppressed.

Output Media:

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

- The suitability for the device to the particular application.
- The need for a hard copy.
- The response time required.
- The location of the users

- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are:

The outputs were needed to be generated as a hot copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achive the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

INPUT STAGES:

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.

- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

INPUT MEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As
Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded upto the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

ERROR DETECTION

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with pop up menus.

USER INTERFACE DESIGN

It is essential to consult the system users and discuss their needs while designing the user interface:

USER INTERFACE SYSTEMS CAN BE BROADLY CLASIFIED AS:

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

USER_INITIATED INTERGFACES

User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

COMPUTER-INITIATED INTERFACES

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

ERROR MESSAGE DESIGN:

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

4.2. PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

SELECTED SOFTWARE

5.1. INTRODUCTION TO .NET Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and Remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive,

object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.
The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

FEATURES OF THE COMMON LANGUAGE RUNTIME

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory

manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

.NET FRAMEWORK CLASS LIBRARY

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Scripted or hosted applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.
- XML Web services.
- Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

CLIENT APPLICATION DEVELOPMENT

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

ASP.NET

Server Application Development

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

SERVER-SIDE MANAGED CODE

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications,

such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

ACTIVE SERVER PAGES.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy

extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.

- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
- **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
- **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

LANGUAGE SUPPORT

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

WHAT IS ASP.NET WEB FORMS?

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

- The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
- The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
- The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form postback to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

CODE-BEHIND WEB FORMS

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

INTRODUCTION TO ASP.NET SERVER CONTROLS

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within

an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both uplevel and downlevel HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

C#.NET

ADO.NET OVERVIEW

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about

the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.
- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.

- **DataSets**. For storing, Remoting and programming against flat data, XML data and relational data.
- **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

Connections:

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

Commands:

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

DataReaders:

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

DATASETS AND DATAADAPTERS:

DataSets

The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with

one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

DATAADAPTERS (OLEDB/SQL)

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing

the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

SQL SERVER

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

SQL SERVER TABLES

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

PRIMARY KEY

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

RELATIONAL DATABASE

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

FOREIGN KEY

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

REFERENTIAL INTEGRITY

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

DATA ABSTRACTION

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

Physical level: This is the lowest level of abstraction at which one describes how the data are actually stored.

Conceptual Level: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

View level: This is the highest level of abstraction at which one describes only part of the database.

ADVANTAGES OF RDBMS

- Redundancy can be avoided
- Inconsistency can be eliminated
- Data can be Shared
- Standards can be enforced
- Security restrictions ca be applied
- Integrity can be maintained
- Conflicting requirements can be balanced
- Data independence can be achieved.

DISADVANTAGES OF DBMS

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

FEATURES OF SQL SERVER (RDBMS)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.
SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

- The row level lock manager

ENTERPRISE WIDE DATA SHARING

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

PORTABILITY

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database sever platform that meets the system requirements.

OPEN SYSTEMS

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server's open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server's Open architecture provides transparent access to data from other relational database and even non-relational database.

DISTRIBUTED DATA SHARING

SQL Server's networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

UNMATCHED PERFORMANCE

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

SOPHISTICATED CONCURRENCY CONTROL

Real World applications demand access to critical data. With most database Systems application becomes "contention bound" – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

NO I/O BOTTLENECKS

SQL Server's fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

SYSTEM DESIGN

6.1. INTRODUCTION

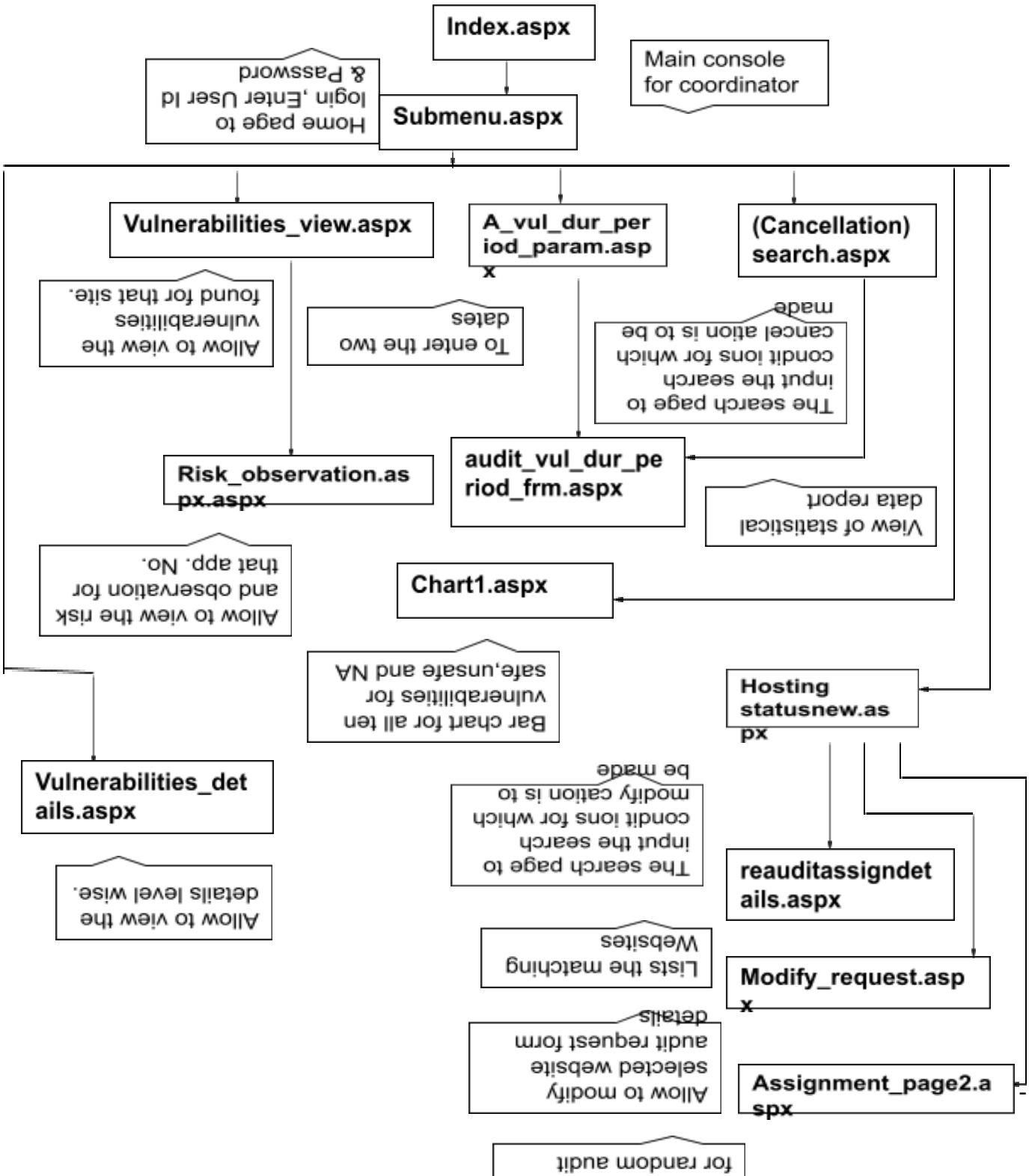
Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

6.2. SYSTEM WORKFLOW

Change the workflow based on your project



6.3. NORMALIZATION

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

Insertion anomaly: Inability to add data to the database due to absence of other data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data inconsistency resulting from data redundancy and partial update

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

FIRST NORMAL FORM:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

SECOND NORMAL FORM:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1) Primary key is a not a composite primary key
2) No non key attributes are present
3) Every non key attribute is fully functionally dependent on full set of primary key.

THIRD NORMAL FORM:

A relation is said to be in third normal form if their exits no transitive dependencies.

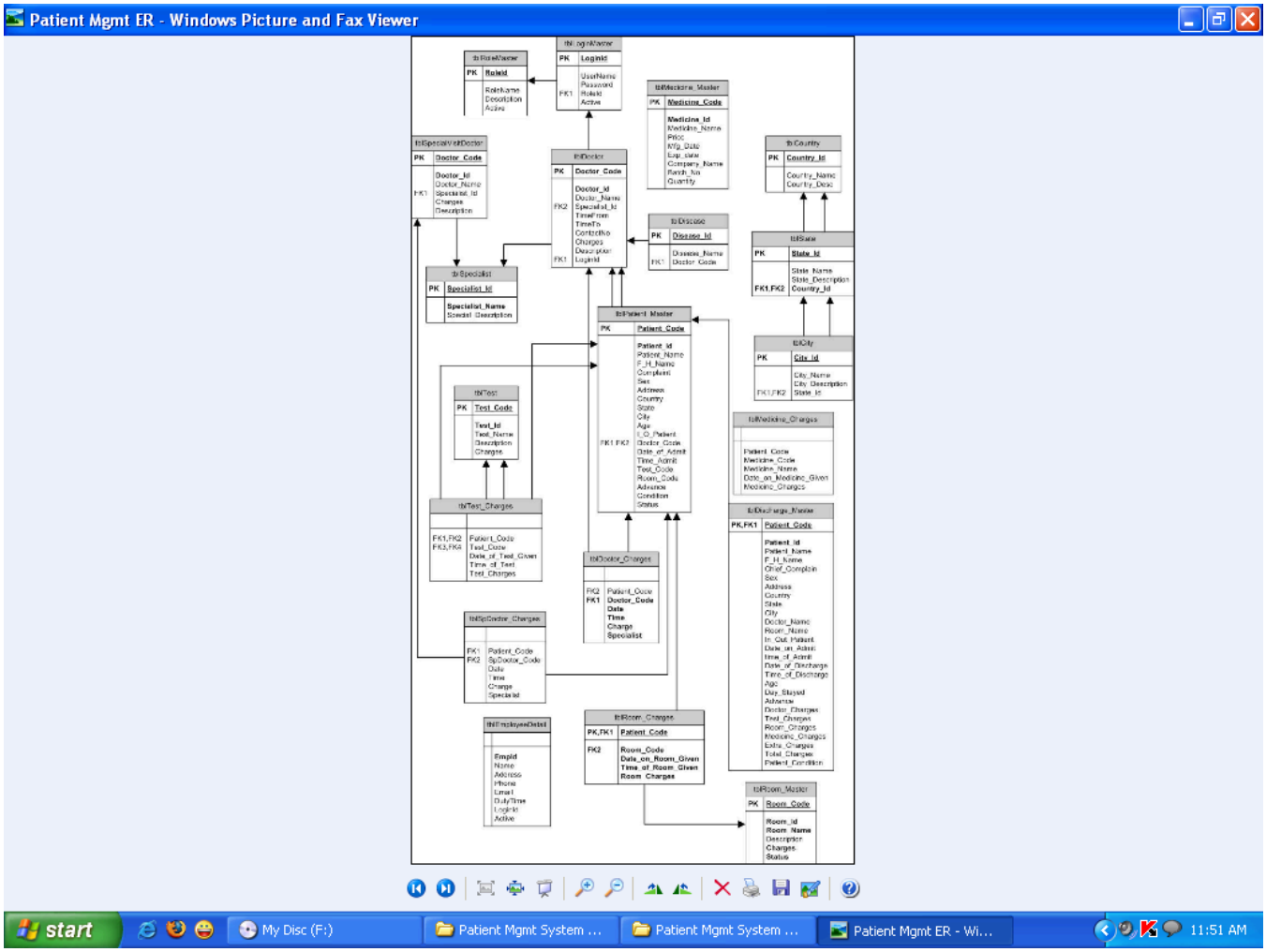
Transitive Dependency: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

6.4. E – R DIAGRAMS

- The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
- The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
- The set of primary components that are identified by the ERD are
  - ◆ Data object
  - ◆ Relationships
  - ◆ Attributes
  - ◆ Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.



6.4. DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow

diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

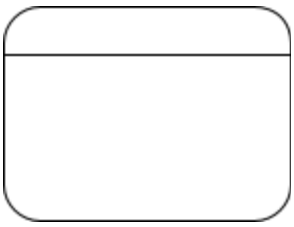
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SAILENT FEATURES OF DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD proecess label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as mush as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with he user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical

model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.
3) A process has a verb phrase label.

DATA STORE

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and /or destination of data.

1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

DATA FLOW

1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an

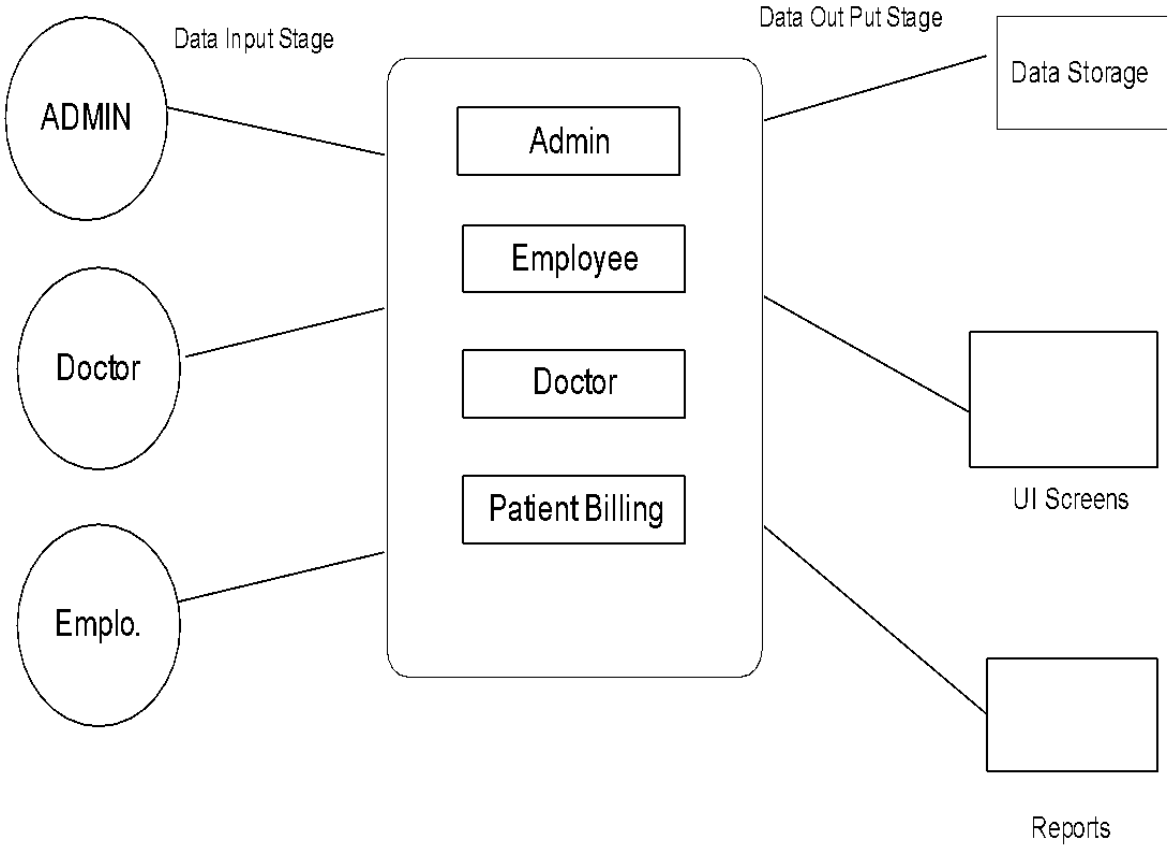
update. The later is usually indicated however by two separate arrows since these happen at different type.

2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

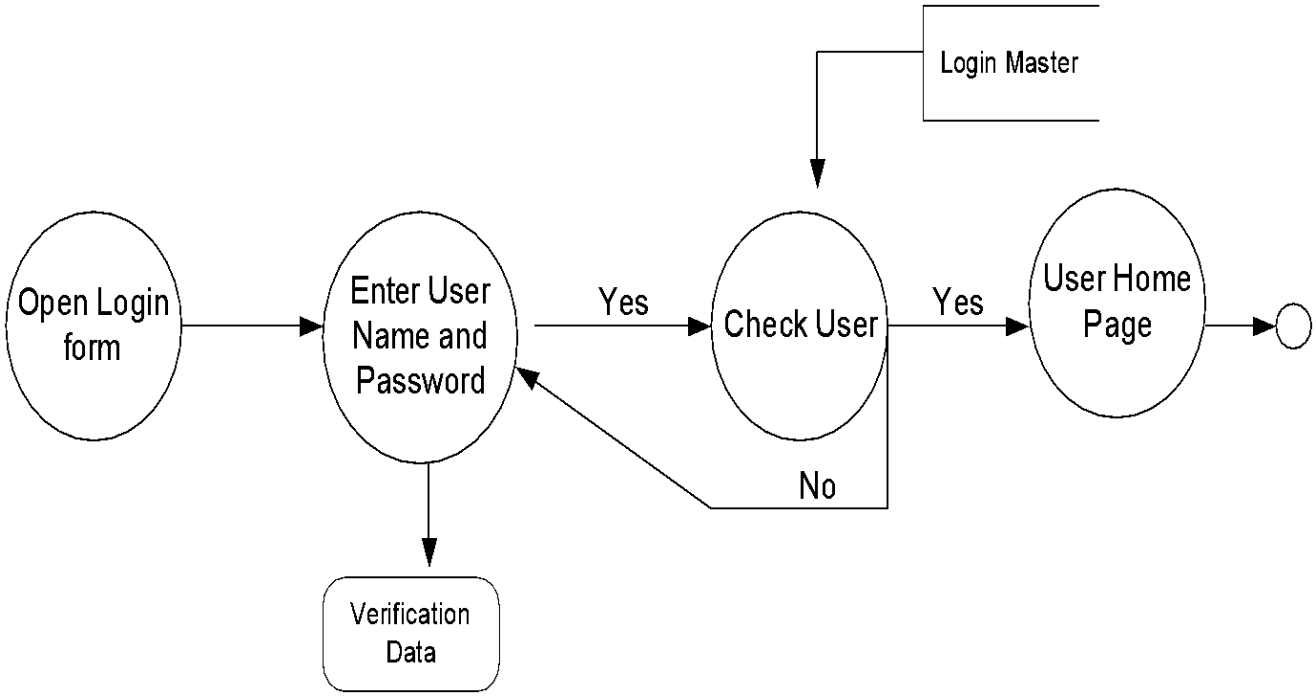
A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

Patient Management DFD's

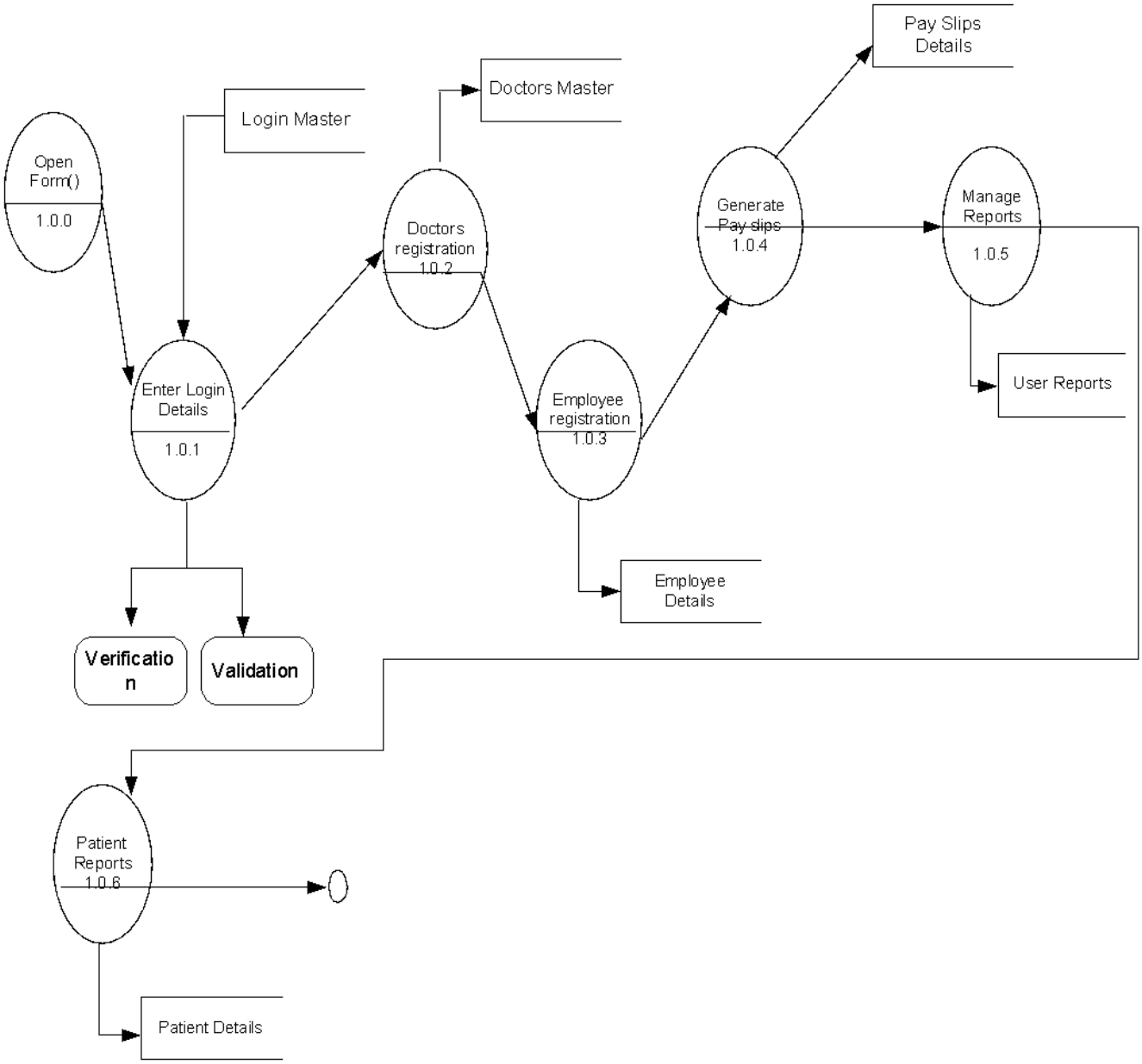
Context Level DFD



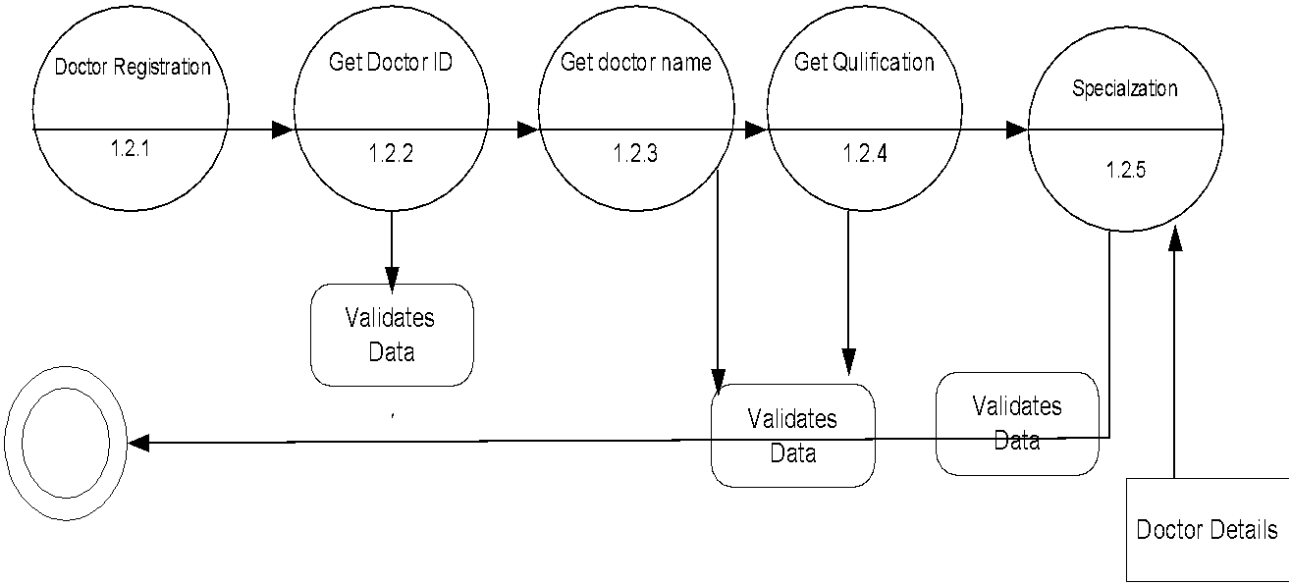
Login DFD


Login Master
Open Login form
Enter User Name and Password
Yes
Check User
Yes
User Home Page
No
Verification Data

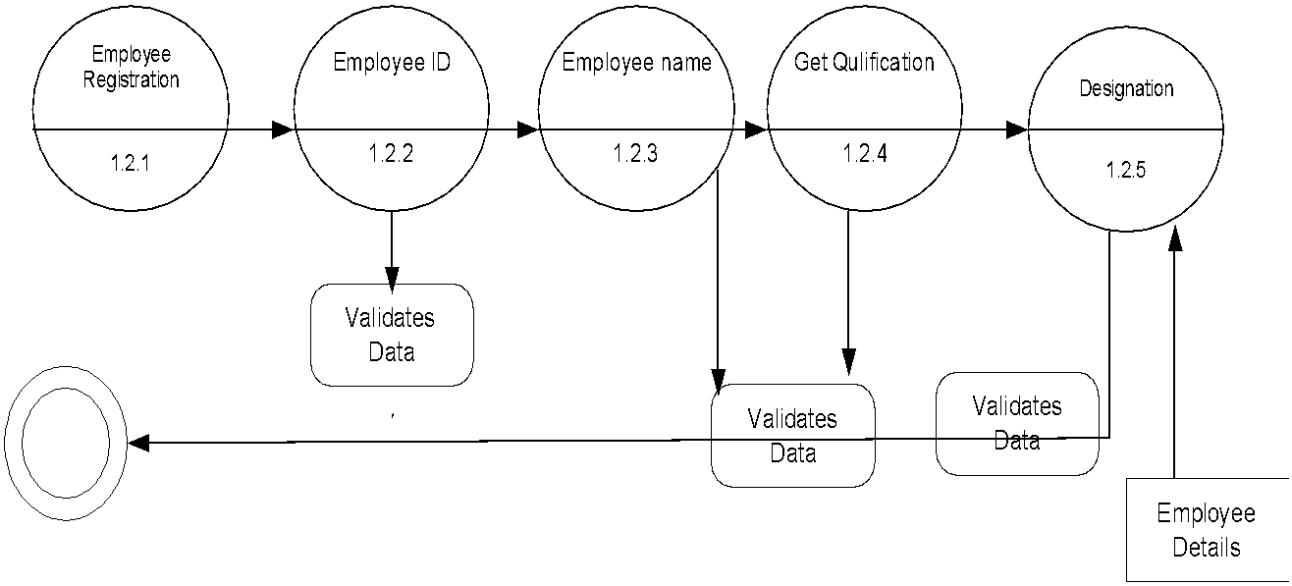
Admin Activities (1st Level)



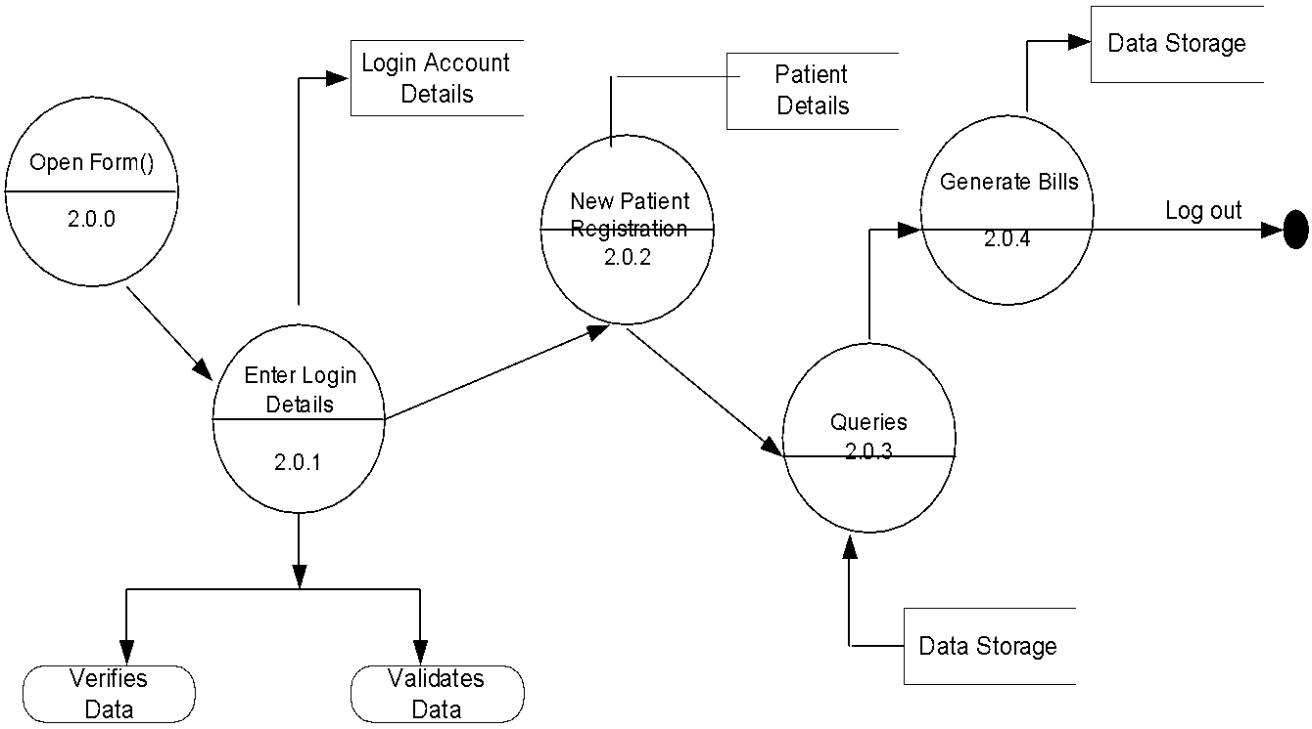
Admin Register Doctors



Admin Register Employee



User (Employee) Activities (2nd Level):


Login Account Details
Patient Details
Data Storage
Open Form()
2.0.0
New Patient Registration
2.0.2
Generate Bills
2.0.4
Log out
Enter Login Details
2.0.1
Queries
2.0.3
Verifies Data
Validates Data
Data Storage

6.6 UML DIAGRAMS:

UNIFIED MODELING LANGUAGE DIAGRAMS

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
- A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.
- User Model View
    i. This view represents the system from the users perspective.
    ii. The analysis representation describes a usage scenario from the end-users perspective.

Structural model view

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structures.

Behavioral Model View

- It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View

- In this the structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

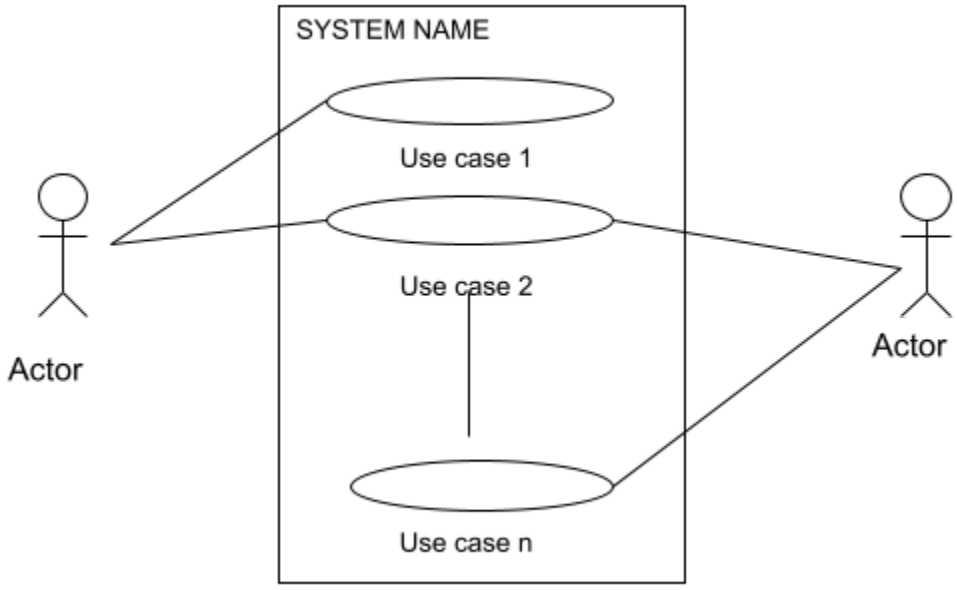
UML is specifically constructed through two different domains they are

- UML Analysis modeling, which focuses on the user model and structural model views of the system?
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

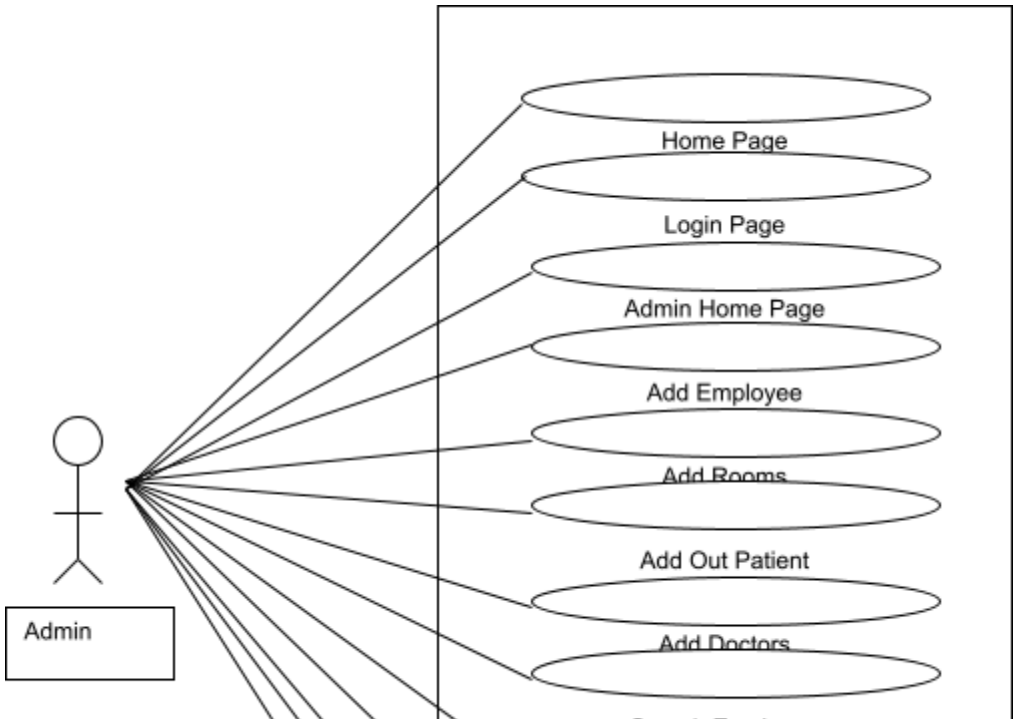
Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

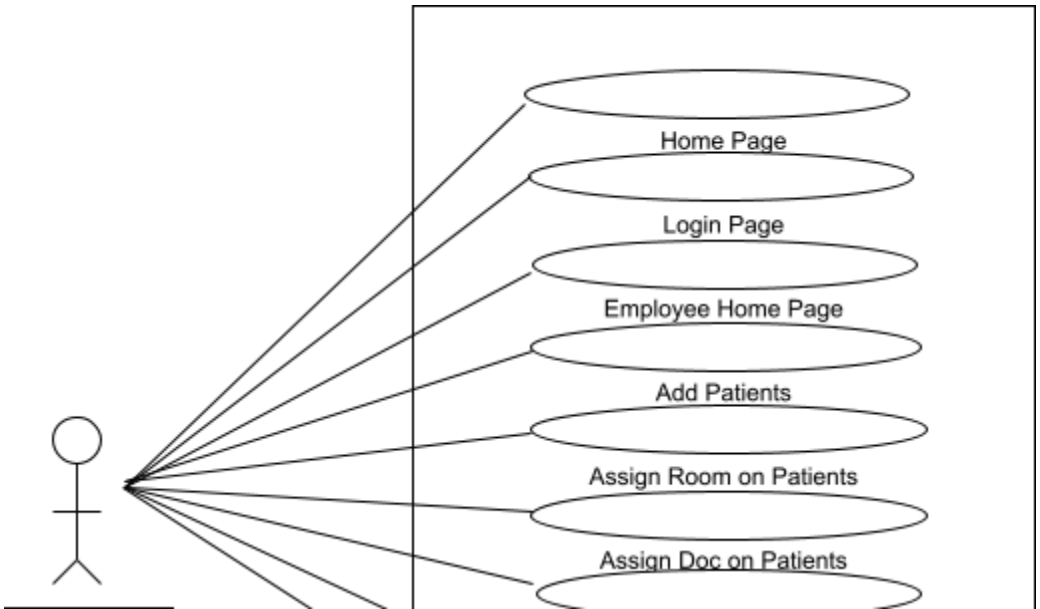
Use case Model



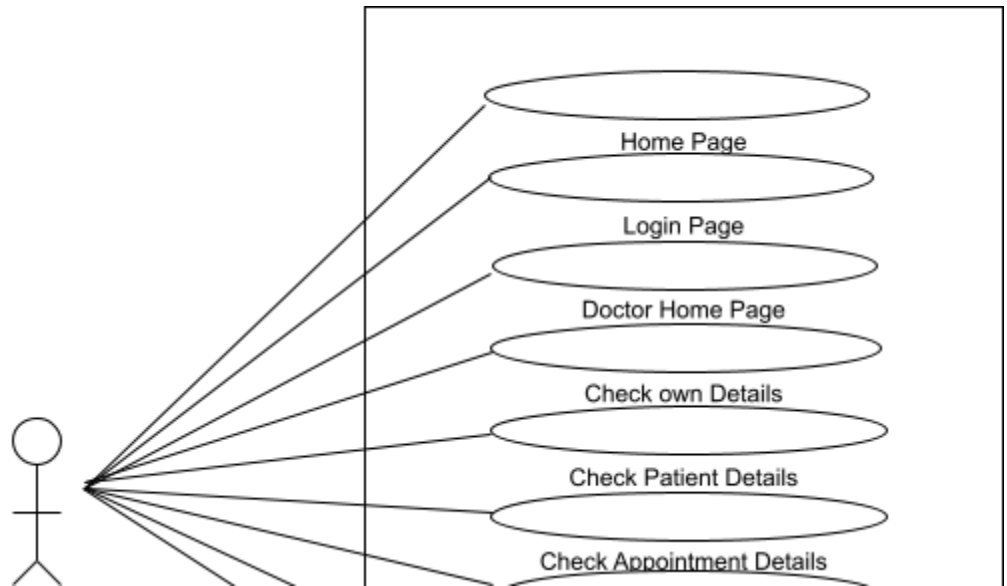
Use Cases of Admin Module



Use Cases of Employee Module



Use Cases of Doctor Module



SYSTEM CODING

SOURCE CODE:

Add Doctor on Patient:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for AddDoctorOnPatient
```

```
/// </summary>
public class AddDoctorOnPatient:Connection
{
   public static DataSet ds;
        public AddDoctorOnPatient()
        {
                //
                // TODO: Add constructor logic here
                //
        }
   string code, doccode, time,specialist;
   public string Code
   {
      get { return code; }
      set { code = value; }
   }
   public string Doccode
   {
      get { return doccode; }
      set { doccode = value; }
   }
   public string Time
   {
      get { return time; }
      set { time = value; }
   }
   public string Specialist
   {
      get { return specialist; }
      set { specialist = value; }
   }
   int charge;
   public int Charge
   {
      get { return charge; }
      set { charge = value; }
   }
   DateTime date;
   public DateTime Date
   {
      get { return date; }
      set { date = value; }
   }
   public void InsertDoctorCharge()
   {
      SqlParameter[] p = new SqlParameter[6];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@doccode", this.doccode);
      p[1].DbType = DbType.String;
      p[2] = new SqlParameter("@date", this.date);
      p[2].DbType = DbType.DateTime;
      p[3] = new SqlParameter("@time", this.time);
```

```
        p[3].DbType = DbType.String;
        p[4] = new SqlParameter("@charge", this.charge);
        p[4].DbType = DbType.Int32;
        p[5] = new SqlParameter("@specialist", this.specialist);
        p[5].DbType = DbType.String;
         SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertDoctorCharges",
p);

    }
    public DataSet ShowDoctorOnPatient()
    {
        ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowDoctorOnPatient");
        return ds;
    }
    public DataSet ShowPatientInfoByDoctor()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@drcode", this.doccode);
        p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientInfoByDoctor",p);
        return ds;
    }
    public DataSet ShowDoctorNameByCode()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@drcode", this.doccode);
        p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowDoctorNameByCode", p);
        return ds;
    }
}
```

Add Patient BL:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for AddPatientBL
```

```
/// </summary>
public class AddPatientBL:Connection
{
   public static DataSet ds;
        public AddPatientBL()
        {
                //
                // TODO: Add constructor logic here
                //
        }
    string code, name, hname, complaint, sex, address, country, state, city, iopatient, doctorcode,
testcode, roomcode, condition, admittime;
   public string Code
   {
      get { return code; }
      set { code = value; }
   }
   public string Name
   {
      get { return name; }
      set { name = value; }
   }
   public string Hname
   {
      get { return hname; }
      set { hname = value; }
   }
   public string Complaint
   {
      get { return complaint; }
      set { complaint = value; }
   }
   public string Sex
   {
      get { return sex; }
      set { sex = value; }
   }
   public string Address
   {
      get { return address; }
      set { address = value; }
   }
   public string Country
   {
      get { return country; }
      set { country = value; }
   }
   public string State
   {
      get { return state; }
      set { state = value; }
   }
   public string City
```

```
    {
        get { return city; }
        set { city = value; }
    }
    public string Iopatient
    {
        get { return iopatient; }
        set { iopatient = value; }
    }
    public string Doctorcode
    {
        get { return doctorcode; }
        set { doctorcode = value; }
    }
    public string Testcode
    {
        get { return testcode; }
        set { testcode = value; }
    }
    public string Roomcode
    {
        get { return roomcode; }
        set { roomcode = value; }
    }
    public string Condition
    {
        get { return condition; }
        set { condition = value; }
    }
    public string Admittime
    {
        get { return admittime; }
        set { admittime = value; }
    }
    int age, advance;
    public int Age
    {
        get { return age; }
        set { age = value; }
    }
    public int Advance
    {
        get { return advance; }
        set { advance = value; }
    }
DateTime admitdate,date1,date2;

    public DateTime Admitdate
    {
        get { return admitdate; }
        set { admitdate = value; }
    }
    public DateTime Date1
```

```
{
   get { return date1; }
   set { date1 = value; }
}
public DateTime Date2
{
   get { return date2; }
   set { date2 = value; }
}

public void AddPatient()
{
   SqlParameter[] p = new SqlParameter[18];
   p[0] = new SqlParameter("@code", this.code);
   p[0].DbType = DbType.String;
   p[1] = new SqlParameter("@name", this.name);
   p[1].DbType = DbType.String;
   p[2] = new SqlParameter("@hname", this.hname);
   p[2].DbType = DbType.String;
   p[3] = new SqlParameter("@complaint", this.complaint);
   p[3].DbType = DbType.String;
   p[4] = new SqlParameter("@sex", this.sex);
   p[4].DbType = DbType.String;
   p[5] = new SqlParameter("@address", this.address);
   p[5].DbType = DbType.String;
   p[6] = new SqlParameter("@country", this.country);
   p[6].DbType = DbType.String;
   p[7] = new SqlParameter("@state", this.state);
   p[7].DbType = DbType.String;
   p[8] = new SqlParameter("@city", this.city);
   p[8].DbType = DbType.String;
   p[9] = new SqlParameter("@age", this.age);
   p[9].DbType = DbType.Int16;
   p[10] = new SqlParameter("@iopatient", this.iopatient);
   p[10].DbType = DbType.String;
   p[11] = new SqlParameter("@doctorcode", this.doctorcode);
   p[11].DbType = DbType.String;
   p[12] = new SqlParameter("@admitdate", this.admitdate);
   p[12].DbType = DbType.DateTime;
   p[13] = new SqlParameter("@admittime", this.admittime);
   p[13].DbType = DbType.String;
   p[14] = new SqlParameter("@testcode", this.testcode);
   p[14].DbType = DbType.String;
   p[15] = new SqlParameter("@roomcode", this.roomcode);
   p[15].DbType = DbType.String;
   p[16] = new SqlParameter("@advance", this.advance);
   p[16].DbType = DbType.Int32;
   p[17] = new SqlParameter("@condition", this.condition);
   p[17].DbType = DbType.String;
   SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpAddPatient", p);
}
public DataSet ShowPatientInfo()
{
```

```
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
        ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfo",
p);
      return ds;
   }
   public DataSet ShowPatientCode()
   {
      ds = new DataSet();
      ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientCode");
      return ds;
   }
   public DataSet ShowDoctorChargeByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowDoctorChargeByCode", p);
      return ds;
   }
   public DataSet ShowTestChargeByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestChargeByCode", p);
      return ds;
   }
   public DataSet ShowRoomChargeByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowRoomChargeByCode", p);
      return ds;
   }
   public DataSet ShowMedicineChargeByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowMedicineChargeByCode", p);
      return ds;
```

```
    }
    public void UpdatePatient()
    {
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@code", this.code);
        p[0].DbType = DbType.String;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpUpdatePatient", p);
    }
    public DataSet ShowPatientInfoByDate()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@date", this.admitdate);
        p[0].DbType = DbType.DateTime;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientInfoByDate", p);
        return ds;
    }
    public DataSet ShowPatientInfoByDoctor()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@drcode", this.doctorcode);
        p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientInfoByDoctor", p);
        return ds;
    }
    public DataSet ShowPatientInfoBetweenDate()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@date1", this.date1);
        p[0].DbType = DbType.DateTime;
        p[1] = new SqlParameter("@date2", this.date2);
        p[1].DbType = DbType.DateTime;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientBetweenDate", p);
        return ds;
    }
    public DataSet ShowPatientCodeByStatus()
    {
        ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientCodeByStatus");
        return ds;
    }
    public void DeletePatientPermanently()
    {
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@code", this.code);
        p[0].DbType = DbType.String;
```

```
                                    SqlHelper.ExecuteNonQuery(con,     CommandType.StoredProcedure,
"SpDeletePatientPermanently", p);
    }

}
```

CITY BL:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using HospitalMgmt.DAL;
using System.Data.SqlClient;
/// <summary>
/// Summary description for CityBL
/// </summary>
public class CityBL:Connection
{
        public CityBL()
        {
                //
                // TODO: Add constructor logic here
                //
        }
    string cityName,cityDesc;
    int stateId;
    public static DataSet ds;


    public string CityName
    {
        get { return cityName; }
        set { cityName = value; }
    }

    public string CityDesc
    {
        get { return cityDesc; }
        set { cityDesc = value; }
    }

    int countryId;
    public int StateId
    {
        get { return stateId; }
        set { stateId = value; }
    }
    public  void InsertCity()
```

```
   {

      SqlParameter[] p = new SqlParameter[3];
      p[0] = new SqlParameter("@CityName", this.cityName);
      p[0].DbType =DbType.String;
      p[1] = new SqlParameter("@CityDesc", this.cityDesc);
      p[1].DbType =DbType.String;
      p[2] = new SqlParameter("@StateId", this.stateId);
      p[2].DbType = DbType.Int16;
      SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "AddCity", p);

   }
   public DataSet ShowAllCity()
   {
      DataSet ds = new DataSet();
      ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "ShowCity");
      return ds;
   }
   public DataSet GetCityByState()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("stateid", this.stateId);
      p[0].DbType = DbType.Int16;
        ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "spGetCityByStateId",
p);
      return ds;
   }
}


DISCARGE PATIENT BL:
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for DischargePatientBL
/// </summary>
public class DischargePatientBL:Connection
{
   public static DataSet ds;
        public DischargePatientBL()
        {
                //
                // TODO: Add constructor logic here
```

```
                //
        }
                                                                                    string
code,name,hname,complain,sex,address,country,state,city,doctorname,roomname,inout,admittime
;
    public string Code
    {
        get { return code; }
        set { code = value; }
    }
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    public string Hname
    {
        get { return hname; }
        set { hname = value; }
    }
    public string Complain
    {
        get { return complain; }
        set { complain = value; }
    }
    public string Sex
    {
        get { return sex; }
        set { sex = value; }
    }
    public string Address
    {
        get { return address; }
        set { address = value; }
    }
    public string Country
    {
        get { return country; }
        set { country = value; }
    }
    public string State
    {
        get { return state; }
        set { state = value; }
    }
    public string City
    {
        get { return city; }
        set { city = value; }
    }
    public string Doctorname
    {
        get { return doctorname; }
```

```
    set { doctorname = value; }
}
public string Roomname
{
    get { return roomname; }
    set
    {
        if (value.ToString() == "")
        {
            roomname = "";
        }
        else
        {
            roomname = value;
        }

    }
}
public string Inout
{
    get { return inout; }
    set { inout = value; }
}
public string Admittime
{
    get { return admittime; }
    set { admittime = value; }
}

string dischargetime, daystayed, condition;
public string Dischargetime
{
    get { return dischargetime; }
    set { dischargetime = value; }
}
public string Daystayed
{
    get { return daystayed; }
    set { daystayed = value; }
}
public string Condition
{
    get { return condition; }
    set { condition = value; }
}
DateTime admitdate,dischargedate;
public DateTime Admitdate
{
    get { return admitdate; }
    set { admitdate = value; }
}
public DateTime Dischargedate
{
```

```
    get { return dischargedate; }
    set { dischargedate = value; }
  }
  int age,advance,doctorcharges,testcharges,roomcharges,medicinecharge,extracharge,totcharge;
  public int Age
  {
    get { return age; }
    set {age = value; }
  }
  public int Advance
  {
    get { return advance; }

    set
    {
      if (value.ToString() == "")
      {
        advance = 0;

      }
      else
      {
        advance = value;
      }

    }
  }
  public int Doctorcharges
  {
    get
    {
      return doctorcharges;
    }
    set
    {
      if (value.ToString() == "")
      {
        doctorcharges = 0;
      }
      else
      {
        doctorcharges = value;
      }
    }
   }
  public int Testcharges
  {
    get
    {
      return testcharges;
    }
    set
    {
```

```
        if (value.ToString() == "")
        {
            testcharges = 0;
        }
        else
        {
            testcharges = value;
        }

    }
}
public int Roomcharges
{
    get
    {
        return roomcharges;
    }
    set
    {
        if (value.ToString() == "")
        {
            roomcharges = 0;
        }
        else
        {
            roomcharges = value;
        }

    }
}
public int Medicinecharge
{
    get { return medicinecharge; }
    set {

        if (value.ToString() == "")
        {
            medicinecharge =0;
        }
        else
        {
            medicinecharge = value;
        }
    }
}
public int Extracharge
{
    get { return extracharge; }
    set
    {
        if (value.ToString() == "")
        {
            extracharge = 0;
```

```
        }
        else
        {
            extracharge = value;
        }

    }
}
public int Totcharge
{
    get { return totcharge; }
    set
    {
        if (value.ToString() == "")
        {
            totcharge = 0;
        }
        else
        {
            totcharge = value;
        }

    }
}
public void InsertDischargePatient()
{
    SqlParameter[] p = new SqlParameter[26];
    p[0] = new SqlParameter("@code", this.code);
    p[0].DbType = DbType.String;
    p[1] = new SqlParameter("@name", this.name);
    p[1].DbType = DbType.String;
    p[2] = new SqlParameter("@hname", this.hname);
    p[2].DbType = DbType.String;
    p[3] = new SqlParameter("@complain", this.complain);
    p[3].DbType = DbType.String;
    p[4] = new SqlParameter("@sex", this.sex);
    p[4].DbType = DbType.String;
    p[5] = new SqlParameter("@address", this.address);
    p[5].DbType = DbType.String;
    p[6] = new SqlParameter("@country", this.country);
    p[6].DbType = DbType.String;
    p[7] = new SqlParameter("@state", this.state);
    p[7].DbType = DbType.String;
    p[8] = new SqlParameter("@city", this.city);
    p[8].DbType = DbType.String;
    p[9] = new SqlParameter("@doctorname", this.doctorname);
    p[9].DbType = DbType.String;
    p[10] = new SqlParameter("@roomname", this.roomname);
    p[10].DbType = DbType.String;
    p[11] = new SqlParameter("@inout", this.inout);
    p[11].DbType = DbType.String;
    p[12] = new SqlParameter("@admitdate", this.admitdate);
    p[12].DbType = DbType.DateTime;
```

```
        p[13] = new SqlParameter("@admittime", this.admittime);
        p[13].DbType = DbType.String;
        p[14] = new SqlParameter("@dischargedate", this.dischargedate);
        p[14].DbType = DbType.DateTime;
        p[15] = new SqlParameter("@dischargetime", this.dischargetime);
        p[15].DbType = DbType.String;
        p[16] = new SqlParameter("@age", this.age);
        p[16].DbType = DbType.Int16;
        p[17] = new SqlParameter("@daystayed", this.daystayed);
        p[17].DbType = DbType.String;
        p[18] = new SqlParameter("@advance", this.advance);
        p[18].DbType = DbType.Int32;
        p[19] = new SqlParameter("@doctorcharges", this.doctorcharges);
        p[19].DbType = DbType.Int32;
        p[20] = new SqlParameter("@testcharges", this.testcharges);
        p[20].DbType = DbType.Int32;
        p[21] = new SqlParameter("@roomcharges", this.roomcharges);
        p[21].DbType = DbType.Int32;
        p[22] = new SqlParameter("@medicinecharge", this.medicinecharge);
        p[22].DbType = DbType.Int32;
        p[23] = new SqlParameter("@extracharge", this.extracharge);
        p[23].DbType = DbType.Int32;
        p[24] = new SqlParameter("@totcharge", this.totcharge);
        p[24].DbType = DbType.Int32;
        p[25] = new SqlParameter("@condition", this.condition);
        p[25].DbType = DbType.String;
                                SqlHelper.ExecuteNonQuery(con,     CommandType.StoredProcedure,
"SpInsertDischargePatient", p);
    }
}
```

DOCTOR MASTER BL:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for DoctorMasterBL
/// </summary>
public class DoctorMasterBL:Connection
{
    public static DataSet ds;
        public DoctorMasterBL()
        {
                //
                // TODO: Add constructor logic here
```

```
                //
        }
    string code,name,desc,time1,time2,contactno,username,password;
     public string Username
     {
        get { return username; }
        set { username = value; }
     }
     public string Password
     {
        get { return password; }
        set { password = value; }
     }
     public string Time1
     {
        get { return time1; }
        set { time1 = value; }
     }
     public string Time2
     {
        get { return time2; }
        set { time2 = value; }
     }
     public string Contactno
     {
        get { return contactno; }
        set { contactno = value; }
     }
     public string Code
     { get { return code; }
        set { code = value; }
     }
     public string Name
     {
        get { return name; }
        set { name = value; }
     }
     public string Desc
     {
        get { return desc; }
        set { desc = value; }
     }
     int id, charge,roleid;
     public int Roleid
     {
        get { return roleid; }
        set { roleid = value; }
     }
     public int Id
     {
        get { return id; }
        set { id = value; }
     }
```

```
    public int Chrge
    {
        get { return charge; }
        set {charge = value; }
    }
    public void InsertDoctor()
    {
        SqlParameter[] p = new SqlParameter[11];
        p[0] = new SqlParameter("@code", this.code);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@name", this.name);
        p[1].DbType = DbType.String;
        p[2] = new SqlParameter("@id", this.id);
        p[2].DbType = DbType.Int16;
        p[3] = new SqlParameter("@time1", this.time1);
        p[3].DbType = DbType.String;
        p[4] = new SqlParameter("@time2", this.time2);
        p[4].DbType = DbType.String;
        p[5] = new SqlParameter("@contactno", this.contactno);
        p[5].DbType = DbType.String;
        p[6] = new SqlParameter("@charge", this.charge);
        p[6].DbType = DbType.Int32;
        p[7] = new SqlParameter("@desc", this.desc);
        p[7].DbType = DbType.String;
        p[8] = new SqlParameter("@uname", this.username);
        p[8].DbType = DbType.String;
        p[9] = new SqlParameter("@password", this.password);
        p[9].DbType = DbType.String;
        p[10] = new SqlParameter("@roleid", this.roleid);
        p[10].DbType = DbType.Int16;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertDoctor", p);
    }
    public DataSet ShowDoctor()
    {
        ds = new DataSet();
        ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctor");
        return ds;
    }
    public DataSet ShowDoctorByID()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@id", this.id);
        p[0].DbType = DbType.Int16;
            ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorByID",
p);
        return ds;
    }
     public void DeleteDoctor()
    {
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@id", this.id);
        p[0].DbType = DbType.Int16;
```

```
      SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpDeleteDoctor", p);
   }
   public DataSet ShowAllDoctor()
   {
      //SqlParameter[] p = new SqlParameter[1];
      //p[0] = new SqlParameter("@id", this.id);
      //p[0].DbType = DbType.Int16;
      ds = new DataSet();
      ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowAllDoctor");
      return ds;
   }
   public DataSet ShowDoctorByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowDoctorInfoByCode", p);
      return ds;
   }
   public void ModifyDoctorDetail()
   {
      SqlParameter[] p = new SqlParameter[8];
      p[0] = new SqlParameter("@doccode", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@name", this.name);
      p[1].DbType = DbType.String;
      p[2] = new SqlParameter("@id", this.id);
      p[2].DbType = DbType.Int16;
      p[3] = new SqlParameter("@time1", this.time1);
      p[3].DbType = DbType.String;
      p[4] = new SqlParameter("@time2", this.time2);
      p[4].DbType = DbType.String;
      p[5] = new SqlParameter("@contact", this.contactno);
      p[5].DbType = DbType.String;
      p[6] = new SqlParameter("@charge", this.charge);
      p[6].DbType = DbType.Int32;
      p[7] = new SqlParameter("@desc", this.desc);
      p[7].DbType = DbType.String;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpModifyDoctorDetail",
p);
   }
}
```

LOGIN INFO BL:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for LoginInfoBL
/// </summary>
public class LoginInfoBL:Connection
{
   public static DataSet ds;
        public LoginInfoBL()
        {
                //
                // TODO: Add constructor logic here
                //
        }
   private string name, password;
   public string Name
   {
      get { return name; }
      set { name = value; }
   }
   public string Password
   {
      get { return password; }
      set { password = value; }
   }
   DateTime date;
   public DateTime Date
   {
      get { return date; }
      set { date = value; }
   }
   public bool CheckAdmininfo()
   {
      SqlParameter[] p = new SqlParameter[2];
      p[0] = new SqlParameter("@name", this.name);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@password", this.password);
      p[1].DbType = DbType.String;
      int count;

count=int.Parse(SqlHelper.ExecuteScalar(con,CommandType.StoredProcedure,"SpCheckAdminInfo"
,p).ToString());
      if (count > 0)
      {
         return true;
      }
      else
      {
         return false;
      }
```

```
    }
    public bool CheckEmployeeinfo()
    {
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@name", this.name);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        int count;
                count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.StoredProcedure,
"SpCheckEmployeeInfo", p).ToString());
        if (count > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public bool CheckDoctorInfo()
    {
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@name", this.name);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        int count;
                count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.StoredProcedure,
"SpCheckDoctorInfo", p).ToString());
        if (count > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public DataSet ShowPatientByDoctor()
    {
        SqlParameter[] p = new SqlParameter[3];
        p[0] = new SqlParameter("@uname", this.name);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        p[2] = new SqlParameter("@date", this.date);
        p[2].DbType = DbType.Date;
        ds = new DataSet();
         ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorByLogin",
p);
        return ds;
    }
```

```
    public DataSet ShowPatientByDoctor1()
    {
        SqlParameter[] p = new SqlParameter[3];
        p[0] = new SqlParameter("@uname", this.name);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        p[2] = new SqlParameter("@date", this.date);
        p[2].DbType = DbType.Date;
        ds = new DataSet();
                        ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure,
"SpShowDoctorByLogin1", p);
        return ds;
    }
    public void ChangeDoctorPassword()
    {
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@uname", this.name);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
                                SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure,
"SpChangeDoctorPassword", p);

   }
    public void ChangeEmployeePassword()
   {
       SqlParameter[] p = new SqlParameter[2];
       p[0] = new SqlParameter("@uname", this.name);
       p[0].DbType = DbType.String;
       p[1] = new SqlParameter("@password", this.password);
       p[1].DbType = DbType.String;
                                    SqlHelper.ExecuteNonQuery(con,CommandType.StoredProcedure,
"SpChangeEmployeePassword", p);

   }
    public DataSet ShowDoctorInfo()
   {
       SqlParameter[] p = new SqlParameter[2];
       p[0] = new SqlParameter("@uname", this.name);
       p[0].DbType = DbType.String;
       p[1] = new SqlParameter("@password", this.password);
       p[1].DbType = DbType.String;
       ds = new DataSet();
       ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorInfo", p);
       return ds;
   }
}

MEDICINE CHARGE BL:
using System;
using System.Data;
using System.Configuration;
```

```
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for MedicineChargeBL
/// </summary>
public class MedicineChargeBL:Connection
{
   public static DataSet ds;
        public MedicineChargeBL()
        {
                //
                // TODO: Add constructor logic here
                //
        }
   string code, medicinecode, name;
   public string Code
   {
      get { return code; }
      set { code = value; }
   }
   public string Medicinecode
   {
      get { return medicinecode; }
      set { medicinecode = value; }
   }
   public string Name
   {
      get { return name; }
      set { name = value; }
   }
   int charge;
   public int Charge
   {
      get { return charge; }
      set { charge = value; }
   }
   DateTime date;
   public DateTime Date
   {
      get { return date; }
      set { date = value; }
   }
   public void InsertMedicineCharge()
   {
      SqlParameter[] p = new SqlParameter[5];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
```

```
      p[1] = new SqlParameter("@medicinecode", this.medicinecode);
      p[1].DbType = DbType.String;
      p[2] = new SqlParameter("@name", this.name);
      p[2].DbType = DbType.String;
      p[3] = new SqlParameter("@date", this.date);
      p[3].DbType = DbType.DateTime;
      p[4] = new SqlParameter("@charge", this.charge);
      p[4].DbType = DbType.Int32;
                                SqlHelper.ExecuteNonQuery(con,     CommandType.StoredProcedure,
"SpInsertMedicineCharges", p);

   }
   public DataSet ShowMedicineByDate()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[2];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@date", this.date);
      p[1].DbType = DbType.DateTime;
                              ds=SqlHelper.ExecuteDataset(con,     CommandType.StoredProcedure,
"SpShowMedicineByDate", p);
      return ds;
   }
   public DataSet ShowMedicineCode()
   {
      ds = new DataSet();
                          ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowMedicineCode");
      return ds;
   }
   public DataSet ShowpatientnameByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
                          ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowpatientnameByCode", p);
      return ds;
   }
   public DataSet ShowMedicineByPatientCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
                          ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowMedicineByPatientCode", p);
      return ds;
   }
   public DataSet ShowAllMedicineInfoByCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
```

```
        p[0] = new SqlParameter("@medcode", this.medicinecode);
                                ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowAllMedicineInfoByCode", p);
        return ds;
    }
}
ROOM CHARGE:
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for RoomChargeBL
/// </summary>
public class RoomChargeBL:Connection
{
    public static DataSet ds;
          public RoomChargeBL()
          {
                  //
                  // TODO: Add constructor logic here
                  //
          }
   string code,roomcode,time;
    public string Code
    {
       get { return code; }
       set { code = value; }
    }
    public string Roomcode
    {
       get { return roomcode; }
       set { roomcode = value; }
    }
    public string Time
    {
       get { return time; }
       set { time = value; }
    }
   int charge;
    public int Charge
    {
       get { return charge; }
       set { charge = value; }
    }
    DateTime date;
```

```
   public DateTime Date
   {
      get { return date; }
      set { date = value; }
   }
   public void InsertRoomCharge()
   {
      SqlParameter[] p = new SqlParameter[5];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@roomcode", this.roomcode);
      p[1].DbType = DbType.String;
      p[2] = new SqlParameter("@date", this.date);
      p[2].DbType = DbType.DateTime;
      p[3] = new SqlParameter("@time", this.time);
      p[3].DbType = DbType.String;
      p[4] = new SqlParameter("@charge", this.charge);
      p[4].DbType = DbType.Int32;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertRoomCharges",
p);

   }
   public DataSet ShowPatientRoomCode()
   {
      ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientRoomCode");
      return ds;
   }
   public DataSet ShowRoomNameByCode()
   {
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.roomcode);
      p[0].DbType = DbType.String;
      ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowRoomNameByCode",p);
      return ds;
   }
   public DataSet ShowPatientInfoByRoomCode()
   {
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.roomcode);
      p[0].DbType = DbType.String;
      ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientInfoByRoomCode", p);
      return ds;
   }
    public void UpdateRoomMaster()
   {
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
```

```
      p[0].DbType = DbType.String;
         SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpUpdateRoomMaster",
p);

   }
}
```

TEST CHARGE BL:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using HospitalMgmt.DAL;
/// <summary>
/// Summary description for TestChargeBL
/// </summary>
public class TestChargeBL:Connection
{
   public static DataSet ds;
        public TestChargeBL()
        {
                //
                // TODO: Add constructor logic here
                //
        }
  string code,testcode,time;
  public string Code
  {
     get { return code; }
     set { code = value; }
  }
  public string Testcode
  {
     get { return testcode; }
     set { testcode = value; }
  }
  public string Time
  {
     get { return time; }
     set { time = value; }
  }
   DateTime date, date1, date2;
   public DateTime Date
   {
      get { return date; }
      set { date = value; }
   }
```

```
    public DateTime Date1
    {
        get { return date1; }
        set { date1 = value; }
    }
    public DateTime Date2
    {
        get { return date2; }
        set { date2 = value; }
    }
    int charge;
    public int Charge
    {
        get { return charge; }
        set { charge = value; }
    }
    public void InsertTestCharge()
    {
        SqlParameter[] p = new SqlParameter[5];
        p[0] = new SqlParameter("@code", this.code);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@testcode", this.testcode);
        p[1].DbType = DbType.String;
        p[2] = new SqlParameter("@date", this.date);
        p[2].DbType = DbType.DateTime;
        p[3] = new SqlParameter("@time", this.time);
        p[3].DbType = DbType.String;
        p[4] = new SqlParameter("@charge", this.charge);
        p[4].DbType = DbType.Int32;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertTestCharges", p);

    }
    public DataSet ShowTestInfoByDate()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[1];
        p[0] = new SqlParameter("@date", this.date);
        p[0].DbType = DbType.DateTime;
                                ds=SqlHelper.ExecuteDataset(con,     CommandType.StoredProcedure,
"SpShowTestInfoByDate", p);
        return ds;
    }
    public DataSet ShowPatientCodeBYTest()
    {
        ds = new DataSet();
                            ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowPatientCodeBYTest");
        return ds;
    }
    public DataSet ShowTestInfoByDatePatient()
    {
        ds = new DataSet();
        SqlParameter[] p = new SqlParameter[2];
```

```
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@date", this.date);
      p[1].DbType = DbType.DateTime;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestInfoByDatePatient", p);
      return ds;
   }
   public DataSet ShowTestInfoByOnlyDate()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@date", this.date);
      p[0].DbType = DbType.DateTime;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestInfoByOnlyDate", p);
      return ds;
   }
   public DataSet ShowTestCodeOnPatient()
   {
      ds = new DataSet();
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestCodeOnPatient");
      return ds;
   }
   public DataSet ShowTestCodeByPateintCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestCodeByPateintCode", p);
      return ds;
   }
   public DataSet ShowTestNameByTestCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[1];
      p[0] = new SqlParameter("@code", this.testcode);
      p[0].DbType = DbType.String;
                        ds    =    SqlHelper.ExecuteDataset(con,    CommandType.StoredProcedure,
"SpShowTestNameByTestCode", p);
      return ds;
   }
   public DataSet ShowTestInfoByTestPatientCode()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[2];
      p[0] = new SqlParameter("@code", this.code);
      p[0].DbType = DbType.String;
      p[1] = new SqlParameter("@testcode", this.testcode);
      p[1].DbType = DbType.String;
```

```
                        ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure,
"SpShowTestInfoByTestPatientCode", p);
      return ds;
   }
   public DataSet SpShowTestBetweenDate()
   {
      ds = new DataSet();
      SqlParameter[] p = new SqlParameter[2];
      p[0] = new SqlParameter("@date1", this.date1);
      p[0].DbType = DbType.DateTime;
      p[1] = new SqlParameter("@date2", this.date2);
      p[1].DbType = DbType.DateTime;
                              ds=SqlHelper.ExecuteDataset(con    CommandType.StoredProcedure,
"SpShowTestBetweenDate", p);
      return ds;
   }
}
```

SYSTEM TESTING AND IMPLEMENTATION

8.1. INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

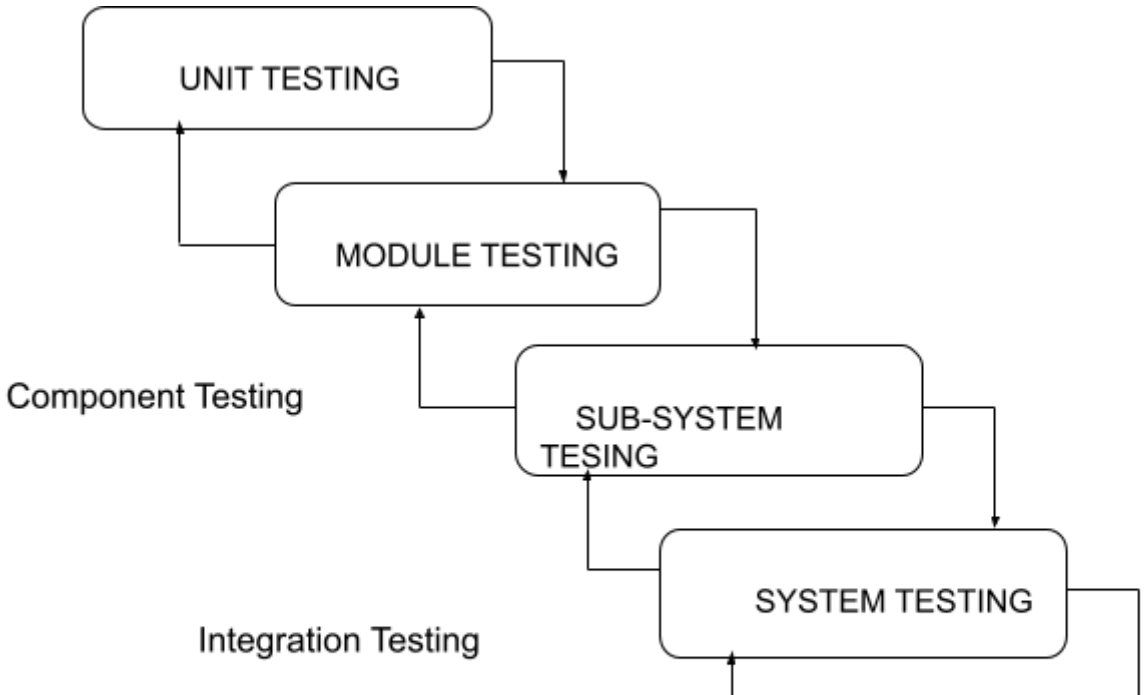
A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm

software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

8.2. STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.



8.3. Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

1. WHITE BOX TESTING

This type of testing ensures that

- All independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

2. BASIC PATH TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:
Use the design of the code and draw correspondent flow graph.
Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or
V(G)=P+1 or
V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

3. CONDITIONAL TESTING

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

4. DATA FLOW TESTING

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

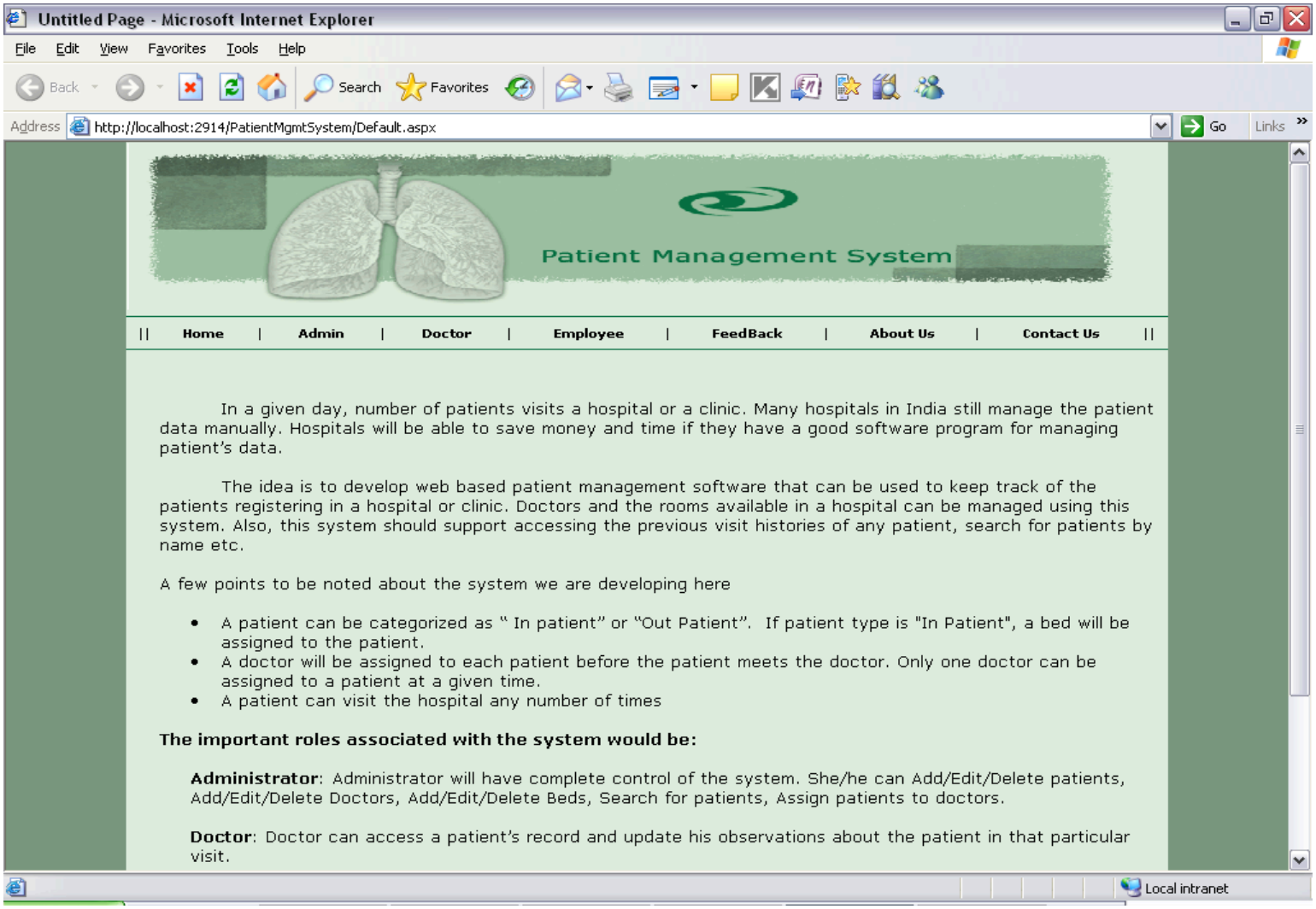
5. LOOP TESTING

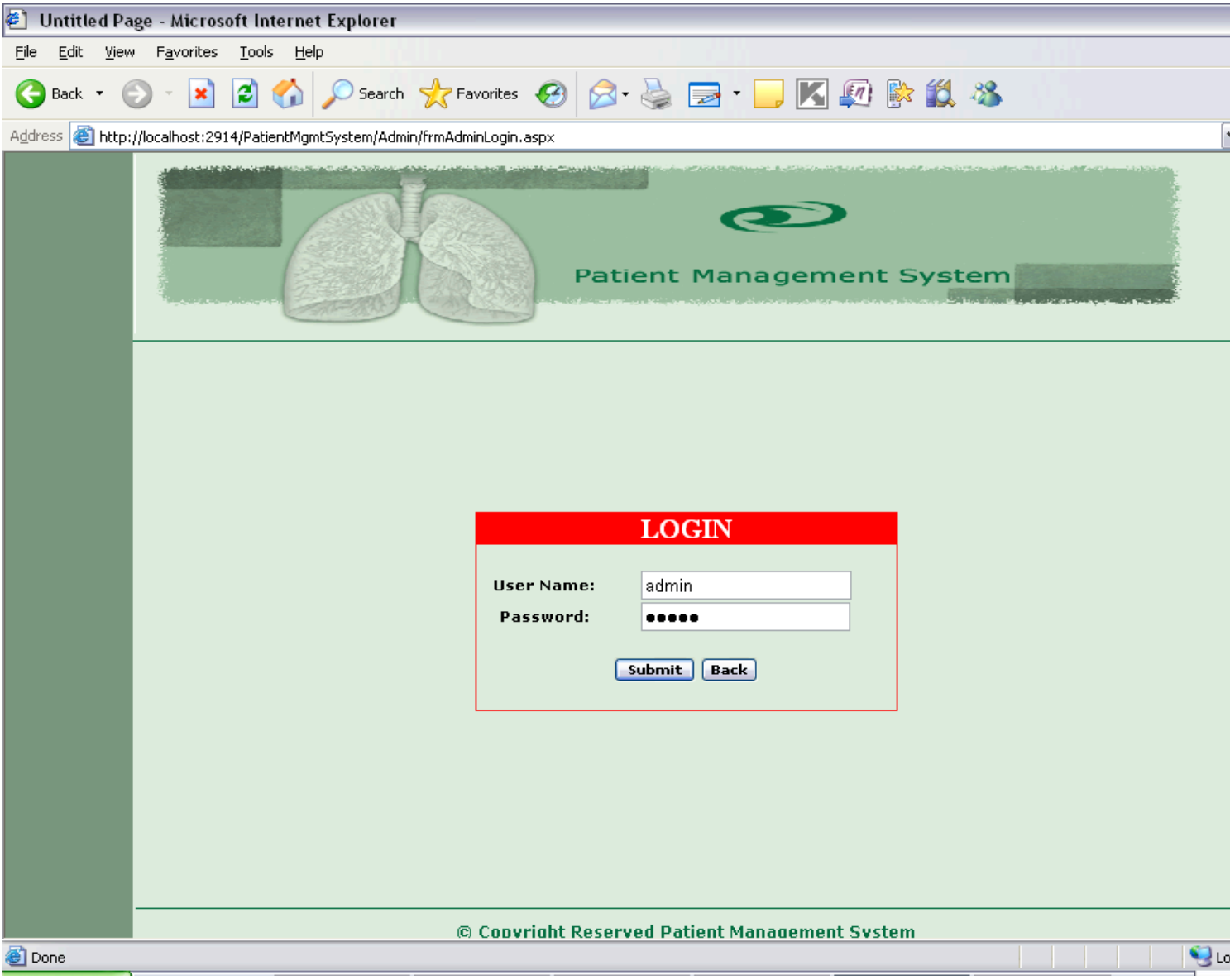
In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

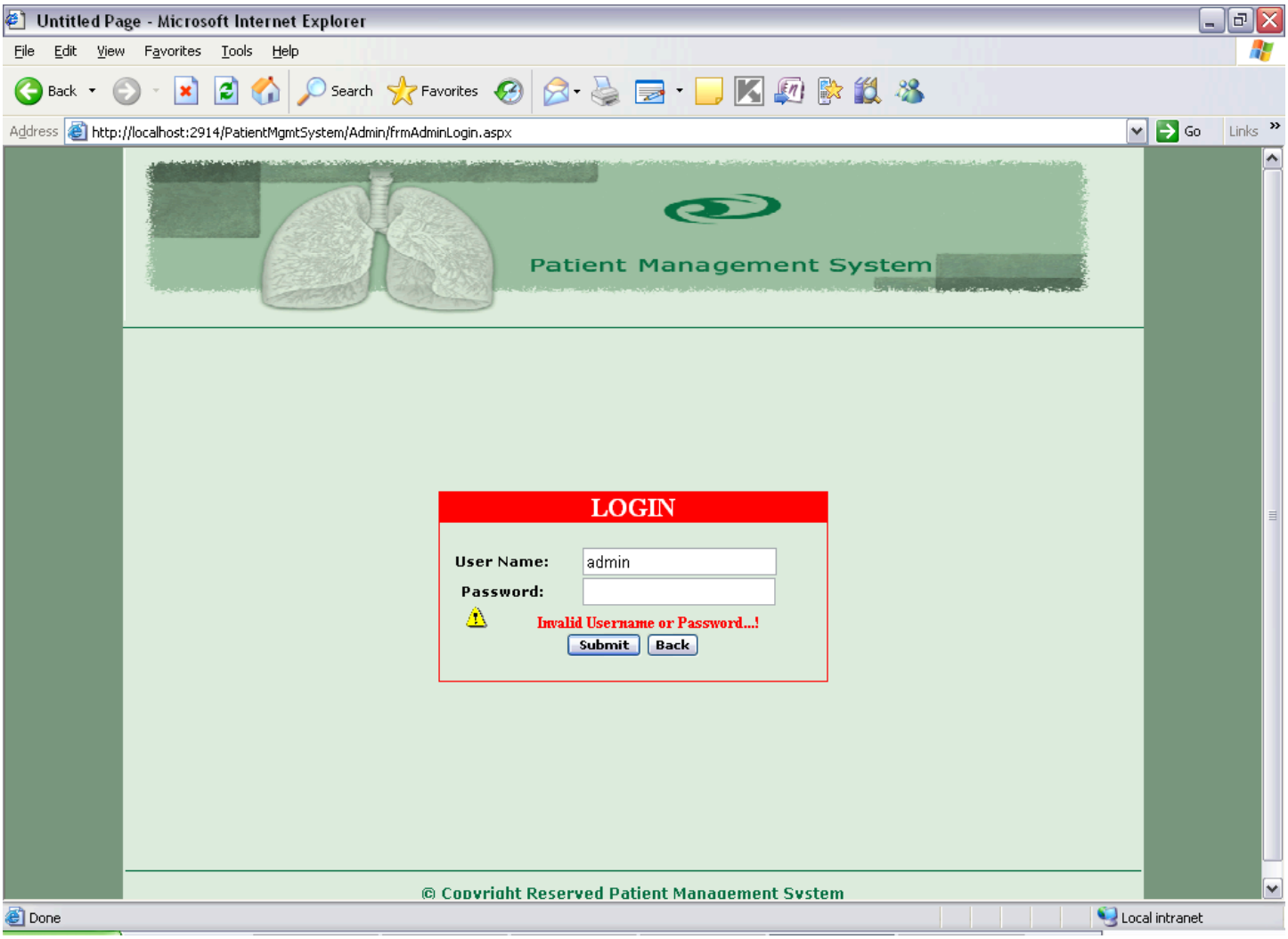
- All the loops were tested at their limits, just above them and just below them.
- All the loops were skipped at least once.
- For nested loops test the inner most loop first and then work outwards.
- For concatenated loops the values of dependent loops were set with the help of connected loop.
- Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

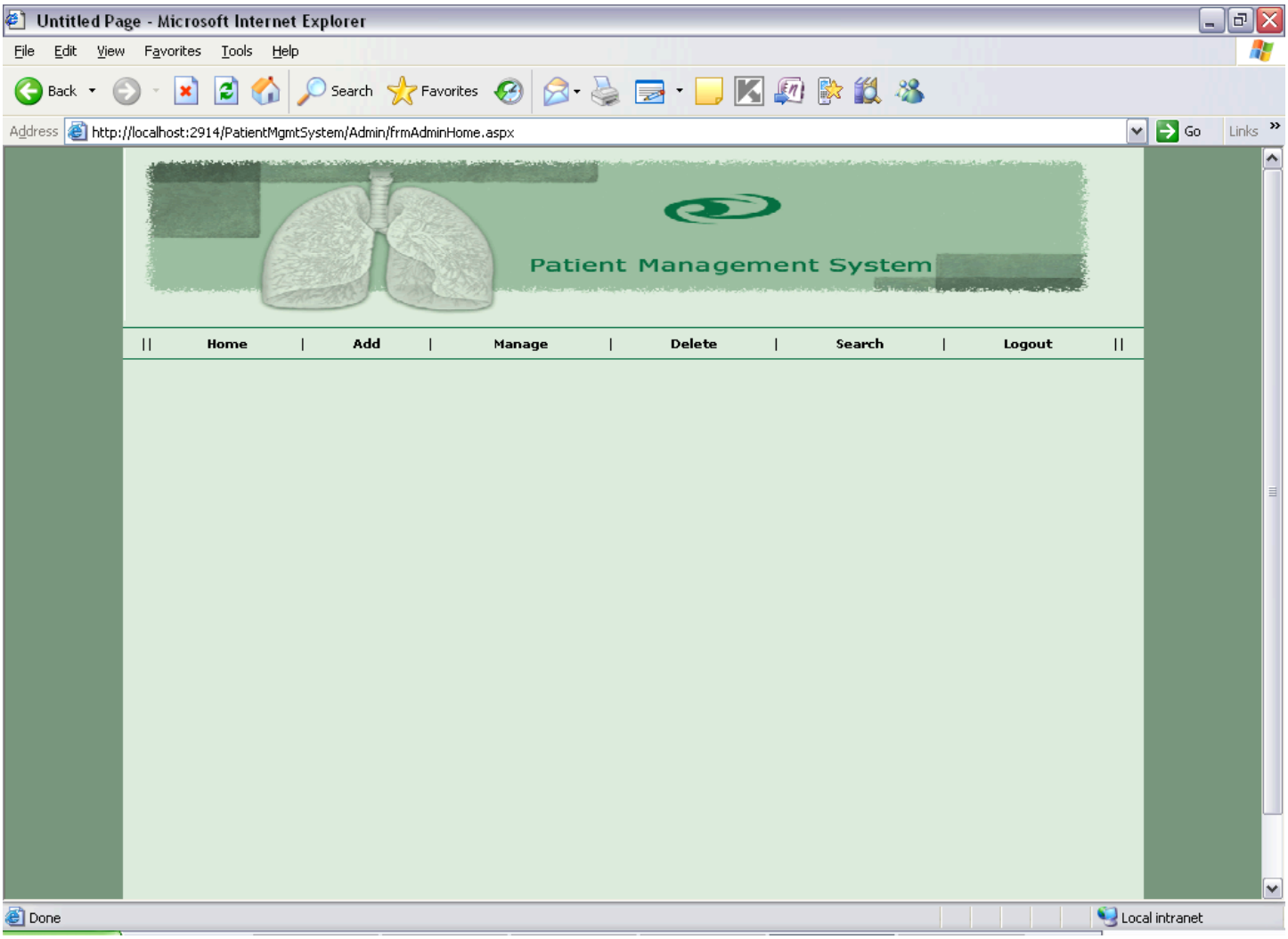
Each unit has been separately tested by the development team itself and all the input have been validated.

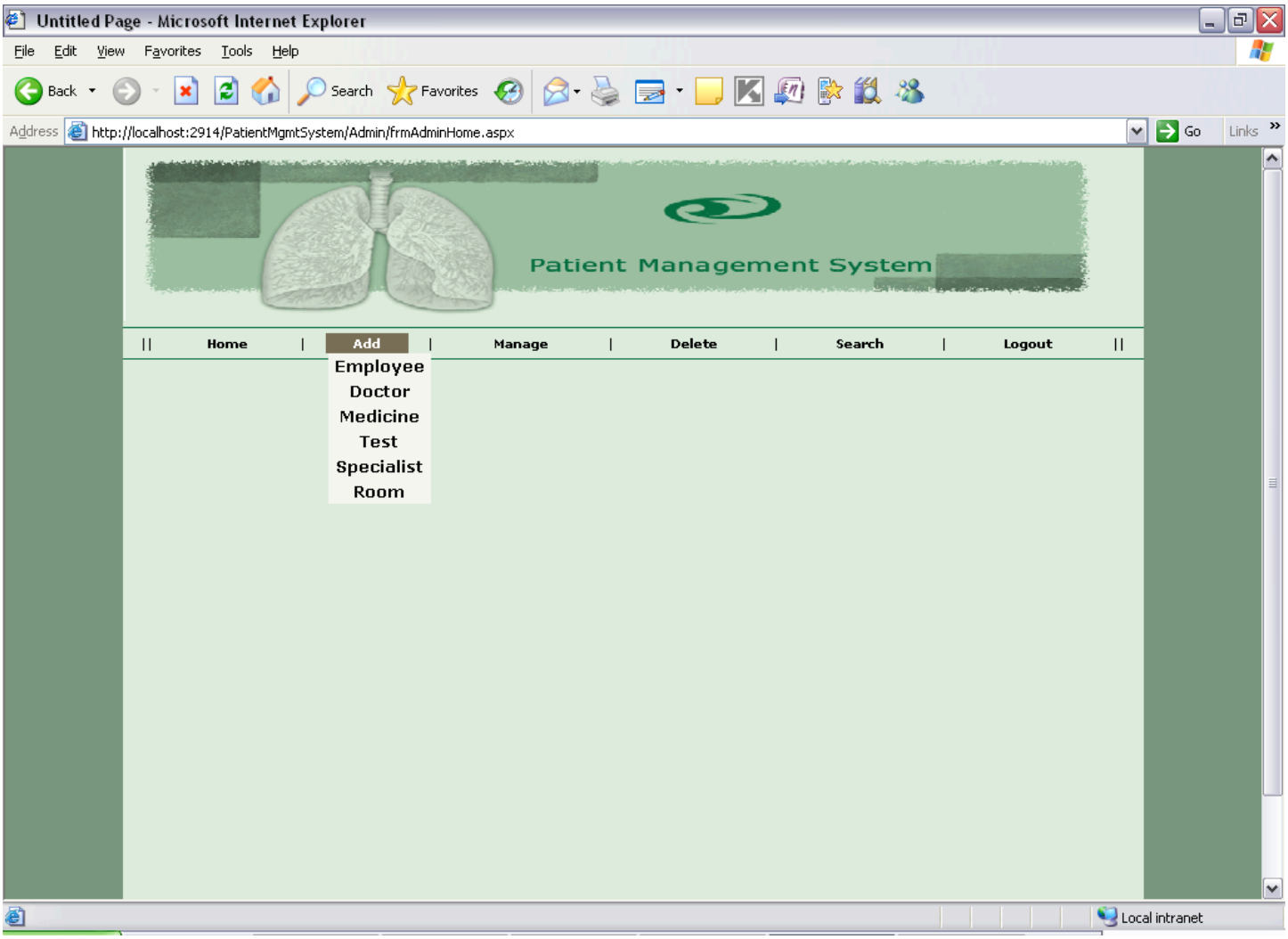
SCREENS

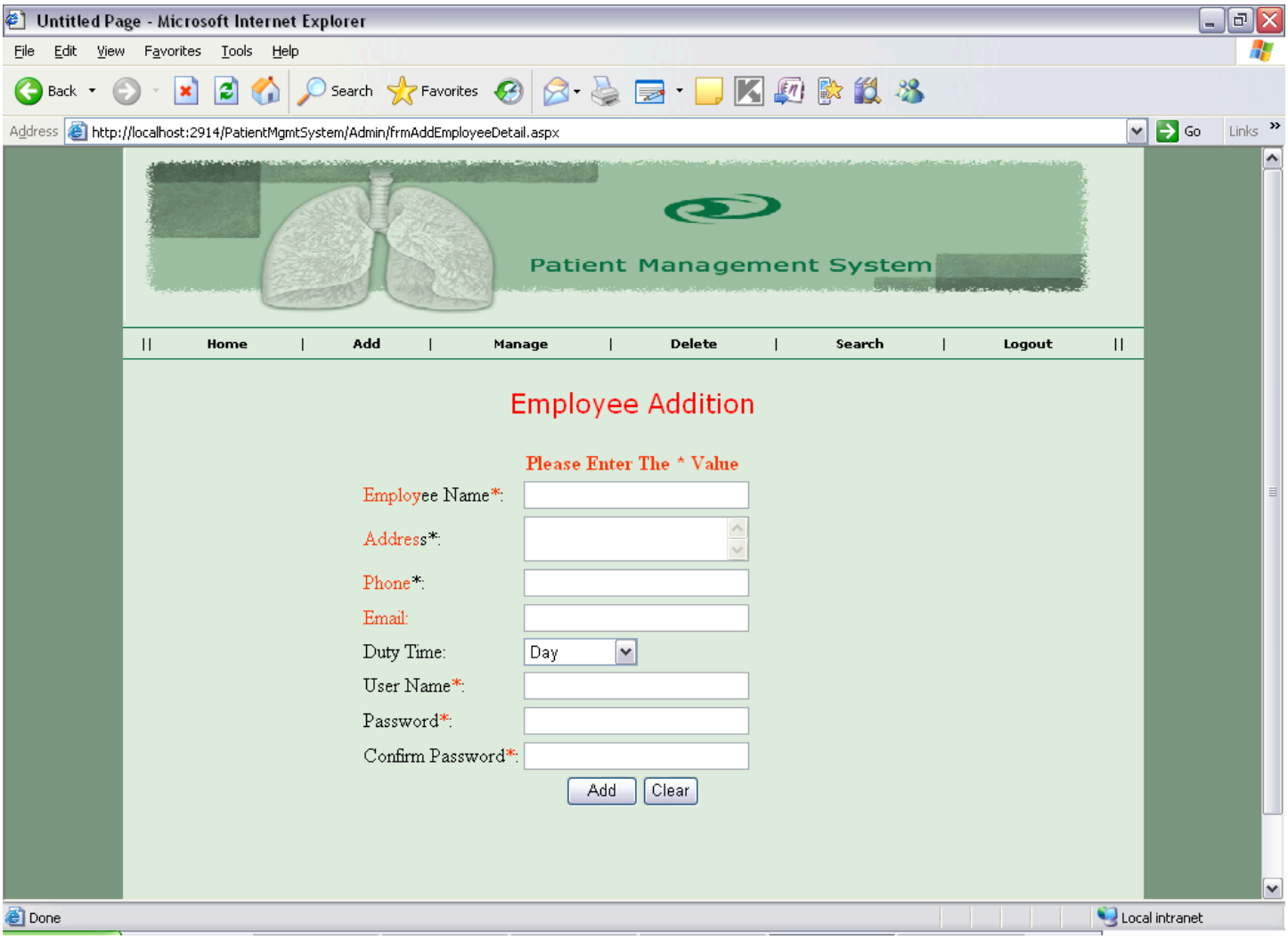

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Default.aspx
Go Links
Patient Management System
|| Home | Admin | Doctor | Employee | FeedBack | About Us | Contact Us ||
In a given day, number of patients visits a hospital or a clinic. Many hospitals in India still manage the patient data manually. Hospitals will be able to save money and time if they have a good software program for managing patient's data.
The idea is to develop web based patient management software that can be used to keep track of the patients registering in a hospital or clinic. Doctors and the rooms available in a hospital can be managed using this system. Also, this system should support accessing the previous visit histories of any patient, search for patients by name etc.
A few points to be noted about the system we are developing here
A patient can be categorized as " In patient" or "Out Patient". If patient type is "In Patient", a bed will be assigned to the patient.
A doctor will be assigned to each patient before the patient meets the doctor. Only one doctor can be assigned to a patient at a given time.
A patient can visit the hospital any number of times
The important roles associated with the system would be:
Administrator: Administrator will have complete control of the system. She/he can Add/Edit/Delete patients, Add/Edit/Delete Doctors, Add/Edit/Delete Beds, Search for patients, Assign patients to doctors.
Doctor: Doctor can access a patient's record and update his observations about the patient in that particular visit.
Local intranet

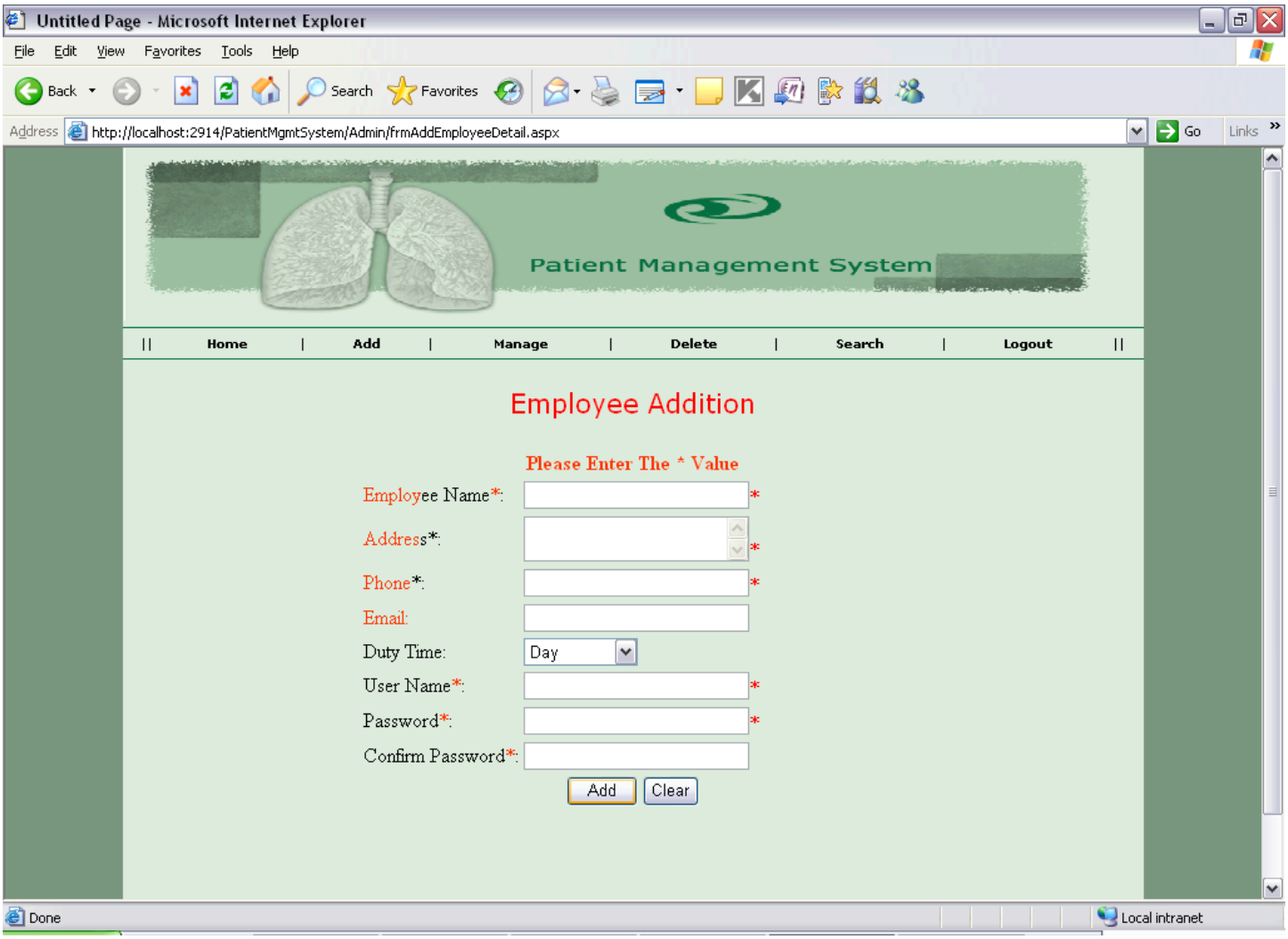

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminLogin.aspx
Patient Management System
LOGIN
User Name: admin
Password:
Submit Back
© Copyright Reserved Patient Management System
Done

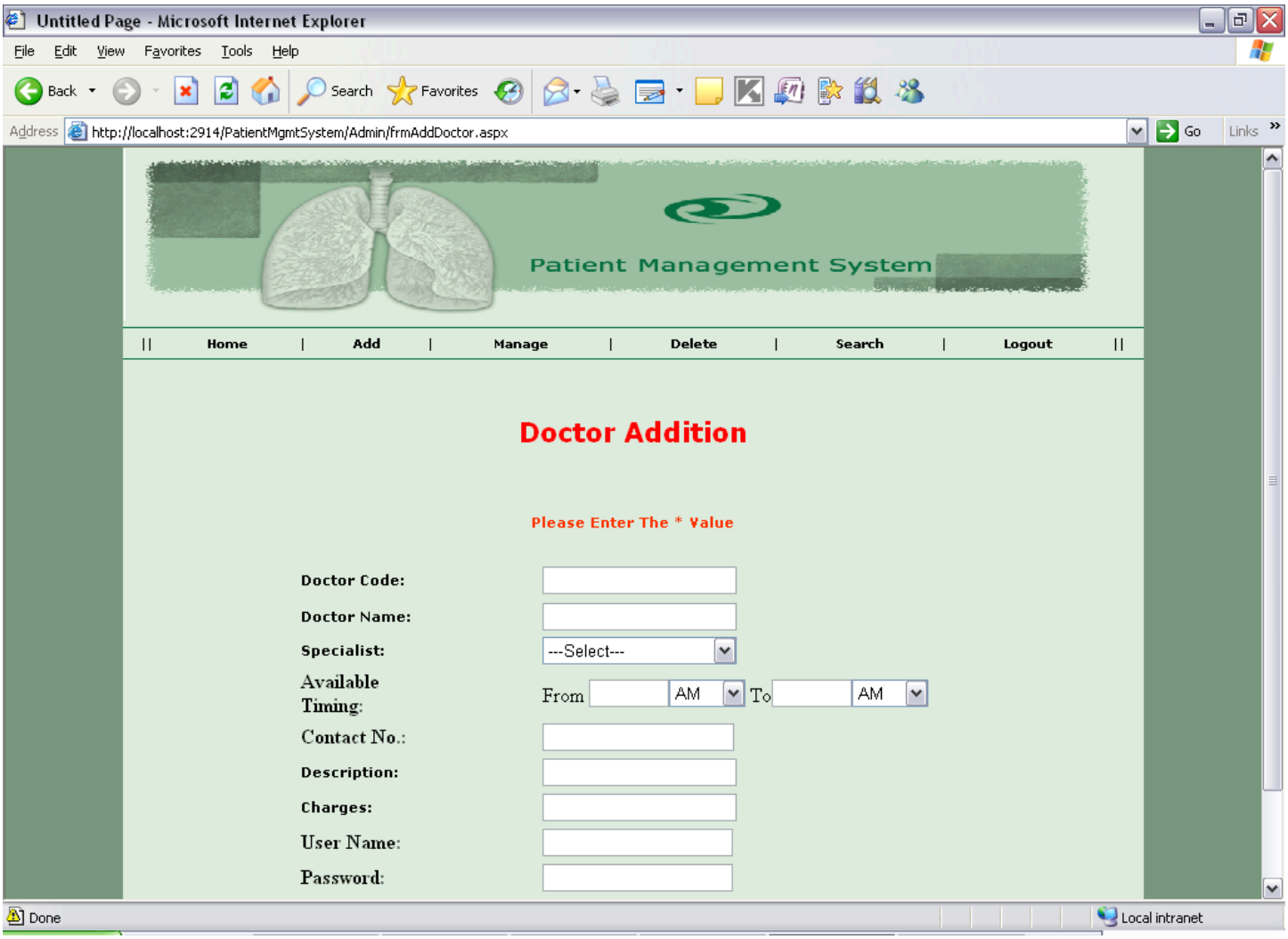

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminLogin.aspx
Go Links
Patient Management System
LOGIN
User Name: admin
Password:
Invalid Username or Password...!
Submit Back
© Copyright Reserved Patient Management System
Done
Local intranet

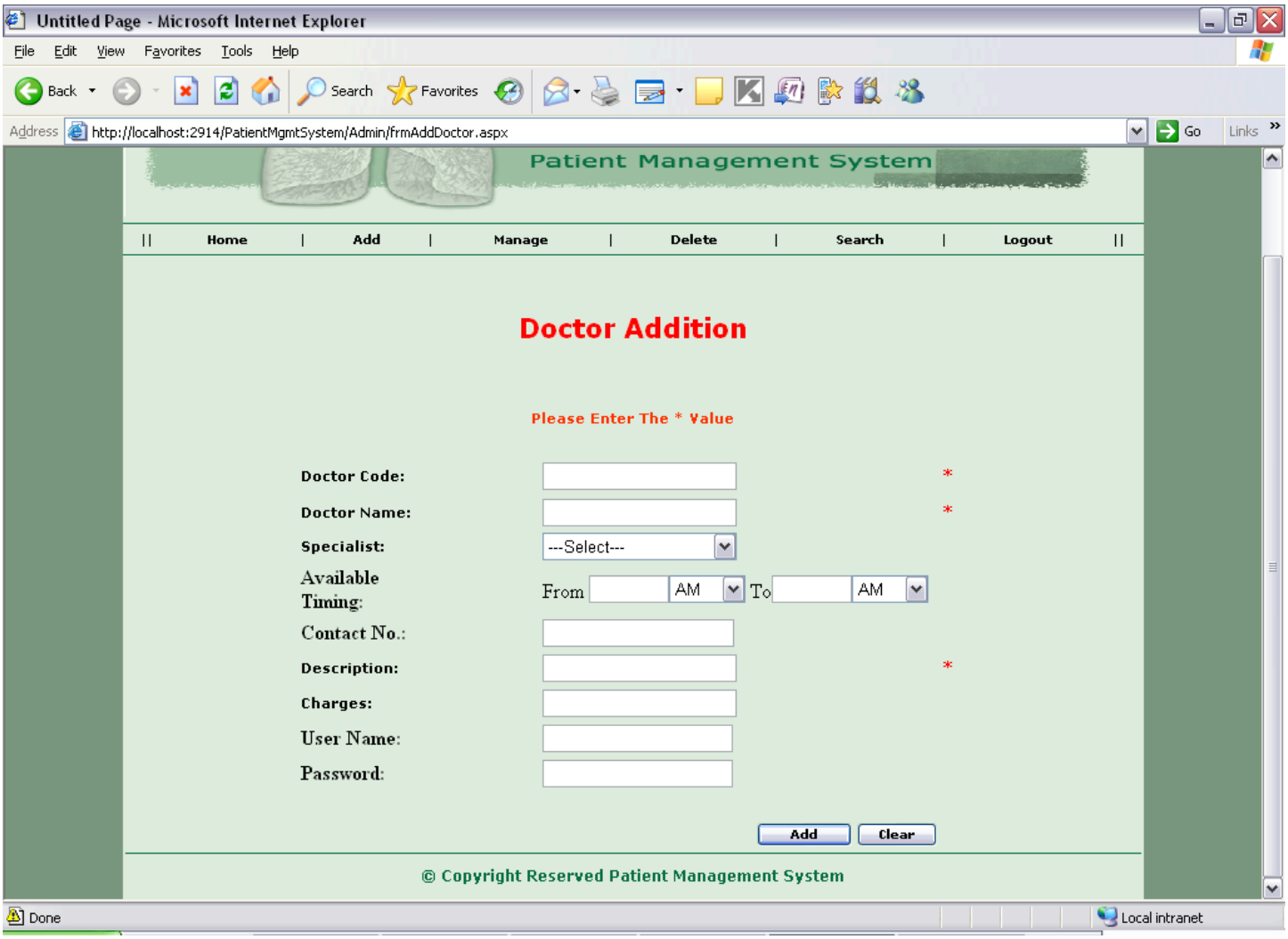

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Done
Local intranet

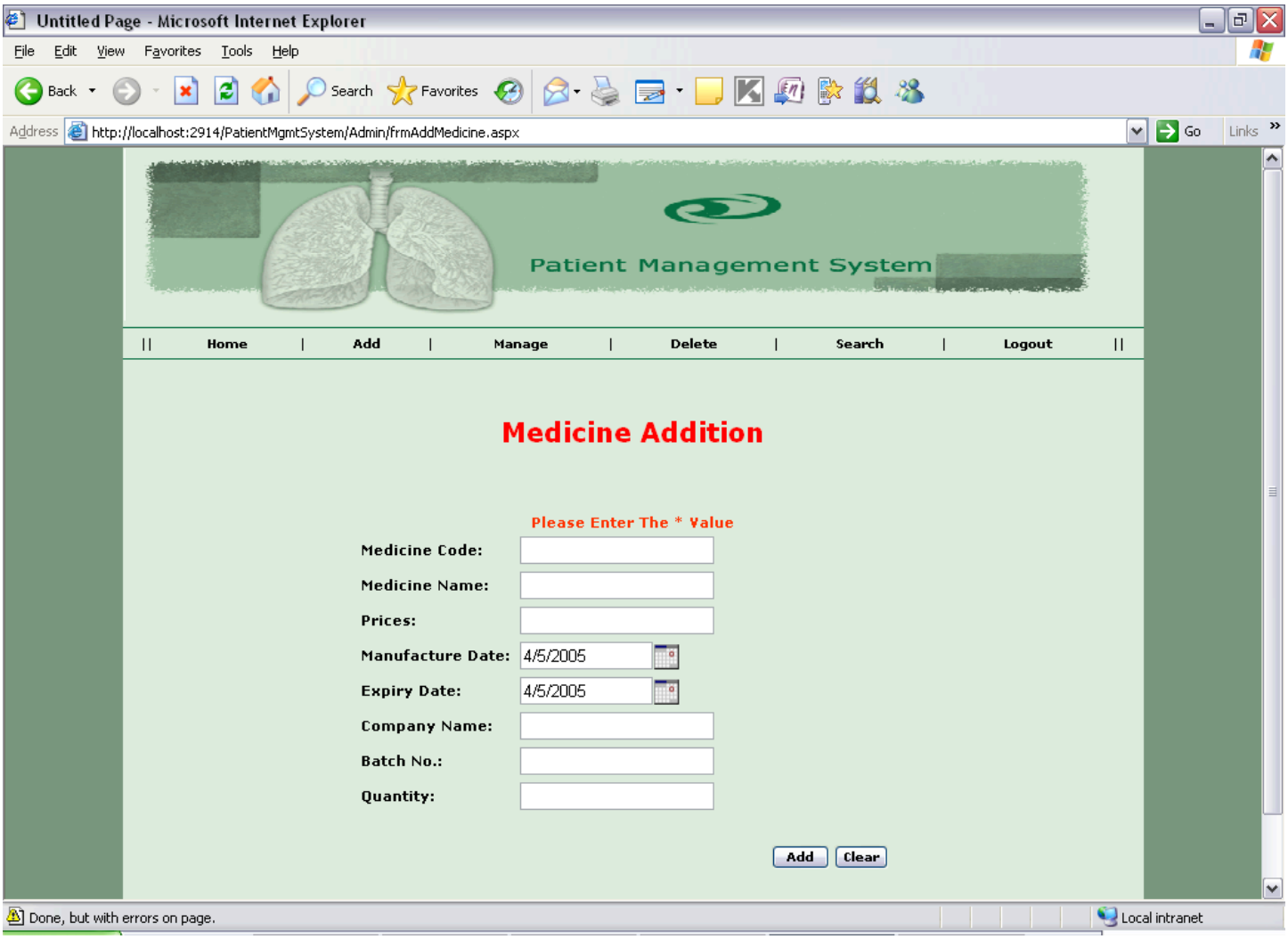

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Employee
Doctor
Medicine
Test
Specialist
Room
Local intranet

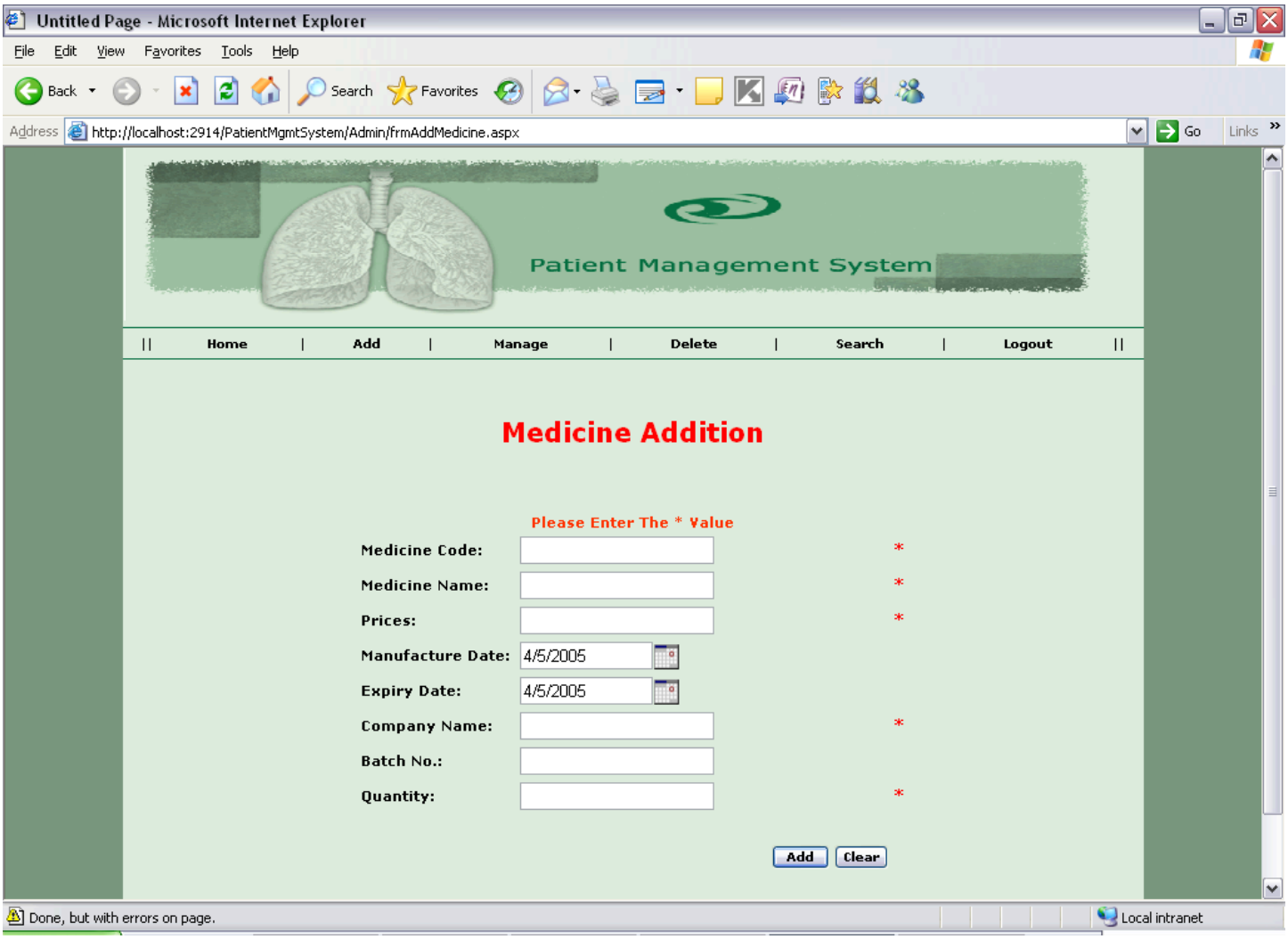

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddEmployeeDetail.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Employee Addition
Please Enter The * Value
Employee Name*:
Address*:
Phone*:
Email:
Duty Time:
Day
User Name*:
Password*:
Confirm Password*:
Add
Clear
Done
Local intranet

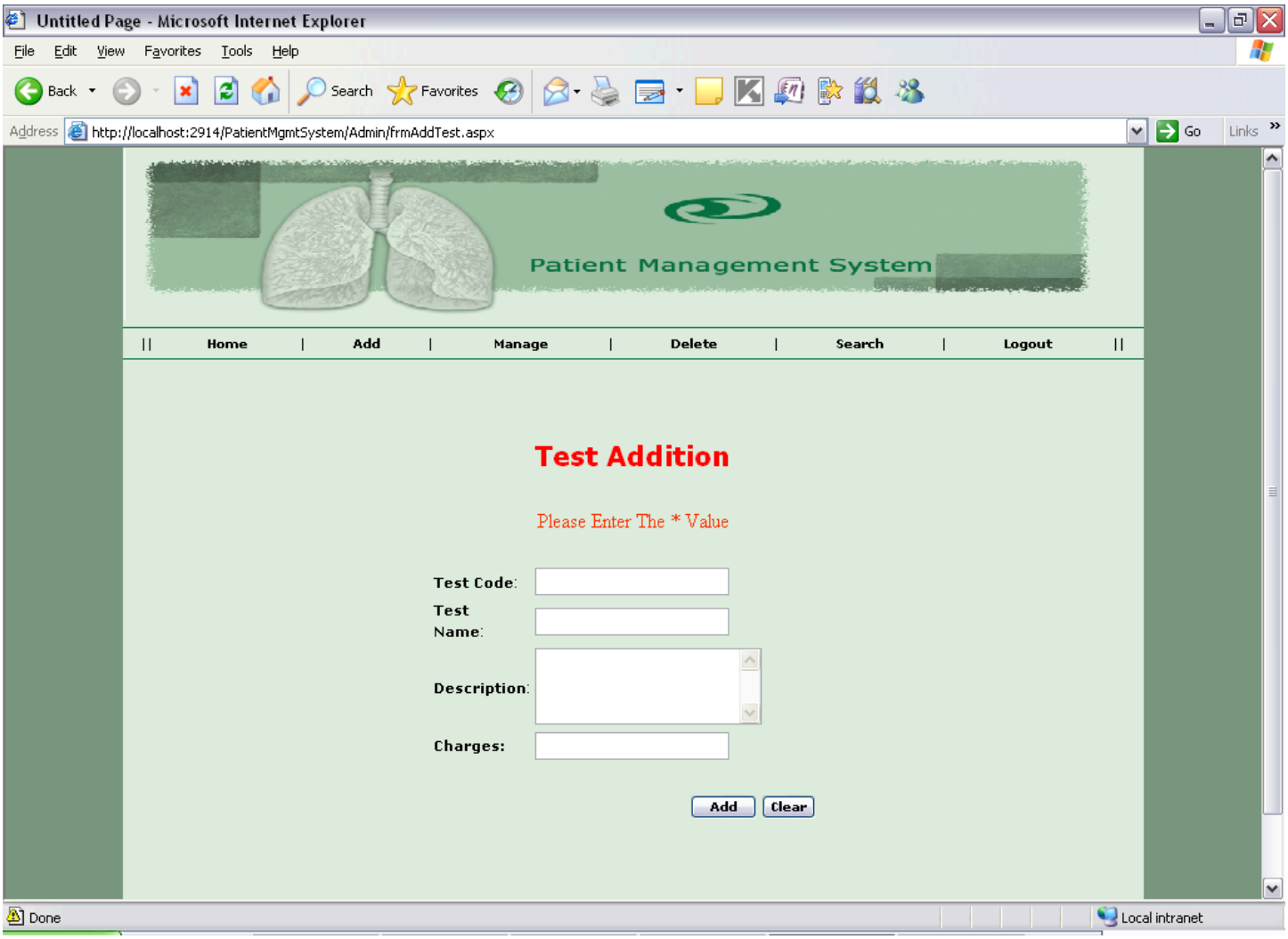

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddEmployeeDetail.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Employee Addition
Please Enter The * Value
Employee Name*:
Address*:
Phone*:
Email:
Duty Time:
Day
User Name*:
Password*:
Confirm Password*:
Add
Clear
Done
Local intranet

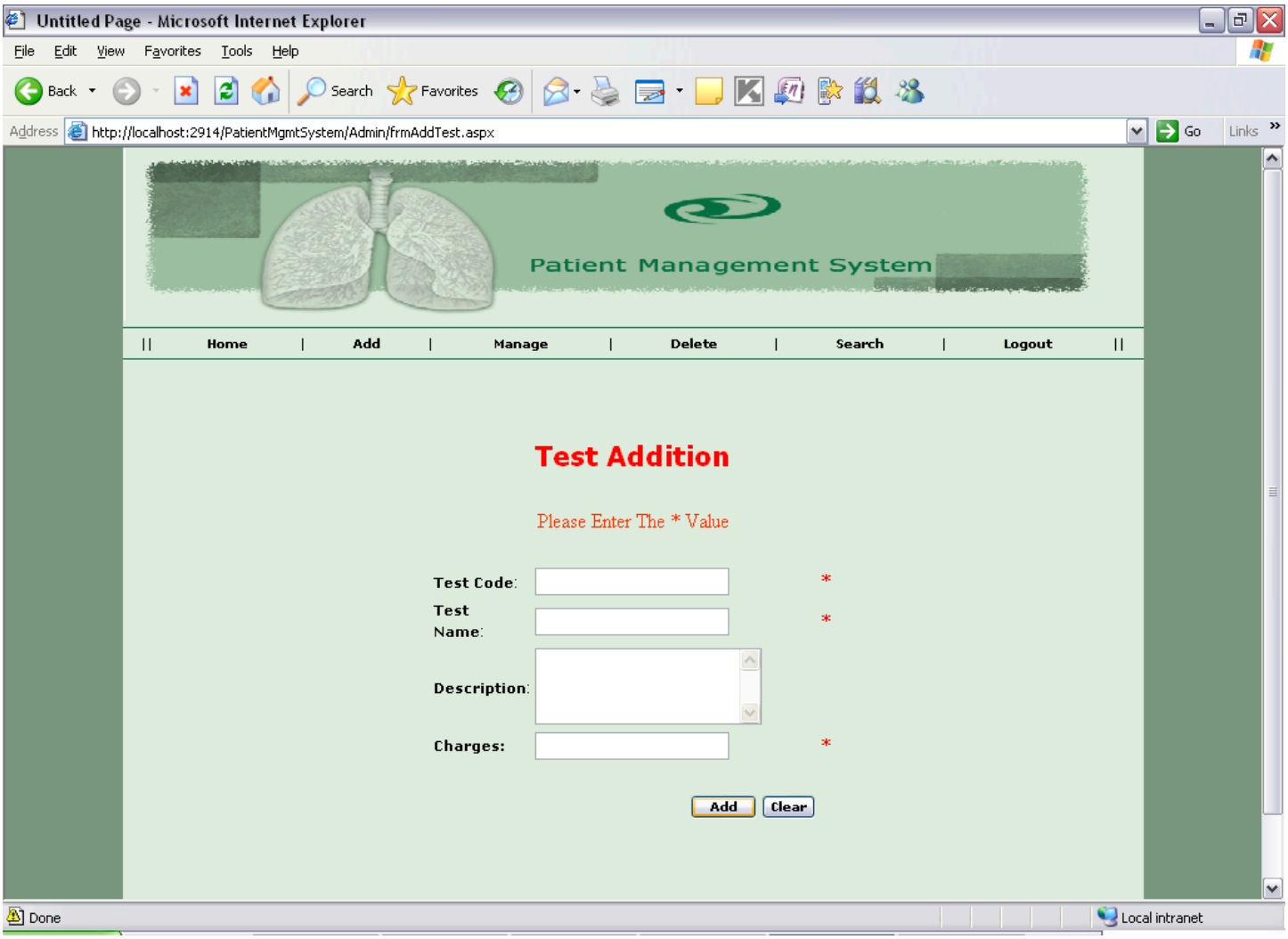

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Doctor Addition
Please Enter The * Value
Doctor Code:
Doctor Name:
Specialist:
---Select---
Available Timing:
From AM To AM
Contact No.:
Description:
Charges:
User Name:
Password:
Done
Local intranet

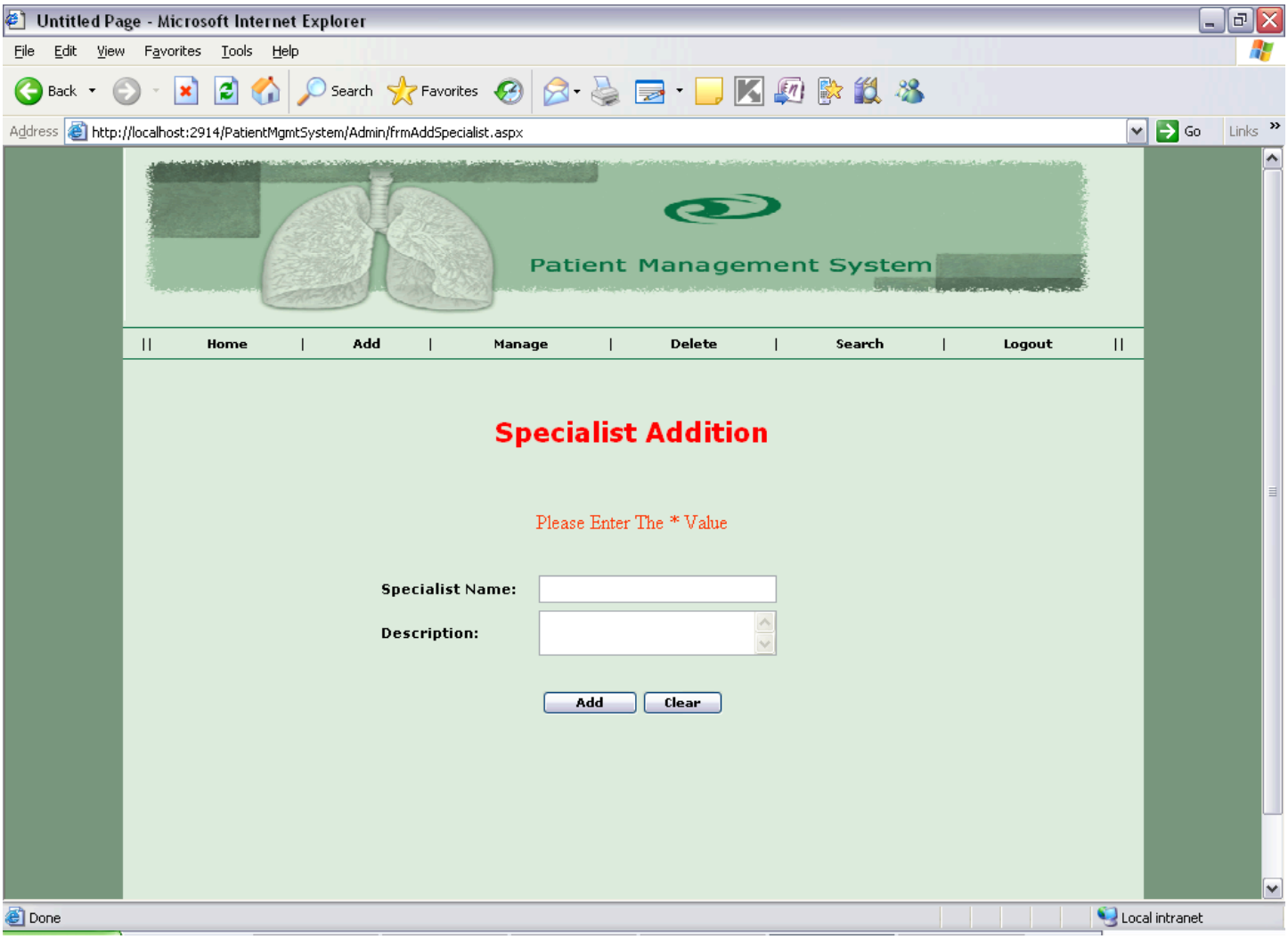

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Doctor Addition
Please Enter The * Value
Doctor Code: *
Doctor Name: *
Specialist: ---Select---
Available Timing: From AM To AM
Contact No.:
Description: *
Charges:
User Name:
Password:
Add Clear
© Copyright Reserved Patient Management System
Done
Local intranet

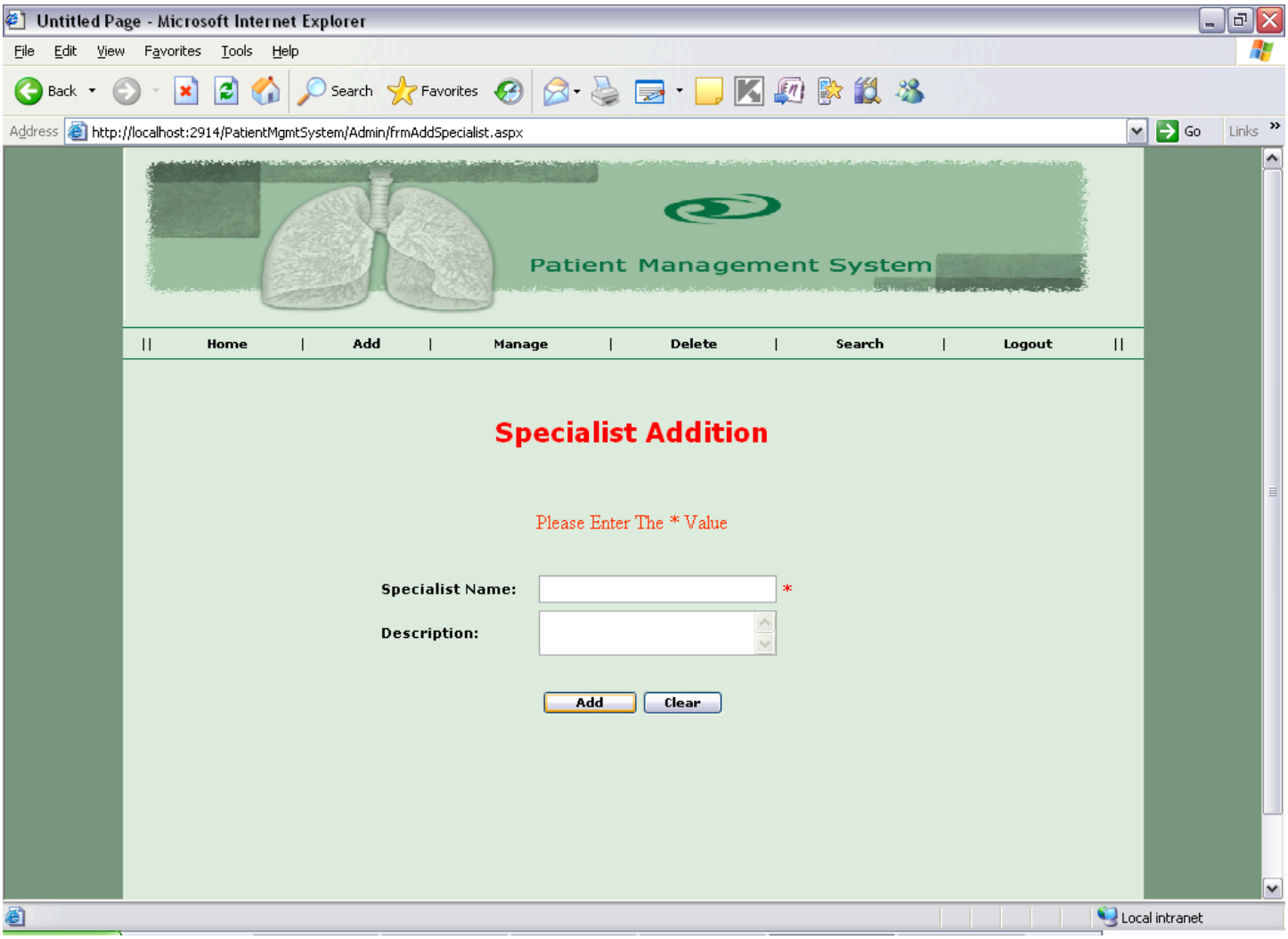

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddMedicine.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Medicine Addition
Please Enter The * Value
Medicine Code:
Medicine Name:
Prices:
Manufacture Date: 4/5/2005
Expiry Date: 4/5/2005
Company Name:
Batch No.:
Quantity:
Add Clear
Done, but with errors on page.
Local intranet

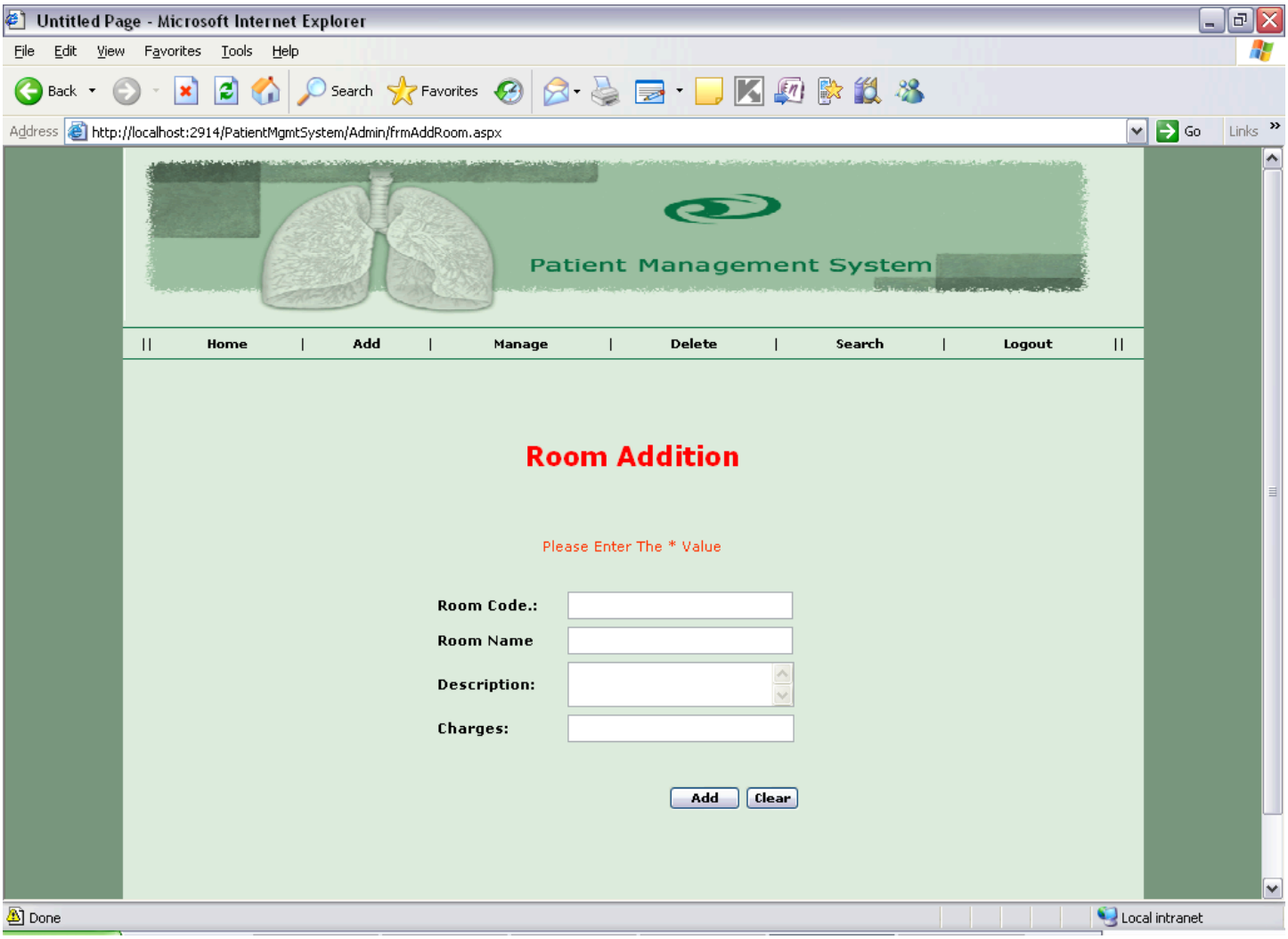

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddMedicine.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Medicine Addition
Please Enter The * Value
Medicine Code: *
Medicine Name: *
Prices: *
Manufacture Date: 4/5/2005
Expiry Date: 4/5/2005
Company Name: *
Batch No.:
Quantity: *
Add Clear
Done, but with errors on page.
Local intranet

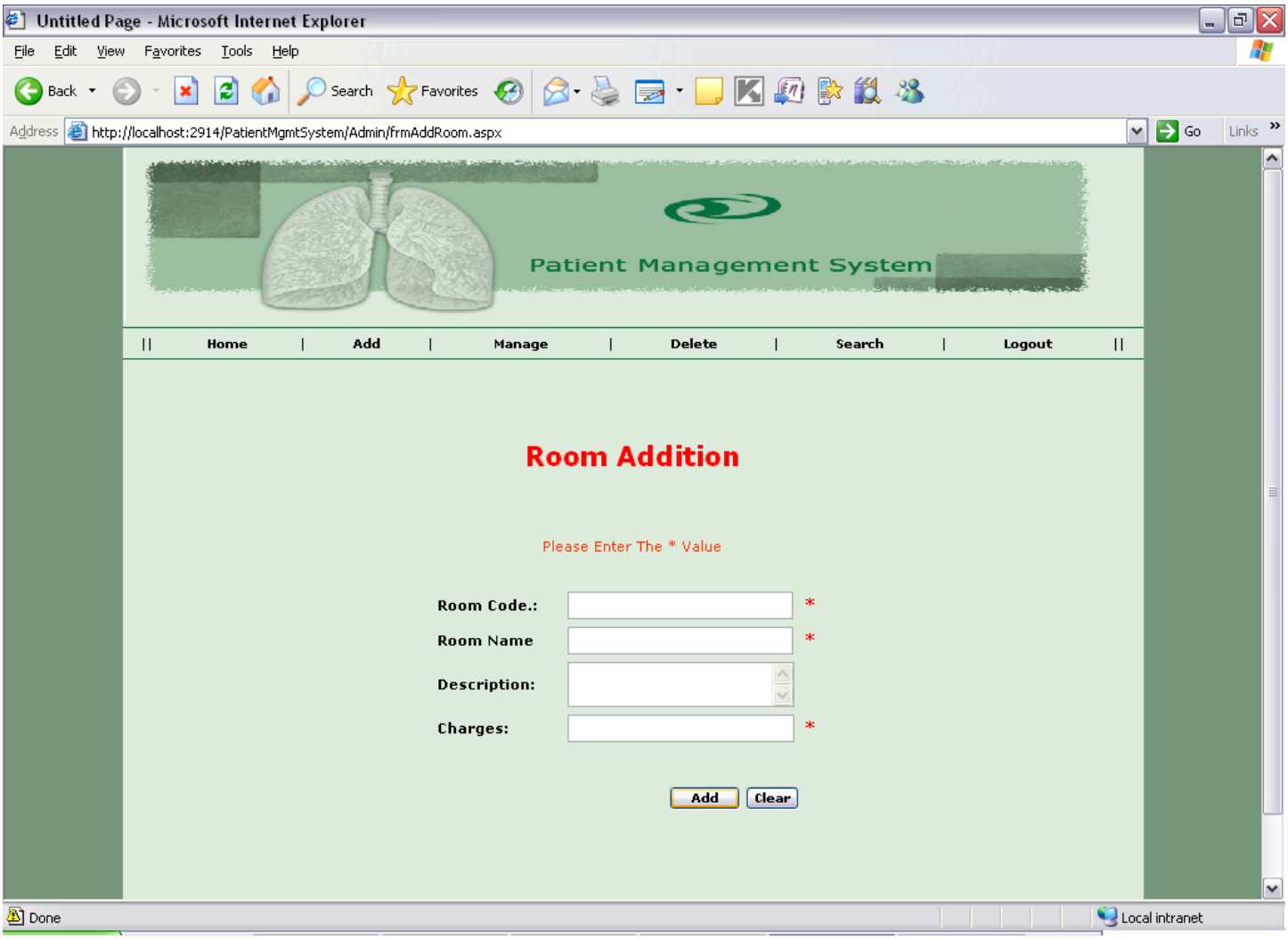

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddTest.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Test Addition
Please Enter The * Value
Test Code:
Test Name:
Description:
Charges:
Add Clear
Done
Local intranet

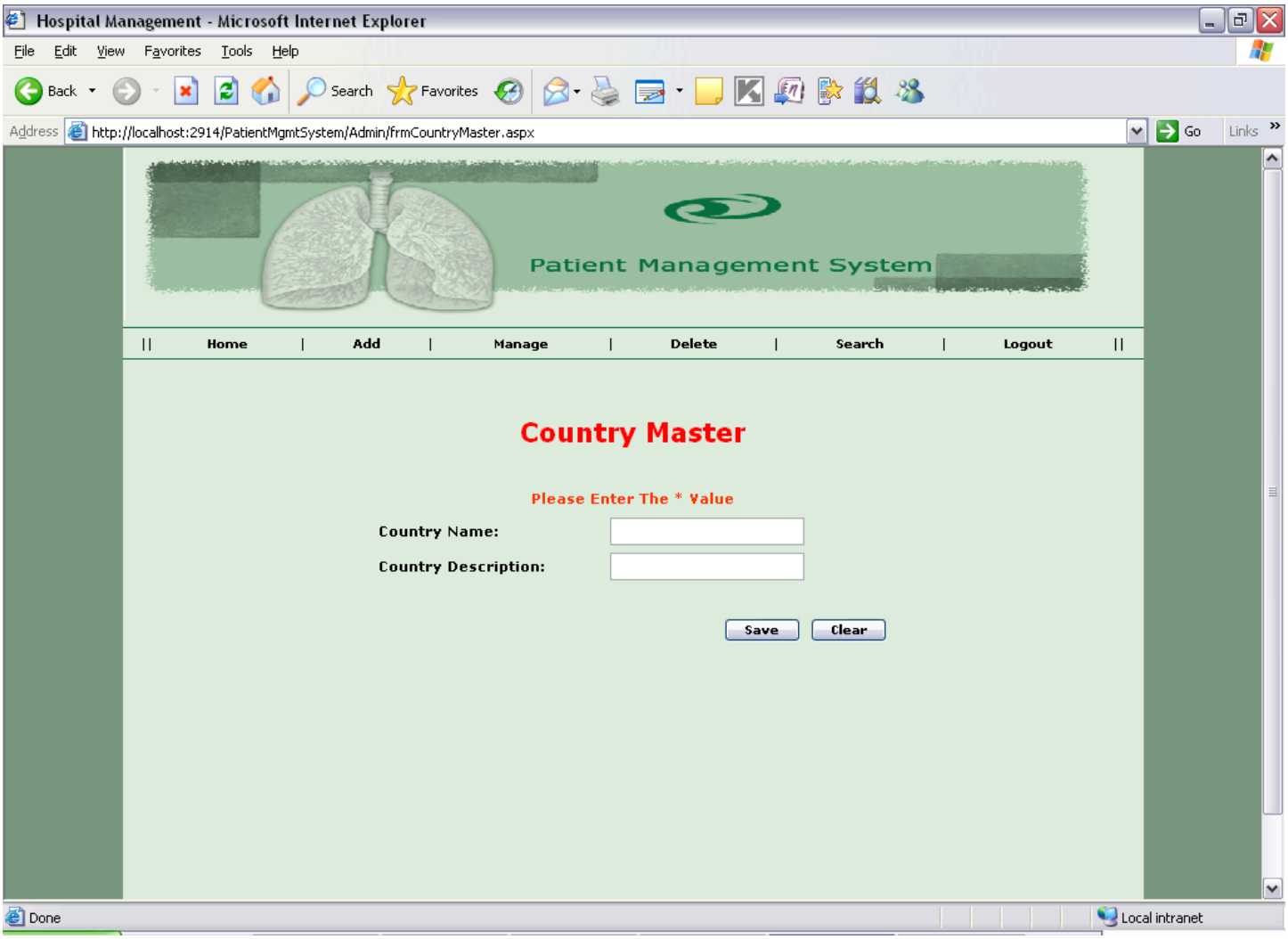

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddTest.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Test Addition
Please Enter The * Value
Test Code:
Test Name:
Description:
Charges:
Add
Clear
Done
Local intranet

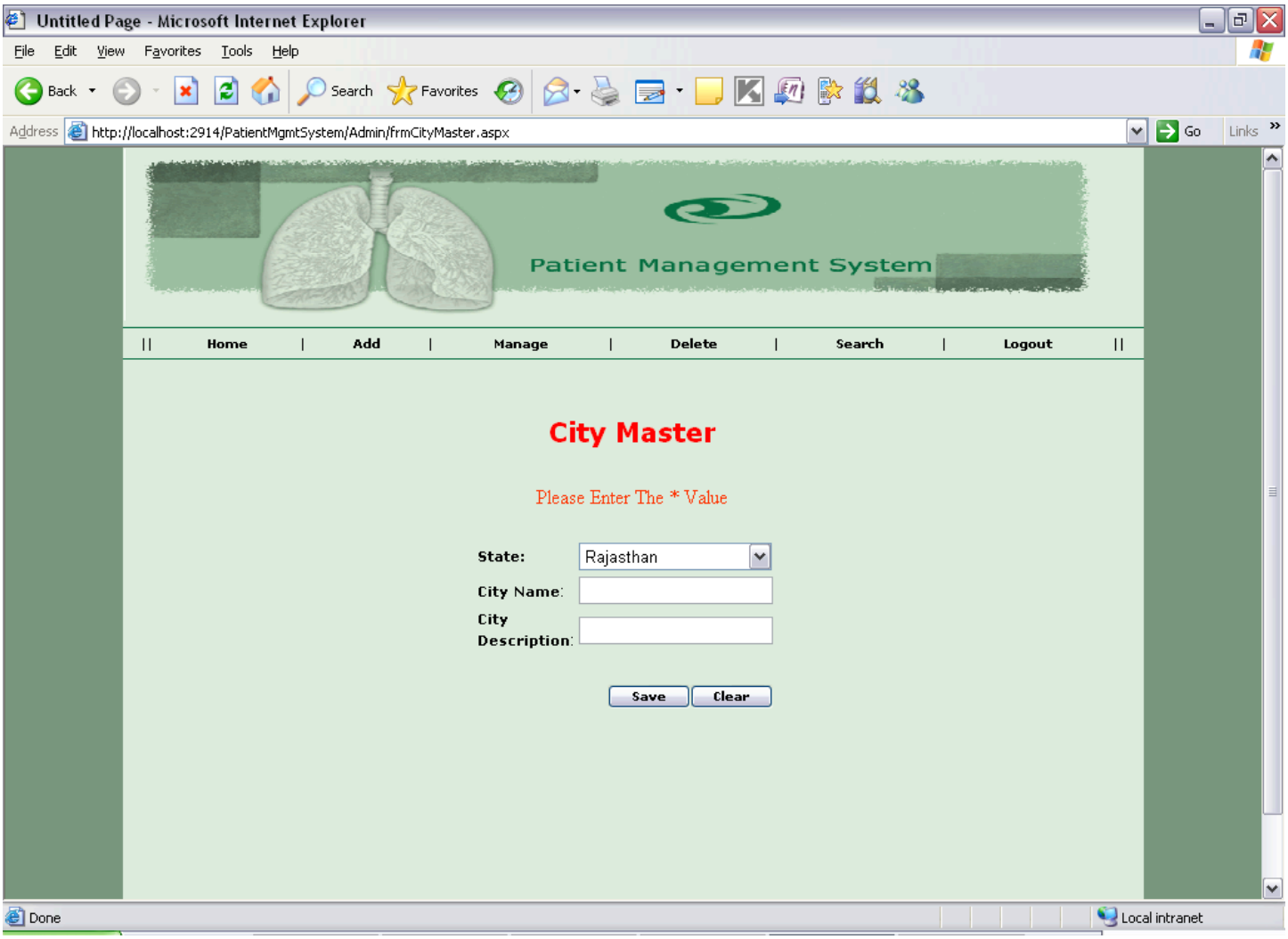

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddSpecialist.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Specialist Addition
Please Enter The * Value
Specialist Name:
Description:
Add Clear
Done
Local intranet

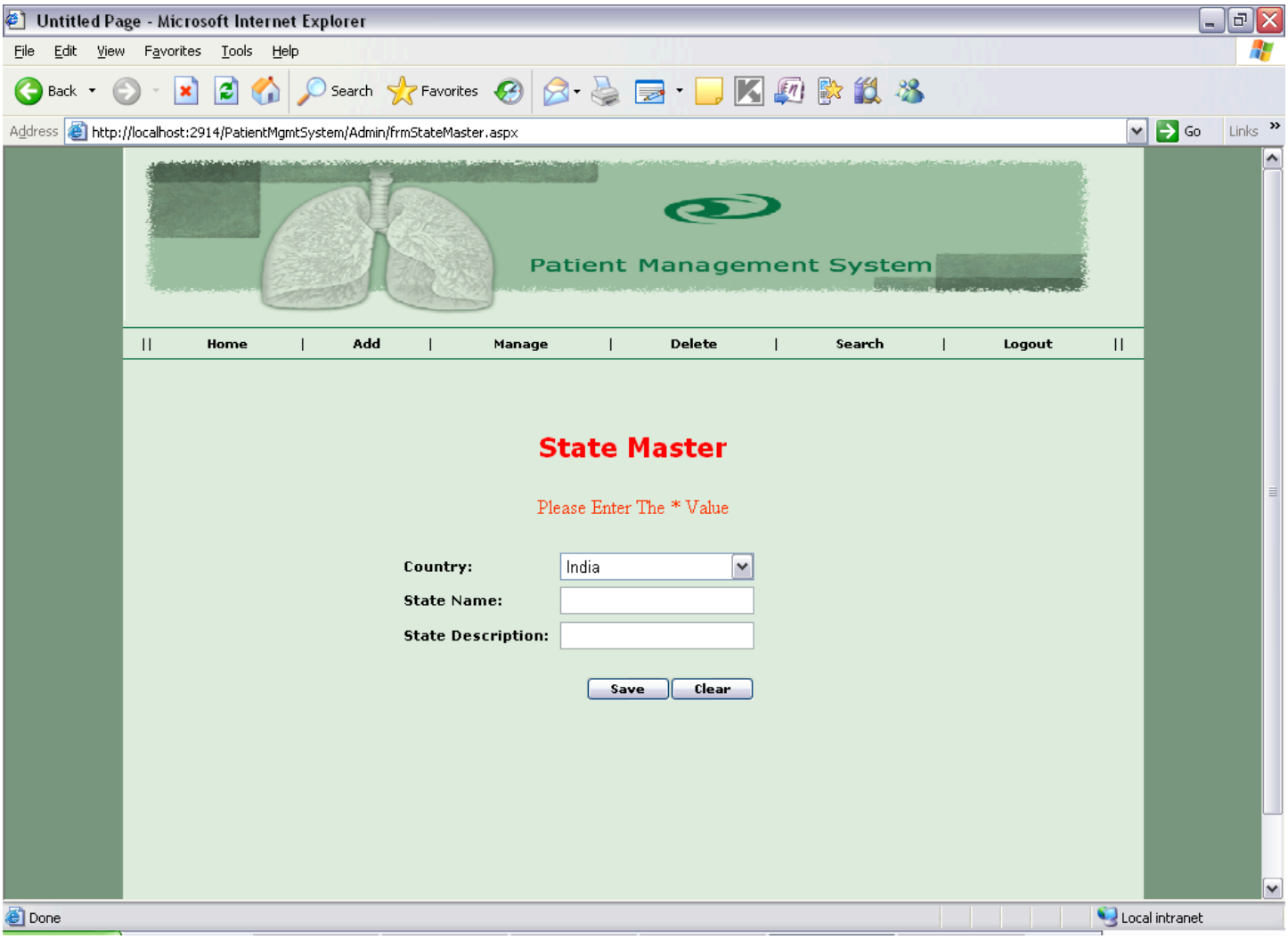

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddSpecialist.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Specialist Addition
Please Enter The * Value
Specialist Name: *
Description:
Add Clear
Local intranet

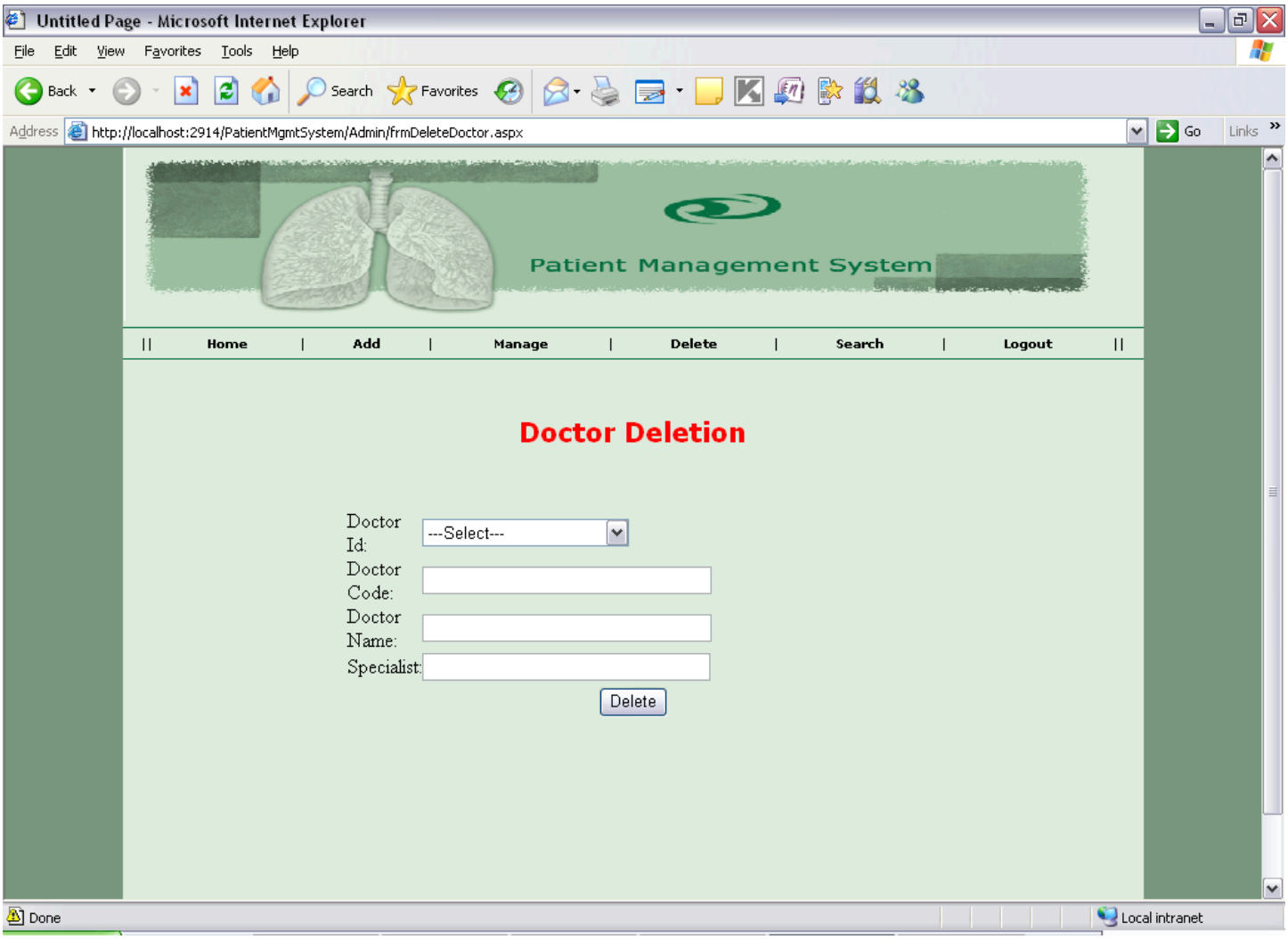

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddRoom.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Room Addition
Please Enter The * Value
Room Code.:
Room Name
Description:
Charges:
Add Clear
Done
Local intranet

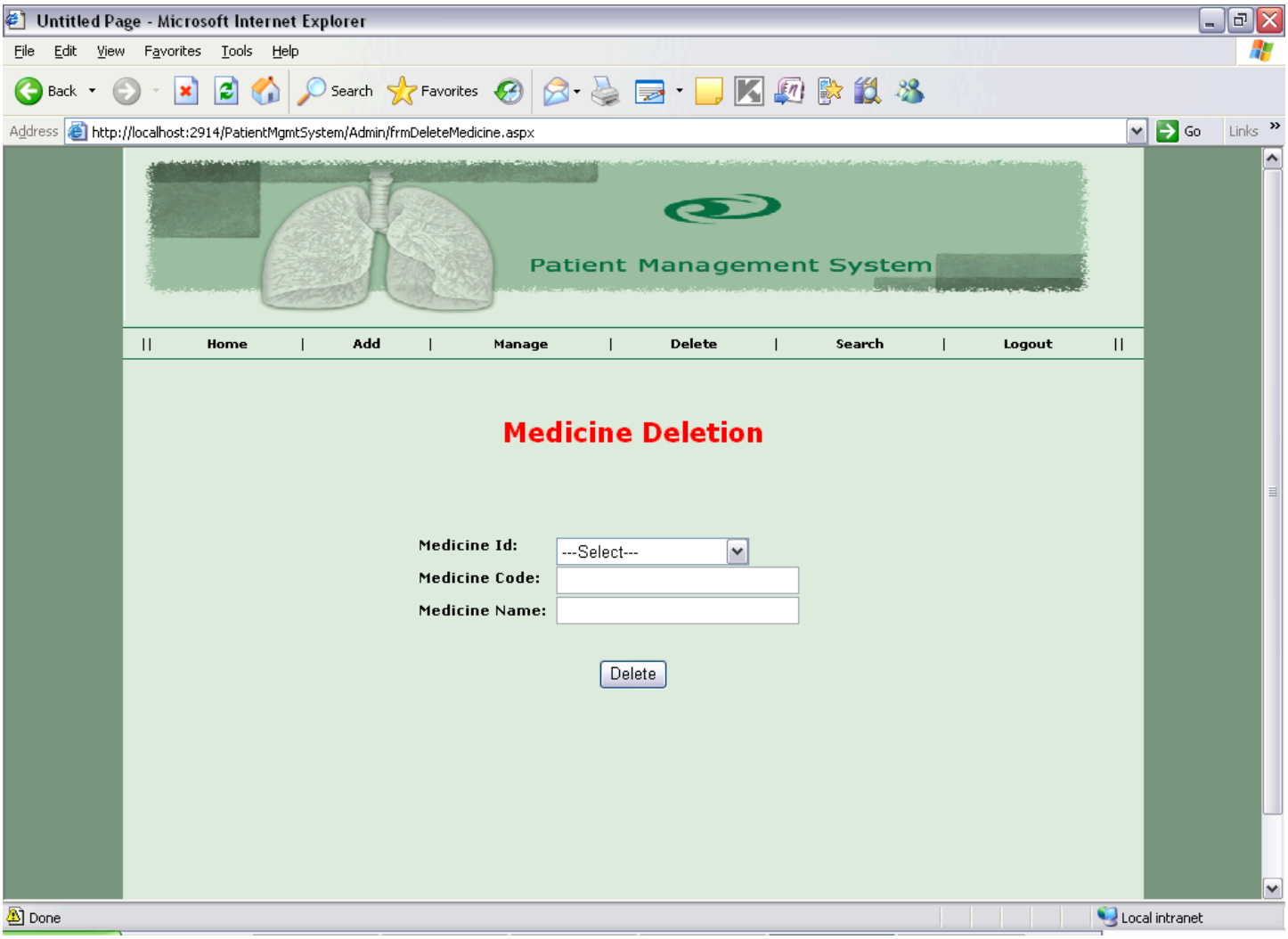

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddRoom.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Room Addition
Please Enter The * Value
Room Code.:
Room Name
Description:
Charges:
Add
Clear
Done
Local intranet

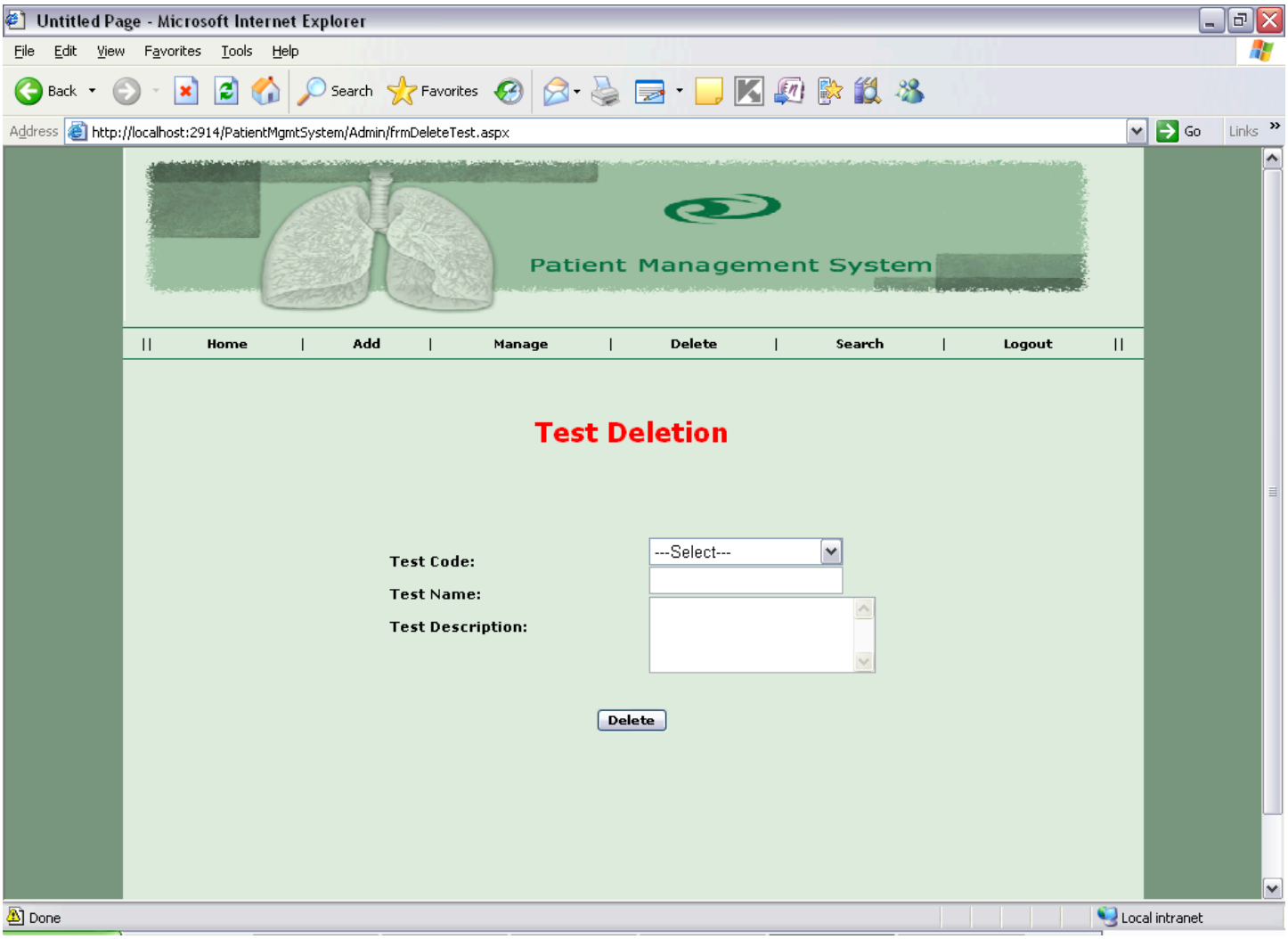

Hospital Management - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmCountryMaster.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Country Master
Please Enter The * Value
Country Name:
Country Description:
Save Clear
Done
Local intranet

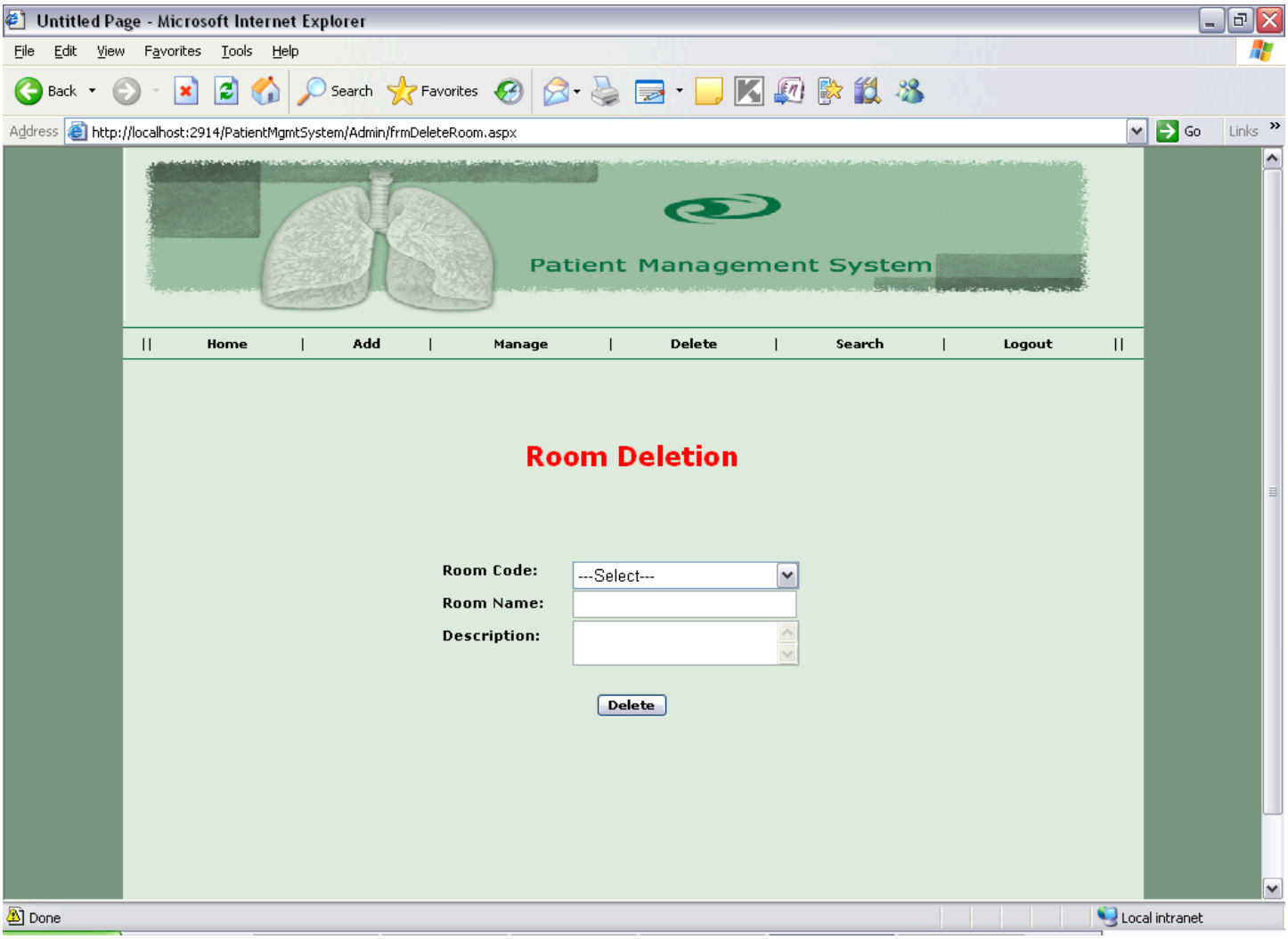

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmCityMaster.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
City Master
Please Enter The * Value
State: Rajasthan
City Name:
City Description:
Save Clear
Done
Local intranet

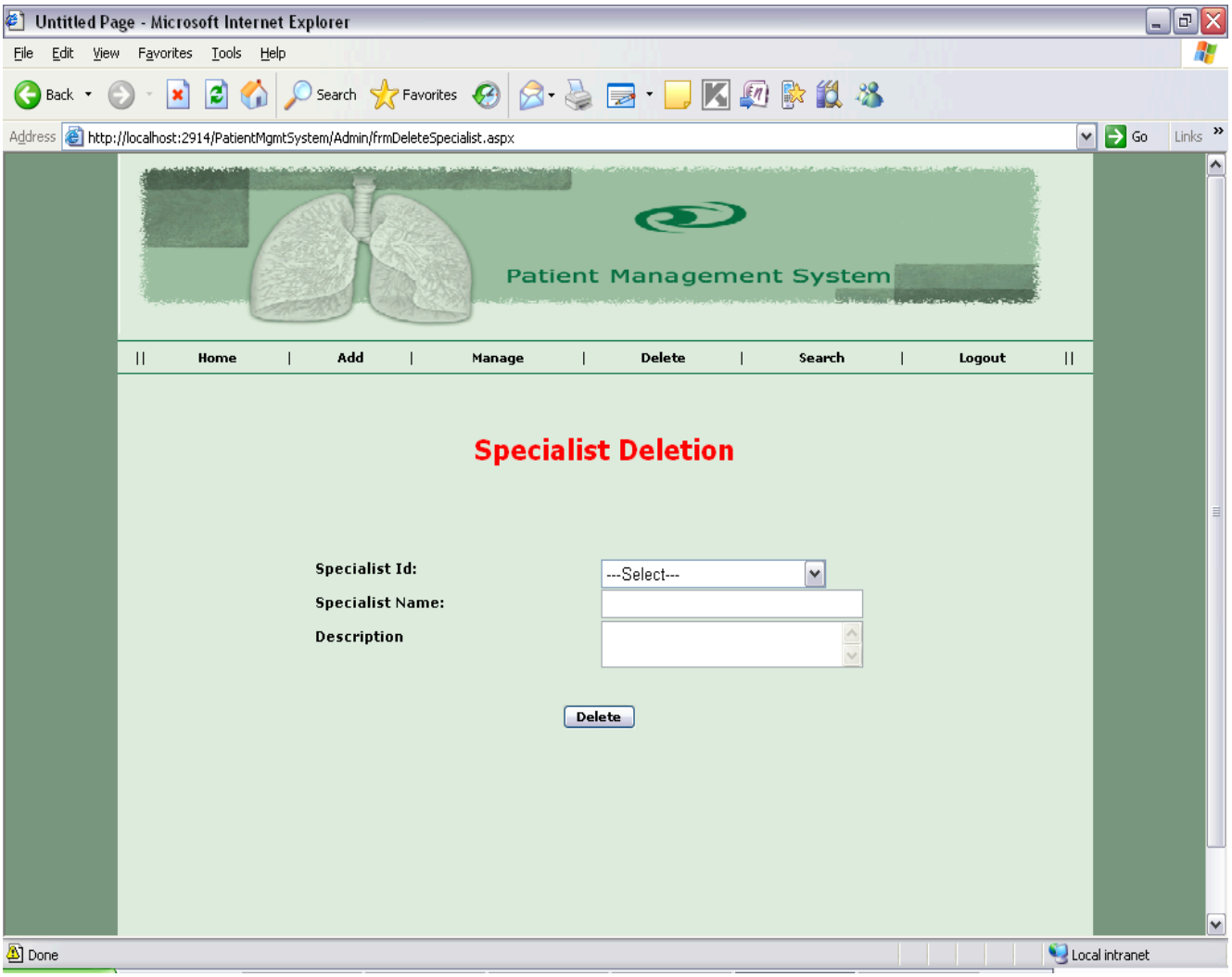

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmStateMaster.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
State Master
Please Enter The * Value
Country: India
State Name:
State Description:
Save Clear
Done
Local intranet

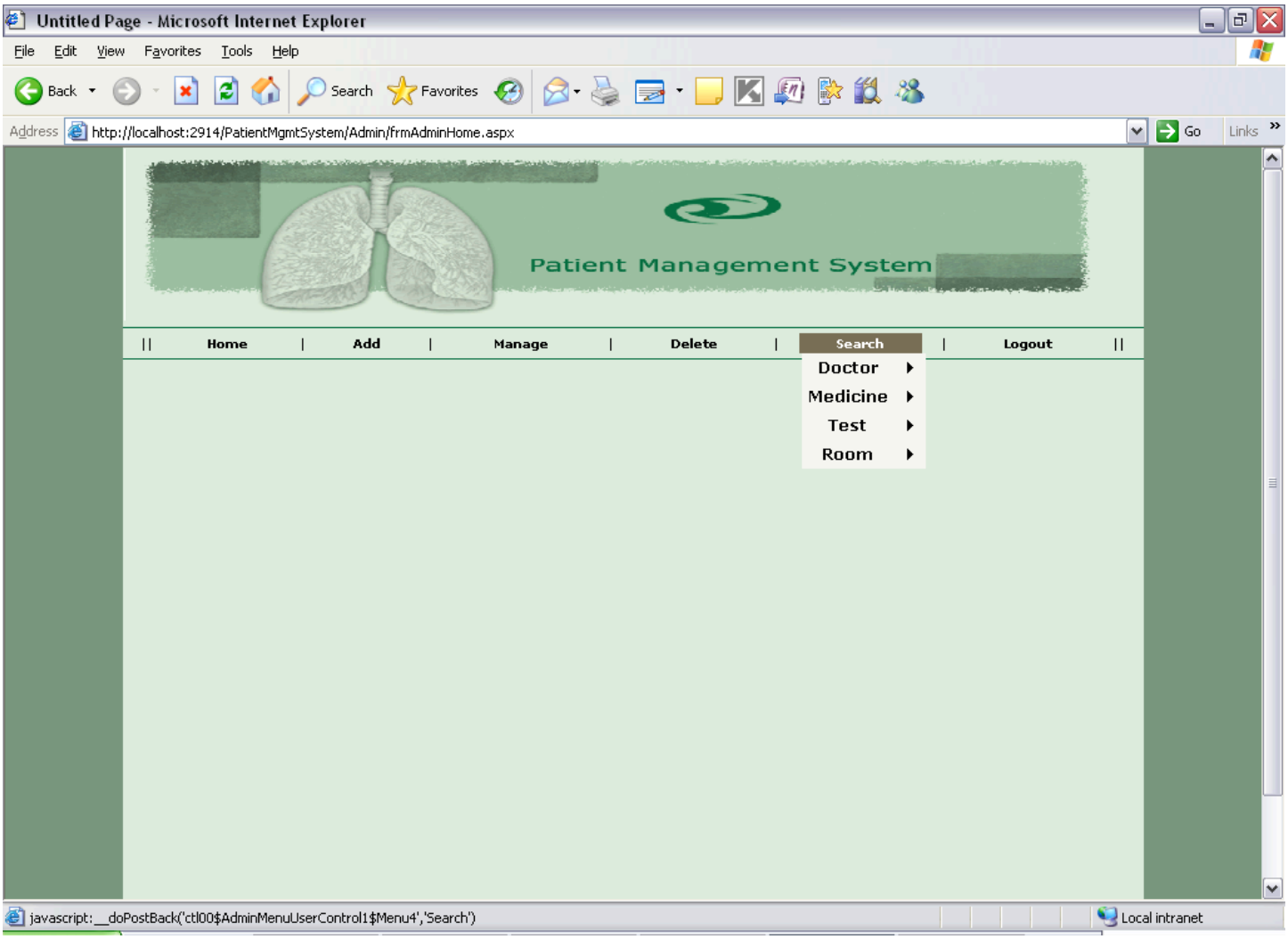

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmDeleteDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Doctor Deletion
Doctor Id:
---Select---
Doctor Code:
Doctor Name:
Specialist:
Delete
Done
Local intranet

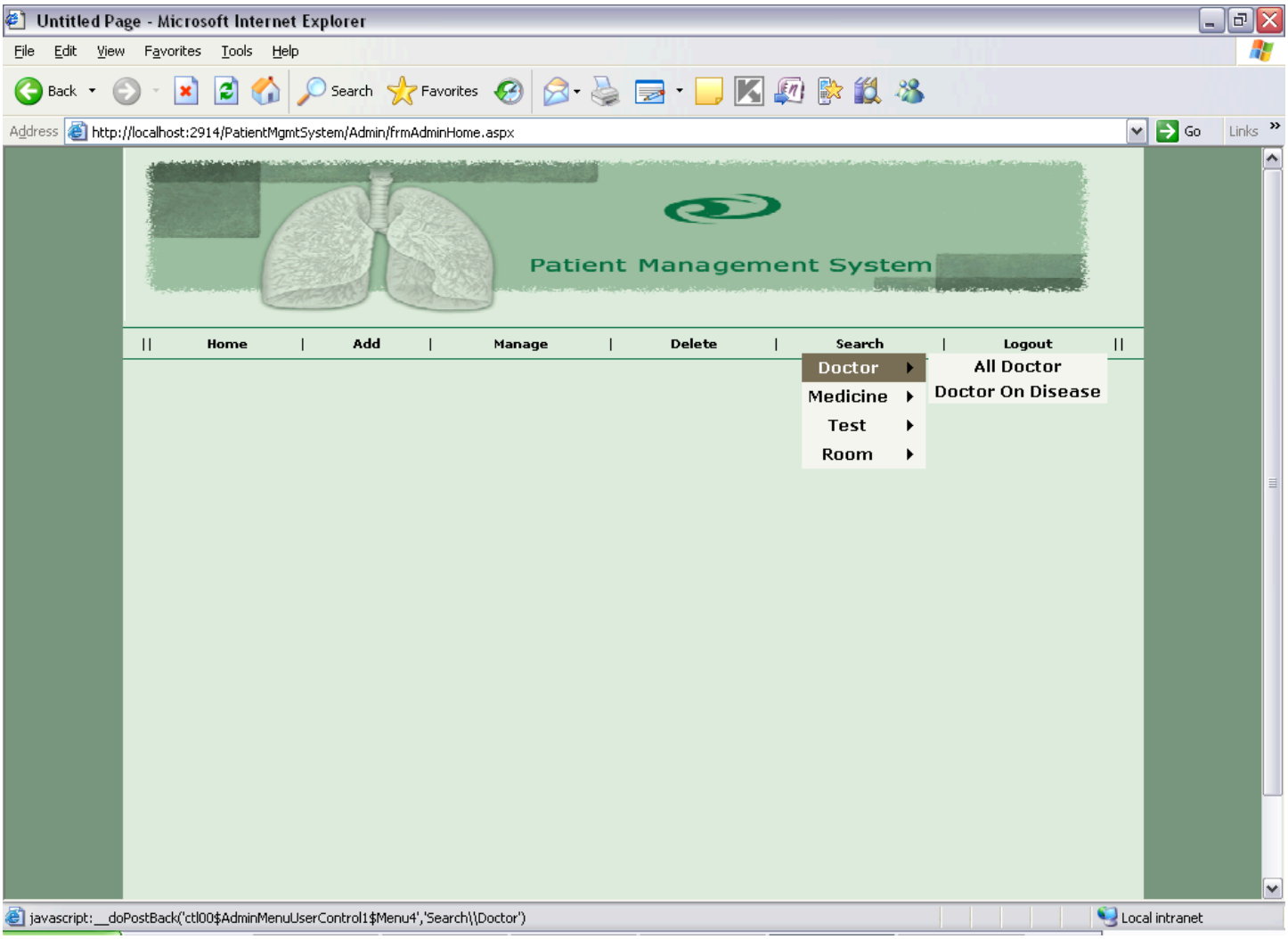

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmDeleteMedicine.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Medicine Deletion
Medicine Id:
---Select---
Medicine Code:
Medicine Name:
Delete
Done
Local intranet

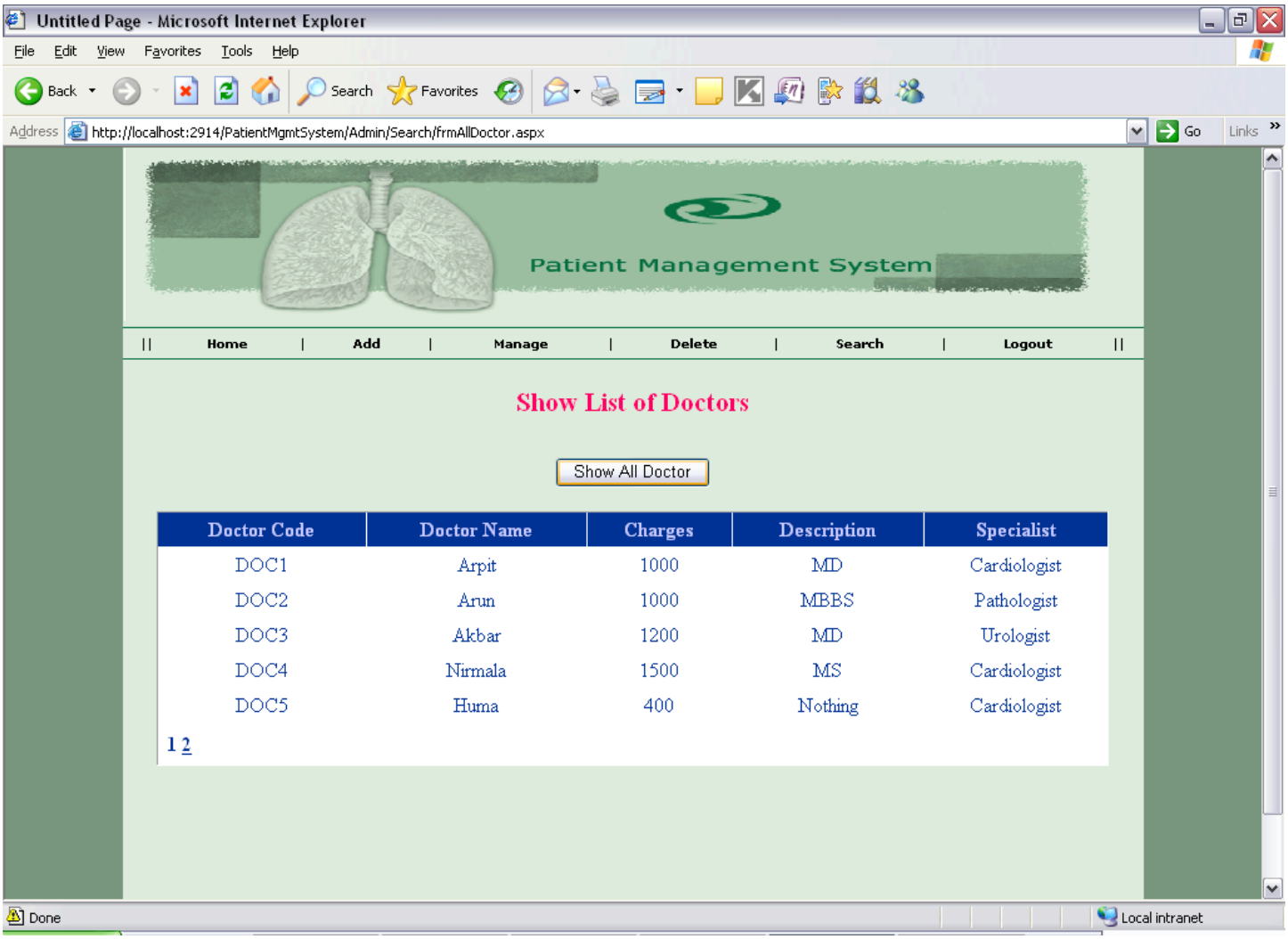

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmDeleteTest.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Test Deletion
Test Code:
---Select---
Test Name:
Test Description:
Delete
Done
Local intranet

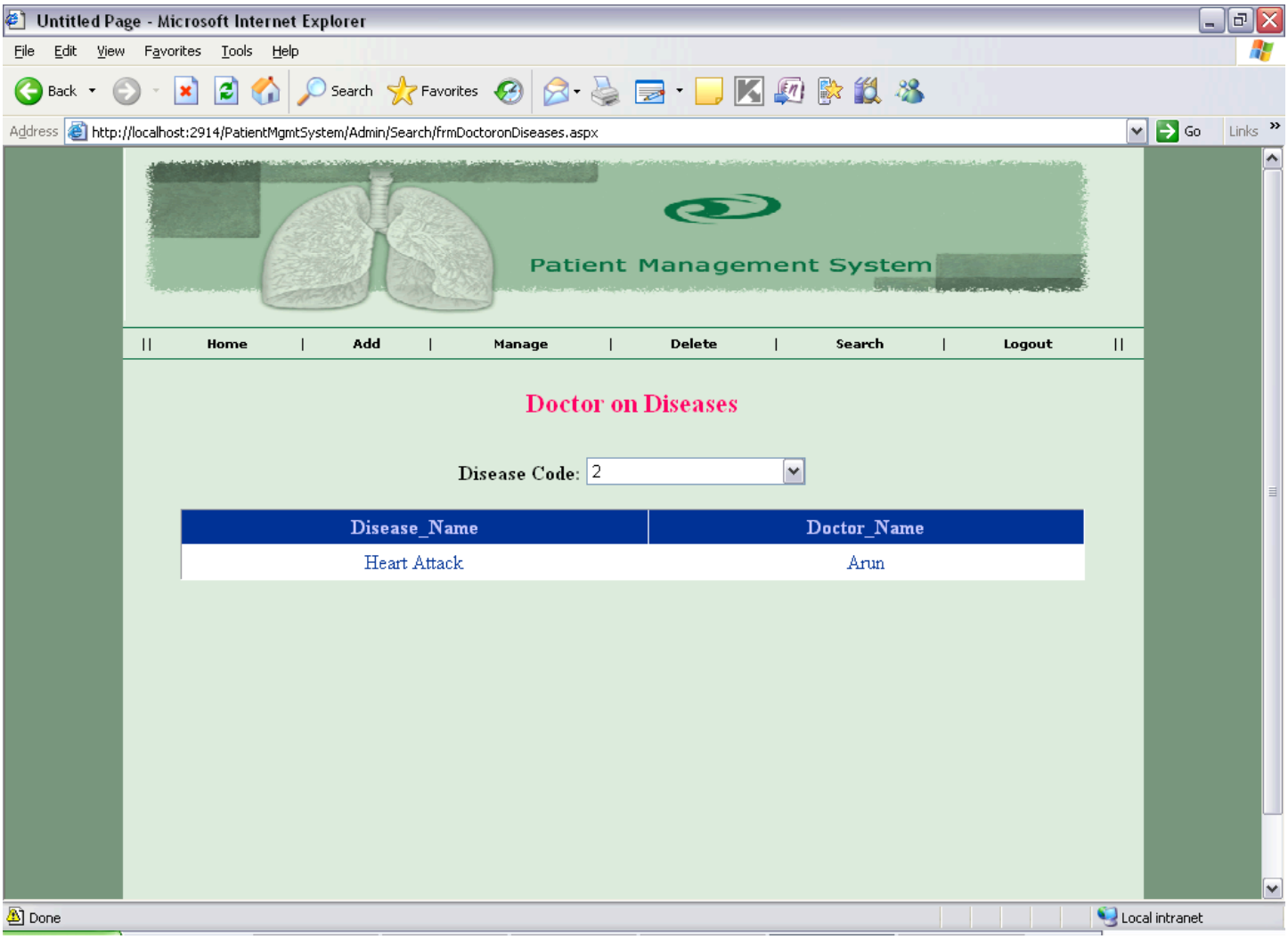

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmDeleteRoom.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Room Deletion
Room Code:
---Select---
Room Name:
Description:
Delete
Done
Local intranet

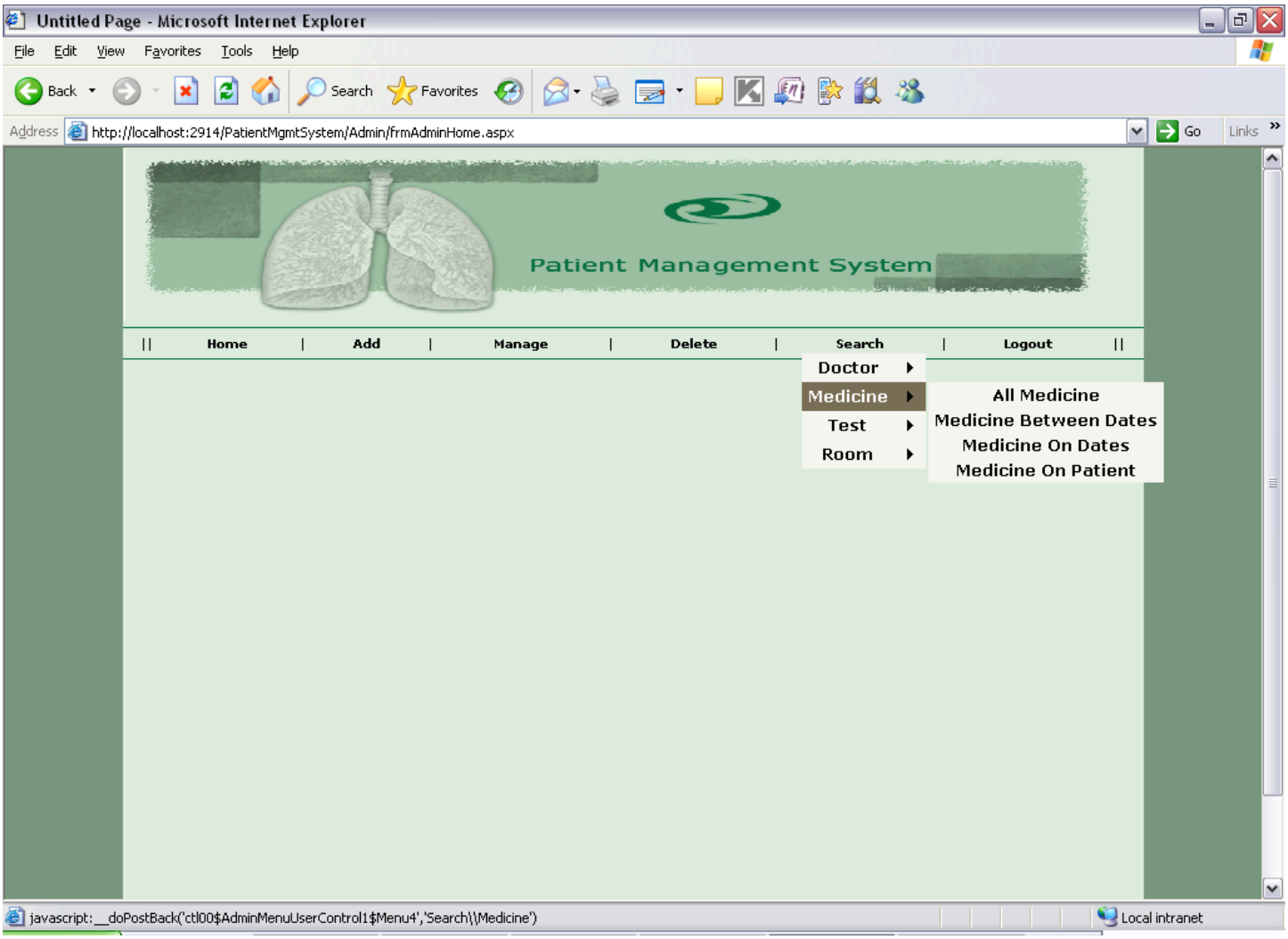

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmDeleteSpecialist.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Specialist Deletion
Specialist Id:
---Select---
Specialist Name:
Description
Delete
Done
Local intranet

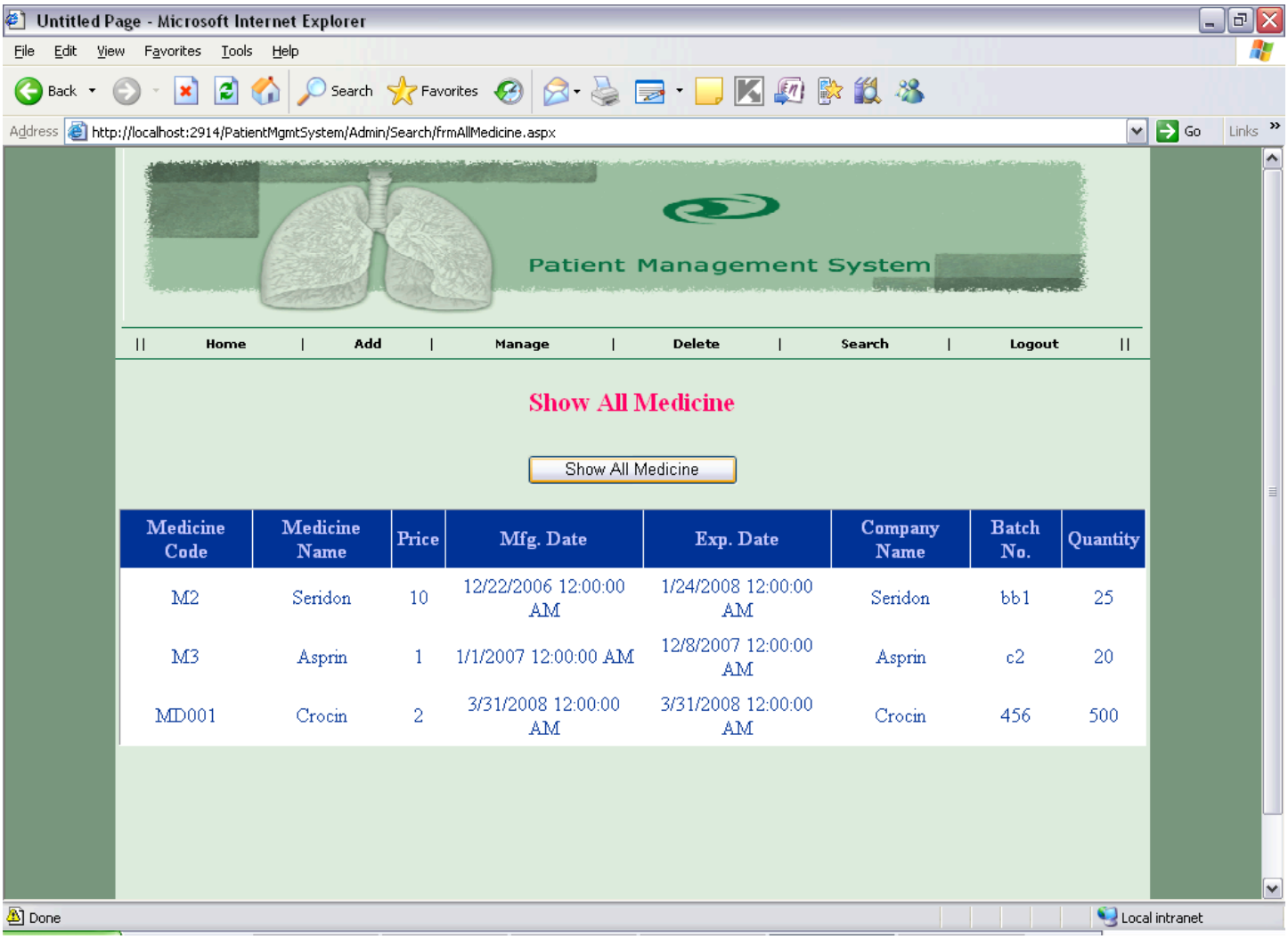

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor
Medicine
Test
Room
javascript:__doPostBack('ctl00$AdminMenuUserControl1$Menu4','Search')
Local intranet

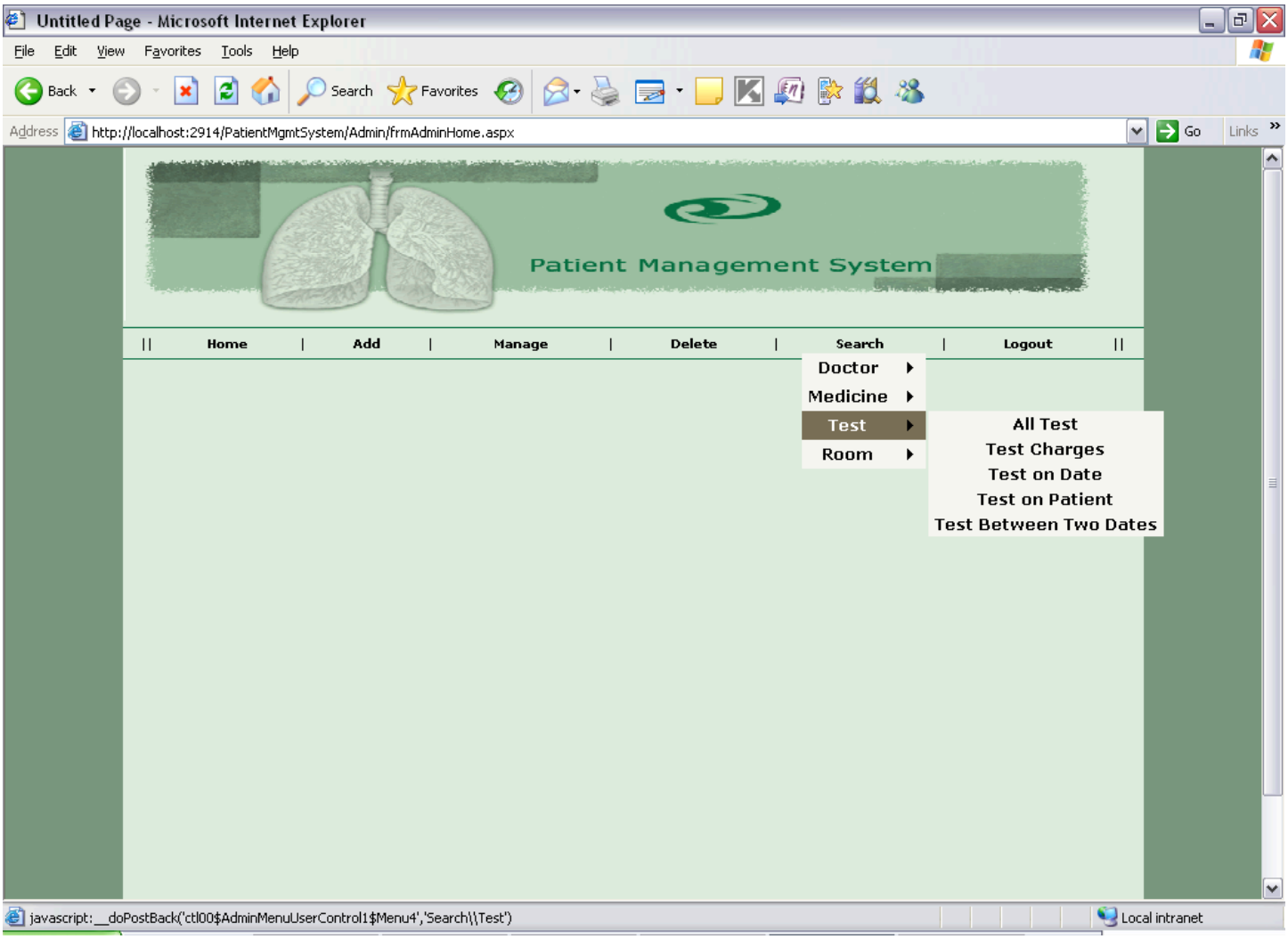

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor
Medicine
Test
Room
All Doctor
Doctor On Disease
javascript:__doPostBack('ctl00$AdminMenuUserControl1$Menu4','Search\\Doctor')
Local intranet

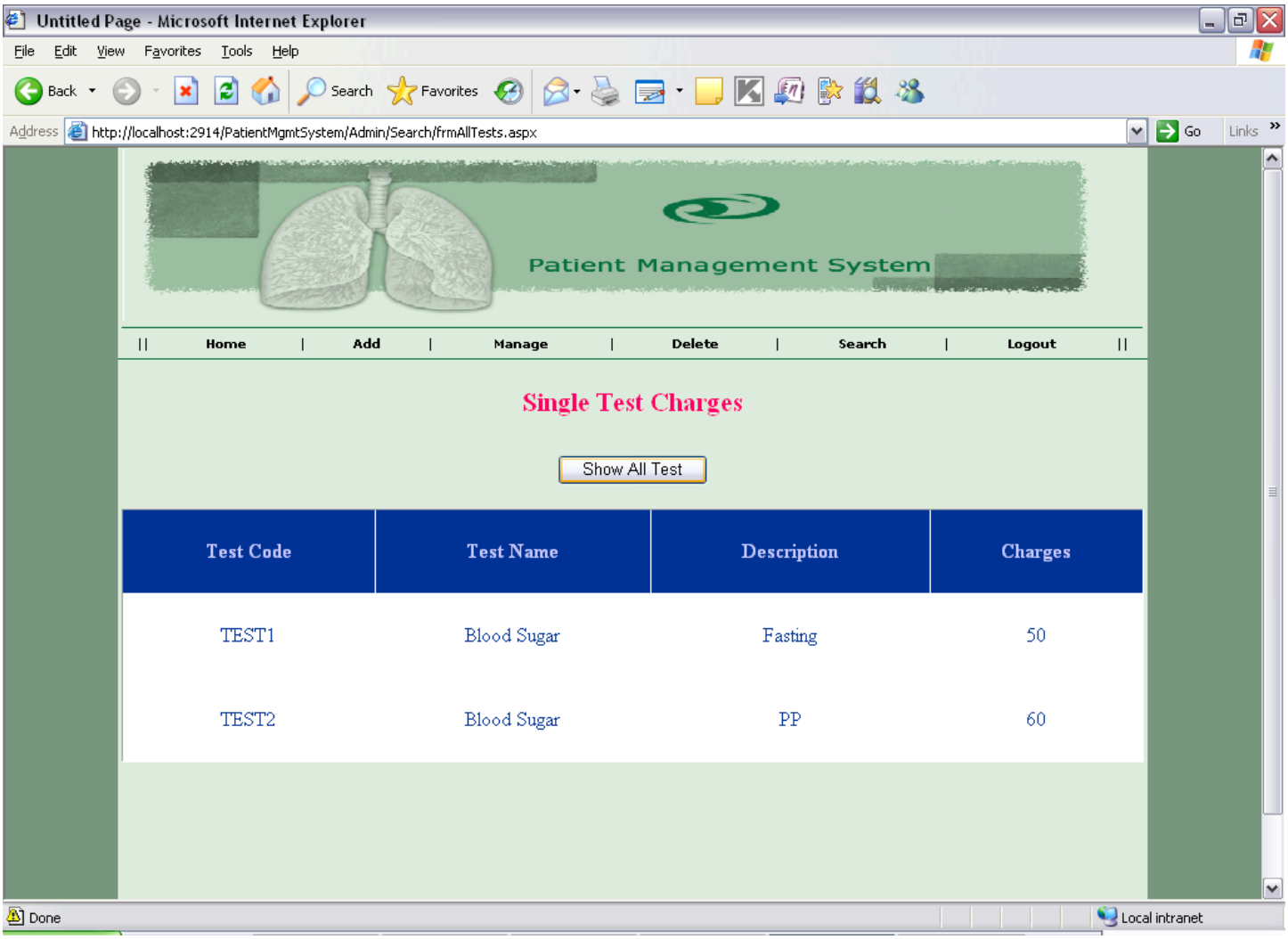

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/Search/frmAllDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Show List of Doctors
Show All Doctor
Doctor Code | Doctor Name | Charges | Description | Specialist
DOC1 | Arpit | 1000 | MD | Cardiologist
DOC2 | Arun | 1000 | MBBS | Pathologist
DOC3 | Akbar | 1200 | MD | Urologist
DOC4 | Nirmala | 1500 | MS | Cardiologist
DOC5 | Huma | 400 | Nothing | Cardiologist
1 2
Done
Local intranet

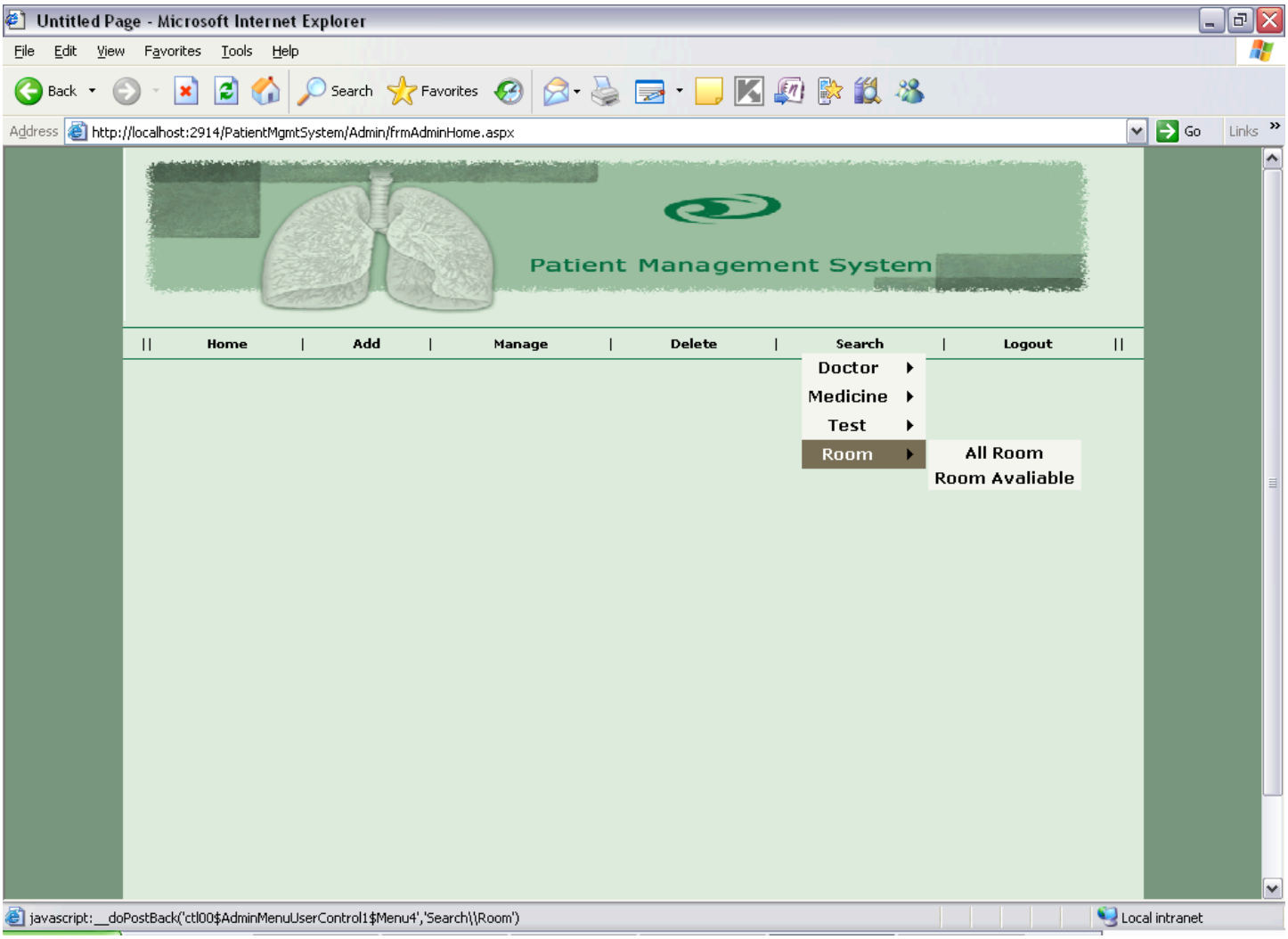

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/Search/frmDoctoronDiseases.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor on Diseases
Disease Code: 2
Disease_Name
Doctor_Name
Heart Attack
Arun
Done
Local intranet

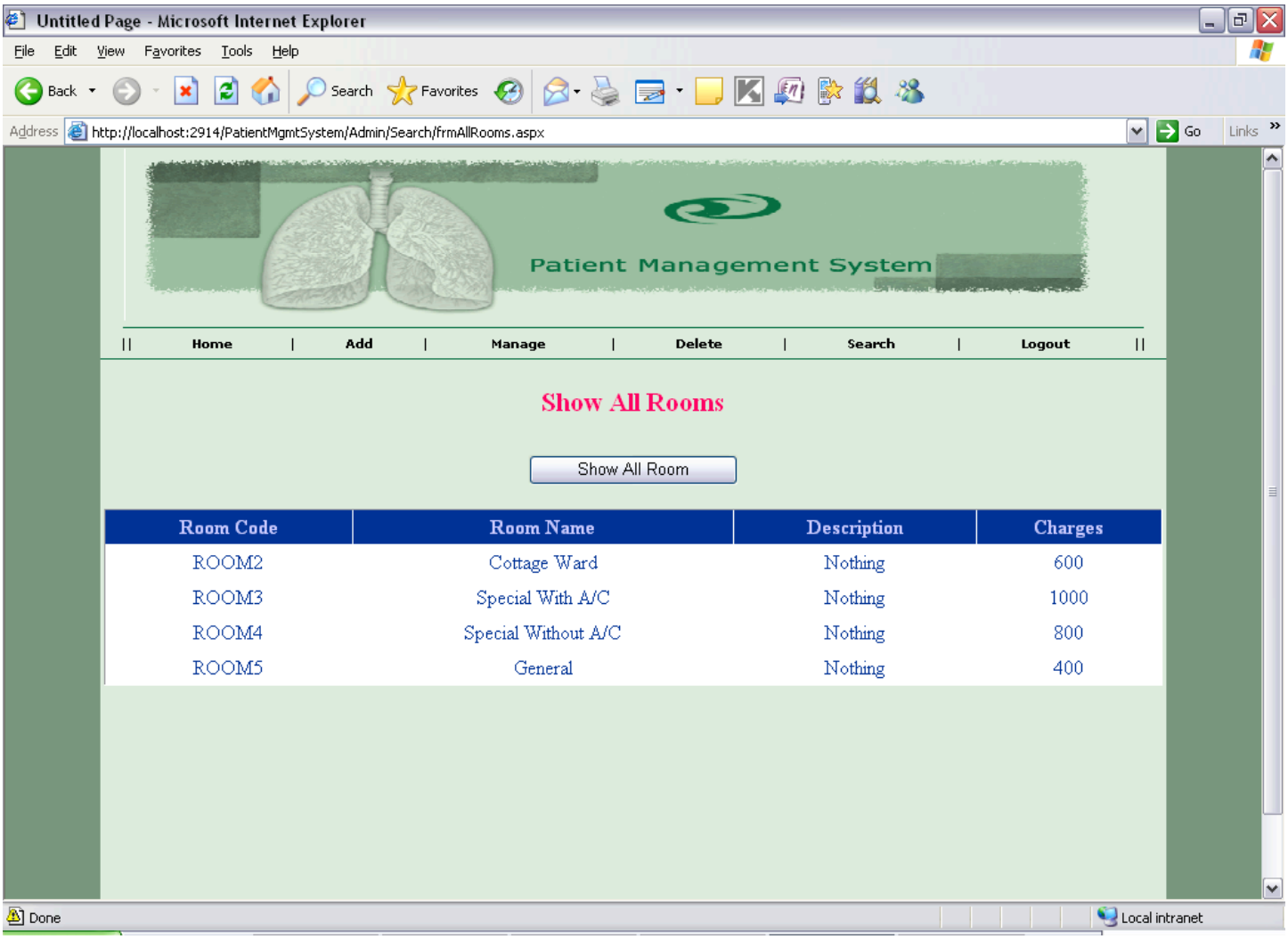

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor
Medicine
Test
Room
All Medicine
Medicine Between Dates
Medicine On Dates
Medicine On Patient
javascript:__doPostBack('ctl00$AdminMenuUserControl1$Menu4','Search\\Medicine')
Local intranet

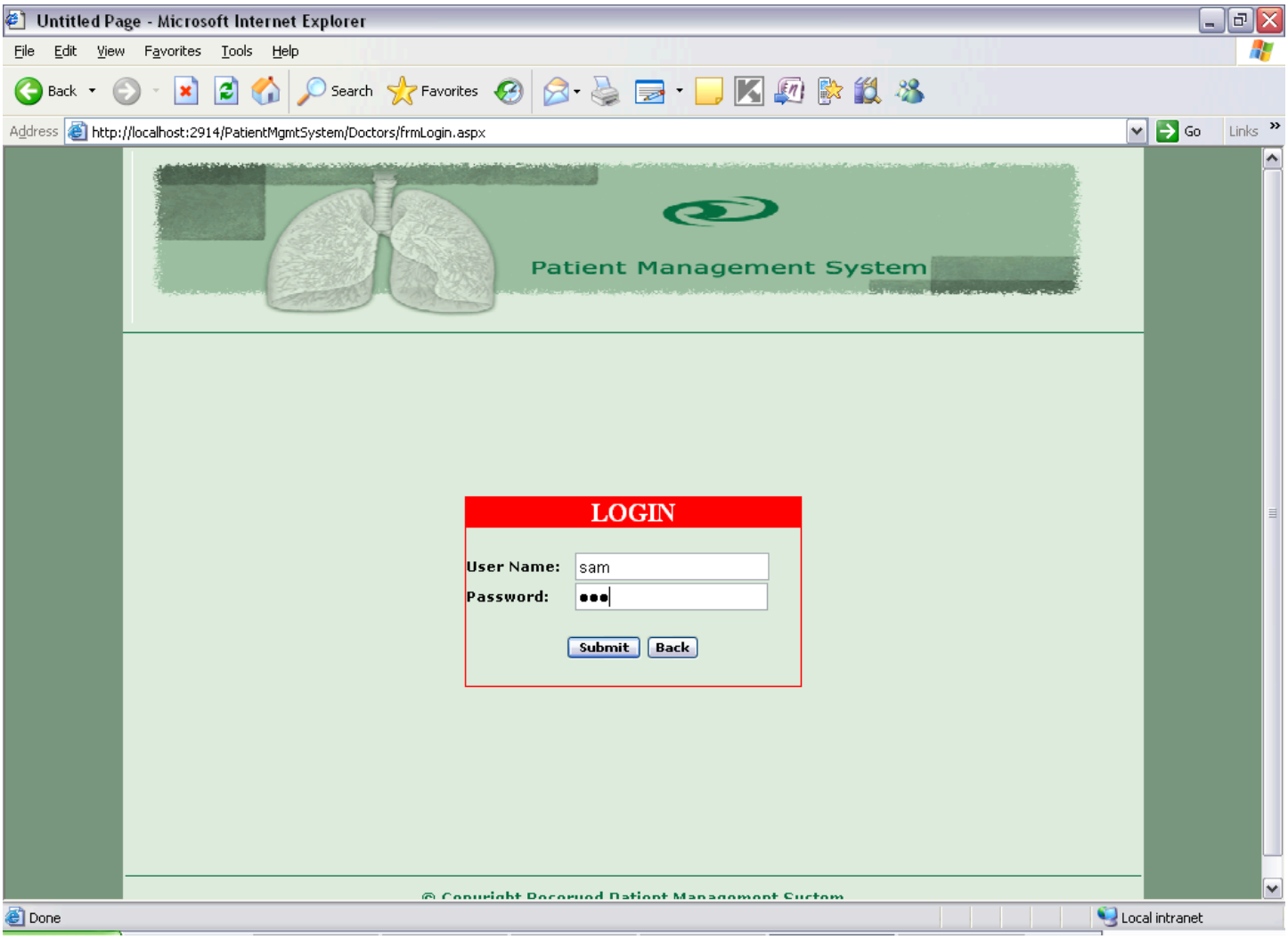

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/Search/frmAllMedicine.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Show All Medicine
Show All Medicine
Medicine Code | Medicine Name | Price | Mfg. Date | Exp. Date | Company Name | Batch No. | Quantity
M2 | Seridon | 10 | 12/22/2006 12:00:00 AM | 1/24/2008 12:00:00 AM | Seridon | bb1 | 25
M3 | Asprin | 1 | 1/1/2007 12:00:00 AM | 12/8/2007 12:00:00 AM | Asprin | c2 | 20
MD001 | Crocin | 2 | 3/31/2008 12:00:00 AM | 3/31/2008 12:00:00 AM | Crocin | 456 | 500
Done
Local intranet

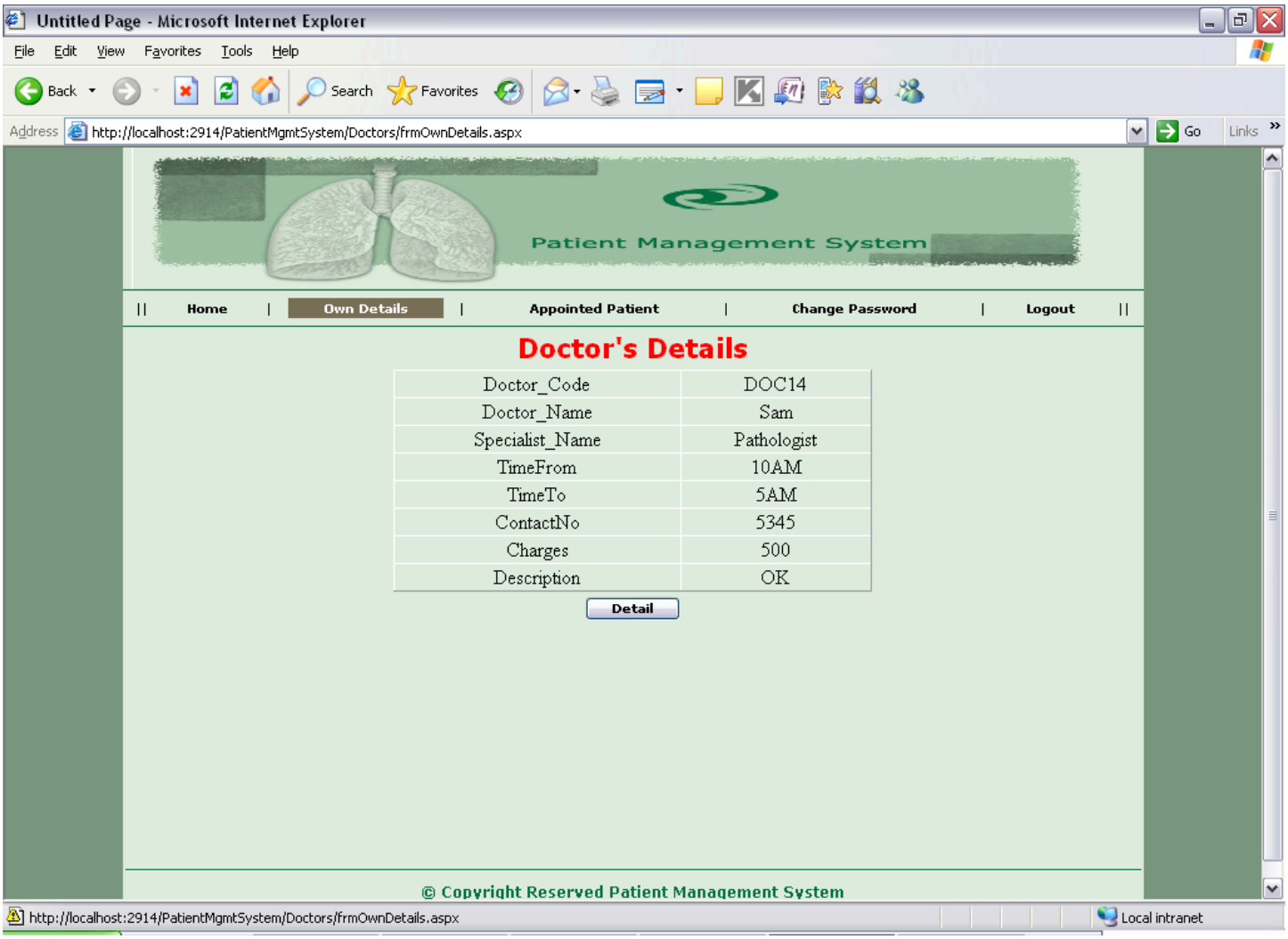

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor
Medicine
Test
Room
All Test
Test Charges
Test on Date
Test on Patient
Test Between Two Dates
javascript:__doPostBack('ctl00$AdminMenuUserControl1$Menu4','Search\\Test')
Local intranet

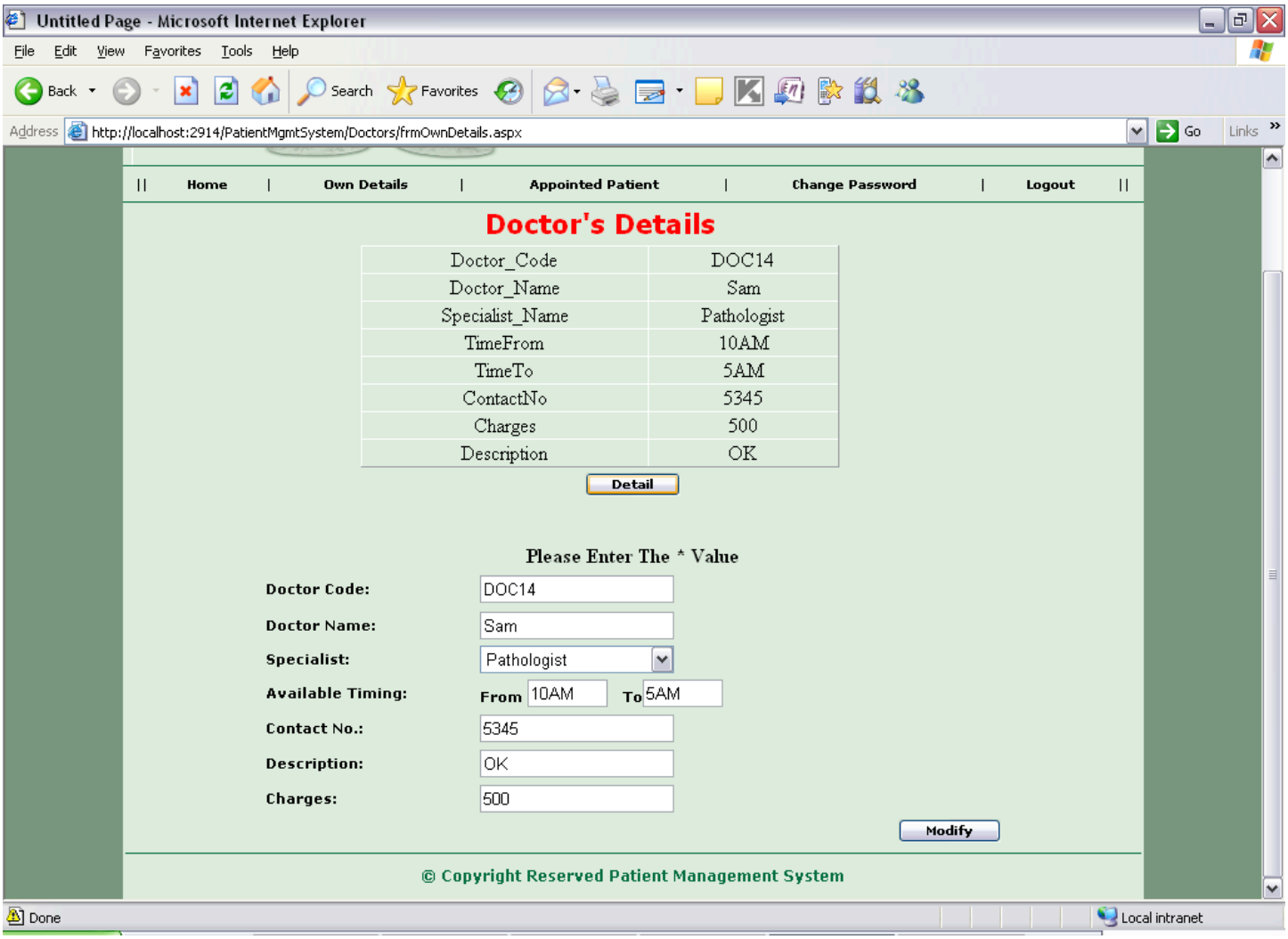

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/Search/frmAllTests.aspx
Go Links
Patient Management System
|| Home | Add | Manage | Delete | Search | Logout ||
Single Test Charges
Show All Test
Test Code
Test Name
Description
Charges
TEST1
Blood Sugar
Fasting
50
TEST2
Blood Sugar
PP
60
Done
Local intranet

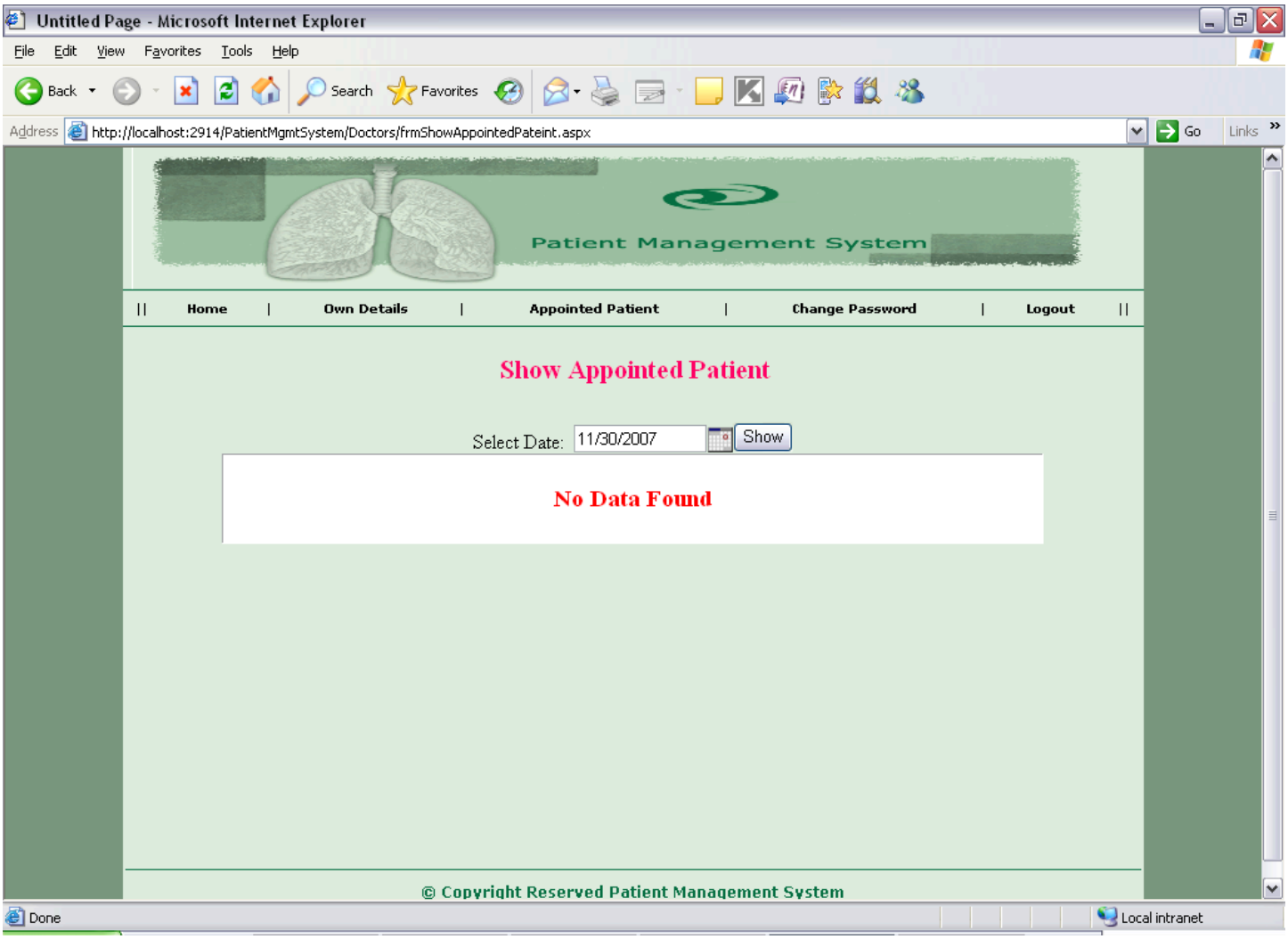

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminHome.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Doctor
Medicine
Test
Room
All Room
Room Avaliable
javascript:__doPostBack('ctl00$AdminMenuUserControl1$Menu4','Search\\Room')
Local intranet

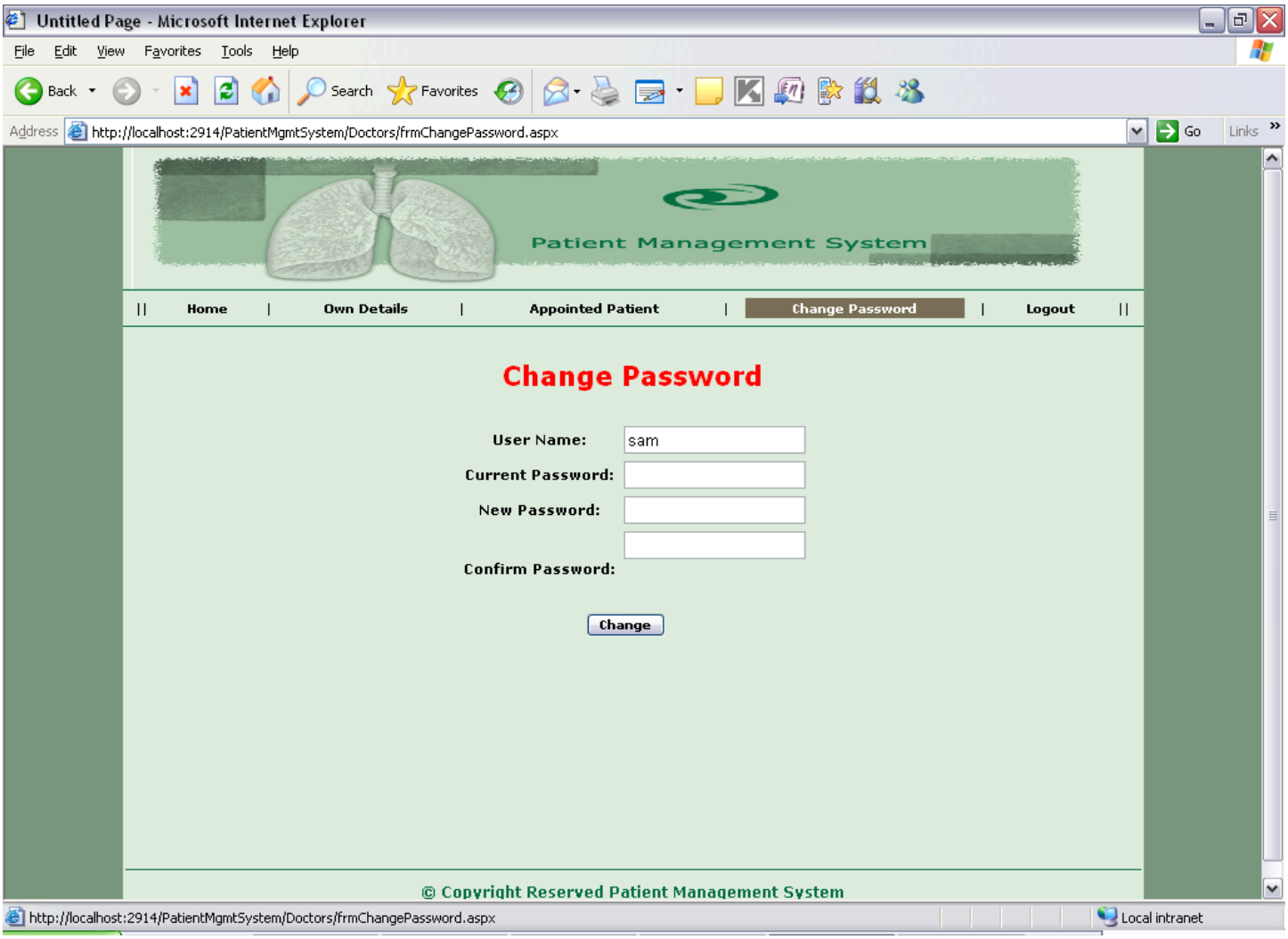

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Admin/Search/frmAllRooms.aspx
Go Links
Patient Management System
Home | Add | Manage | Delete | Search | Logout
Show All Rooms
Show All Room
Room Code | Room Name | Description | Charges
ROOM2 | Cottage Ward | Nothing | 600
ROOM3 | Special With A/C | Nothing | 1000
ROOM4 | Special Without A/C | Nothing | 800
ROOM5 | General | Nothing | 400
Done
Local intranet

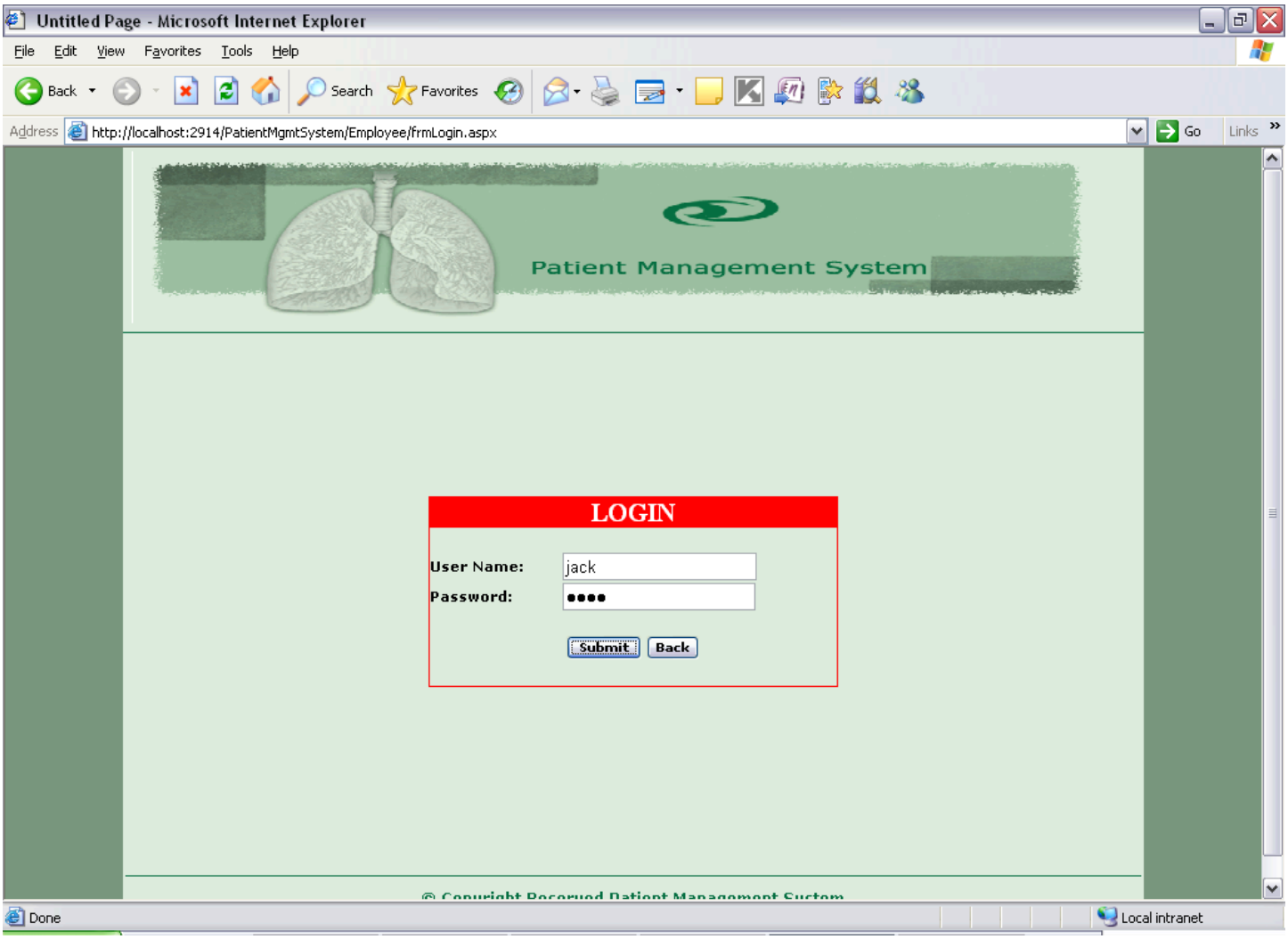

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Doctors/frmLogin.aspx
Go Links
Patient Management System
LOGIN
User Name: sam
Password:
Submit Back
Done
Local intranet

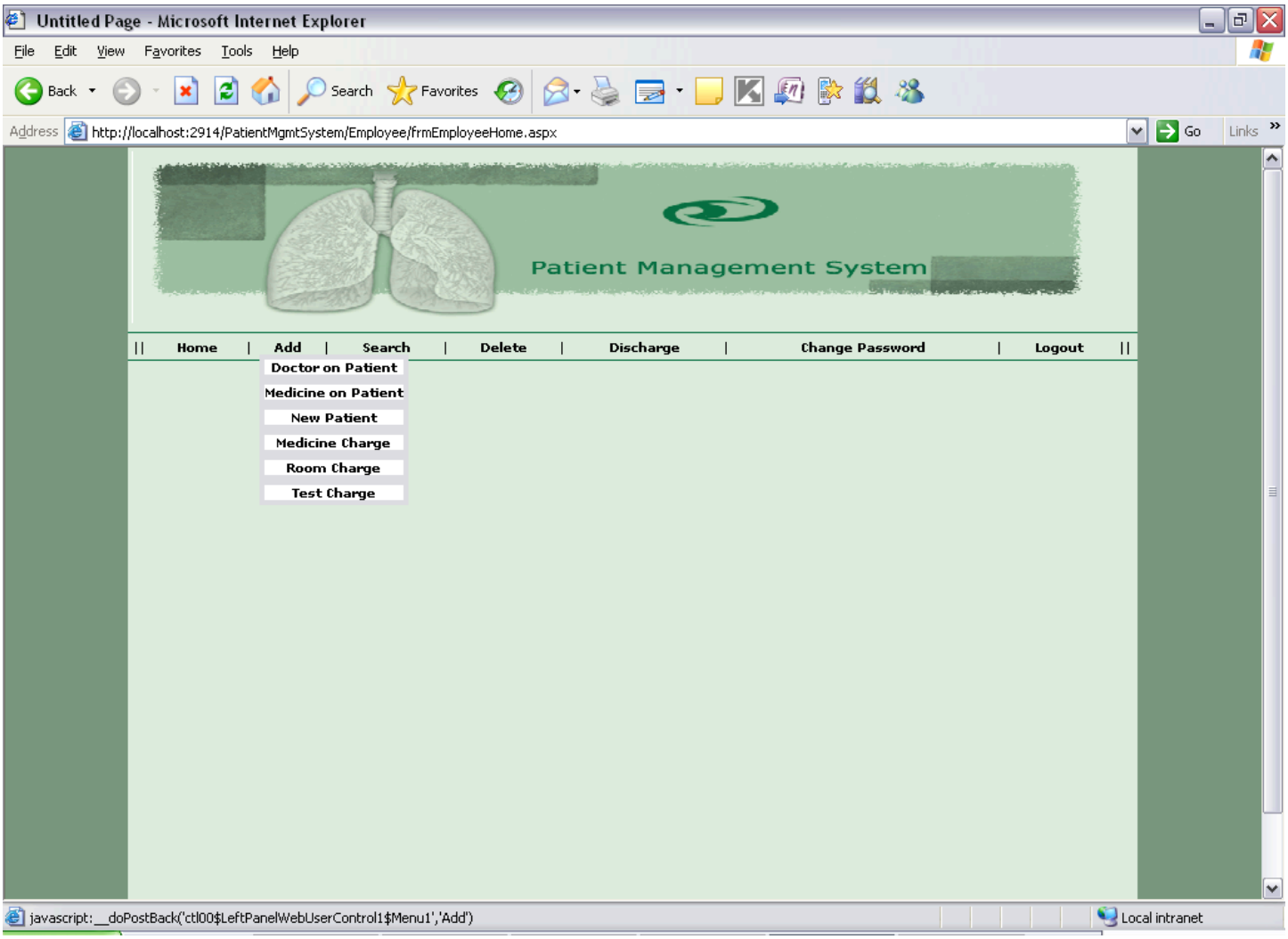

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Doctors/frmOwnDetails.aspx
Go Links
Patient Management System
|| Home | Own Details | Appointed Patient | Change Password | Logout ||
Doctor's Details
Doctor_Code DOC14
Doctor_Name Sam
Specialist_Name Pathologist
TimeFrom 10AM
TimeTo 5AM
ContactNo 5345
Charges 500
Description OK
Detail
© Copyright Reserved Patient Management System
http://localhost:2914/PatientMgmtSystem/Doctors/frmOwnDetails.aspx
Local intranet

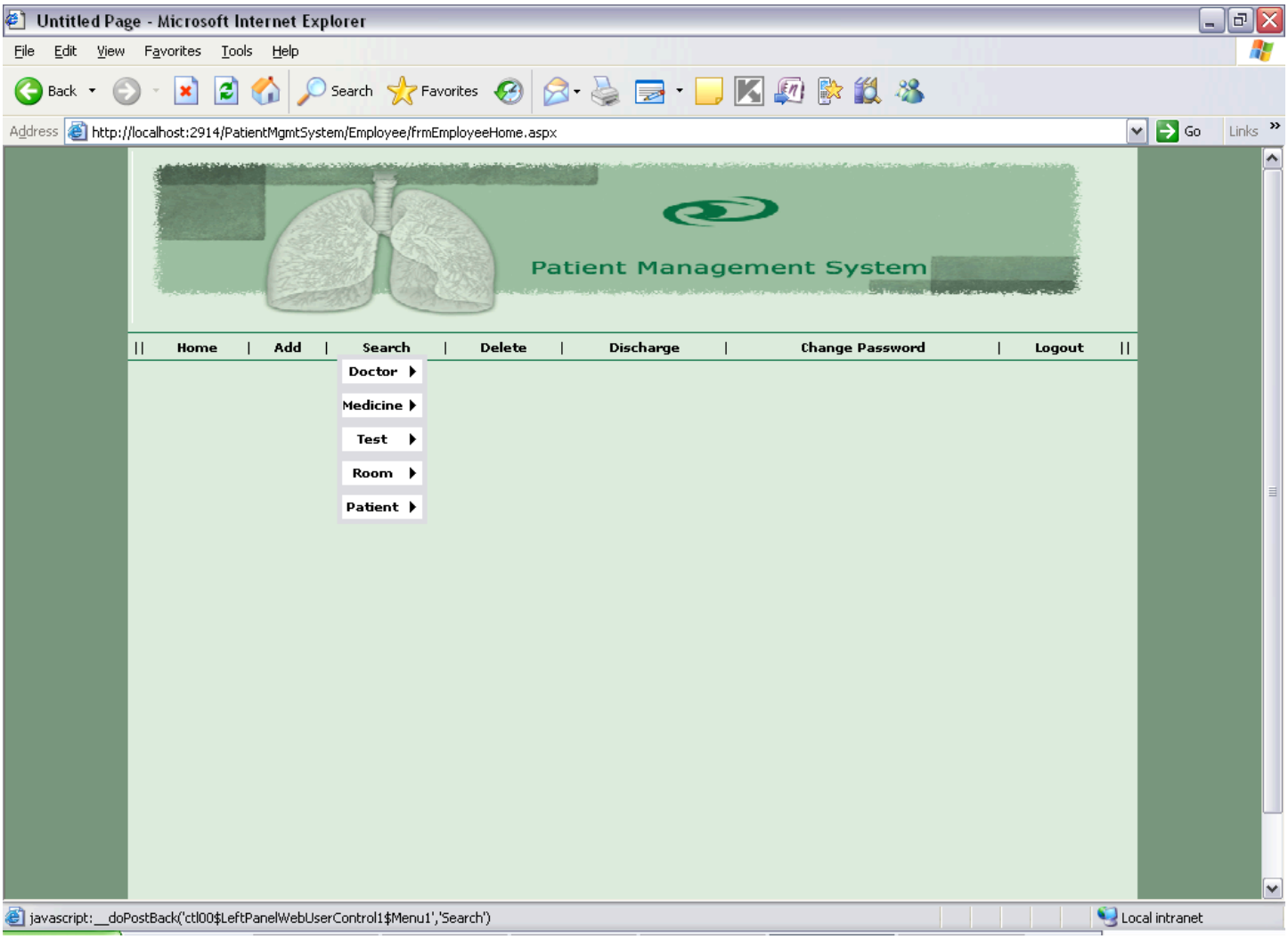

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Doctors/frmOwnDetails.aspx
Go Links
|| Home | Own Details | Appointed Patient | Change Password | Logout ||
Doctor's Details
Doctor_Code DOC14
Doctor_Name Sam
Specialist_Name Pathologist
TimeFrom 10AM
TimeTo 5AM
ContactNo 5345
Charges 500
Description OK
Detail
Please Enter The * Value
Doctor Code: DOC14
Doctor Name: Sam
Specialist: Pathologist
Available Timing: From 10AM To 5AM
Contact No.: 5345
Description: OK
Charges: 500
Modify
© Copyright Reserved Patient Management System
Done
Local intranet

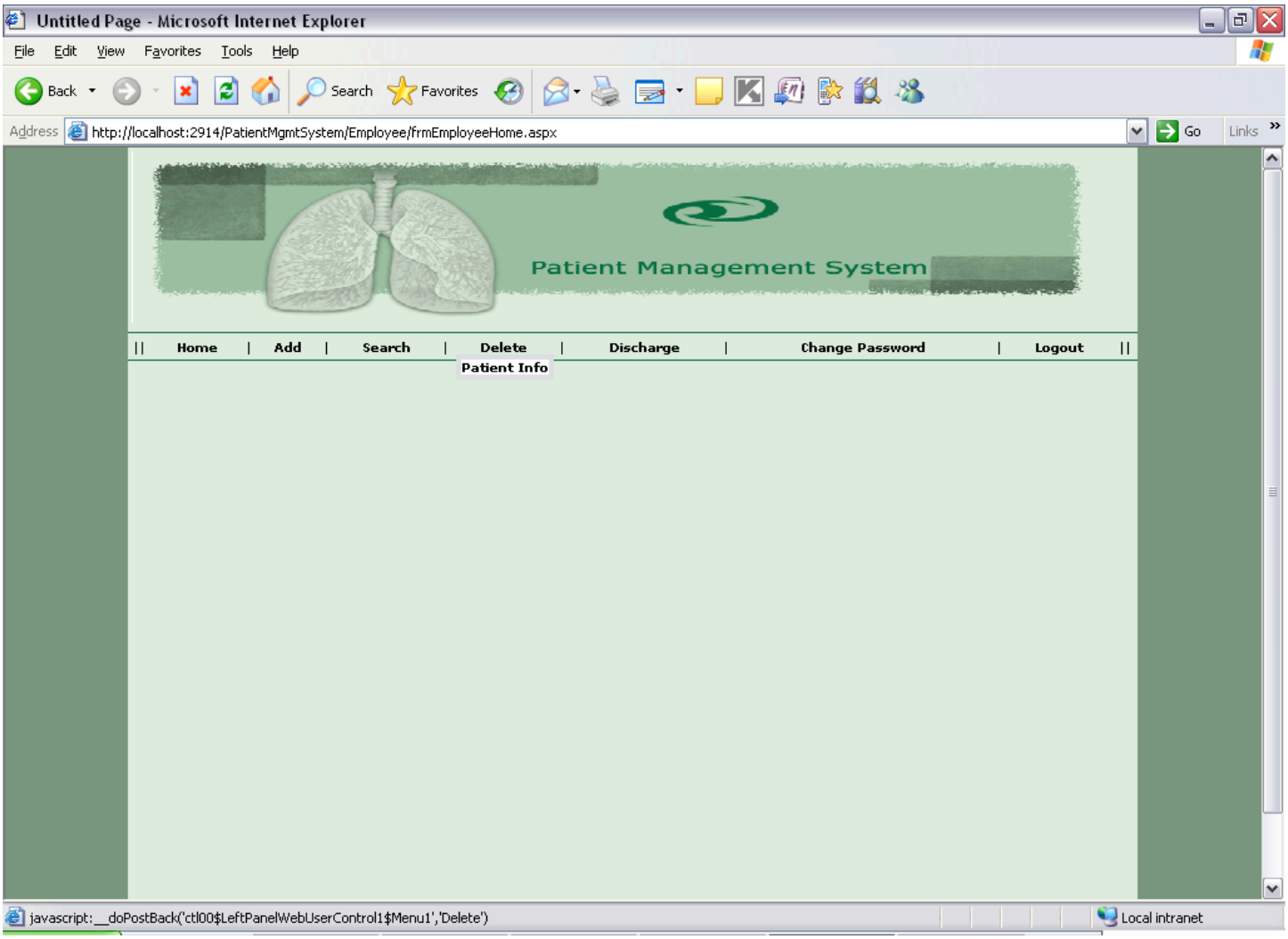

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Doctors/frmShowAppointedPateint.aspx
Go Links
Patient Management System
|| Home | Own Details | Appointed Patient | Change Password | Logout ||
Show Appointed Patient
Select Date: 11/30/2007 Show
No Data Found
© Copyright Reserved Patient Management System
Done
Local intranet

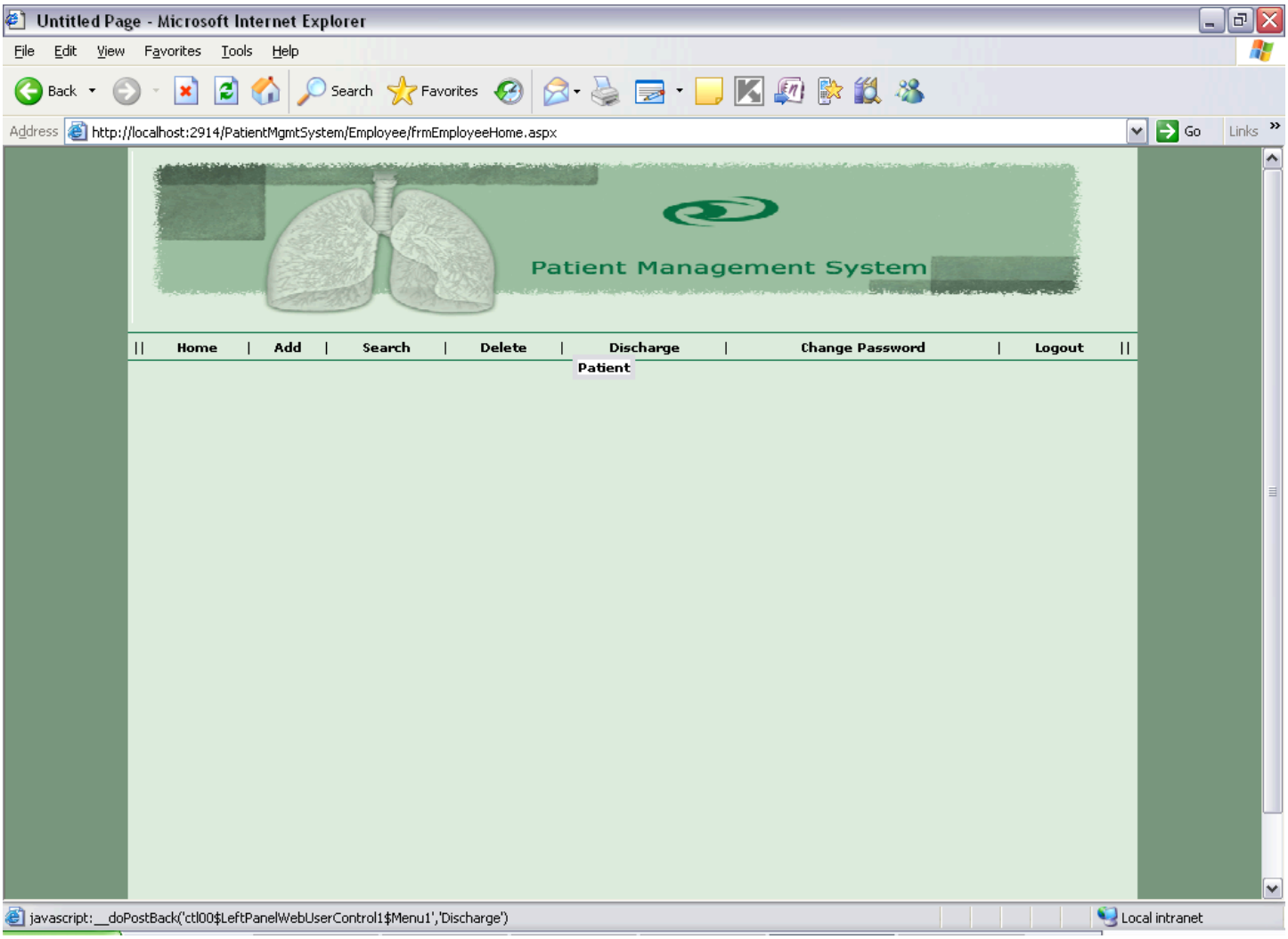

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Doctors/frmChangePassword.aspx
Go Links
Patient Management System
|| Home | Own Details | Appointed Patient | Change Password | Logout ||
Change Password
User Name: sam
Current Password:
New Password:
Confirm Password:
Change
© Copyright Reserved Patient Management System
http://localhost:2914/PatientMgmtSystem/Doctors/frmChangePassword.aspx
Local intranet

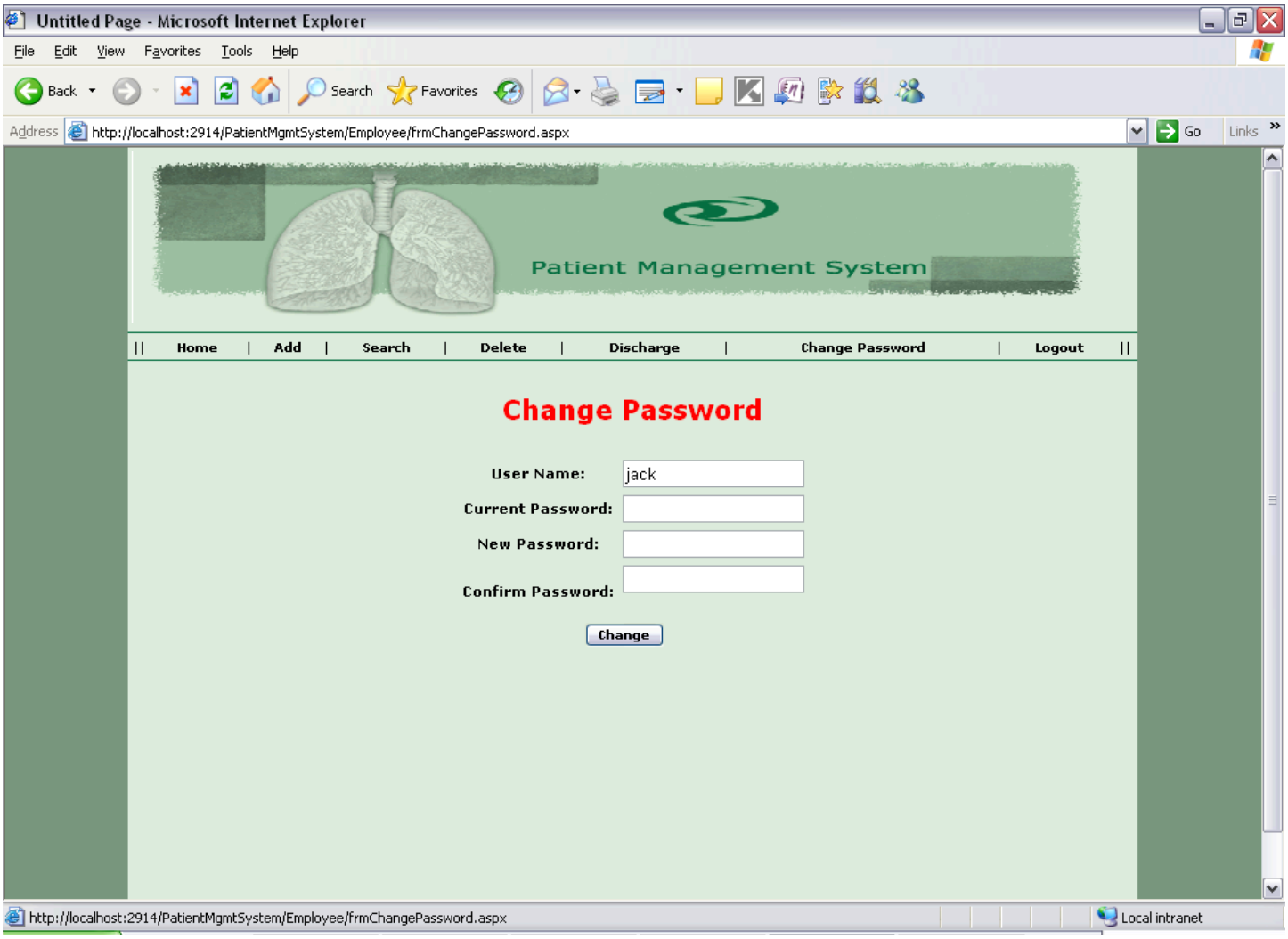

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmLogin.aspx
Go Links
Patient Management System
LOGIN
User Name: jack
Password:
Submit Back
Done
Local intranet

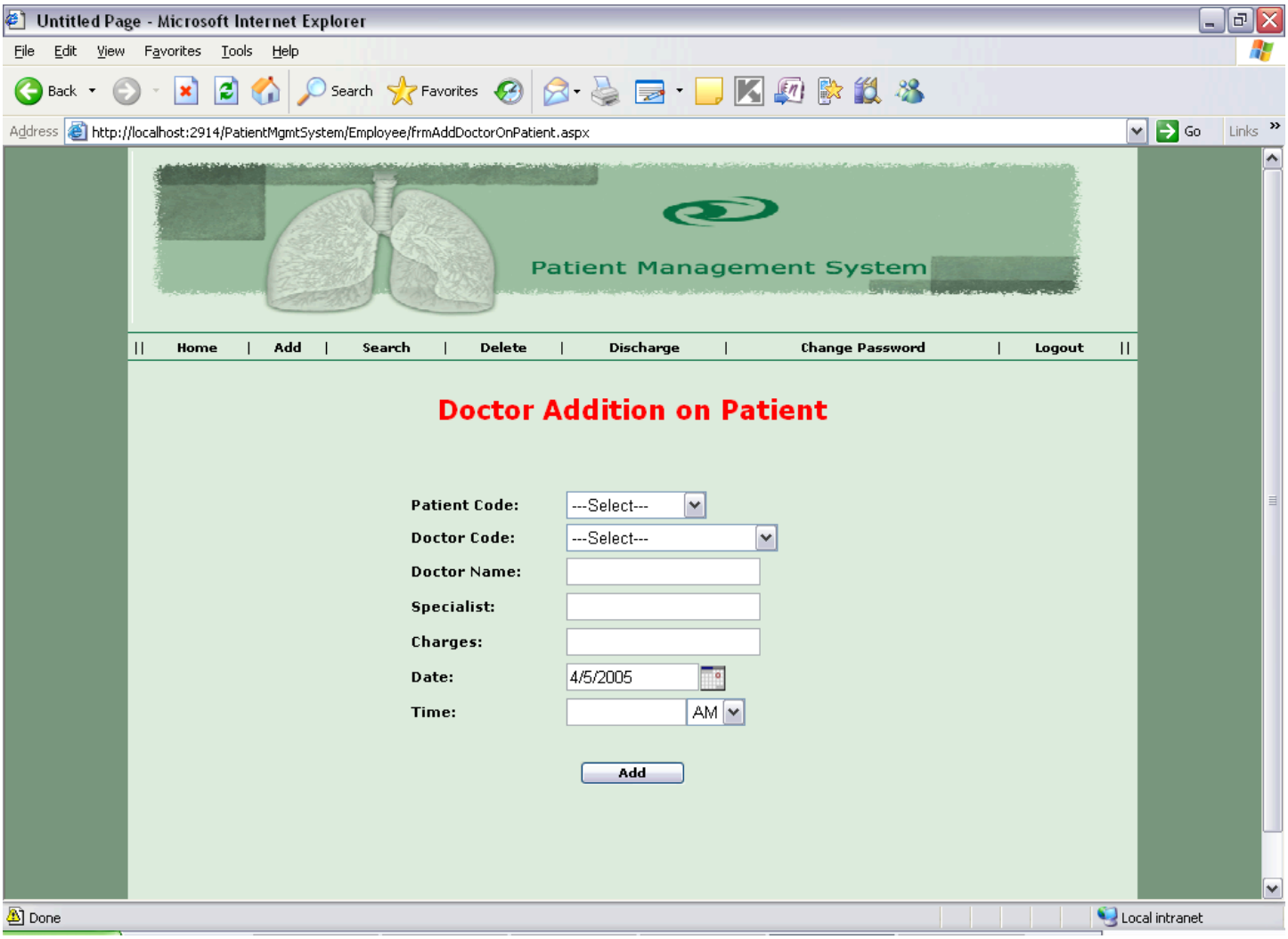

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmEmployeeHome.aspx
Go Links
Patient Management System
Home | Add | Search | Delete | Discharge | Change Password | Logout
Doctor on Patient
Medicine on Patient
New Patient
Medicine Charge
Room Charge
Test Charge
javascript:__doPostBack('ctl00$LeftPanelWebUserControl1$Menu1','Add')
Local intranet

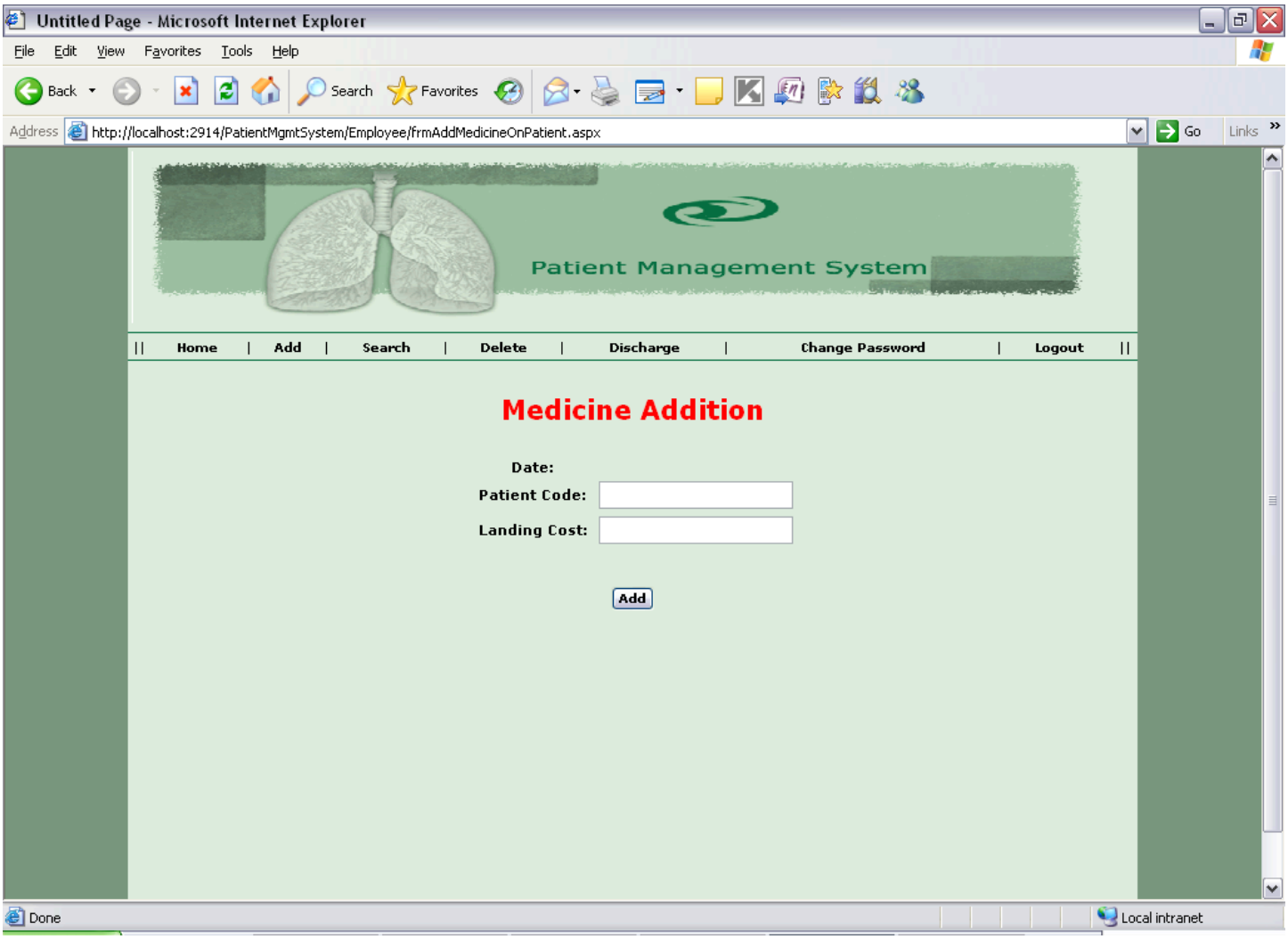

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmEmployeeHome.aspx
Go Links
Patient Management System
Home | Add | Search | Delete | Discharge | Change Password | Logout
Doctor
Medicine
Test
Room
Patient
javascript:__doPostBack('ctl00$LeftPanelWebUserControl1$Menu1','Search')
Local intranet

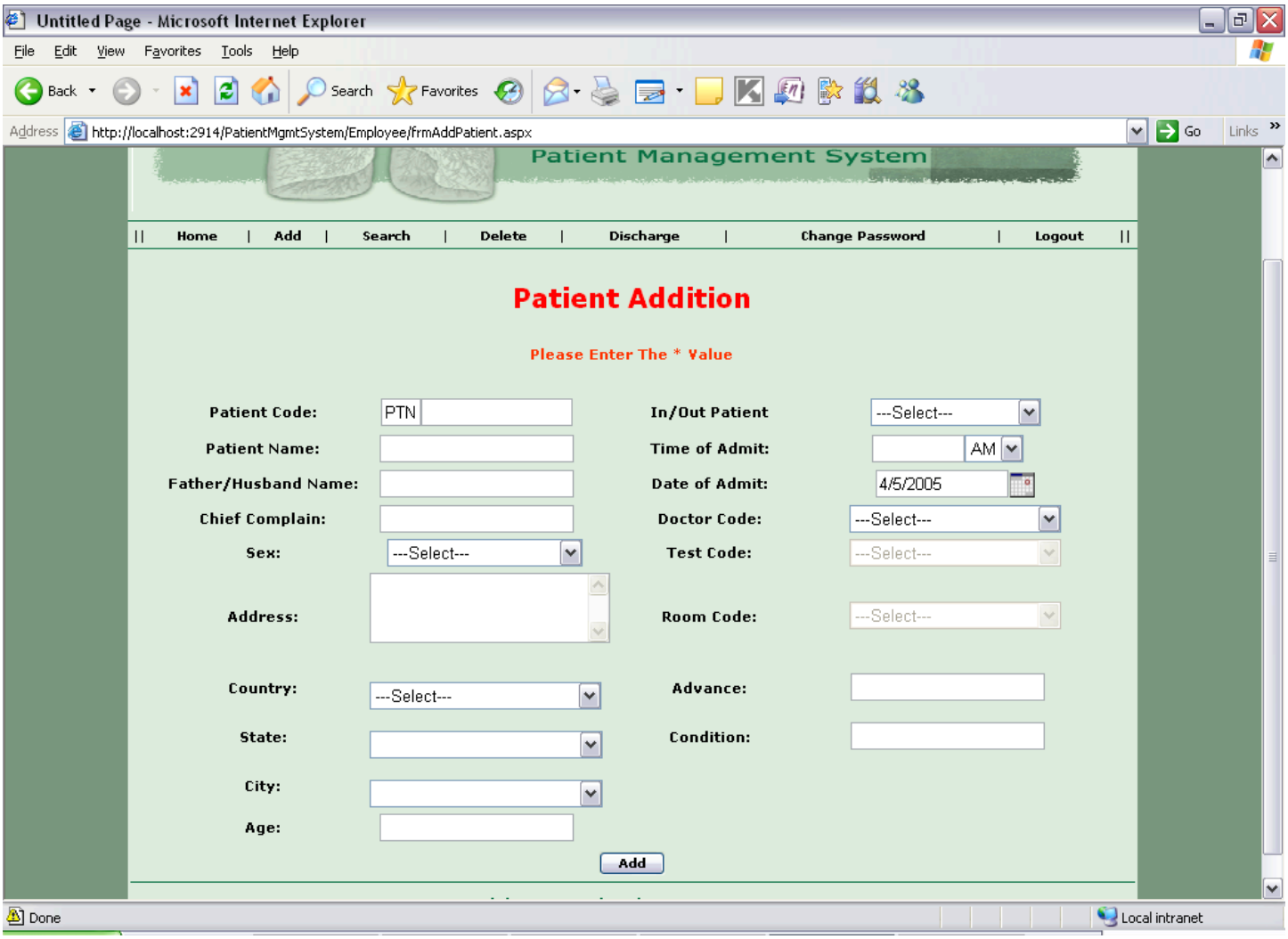

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmEmployeeHome.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Patient Info
javascript:__doPostBack('ctl00$LeftPanelWebUserControl1$Menu1','Delete')
Local intranet

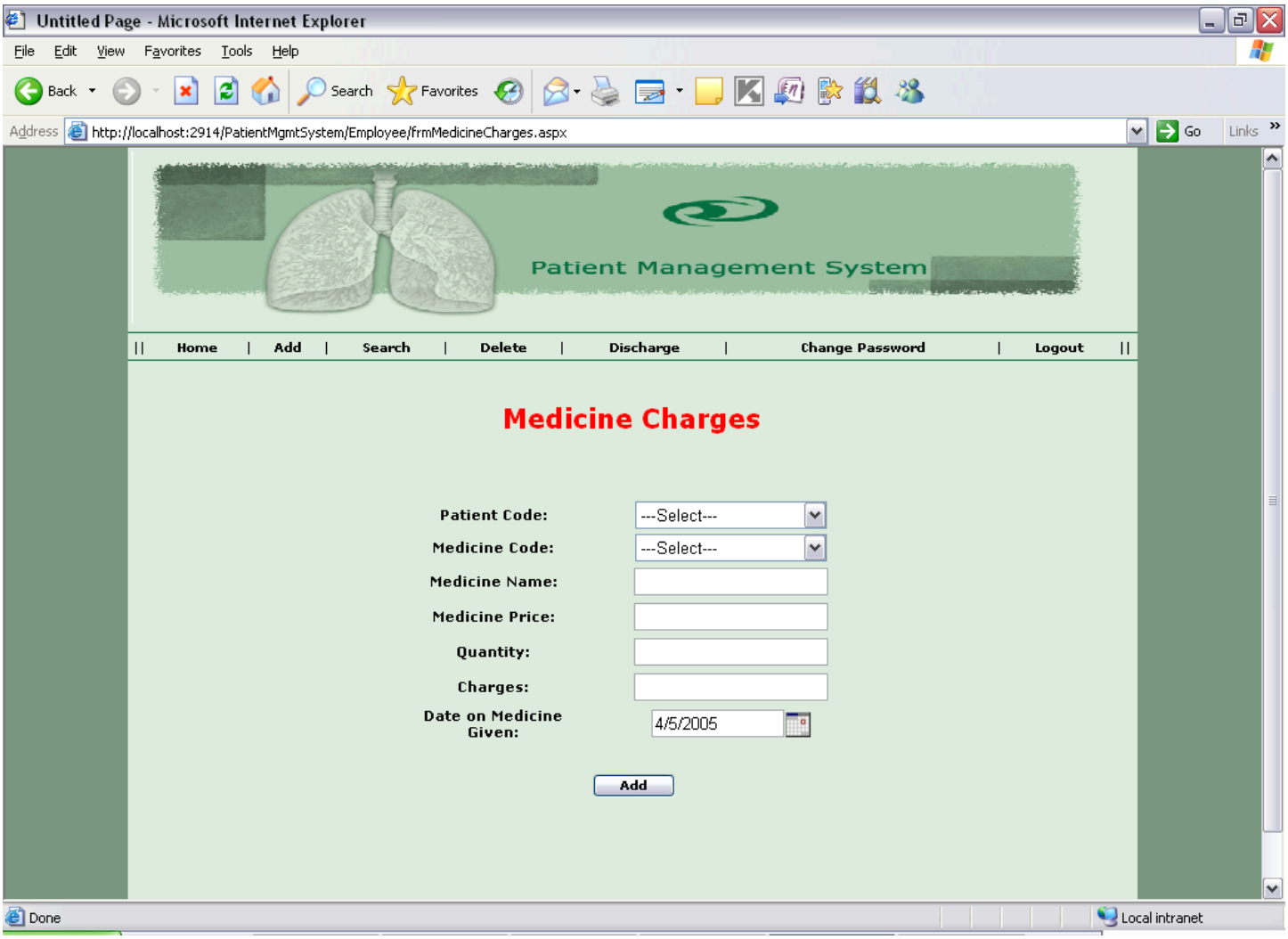

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmEmployeeHome.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge Patient | Change Password | Logout ||
javascript:__doPostBack('ctl00$LeftPanelWebUserControl1$Menu1','Discharge')
Local intranet

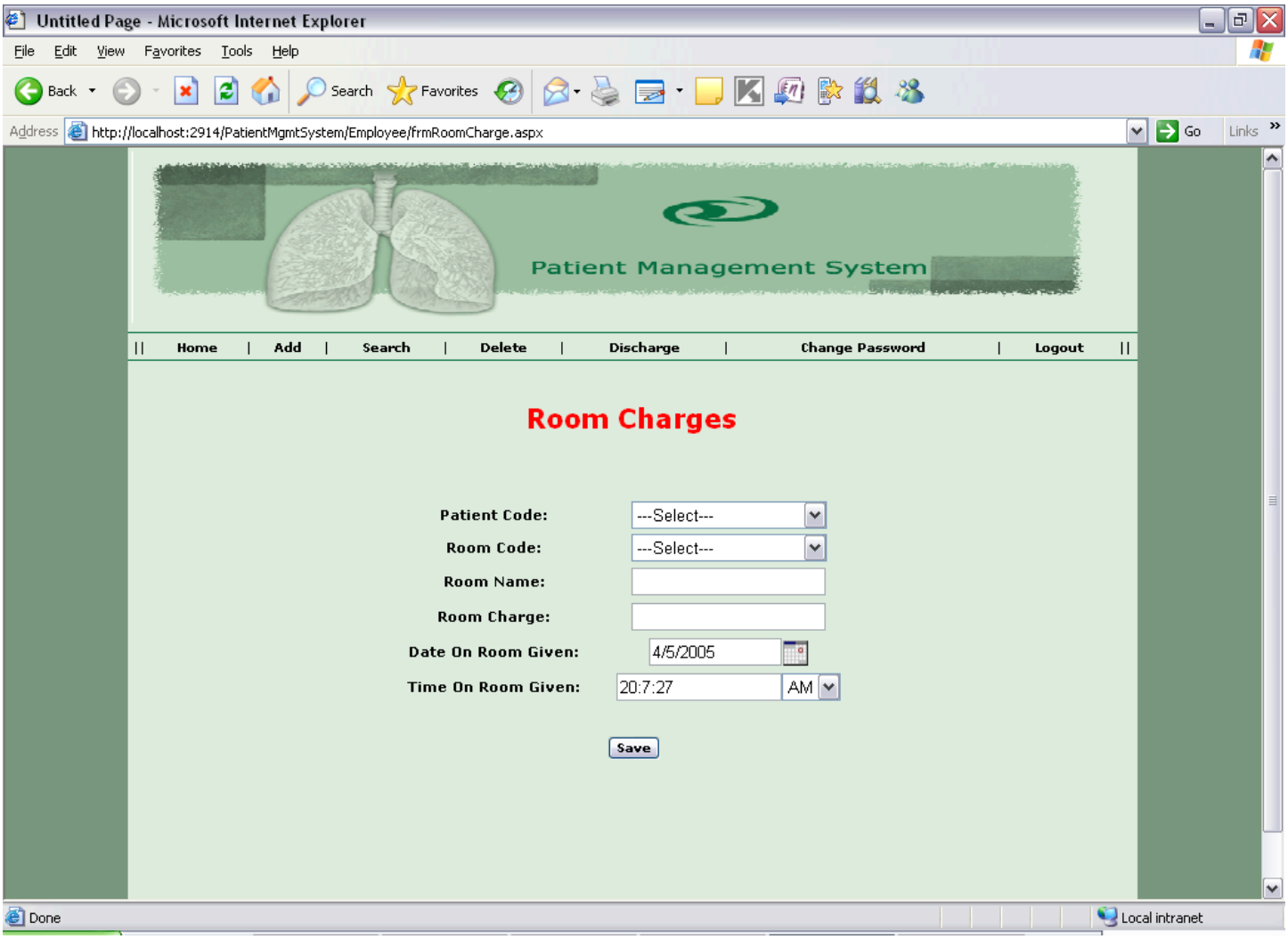

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmChangePassword.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Change Password
User Name: jack
Current Password:
New Password:
Confirm Password:
Change
http://localhost:2914/PatientMgmtSystem/Employee/frmChangePassword.aspx
Local intranet

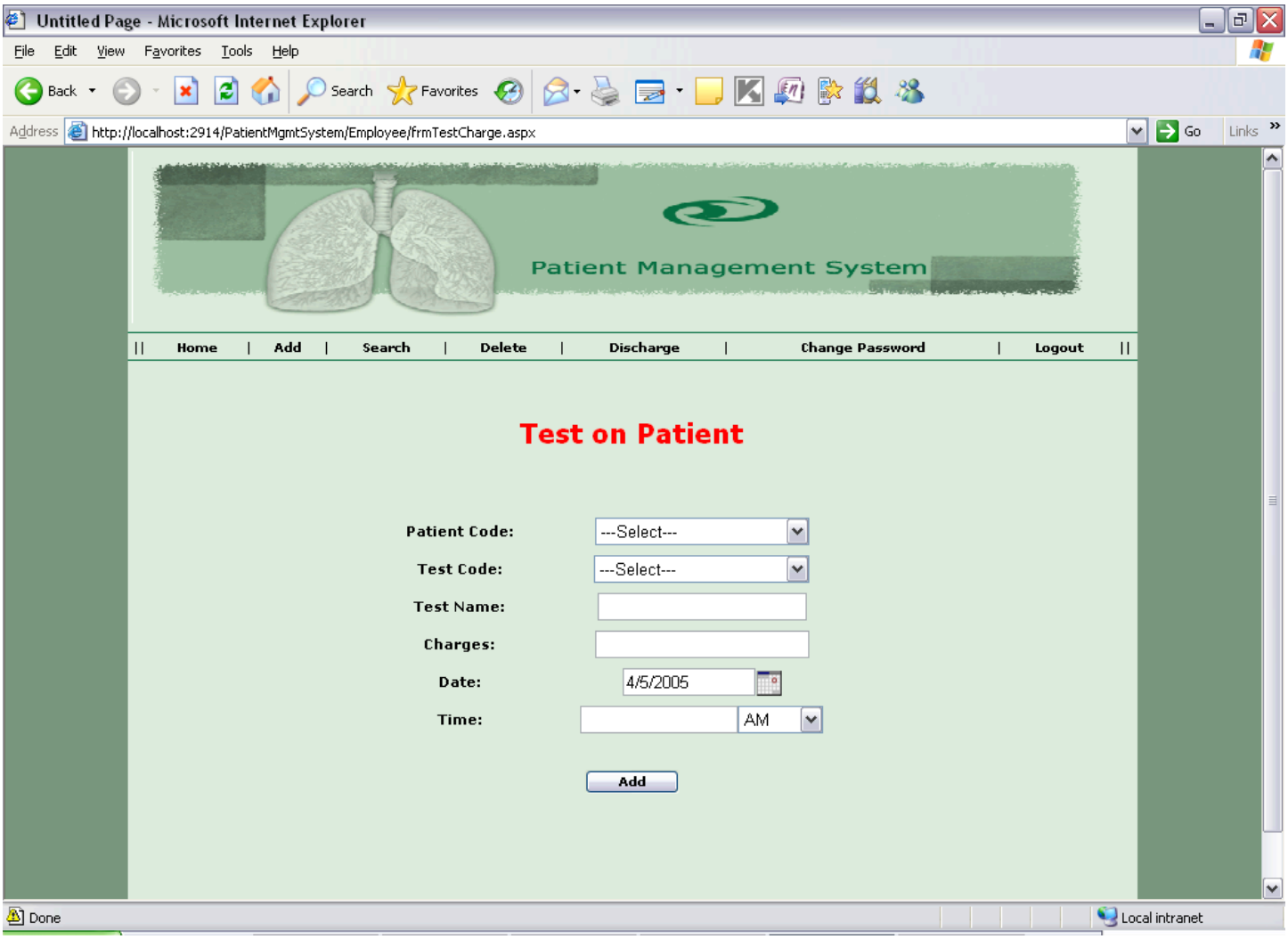

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmAddDoctorOnPatient.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Doctor Addition on Patient
Patient Code: ---Select---
Doctor Code: ---Select---
Doctor Name:
Specialist:
Charges:
Date: 4/5/2005
Time: AM
Add
Done
Local intranet

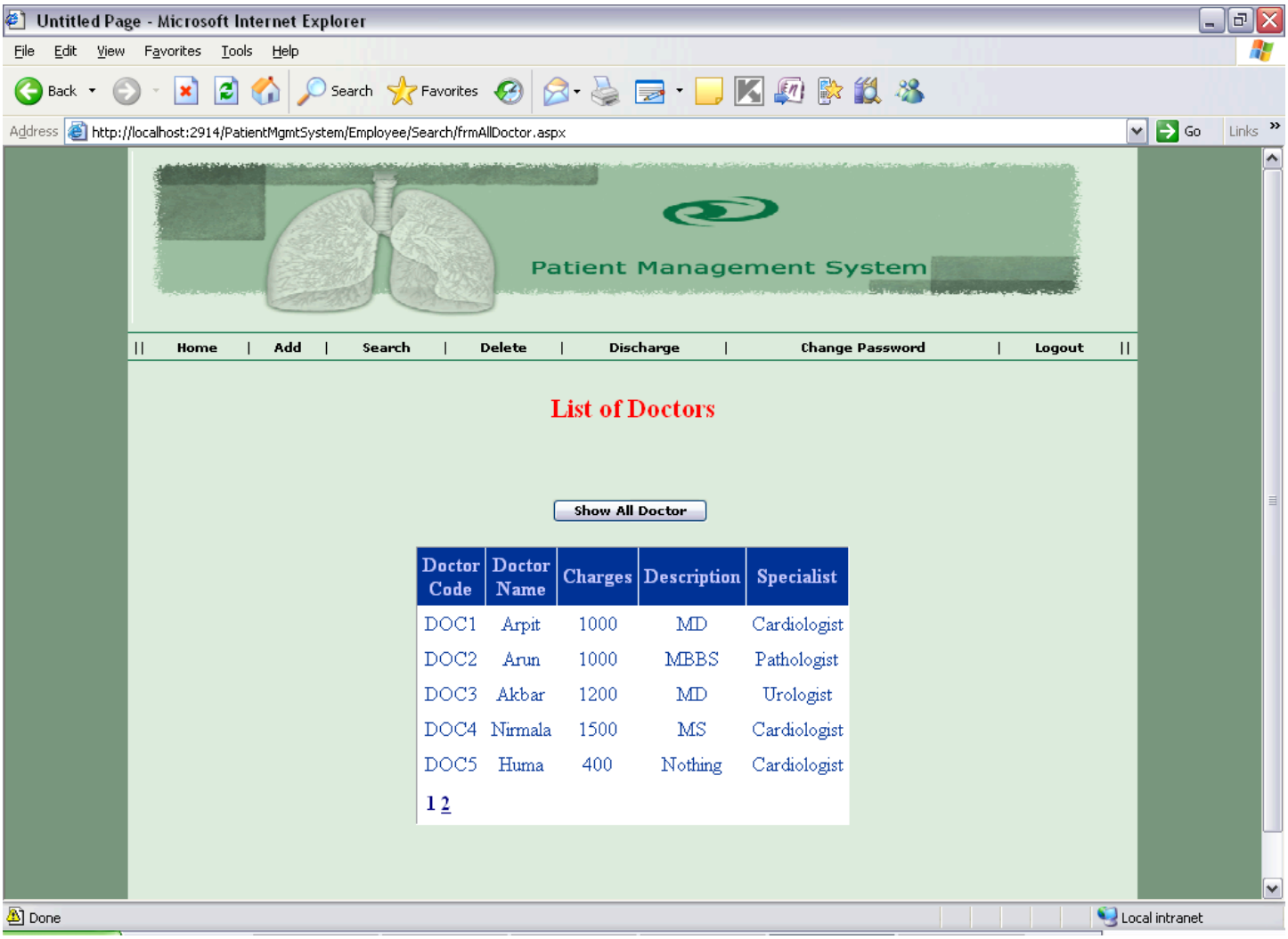

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmAddMedicineOnPatient.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Medicine Addition
Date:
Patient Code:
Landing Cost:
Add
Done
Local intranet

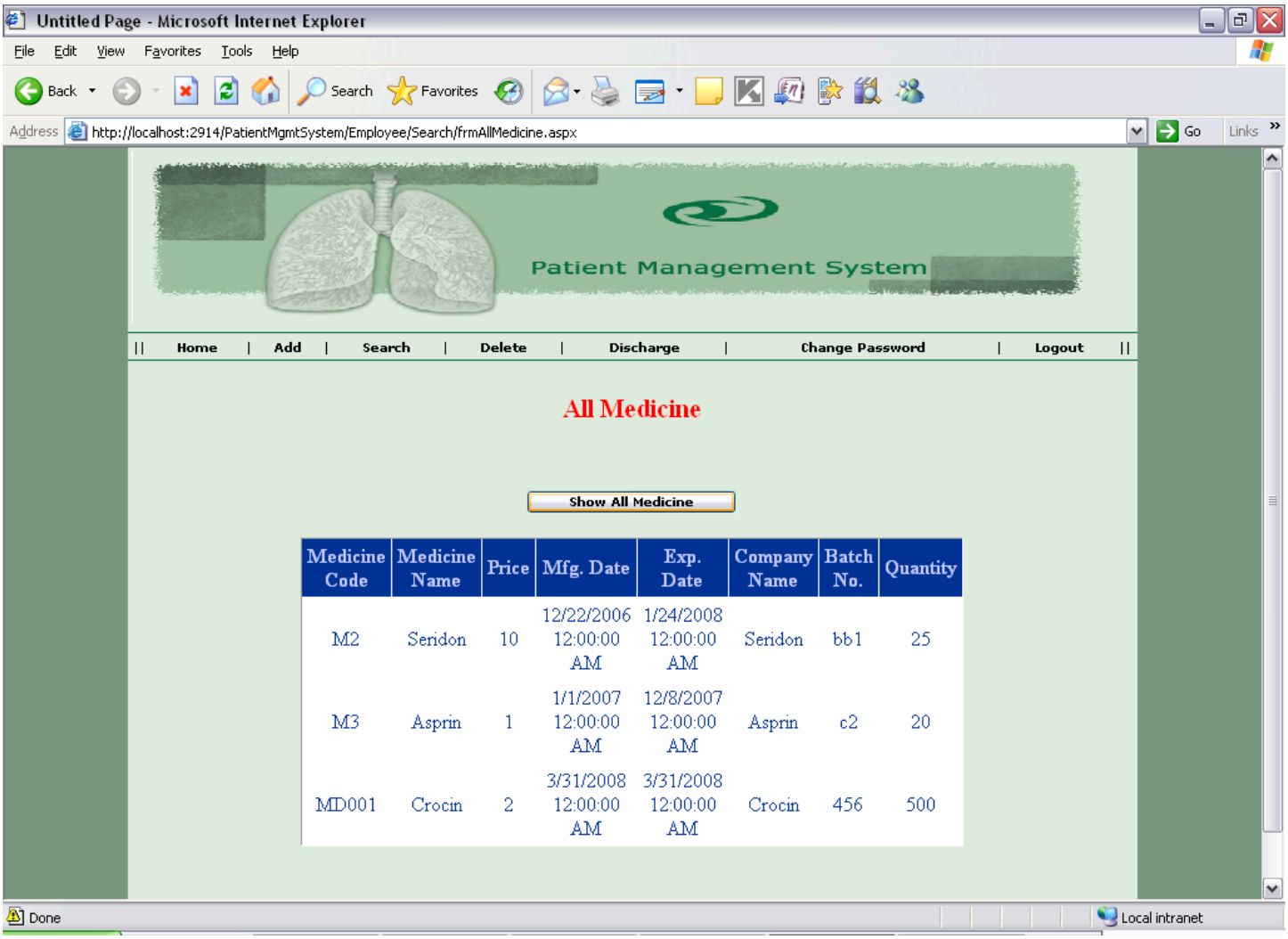

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmAddPatient.aspx
Go Links
Patient Management System
Home | Add | Search | Delete | Discharge | Change Password | Logout
Patient Addition
Please Enter The * Value
Patient Code: PTN
Patient Name:
Father/Husband Name:
Chief Complain:
Sex: ---Select---
Address:
Country: ---Select---
State:
City:
Age:
In/Out Patient ---Select---
Time of Admit: AM
Date of Admit: 4/5/2005
Doctor Code: ---Select---
Test Code: ---Select---
Room Code: ---Select---
Advance:
Condition:
Add
Done
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmMedicineCharges.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Medicine Charges
Patient Code:
---Select---
Medicine Code:
---Select---
Medicine Name:
Medicine Price:
Quantity:
Charges:
Date on Medicine Given:
4/5/2005
Add
Done
Local intranet

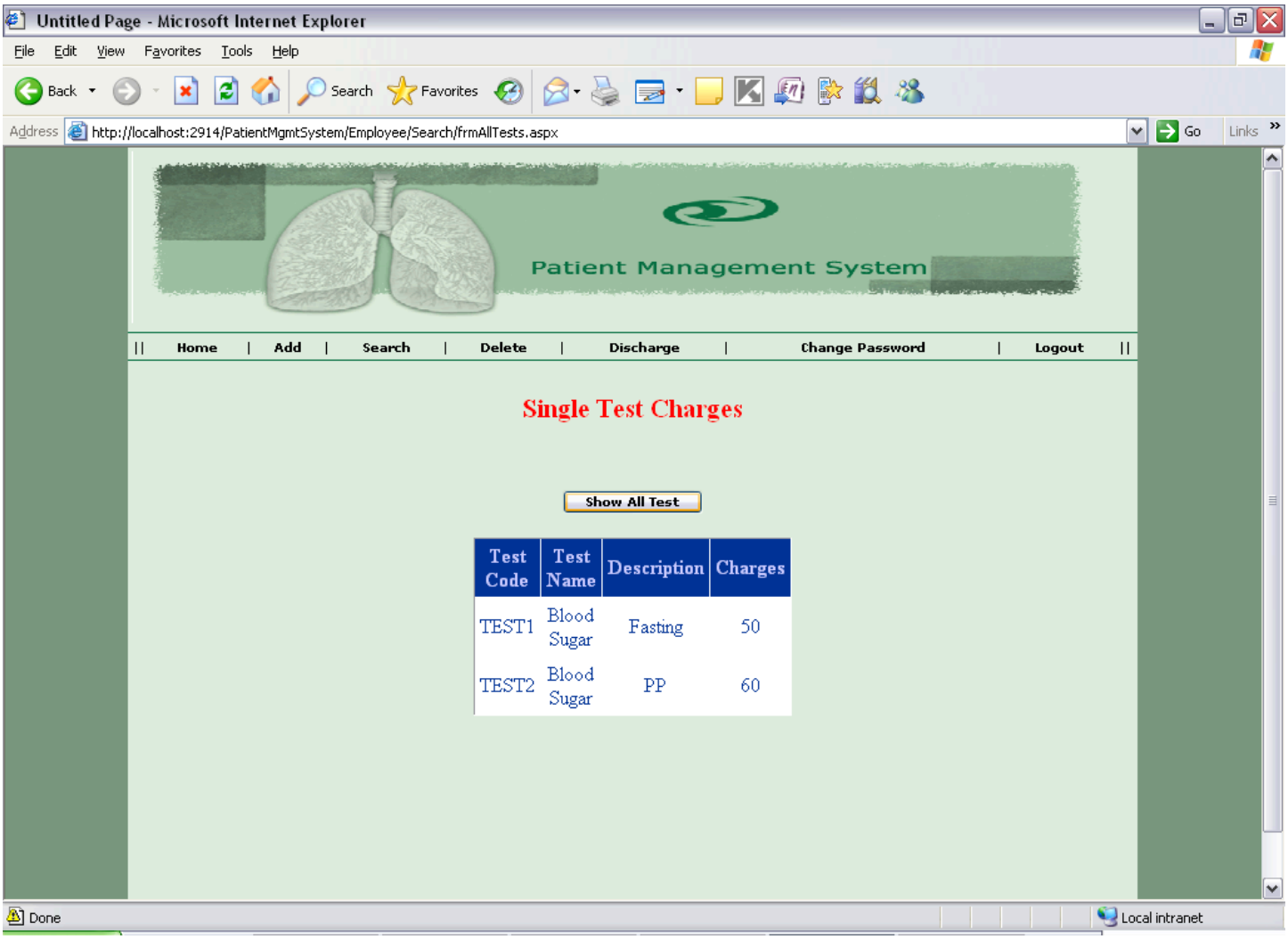

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmRoomCharge.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Room Charges
Patient Code: ---Select---
Room Code: ---Select---
Room Name:
Room Charge:
Date On Room Given: 4/5/2005
Time On Room Given: 20:7:27 AM
Save
Done
Local intranet

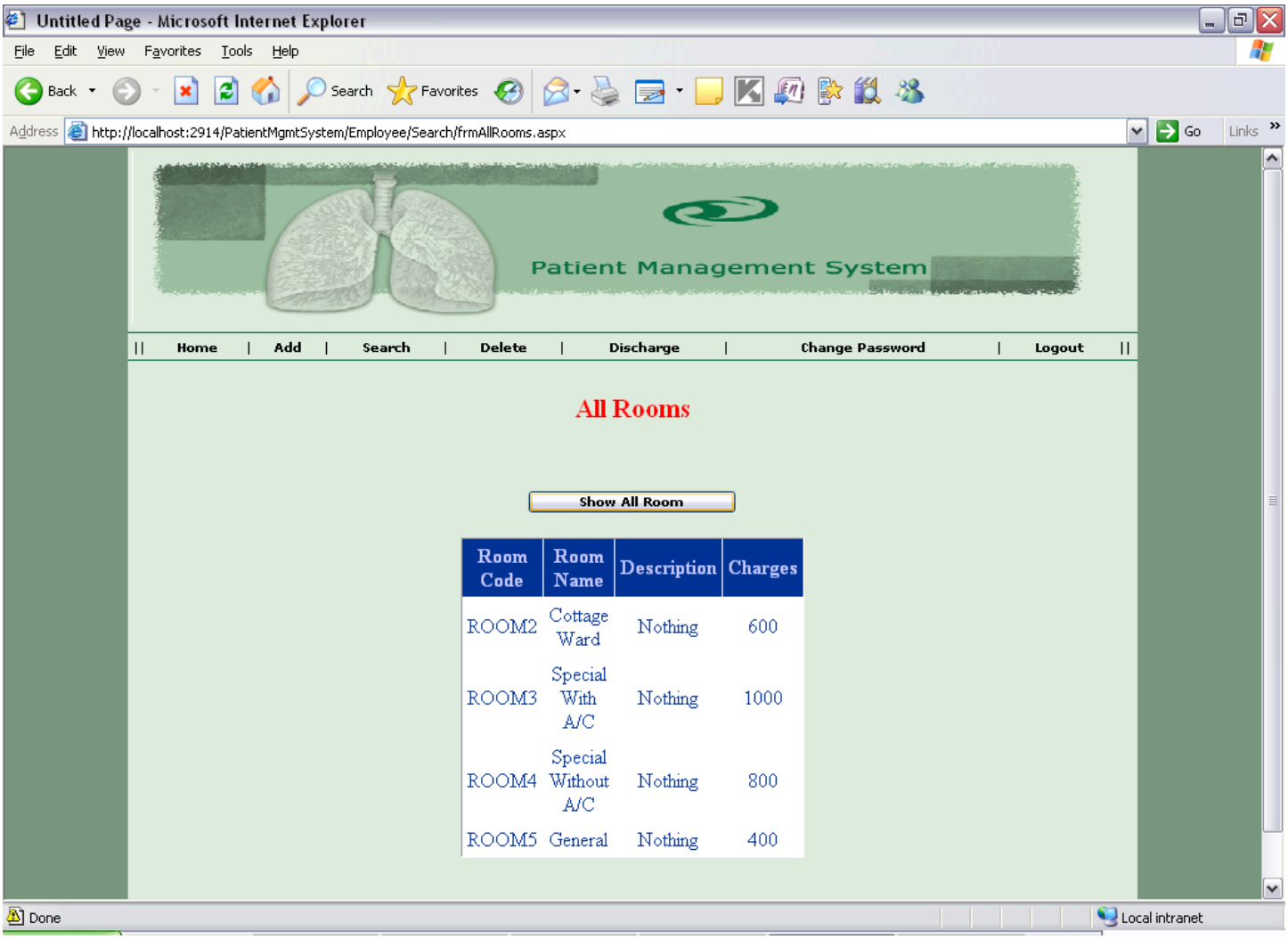

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmTestCharge.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Test on Patient
Patient Code: ---Select---
Test Code: ---Select---
Test Name:
Charges:
Date: 4/5/2005
Time: AM
Add
Done
Local intranet

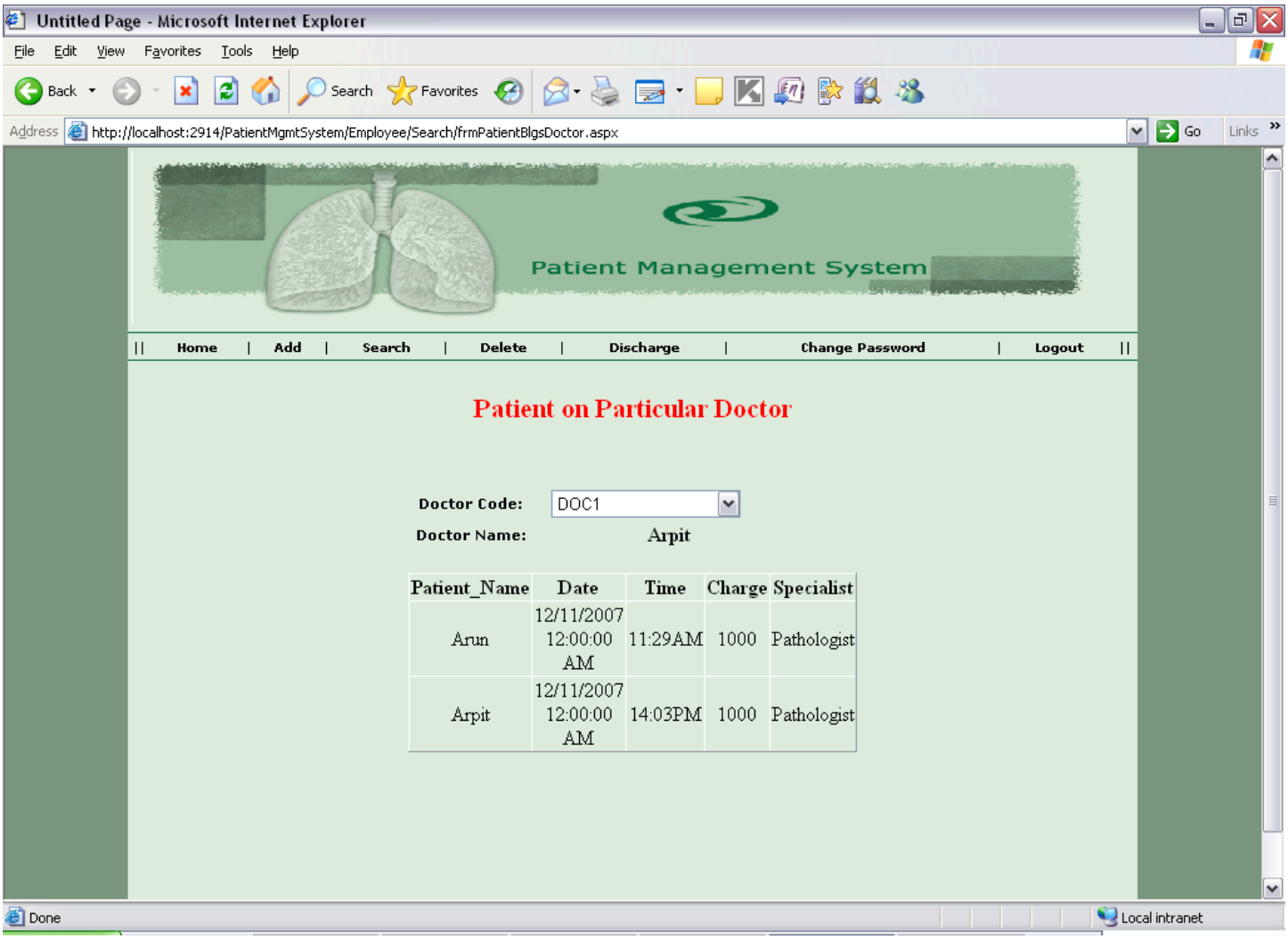

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/Search/frmAllDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
List of Doctors
Show All Doctor
Doctor Code | Doctor Name | Charges | Description | Specialist
DOC1 | Arpit | 1000 | MD | Cardiologist
DOC2 | Arun | 1000 | MBBS | Pathologist
DOC3 | Akbar | 1200 | MD | Urologist
DOC4 | Nirmala | 1500 | MS | Cardiologist
DOC5 | Huma | 400 | Nothing | Cardiologist
1 2
Done
Local intranet

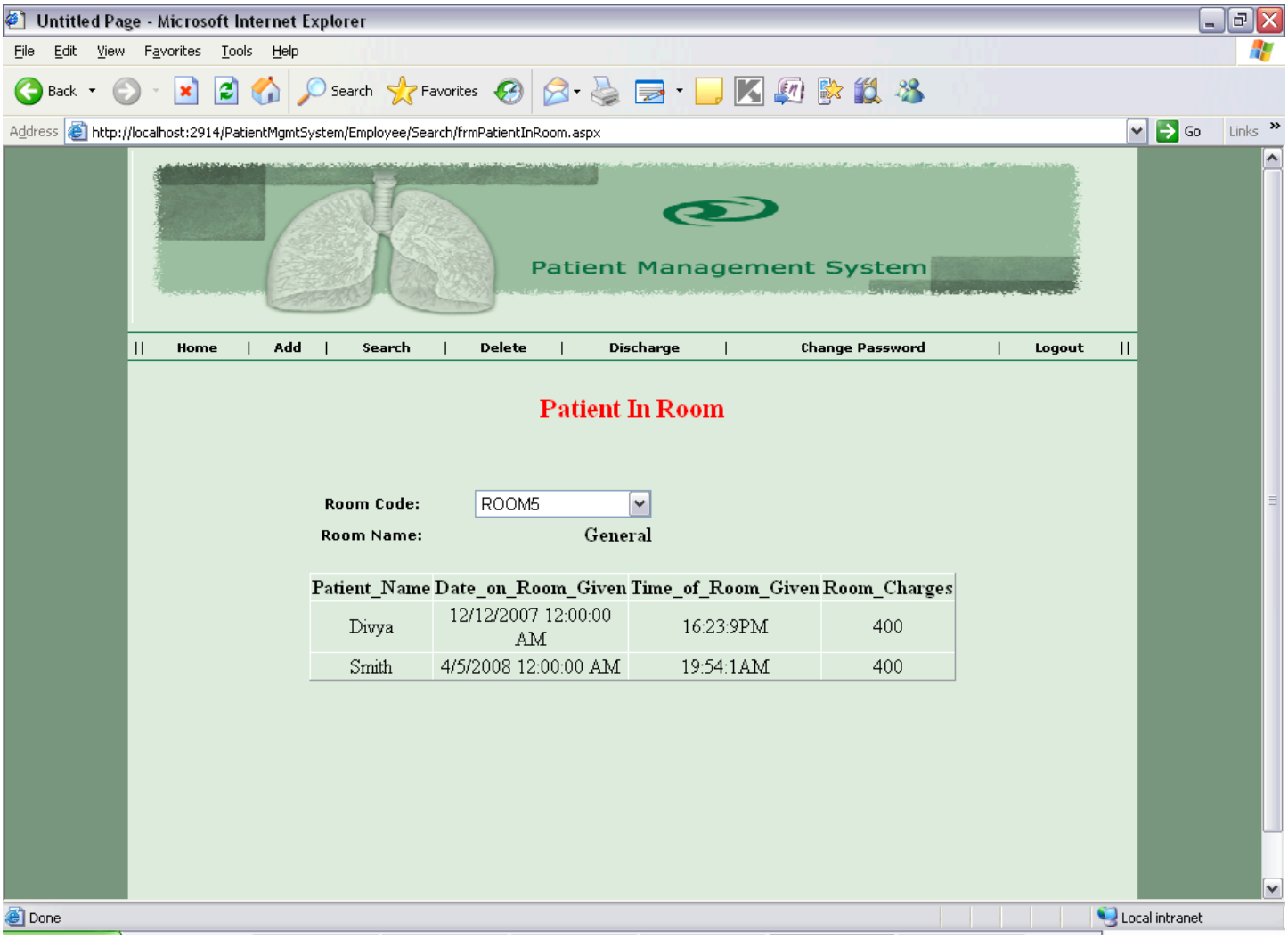


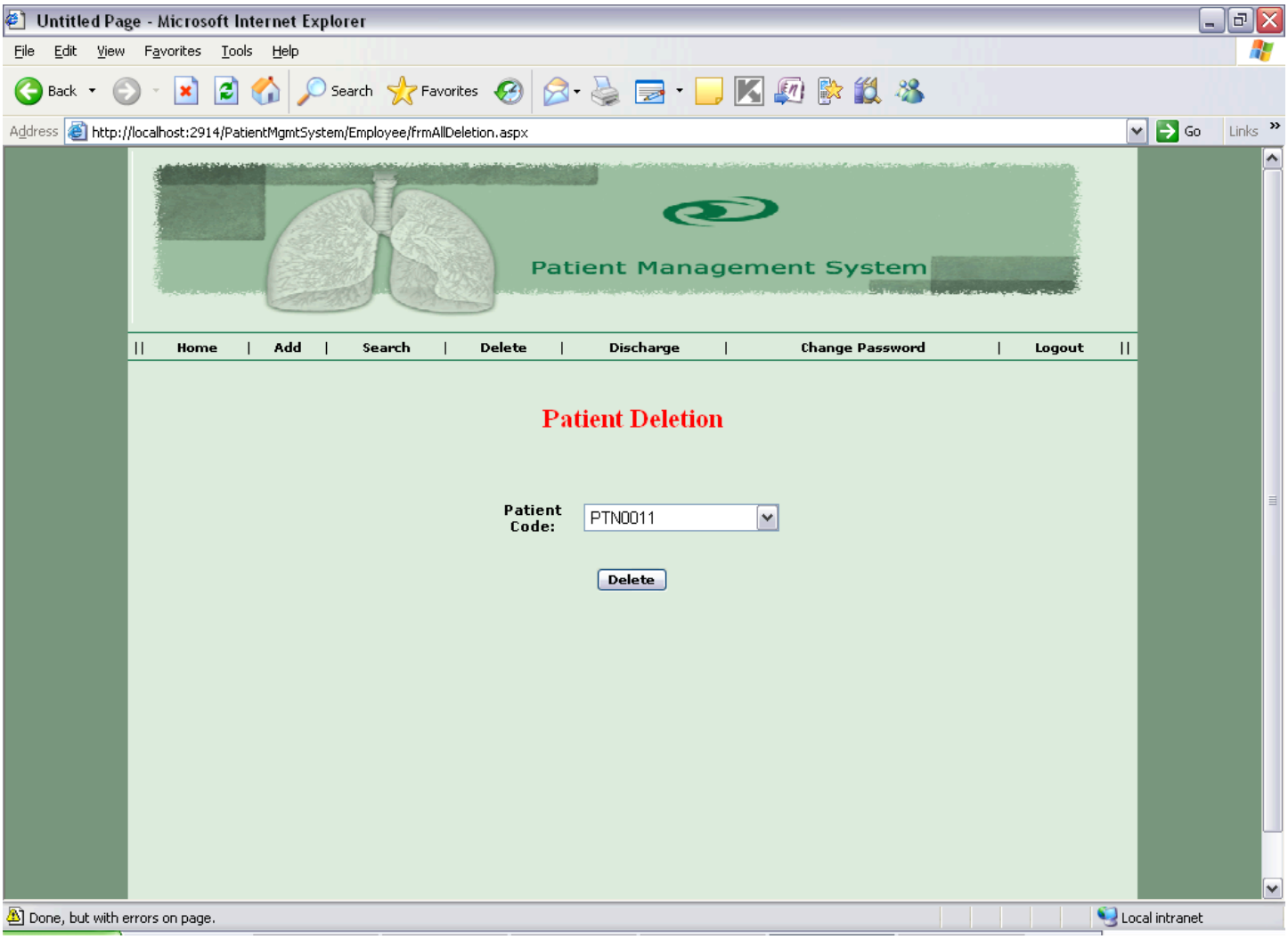
Medicine Code	Medicine Name	Price	Mfg. Date	Exp. Date	Company Name	Batch No.	Quantity
M2	Seridon	10	12/22/2006 12:00:00 AM	1/24/2008 12:00:00 AM	Seridon	bb1	25
M3	Asprin	1	1/1/2007 12:00:00 AM	12/8/2007 12:00:00 AM	Asprin	c2	20
MD001	Crocin	2	3/31/2008 12:00:00 AM	3/31/2008 12:00:00 AM	Crocin	456	500

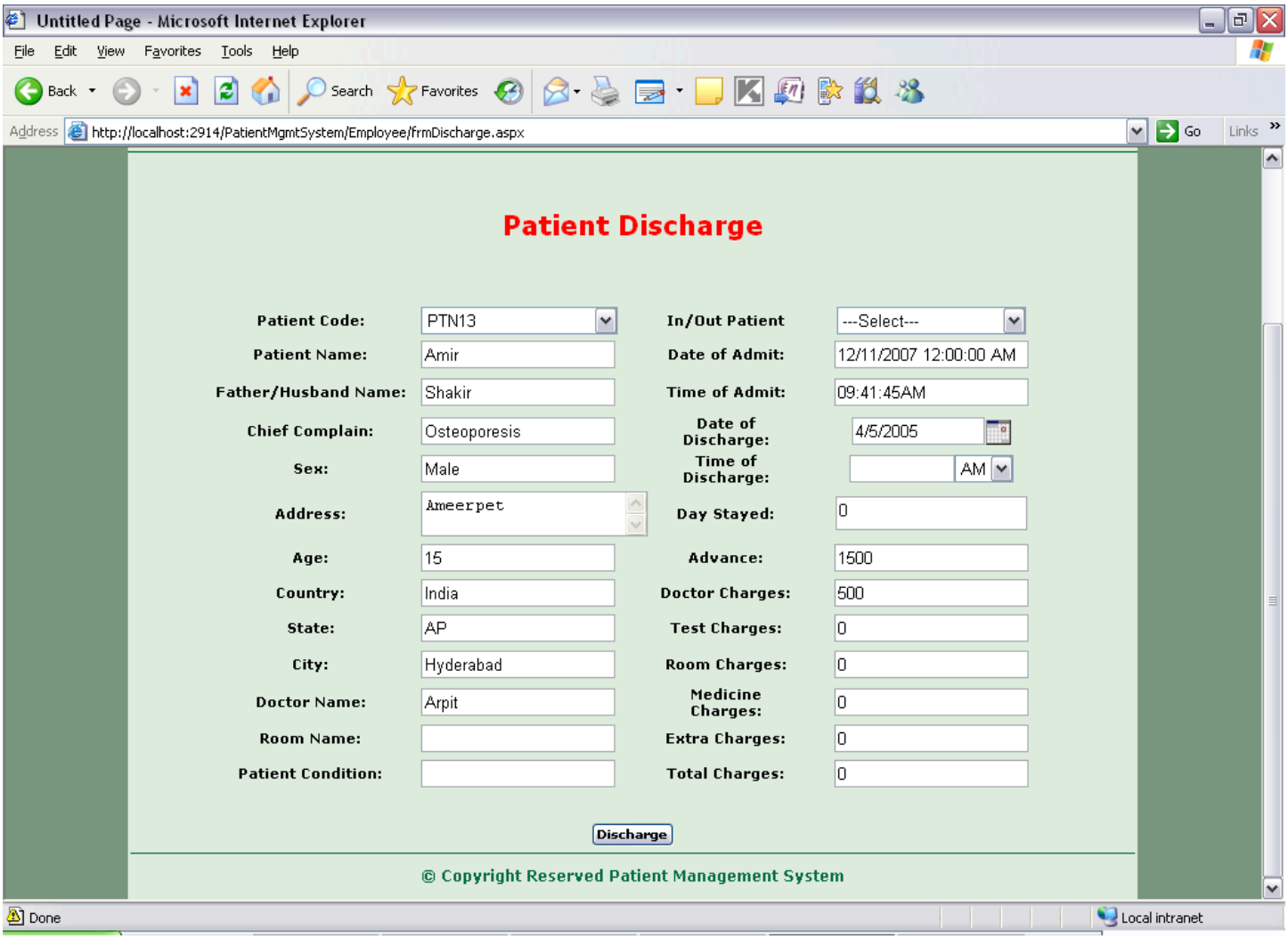

Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/Search/frmAllTests.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Single Test Charges
Show All Test
Test Code
Test Name
Description
Charges
TEST1
Blood Sugar
Fasting
50
TEST2
Blood Sugar
PP
60
Done
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/Search/frmAllRooms.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
All Rooms
Show All Room
Room Code | Room Name | Description | Charges
ROOM2 | Cottage Ward | Nothing | 600
ROOM3 | Special With A/C | Nothing | 1000
ROOM4 | Special Without A/C | Nothing | 800
ROOM5 | General | Nothing | 400
Done
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/Search/frmPatientBlgsDoctor.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Patient on Particular Doctor
Doctor Code: DOC1
Doctor Name: Arpit
Patient_Name Date Time Charge Specialist
Arun 12/11/2007 12:00:00 AM 11:29AM 1000 Pathologist
Arpit 12/11/2007 12:00:00 AM 14:03PM 1000 Pathologist
Done
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/Search/frmPatientInRoom.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Patient In Room
Room Code: ROOM5
Room Name: General
Patient_Name Date_on_Room_Given Time_of_Room_Given Room_Charges
Divya 12/12/2007 12:00:00 AM 16:23:9PM 400
Smith 4/5/2008 12:00:00 AM 19:54:1AM 400
Done
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmAllDeletion.aspx
Go Links
Patient Management System
|| Home | Add | Search | Delete | Discharge | Change Password | Logout ||
Patient Deletion
Patient Code:
PTN0011
Delete
Done, but with errors on page.
Local intranet


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:2914/PatientMgmtSystem/Employee/frmDischarge.aspx
Go Links
Patient Discharge
Patient Code: PTN13
Patient Name: Amir
Father/Husband Name: Shakir
Chief Complain: Osteoporesis
Sex: Male
Address: Ameerpet
Age: 15
Country: India
State: AP
City: Hyderabad
Doctor Name: Arpit
Room Name:
Patient Condition:
In/Out Patient ---Select---
Date of Admit: 12/11/2007 12:00:00 AM
Time of Admit: 09:41:45AM
Date of Discharge: 4/5/2005
Time of Discharge: AM
Day Stayed: 0
Advance: 1500
Doctor Charges: 500
Test Charges: 0
Room Charges: 0
Medicine Charges: 0
Extra Charges: 0
Total Charges: 0
Discharge
© Copyright Reserved Patient Management System
Done
Local intranet

System Maintainance

9.1. Introduction

The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural

Disaster is known as System Security.

System Security can be divided into four related issues:

- Security
- Integrity
- Privacy
- Confidentiality

SYSTEM SECURITY refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

DATA SECURITY is the protection of data from loss, disclosure, modification and destruction.

SYSTEM INTEGRITY refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

PRIVACY defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

CONFIDENTIALITY is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

9.2. SECURITY IN SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls:

CLIENT SIDE VALIDATION

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

- VBScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
- Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
- Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

SERVER SIDE VALIDATION

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

- Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.

- User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
- Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User-name, passwords and permissions are controlled o the server side.
- Using server side validation, constraints on several restricted operations are imposed.

CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in ASP.NET and VB.NET web based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with **"PATIENT MONITORING SYSTEM".** It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

BENEFITS:

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- It's a web-enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is

concerned. That is, we can sat that the project is user friendly which is one of the primary concerns of any good project.

- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
- Allocating of sample results becomes much faster because at a time the user can see the records of last years.
- Easier and faster data transfer through latest technology associated with the computer and communication.
- Through these features it will increase the efficiency, accuracy and transparency,

LIMITATIONS:

- The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
- Training for simple computer operations is necessary for the users working on the system.

FUTURE ENHANCEMENTS

- This System being web-based and an undertaking of Cyber Security Division, needs to be thoroughly tested to find out any security gaps.

- A console for the data centre may be made available to allow the personnel to monitor on the sites which were cleared for hosting during a particular period.

- Moreover, it is just a beginning; further the system may be utilized in various other types of auditing operation viz. Network auditing or similar process/workflow based applications...

DATA DICTONARY

After carefully understanding the requirements of the client the the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

tblEmployeeDetail

Column Name	Data Type	Length	Allow Nulls
EmpId	int	4	
Name	varchar	60	✓
Address	varchar	150	✓
Phone	varchar	20	✓
Email	varchar	25	✓
DutyTime	varchar	25	✓
LoginId	int	4	✓
Active	tinyint	1	✓

tblMedicine_Charges

Column Name	Data Type	Length	Allow Nulls
Patient_Code	varchar	50	✓
Medicine_Code	varchar	50	✓
Medicine_Name	varchar	50	✓
Date_on_Medicine_Gi	datetime	8	✓
Medicine_Charges	numeric	9	✓

tblCountry

Column Name	Data Type	Length	Allow Nulls
Country_Id	int	4	
Country_Name	varchar	50	✓
Country_Desc	varchar	100	✓

tblMedicine_Master

Column Name	Data Type	Length	Allow Nulls
Medicine_Id	int	4	
Medicine_Code	varchar	50	
Medicine_Name	varchar	50	✓
Price	decimal	9	✓
Mfg_Date	datetime	8	✓
Exp_date	datetime	8	✓
Company_Name	varchar	100	✓
Batch_No	varchar	50	✓
Quantity	numeric	9	✓

tblState

Column Name	Data Type	Length	Allow Nulls
State_Id	int	4	
State_Name	varchar	50	✓
State_Description	varchar	100	✓
Country_Id	int	4	

tblCity

Column Name	Data Type	Length	Allow Nulls
City_Id	int	4	
City_Name	varchar	50	✓
City_Description	varchar	100	✓
State_Id	int	4	✓

tblRoom_Charges

Column Name	Data Type	Length	Allow Nulls
Patient_Code	varchar	50	
Room_Code	varchar	50	
Date_on_Room_Giver	datetime	8	
Time_of_Room_Giver	varchar	50	
Room_Charges	numeric	9	

BIBLIOGRAPHY

- **FOR .NET INSTALLATION**

  www.support.mircosoft.com

- **FOR DEPLOYMENT AND PACKING ON SERVER**

  www.developer.com

  www.15seconds.com

- **FOR SQL**

  www.msdn.microsoft.com

- **FOR ASP.NET**

  www.msdn.microsoft.com/net/quickstart/aspplus/default.com

  www.asp.net

  www.fmexpense.com/quickstart/aspplus/default.com

  www.asptoday.com

  www.aspfree.com

  www.4guysfromrolla.com/index.aspx